Elicitation and management of user requirements in market-driven software development

Natt och Dag, Johan

2002

Elicitation and Management of User Requirements in Market-Driven Software Development

Johan Natt och Dag

LUND UNIVERSITY

Department of Communication Systems
Lund Institute of Technology

ISSN 1101-3931
ISRN LUTEDX/TETS--1054--SE+158P

Printed in Sweden
KFS AB
Lund 2002

To my mother, Catharina, and my sister, Ann.

This thesis is submitted to Research Board FIME – Physics, Informatics, Mathematics and Electrical Engineering – at Lund Institute of Technology (LTH), Lund University, in partial fulfilment of the requirements for the degree of Licentiate of Technology in Software Engineering.

Contact Information:

Johan Natt och Dag
Department of Communication Systems
Lund University
P.O. Box 118
SE-221 00 LUND
Sweden

Tel: +46 46 222 08 83
Fax: +46 46 14 58 23
E-mail: johan.nattochdag@telecom.lth.se
Web: http://www.telecom.lth.se/Personal/johannod

Abstract

Market-driven software development companies experience challenges in requirements management that many traditional requirements engineering methods and techniques do not acknowledge. Large markets, limited contact with end users, and strong competition forces the market-driven software development company to constantly invent new, selling requirements, frequently release new versions with an accompanying pressure of short time-to-market, and take both the technical and financial risks of development.

This thesis presents empirical results from case studies in requirements elicitation and management at a software development company. The results include techniques to explore, understand, and handle bottlenecks in the requirements process where requirements continuously arrive at a high rate from many different stakeholders. Through simulation of the requirements process, potential bottlenecks are identified at an early stage, and fruitless improvement attempts may be avoided.

Several techniques are evaluated and recommended to support the market-driven organisation in order to increase software quality and avoid process overload situations. It is shown that a quick and uncomplicated in-house usability evaluation technique, an improved heuristic evaluation, may be adequate to get closer to customer satisfaction. Since needs and opportunities differ between markets, a distributed prioritisation technique is suggested that will help the organisation to pick the most cost-beneficial and customer satisfying requirements for development. Finally, a technique based on automated natural language analysis is investigated with the aim to help resolve congestion in the requirements engineering process, yet retaining ideas that may bring a competitive advantage.

Acknowledgements

This work was partly funded by VINNOVA under grant for LUCAS – the Center of Applied Software Research at Lund University. Telelogic AB have provided invaluable support; their personnel have generously contributed with time and data.

I cannot fully express the gratitude I feel to the persons that have helped, supported, and inspired me the last two years. I am happy that the founder of our research group and my first supervisor, *Prof. Claes Wohlin*, attracted my attention to PhD studies in Software Engineering. And without *Assist. Prof. Björn Regnell*, my colleague, supervisor, coach and friend, I would never have managed this far. His deep knowledge, commitment, and support have certainly made my work exciting, inspiring, and fun.

Invaluable is also the support from *Assoc. Prof. Per Runeson*, head of our Software Engineering Research Group and my second current supervisor. He makes me enjoy working in and for our department as well as in the Center for Applied Software Research (LUCAS). Warm thanks to *Assist. Prof. Martin Höst* whose competence and experience in research inspire and help me. I would also like to thank all my colleagues who influence and help me, both at work and after work. In particular I would like to thank *Tekn. Lic. Håkan Petersson*, *Tekn. Lic. Thomas Thelin*, *Tekn. Lic. Enrico Johansson*, *Thomas Olsson*, *Josef Nedstam*, *Daniel Karlström*, *Lena Karlsson*, *Dr. Maria Kihl, Assist. Prof. Christian Nyberg* and *Tekn. Lic. Niklas Widell.* Many thanks to *Prof. Ulf Körner*, head of our department, for providing and enabling an open and inspiring environment, and to *Ingrid Nilsson*, who, always with a smile, makes my administration issues a whole lot easier.

Countless thanks to my industrial partners for their valuable contribution, making my work worthwhile and the research results more credible: *Per Beremark* at Appium AB, *Ofelia S. Madsen* at C-Technologies AB, *Michael Andersson* and *Thomas Hjelm* at Telelogic AB and *Dr. Joachim Karlsson* at Focal Point AB. Also, a very special thanks to all the researchers I have met around the world for inspiring discussions. In particular I would like to thank my colleagues and friends *Dr. Aybüke Aurum* and *Kerstin Lindmark*, who both have taken the time to proofread the introduction. Also thanks to *Prof. Pierre Nugues* for checking Section 1.4. Special thanks to *Dr. Pär Carlshamre*, *Åsa G. Dahlstedt* and *Dr. Anne Persson* for their assistance, constructive input and fruitful discussions. For an inspiring and excellent REFSQ'01 workshop I would also like to thank *Prof. Andreas L. Opdahl, Dr. Camille Ben Achour-Saliensi* and *Dr. Matti Rossi*.

I would like to express my warmest appreciation to all my friends who have supported me through joy and despair and who make my life richer. Thanks to *Ted* for listening to me. Thanks to *Ofelia* for her never-ending care. Thanks to *Lena & Jonas*, *Christina*, *Klara*, *Marie*, *Urban*, *Tobias, Pernilla* and *Kjell* for coping with me.

A billion thanks to *mum* for all your love and for helping me out when times have been difficult. Thanks to *Kaj* for taking care of her. And thanks to *Ann*, the greatest sister a brother can ask for.

Contents

List of papers

The following papers are included in the thesis:

[I] **Exploring bottlenecks in market-driven requirements management processes with discrete event simulations**
Martin Höst, Björn Regnell, Johan Natt och Dag, Josef Nedstam, & Christian Nyberg.
Journal of Systems and Software, 59, 323-332. 2001.

[II] **An industrial case study of usability engineering in market-driven packaged software development**
Johan Natt och Dag, Björn Regnell, Ofelia S. Madsen, & Aybüke Aurum. M. J. Smith, G. Salvendy, D. Harris & R. J. Koubek (Eds.), *Proceedings of HCI International: Vol 1. Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents and Virtual Reality* (pp. 425-429). Mahwah, NJ: Erlbaum. 2001.

[III] **An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software**
Björn Regnell, Martin Höst, Johan Natt och Dag, Per Beremark, Thomas Hjelm.
Requirements Engineering, 6, 51–62. 2001.

[IV] **A feasibility study of automated natural language requirements analysis in market-driven development**
Johan Natt och Dag, Björn Regnell, Pär Carlshamre, Michael Andersson, & Joachim Karlsson.
Requirements Engineering, 7, 20-33. 2002.

Related publications

The following papers are related but not included in the thesis:

[V] **Exploring bottlenecks in market-driven requirements management processes with discrete event simulation**
Martin Höst, Björn Regnell, Johan Natt och Dag, Josef Nedstam, & Christian Nyberg.
Paper presented at the Software Process Simulation Modeling Workshop, London, UK. July, 2000.
This paper is an earlier version of paper I. It was selected and extended for a special issue of *Journal of Systems and Software*.

[VI] **An industrial case study of usability evaluation**
Johan Natt och Dag, & Ofelia S. Madsen.
Master's Thesis, Report No. CODEN:LUTEDX (TETS-5390)/1-190/ (2000)&local 8. Lund, Sweden: Lund University, Department of Communication Systems. 2000.
This thesis is the basis for the research presented in paper II. It contains more elaborate background information and more research data.

[VII] **Visualization of agreement and satisfaction in distributed prioritization of market requirements**
Björn Regnell, Martin Höst, Johan Natt och Dag, Per Beremark, & Thomas Hjelm.
A. L. Opdahl, K. Pohl & M. Rossi (Eds.), *Proceedings of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality* (pp. 125-136). Essen, Germany: Essener Informatik Beiträge. 2000.
This paper is an earlier version of paper III. It was selected and extended for publication in a special issue of *Requirements Engineering*.

[VIII] **Evaluating automated support for requirements similarity analysis in market-driven development**
Johan Natt och Dag, Björn Regnell, Pär Carlshamre, Michael Andersson, & Joachim Karlsson.
C. Ben Achour-Saliensi, A. L. Opdahl, K. Pohl, M. Rossi (Eds.), *Proceedings of the Seventh International Workshop on Requirements Engineering: Foundations for Software Quality* (pp. 190-201). Essen, Germany: Essen Informatik Beiträge, 2001.
This paper is an earlier version of paper IV. It was selected and extended for publication in a special issue of *Requirements Engineering*.

[IX] **An industrial survey of requirements interdependencies in software release planning**
Pär Carlshamre, Kristian Sandahl, Mikael Lindvall, Björn Regnell, & Johan Natt och Dag.
Proceedings of the Fifth IEEE International Symposium on Requirements Engineering (pp. 84-91). Los Alamitos, CA: IEEE Computer Society Press. 2001.
This paper contains an automated analysis of interrelationships between requirements. The techniques used are those in Paper IV.

[X] **Requirements mean decisions! – Research issues for understanding and supporting decision-making in requirements engineering**
Björn Regnell, Barbara Paech, Aybüke Aurum, Claes Wohlin, Allen Dutoit, & Johan Natt och Dag.
Proceedings of the First Swedish Conference on Software Engineering Research and Practise (pp. 49-52) (Report No. 2001:10). Ronneby, Sweden: Blekinge Institute of Technology, Department of Software Engineering and Computer Science. 2001.
This paper presents research issues with focus on requirements engineering as a decision-making process.

[XI] **Market-driven requirements engineering challenges: an industrial case study of a process performance declination**
Robert Booth, Björn Regnell, Aybüke Aurum, Ross Jeffery, & Johan Natt och Dag.
A. Aurum & R. Jeffery (Eds.), *Proceeding of the Sixth Australian Workshop on Requirements Engineering* (pp. 41-47). Sydney, Australia: University of New South Wales, The Centre for Advanced Software Engineering Research. 2001.
This paper presents a second study of the requirements engineering process at Telelogic AB.

[XII] **Challenges in market-driven requirements engineering - an industrial interview study**
Lena Karlsson, Åsa G. Dahlstedt, Johan Natt och Dag, Björn Regnell, & Anne Persson.
Manuscript submitted for publication, 2002.
This paper presents preliminary results from a survey of market-driven sofware development companies in Sweden.

Introduction

Without much notice to the uninitiated, the importance of software has grown tremendously during the last 40 years. Industrial as well as developing societies and economies depend considerably on good-working software systems; power plants, hospitals, aviation, communications systems, cars – they all rely on software in order to work properly. And the dependencies keep growing. As the operations manager at Ericsson answers to the question on which competences they will need: "Radio skills are always good. Signal processing, microwave techniques. But there will also be even more software" (Ahlbom, 2002).

Still, software engineering is a young practice and an even younger research area compared to most engineering disciplines. Up until just about a decade ago it was not even considered a true engineering discipline (Shaw, 1990). After the software crisis in the late sixties, the software engineering community has theorized and promoted a vast number of models, methods, techniques, and guidelines to aid software developers to handle the difficulties in developing increasingly complex software systems with acceptable quality.

Research and industry have faced many successes, but the growth in usage and complexity of software systems keep revealing new challenges at a pace that seems hard to keep up with. Although the current body of knowledge in software engineering is quite extensive, novel ideas, whether based on previous convictions or not, are still needed.

As software systems are developed in order to support and aid the human being, those systems are assumed to do precisely that. Unfortunately, it is still a rare case to find a completely satisfied end user. One reason is that the quality level that is considered acceptable is dependent on both the usage and the application domain. End users fall into different categories having different demands. For example, the technical end user may accept a less appealing graphical user interface, provided that the functionality is satisfactory, while the non-technical end user may also demand that the software system is easy to learn and easy to use.

If software is developed for a single user or a single company the diversity issue may be less problematic, yet present. The customer is well-defined and consensus between customer and developer may eventually be reached through negotiation. The difficulties, however, may become more clear for companies developing software for large markets that comprise many different kinds of users: occasional and frequent users, technical and non-technical users, novice and expert users, etc. The variety in users' needs calls for a balance between the needs taken into consideration and those rejected. This balance is extremely hard to reach as it is influenced by a number of factors: users' needs constantly change, users never become satisfied, new technologies are misjudged, believes turn into truths, time is critical, and timing is crucial, to mention a few.

This thesis concentrates on large-scale software development for large markets and how developing companies in the market-driven situation may find a representative collection of users' needs. It shows that this collection must and may be reduced to a reasonable satisfactory and manageable set of needs that the software shall fulfil – satisfactory on behalf of the end user and manageable on behalf of the developing company.

This introduction is organised as follows: In Chapter 1, the focus of the presented research is described, and the specific concepts addressed in the thesis are introduced and explained. In Chapter 2 the practised research methodology is presented and the research methods and questions and validity issues are further described. A summary of the research results, main contributions, and the identified threats to validity, together with the abstracts of the papers included in the thesis, are presented in Chapter 3. In Chapter 4 several issues for further work are suggested and a plan is presented covering at least two years of research impelled by the results from this thesis.

1. Research focus

The research and associated results presented in this thesis apply to the field of *software engineering*, in which methods, techniques, and tools are utilized to overcome the challenges in development and maintenance of complex software systems (Sommerville, 2001). About 15 years ago a sub-discipline within software engineering emerged due to specific challenges in handling customers' wishes and needs (Sommerville & Sawyer, 1997). The sub-discipline, termed *requirements engineering* (RE), mainly focuses on the first stage of software development where customers' wishes and needs, i.e. the *requirements*, are collected, analysed and selected before proceeding with software design, implementation, verification and validation.

The last years, a new form of development called *market-driven development* or *packaged software development* (Sawyer, Sommerville, & Kotonya, 1999), have gained increasing importance as software development companies turn to new and larger markets. The approach affects requirements engineering techniques, as there is very limited negotiation with end users. Instead, many requirements have to be invented within the developing company (Potts, 1995). The absence of negotiations may also be desirable. The market-driven development company often have competitors that may only be defeated by secretly developing successful solutions. The competition also puts a schedule constraint where short *time-to-market* is crucial (Sawyer, 2000). Constantly striving to be ahead of competitors, the market-driven development company therefore frequently delivers new and improved releases of a software system in order to keep old customers satisfied and to win new ones (Potts, 1995; Carlshamre & Regnell, 2000).

The characteristic differences between traditional, or bespoke, software development and market-driven software development have been summarized by Carlshamre (Carlshamre, 2002). His findings (derived from Kamsties, Hörmann, & Schlich, 1998; Keil & Carmel, 1995; Lubars, Potts, & Richter, 1993; Novorita & Grube, 1996; Potts, 1995; Yeh, 1992) with further additions (from Lubars et al.; Robertson & Robertson, 1999) are found in Table 1. From the table it can be found that fundamental organisational issues, such as the primary goal, the success measurements and the product life cycle, are very unlike. The differences are so all-pervading that many traditional requirements engineering practices are unusable for the market-driven company. In

Table 1. Comparison of traditional software development and market-driven software development characteristics (based on Carlshamre, 2002; Lubars et al., 1993; Robertson & Robertson, 1999).

Characteristics	Bespoke development	Market-driven development
Primary goal	Compliance to requirements specification.	Time-to-market.
Measure of success	Satisfaction, acceptance.	Sales, market share, product reviews.
Life cycle	One release, then maintenance.	Several releases, as long as there is a market for the product.
Requirements conception	Elicited, analysed, validated.	Invented.
Requirements specification	Used as a contract between customer and supplier.	Rarely exists or much less formal. Requirements communicated verbally.
Users[a]	Known or easily identifiable.	Difficult to identify or initially unknown.
Customer[a]	Software orderer. Contract negotiator.	Agents for different markets. Key customers may get tailored software.
Physical distance to users	Usually small.	Usually large.
Main stakeholder	Customer organization.	Developing organization.
Specific RE issues	Elicitation, modelling, validation, conflict resolution.	Managing a steady stream of new requirements. Prioritizing, cost-estimating, release planning.
Developer's association with the software	Short-term (until end of project).	Long-term, promoting e.g. investment in maintainability.
Validation	Ongoing process.	Very late, e.g., at trade affairs.
RE standards and explicit methods	More common.	Rare.
Iterative techniques	Less common.	More common.
Domain expertise available on the development team	More common.	Less common (product development often breaks new ground).

a. The terms *user* and *customer* are here further elaborated compared to Carlshamre (2002).

Section 1.1, the characteristics of requirements engineering in market-driven software development and related challenges are further elaborated.

One of the specific RE issues (see Table 1) is the steady stream of incoming requirements. For large companies, the flow may average up to several requirements a day, arriving from many different stakeholders in the organisation (Regnell, Beremark, & Eklund, 1998). This puts a high pressure on the people responsible for analysing and finding 'good' requirements. Bottlenecks are likely to appear in the requirements management process, and exposing these bottlenecks before they appear is highly desirable. One way to expose potential bottlenecks may be to simulate the current process. By using historical data about the process and by building a model of the actual process, simulations may reveal what will happen in the future. In Paper I, discrete-event simulation (Banks, Carson, & Nelson, 1996) is used to provide a better understanding of a requirements engineering process and to unveil its potential bottlenecks. The possibilities and advantages of simulation are further elaborated in Section 1.3.

Since one of the challenges of developing software for larger markets is to satisfy the end user albeit contact with the end user is limited, there is a need for techniques that validate that the product is usable and help to reveal tasks that the end user may feel cumbersome or some functionality that the end user may feel is missing. Therefore, for the market-driven organisation, usability evaluation techniques that require no actual end users are highly desirable. From the field of human-computer interaction, *usability engineering* has emerged to address the specific issues concerned with the often-misunderstood concept of usability. A number of methods and techniques have been proposed, of which several are difficult to adopt in the market-driven development organisation (Natt och Dag & Madsen, 2000). In Paper II this has been addressed by evaluating a slightly improved heuristic evaluation (Nielsen, 1994) to see if it may give valid results when conducted in-house. The results are compared with results from a survey using the SUMI questionnaire (Kirakowski & Corbett, 1996), in which end users are asked about their opinions and feelings about the software application. In Section 1.2, usability engineering is further motivated and explained, as well as the term usability.

Usability evaluations may not be sufficient to determine what will satisfy end users and customers. For companies developing software for a

worldwide market place, needs and opportunities may differ between market segments, and collecting these concerns would provide valuable support for decision-making. The information may make it easier for the developing organisation to comply with business goals and to please customers and end users. Paper III presents techniques to first let each market provide a prioritized list of requirements and then visualize the differences and similarities in these priorities.

More efficient processes may be reached in different ways. One way is to relieve the burden on people working in the process (see Paper I). Having easy-to-use, supportive computer tools that automate certain tasks is a dream that many companies would love to see coming true. In market-driven development companies, requirements are often managed using a database in which requirements are entered in natural language. Consequently, automation of tasks related to natural language may be

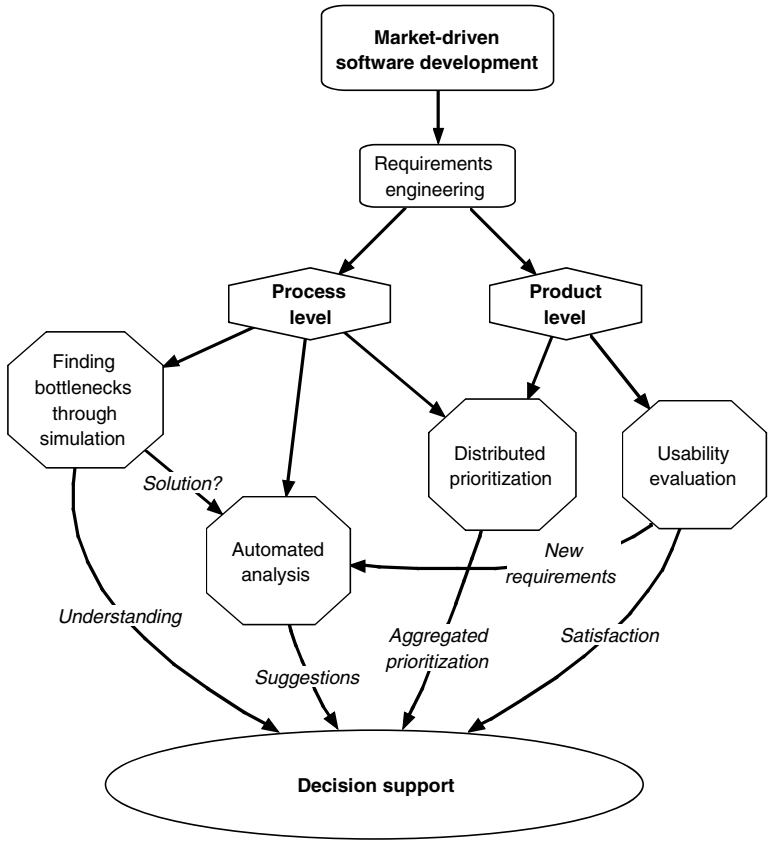

Figure 1. *Concept map of the research focus.*

addressed through natural language processing techniques. By automatically analysing requirements based on linguistic content, potential duplicates may be found and *suggested* for removal. The control should still be in the hands of the requirements analyst. In Section 1.4 natural language processing is described and related techniques are presented, and in Paper IV, techniques from the field of information retrieval, related to natural language processing, are tested on actual requirements from the industry.

The research focus is illustrated in Figure 1 using a concept map. It shows how the different topics addressed in this thesis are related and that they all aim at giving decision support in requirements engineering.

The above review of the research focus of this thesis reveals the multi-disciplinary property of software engineering. It is the author's belief that software engineering in general and requirements engineering in particular may benefit from applying well known principles from other research fields. The presented research may, at least partially, support this belief.

1.1 Requirements engineering

When requirements engineering still was in its infancy, the discipline stipulated to write a perfect specification describing what the resulting software system should accomplish. "Perfect" essentially meant fulfilling the quality attributes listed in Table 2 (Davis, 1993), and obliged developers to be very rigorous. For example, to make the requirements specification both understandable by the customer and complete can take some time with a complex system. The tackling was nevertheless wise since one reason for the software crisis was a lack of control when systems became too complex. By accurately describing what the system should do, the requirements specification could act as an agreement, and even as a formal contract, between the customer and the software developer. Solutions were at first banned from the specification and it had to be finalized before any successive work in the development process was initiated. This *waterfall development process model*, illustrated in Figure 2, was a first solution to the chaotic development situation and was strongly advocated.

The basic idea of the waterfall model is still used but the model has gone through many refinements and is now mainly a constituent of other models. For example, a new compound development strategy, extreme

Table 2. Quality attributes for the software requirements specification (derived from Davis, 1993).

Attribute	Description
Correct	Every requirement represents something required by the system.
Unambiguous	Every requirement has only one interpretation.
Complete	Everything the software is supposed to do is included.
Verifiable	There exists a cost-effective process with which a person or machine can check that the actual as-built software product meets every requirement.
Consistent	No requirements in a given subset within the specification conflict with each other.
Understandable by customers	Requirements should be negotiated in a form that suits the customer or user who usually do not understand formal methods.
Modifiable	Structure and style are such that any necessary changes to the requirements can be easily, completely, and consistently executed.
Traced	The origin of each requirement is clear.
Traceable	The specification is written to facilitate referencing of each requirement.
Design independent	The specification does not imply a specific software architecture or algorithm.
Annotated	The necessity of each requirements is denoted essential, desirable or optional. Volatility is indicated by a textual annotation.
Concise	Given two specifications of the same system, each exhibiting identical levels of all the above qualities, the shorter specification is the better one.
Organised	Requirements are easy to locate.

programming (XP), incorporates parts of the waterfall model in extremely small increments (Beck, 2000). The benefits of the waterfall model are utilized, such as its straight-forwardness, while some of the drawbacks are avoided, such as the need for heavy documentation and the lack of support for parallel activities, user involvement, and quick results.

In line with the refinement of the waterfall model and due to new and changing software development paradigms, the attitude arguing that all the quality attributes should be fulfilled has lost ground. The incremental

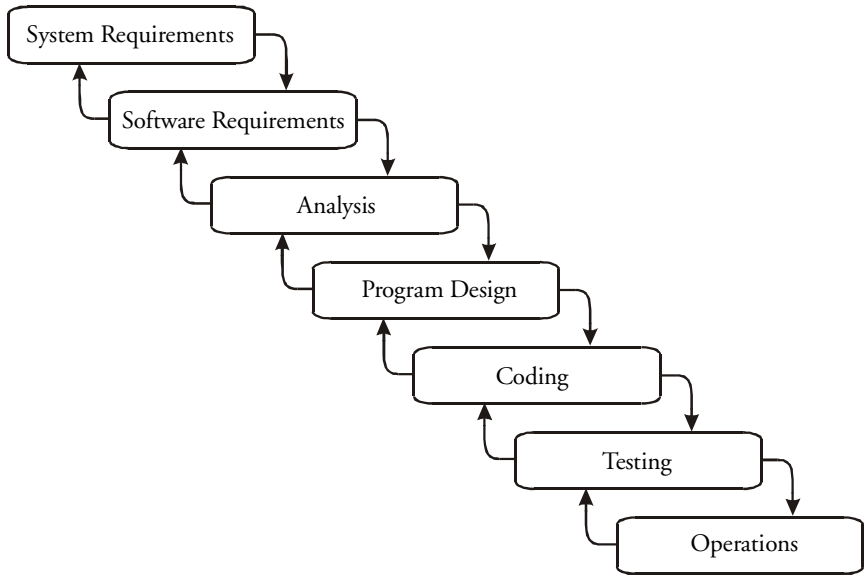

Figure 2. *The classical waterfall development process model where software requirements are to be specified in a first, separate stage (Royce, 1970).*

and evolutionary development models have affected requirements engineering and it has been realised that completeness sometimes is impossible to achieve (Siddiqi & Shekaran, 1996; Goguen, 1996). Yet, requirements quality is considered important to enable full control of the development. And the initial objectives of requirements engineering remain:

1. To understand the problem that the system is supposed to solve
2. To select and document the requirements on the system

Requirements engineering in market-driven software development

Requirements engineering in market-driven development have difficulties with the majority of the quality attributes in Table 2 due to its specific characteristics (see Table 1). There are three main, interlinked reasons: (1) the time constraint, (2) the constant arrival of new requirements during the whole development process, and (3) the need to promptly deliver new improved software releases.

The time-to-market constraint may insist upon requirements being implemented before all quality attributes have been properly checked. A

quality attribute such as completeness is not always prioritized when requirements at an early stage are found to bring competitive advantage.

Requirements continuously arrive from several different sources, called *stakeholders*, such as the marketing department, usability architects, support, developers, etc. This makes it virtually impossible to write a correct requirements specification before proceeding to a subsequent phase in the development process. Rather, requirements are stored in a database and there is a strong focus in the requirements engineering process on managing the evolution of requirements and assuring their quality. Although it is difficult, the person responsible for managing the requirements, the *requirements manager*, tries to make sure that all requirements live up to the quality attributes to a reasonable extent. Requirements management has three main concerns (Kotonya & Sommerville, 1997):

- managing changed and agreed requirements
- managing relationships between requirements
- managing dependencies between requirements and other documentation produced during the software engineering process

A European survey of 4,000 companies has shown that management of requirements was one of the major problem areas in software development (ibid.). Thus, research in requirements engineering may, most likely, still be needed. For the market-driven development organisation, the particular need of proper management of requirements is more obvious than for traditional requirements engineering (see Table 1).

For the companies to stay ahead of competitors, new versions of the software have to be released as soon as there is a major improvement available. Again, the time-constraint leads to special demands. It is sometimes desirable to release new versions more frequently than it is possible, with acceptable quality, to develop. The company being the target of the research presented in this thesis has solved this by scheduling the activities in the development process in parallel (Regnell et al., 1998). In Figure 3 the requirements management processes is showed. By pipelining the releases, requirements management can be a continuous activity and releases may be delivered more often. The figure shows that it takes 14 months to develop a new release, while a new release nevertheless can be delivered every six months.

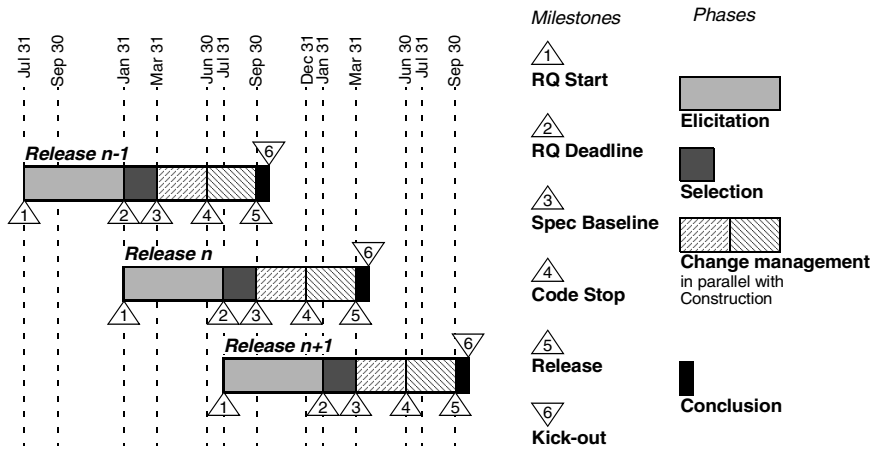

Figure 3. *The REPEAT requirements management process enabling a new release every six months although development of a release takes 14 months (Regnell et al., 1998).*

Still, the solution to the release-delivery problem unfortunately makes the situation even worse for the requirements manager. The process makes it possible to postpone incoming requirements as well as sending important requirements to the release currently implemented. The difficulty is to decide which requirements are to be developed in the current release, which requirements should be postponed, and which requirements should be regarded so important that they are to be selected for implementation into the release soon to be delivered.

To aid the decisions, requirements are given different priorities, which implies that they must at least have been partially analysed. Requirements prioritization may be conducted in at least two ways:

- *Direct assignment*
  A predefined scale is used to classify requirements as they are analysed. For example, a scale ranging from 1 to 3 may be used to classify requirements as having high, medium or low priority.

- *Pair-wise prioritization*
  Two requirements are compared at a time to make a fairly accurate judgement about which one should be prioritized above the other. In one systematic approach, the analytical hierarchy process (Saaty, 1980), all requirements are pair-wise compared to one another, needing $n \cdot (n-1)/2$ comparisons for n requirements. The result is a prioritized list. The method has shown to be efficient, informative, and accurate for prioritizing software requirements (Karlsson,

1998). It is also possible to randomly neglect half of the prioritizations without significant accuracy loss (Carmone, Kara, & Zanakis, 1997).

In Paper III a variant of the first, prioritization using fictitious money, is used in a distributed setting. Several stakeholders are asked to prioritize the same set of high-level requirements and the different prioritizations are then consolidated in order to find an appropriate prioritization with respect to business goals and user satisfaction.

Unfortunately, many of the general challenges in traditional requirements engineering are adopted by the market-driven development organisation, e.g. requirements are erroneous, errors are detected late, ambiguities are difficult to resolve, etc. In addition, there are several new challenges that may be subjects for further research.

1.2 Usability engineering

Usability engineering has its roots in the concept of man-machine interaction (MMI), a term that was more frequently used after the technology explosion of the 1970's. Aspects such as psychology were given new ground and became a general concern to both researchers and system designers. MMI begun the exploration of the potential to make user-friendly systems but was far too narrow in identifying the central concerns for creating these systems. In practice, user-friendliness all too often meant tidying up the screen displays and making them more aesthetically satisfying (Preece et al., 1994).

In the mid-eighties, when computers were introduced on a wide front, there was a natural shift from MMI to a new field named human-computer interaction (HCI). HCI developed as a way of focusing on the specific interaction between humans and computers and a wide range of subjects became part of the development of HCI, such as psychology, cognitive science, and sociology, together with the more traditional subjects: computer science, computer engineering, and graphical design. This made HCI an interdisciplinary subject and extremely difficult to master (Concejero et al., 1996).

From the comprehensive field of HCI a new field emerged called *usability*, in which the primary focus is on the users and their acceptance of computer systems. Many different textual definitions of usability exists

(ISO; IEEE), but it is preferably defined through five usability attributes (Nielsen, 1993; Wixon & Wilson, 1997):

- *Learnability*
  Learnability is the degree of how easy it is to learn how a new software system works. In the market-driven software development organisation this may be a crucial usability attribute to be acknowledged when turning to new markets. It is important to recognize that the learning of a system usually takes place during usage.

- *Rememberability*
  The user should have little problem remembering how the software is operated. Many users fall into the category of "sporadic users", who are neither experts nor novices. They use the software from time to time, irregularly, and the software should help the user remember how a certain task is performed.

- *Efficiency*
  When the user has learnt how to operate the software, productivity should be as high as possible. Further, the application must be perceived to execute fast enough to keep up with the user. The feature is denoted *perceived* execution speed, since it does not necessarily refer to the actual speed but rather to the extent to which the user feels that she is in control.

- *Reliability*
  If the user nevertheless makes mistakes, the software should be able to recover from them. The most severe kinds of errors are those not discovered by the user, leading to inadequate work or destroyed data. The software should help the user avoid these situations.

- *Satisfaction*
  The user should be pleased when using the software, and should be subjectively satisfied. The satisfaction attribute is often used in software reviews as it states how pleasant the user may think the software is to use. Satisfaction is a purely subjective attribute.

The definition, using the attributes above, should help make it clear that usability covers much more than just the graphical design of the user interface. Still, usability is just one part of an even bigger picture, which is

best explained through *system acceptability*, a concept which is described below.

System acceptability

To be able to sell a product there has to be some users who have needs that the product fulfills. If not, the product will likely not be on the market for very long. If the user is satisfied, she accepts the way the software is developed or designed. This may be referred to as system acceptability (Nielsen, 1993).

System acceptability comprises many different attributes. A model of these attributes is shown in Figure 4, in which the usability attribute is found as well. From this viewpoint, a software system is acceptable when it is both socially acceptable and practically acceptable. For example, software may not be socially acceptable if it is intended to find out who of the company employees uses the bathroom or the coffee machine most times during a day. It may nevertheless be considered practically acceptable if it is excellent in its performance of identifying the user and reporting it correctly.

Practical acceptability can then be further investigated and be found to constitute attributes well-known to the software developer, such as cost, compatibility, robustness, etc., in addition to usefulness. Usefulness comprises utility and usability, and is basically the issue of whether the software can be used to achieve a desired goal. Utility, in turn, is the question whether the functionality of the system can do what is needed, and finally, usability is the question of how well users can use that functionality. Thereby, usability applies to all aspects of the software system with which a human interacts (Nielsen, 1993).

Usability also affects and is affected by the functionality of the software product. Software is usually developed with a certain kind of functionality and new versions are released with what is believed to be increased functionality, new features, and improved features. What has to be remembered is that the user must also be able to use that functionality. Still, it does not matter if usability is increased if there is not enough functionality. Without the functionality the software will not be usable anyway. Thus, functionality and usability are complementary characteristics (Goodwin, 1987).

For the market-driven organisation there is an important economical aspect. If a product cannot be sold no money will be made. If the product

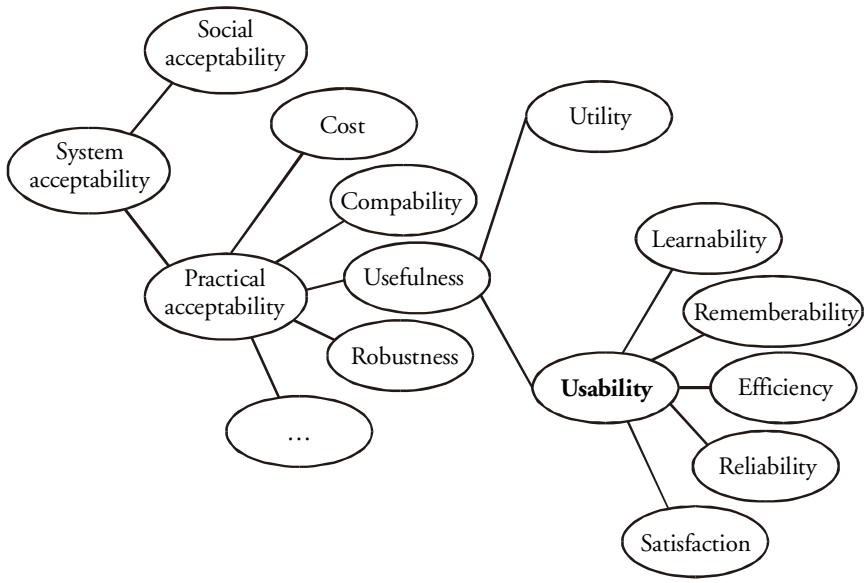

Figure 4. *Model of attributes of system acceptability (Nielsen, 1993)*

is not good enough it will not sell. The methods and techniques from experiences of usability engineering help to make a better product and also save money (Bias, 1994).

Paper II concentrates on two usability evaluation techniques that are very easy to use. They give complementary data, the first providing input to the requirements engineering process in the form of new requirements, and the second bringing quantitative data to confirm the focus of the suggested improvements.

1.3 Process simulation

Simulation may be applied to a vast number of areas to imitate the operation of a real-world process or system over time (Banks et al., 1996). It may be executed either by hand or by computer to generate artificial historical data, which is observed to investigate the plausible behaviour of the real system. The behaviour is studied by developing a simulation model, which describes the system through mathematical, logical and symbolic relationships between objects in the system that are of interest. A useful model always simplifies and idealizes and the boundaries between the system and the model are rather arbitrary defined. However, the usefulness is dependent of the possibility to practically determine all its

relevant behaviour: analytically, numerically, or by running the model with certain inputs and observe the outputs (Bratley, Fox, & Schrage, 1987).

The purposes of simulation are many (Naylor, Balintfy, Burdick, & Chu, 1966, pp. 8–9; Banks et al., 1996, p. 4):

1. Enable the study of, and experimentation with, the internal interaction of or within a complex system.
2. Simulate informational, organisational and environmental changes and observe the effects of alterations.
3. Provide knowledge from designing a simulation model that may be of great value towards suggestion of improvements to the system.
4. Obtain insight into the questions of which variables are most important and how variables interact.
5. Pedagogically reinforce analytical solution methodologies.
6. Experiment with new designs or policies prior to implementation, so as to prepare for what may happen.
7. Verify analytical solutions.

The appealing property of simulation, to mimic what does or may happen in a real system, makes it an attractive approach with several benefits (Pegden, Shannon, & Sadowski, 1995, p. 9):

1. New policies, operating procedures, decision rules, organizational structures, and the like, can be explored without disrupting ongoing operations.
2. New hardware designs, physical layouts, software programs, transportation systems, etc., can be tested before committing resources to their acquisition and/or implementation.
3. Hypotheses about how or why certain phenomena occur can be tested.
4. Time can be controlled; it can be compressed, expanded, etc., allowing to speed up or slow down a phenomenon for study.
5. Insight can be gained about which variables are most important for performance and how these interact.

6. Bottlenecks in material, information and product flow can be identified.

7. A simulation study can prove invaluable to understanding how the system actually operates as opposed to how everyone thinks it operates.

8. New situations, about which there is limited knowledge and experience, can be manipulated in order to prepare for theoretical future events. Simulation's great strength lies in its ability to enable the exploration of "what if" questions.

The research presented in Paper I, involving modeling and simulation of a requirements process, mainly aimed at studying the internal interaction within the requirements process (purpose 1) and to simulate the effects of informational and organisational changes to the process (purpose 2). A primary goal was also a better understanding of the system in order to suggest improvements (purpose 3). Finally, by showing how a simulation model of the requirements process may look like, the organisation under study was enabled to experiment prior to implementation (purpose 6). The identified advantages for choosing simulation were the possibility to explore the information flow and new process policies (advantage 1), to reveal bottlenecks (advantage 6), to understand how the process behaved (advantage 7) and to answer what would happen if certain changes were made in the process (advantage 8). More information may be found in Paper I.

However, simulation also has a few disadvantages (Pegden et al., 1995, p. 9). Firstly, model building requires special training and experience. Two models that are constructed by two competent individuals may have similarities but is highly unlikely to be the same. Secondly, simulation results may be difficult to interpret, as it may be hard to determine whether the output depends on randomness or system interrelationships. Thirdly, simulation modeling and analysis can be time consuming and expensive. If enough resources are not assigned, the model or analysis may be insufficient. Fourthly, a disadvantage identified by Banks et al., 1996, p. 5), simulation is sometimes used when an analytical solution is possible, or even preferable. Solvable queuing models may be used in some circumstances.

Thanks to vendors of simulation software, model packages and thorough analysis tools are available to address the disadvantages. Also,

simulation may continually be performed even faster, thanks to advances in hardware.

Simulation models

Simulation models may be classified in several ways. One general classification scheme is shown in Figure 5 (Gordon, 1969; Banks et al., 1996). The two main models are the *physical* and the *mathematical*. The physical model is self-explanatory; the mathematical model uses symbolic notation and mathematical equations to represent a system. These models may then be further classified into *static* or *dynamic*. The static simulation model, also referred to as Monte Carlo simulation, represents a system at a particular point in time, i.e. time is not a variable. In contrast, the dynamic model represents systems as they change over time. A mathematical model may be dealt with either *numerically* or *analytically*. The analytical approach is mostly used for optimization models to solve a problem. Simulation models, on the other hand, are 'run' rather than solved.

A third important distinction is made when describing simulation models to tell whether the model contains random variables or not. A model is said to be *deterministic* if it has a known set of inputs, which

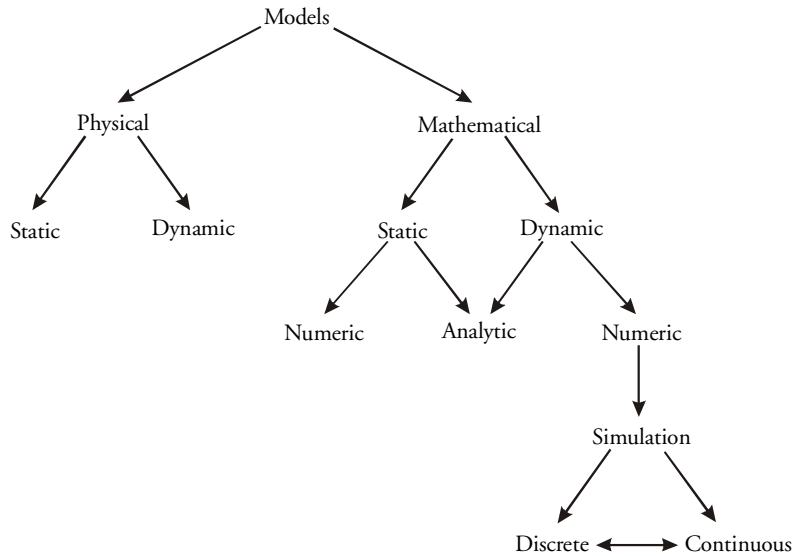

Figure 5. *General classification scheme of simulation models (combined from Gordon, 1969; Banks et al., 1996)*

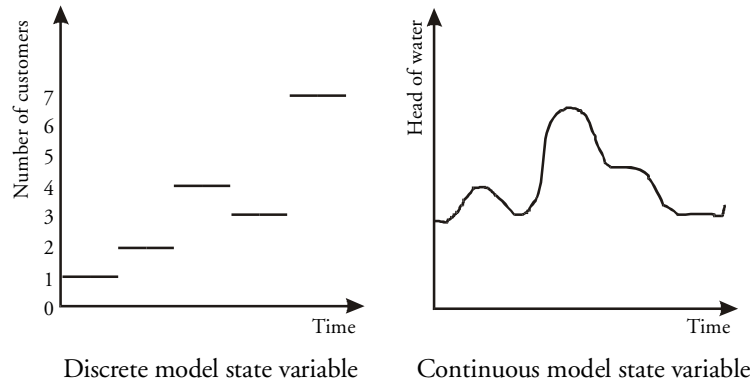

Figure 6. *Difference between the discrete and the continuous simulation models (Banks et al., 1996)*

result in a known set of outputs, i.e. no random variables. The opposite model, the *stochastic* simulation model, has one or more random variables. The output is then also random and must be treated accordingly as statistical estimates of averages over time.

Finally, models may be classified into *discrete* or *continuous* models. In a discrete simulation model, changes in the simulation happen only at discrete points in time. In the continuous simulation model, the variables change continuously over time. The difference is illustrated in Figure 6. Although the models are categorized into discrete and continuous, it is rare to find systems in the real world to fall exclusively in these two categories. "But since one type of change predominates for most systems, it will usually be possible to classify a system as being either discrete or continuous" (Law & Kelton, 1991, p. 4). Discrete models are not always used to model discrete systems, continuous models are not always used to model continuous systems, and simulation models may even be integrated (Zeigler, Praehofer, & Kim, 2000).

For the research presented in this thesis computers were utilized to simulate a discrete, dynamic and stochastic queuing network model of a requirements process. Queuing models may be used to calculate performance in systems that can be described through a number of servers and associated queues and are very well suited to reveal congestion (Banks et al., 1996; Gordon, 1969). An illustration of a simple queuing system is found in Figure 7, where a single server is serving the units of a calling population. When the server is busy, the units in the calling population, one at a time in a random fashion, joins the waiting line. The system is described by its calling population, the nature of arrivals and servers, the

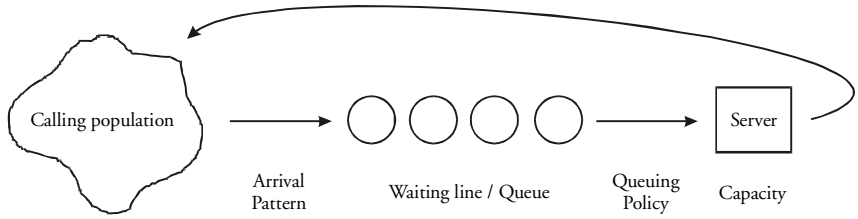

Figure 7. *A simple queuing system.*

capacity of the system (how many units can be served simultaneously), and the queuing discipline (how is the next unit to be served selected). Simple queuing models may, as mentioned when reviewing the disadvantages of simulation, be solved analytically but when they become too complex, simulation is the only way to draw proper conclusions about the system. For further details about the queuing network model used in the research, see Paper I.

1.4 Natural language processing

Language processing techniques emerged during the Second World War when computers were utilized to break message codes (Jurafsky & Martin, 2000). Since then a number of overlapping fields have emerged: computational linguistics, natural language processing (NLP), speech recognition, and computational psycholinguistics. Although the fields have more or less merged, NLP, which emerged in the field of computer science, is the main interest of the research presented in this thesis.

The quest of NLP is to deliver well-engineered systems that rely on the use of natural language. Such systems may serve as front-ends to databases and allow users to enter queries in natural language rather than having to learn a database query language, or to produce programs automatically from a natural language description. This is accomplished through the use of well-defined concepts within linguistics and computer science:

- *Morphology*
  The study of the meaningful components of words.

- *Syntax*
  The study of the structural relationships between words.

- *Semantics*
  The study of meaning.
- *Pragmatics*
  The study of how language is used to accomplish a goal.
- *Discourse*
  The study of linguistic units larger than a single utterance.

The task in language processing is to resolve ambiguity in these categories. This is accomplished through lexical or syntactic disambiguation, which utilizes various techniques, such as part-of-speech tagging, word sense disambiguation and probabilistic parsing (Jurafsky & Martin, 2000).

For the market-driven development organisation, NLP may be a means of finding support in handling requirements. Requirements are still often written in natural language and although many requirements activities rely on verbal communication and negotiations, requirements carry information that somehow must be managed. Further, it becomes more and more common for market-driven organisation to store requirements in some form of repository, i.e. a database. Thus, NLP techniques may be of even higher interest in such settings.

Although the boundaries are fuzzy, natural language processing may be subdivided into statistical and non-statistical NLP. Statistical NLP can be said to comprise all quantitative approaches to automated language processing, including probabilistic modeling, information theory, and linear algebra (Manning & Schütze, 2000). Further, for the purposes of the research presented in this thesis, yet another closely related field has been explored: information retrieval (IR). Information retrieval primarily deals with the representation, storage, organisation of, and access to information items (Baeza-Yates & Ribeiro-Neto, 1999). Thus, information retrieval is a highly interesting field when trying to manage requirements.

In traditional information retrieval, the information items, called documents, are indexed to provide a foundation for retrieving the particular document. The index consists of index terms, which in a restricted sense are keywords or groups of related words. By querying an information retrieval system, i.e. requesting a document by submitting one or more keywords, relevant documents are returned based on a match between the submitted queries and the available index terms. The described scenario is a rather simplified illustration of an information

retrieval system. Behind the curtains, advanced ranking algorithms are used to return the most relevant document.

The matching techniques in information retrieval may be used to compare requirements and reveal similarities and relationships between requirements. By complementing these techniques with semantic parsing techniques from NLP, a rather high matching precision may be reached. Unfortunately, the available techniques have not solved many of the problems of NLP and a variety of approaches have been taken to try to automatically handle requirements. An extensive survey of related work within this field may be found in Paper IV. Progress has been made and there is a better understanding of some of the limitations of the methods that have been used. Thus, research will move on to investigate other promising techniques and application areas.

Paper IV presents a baseline of an evaluation of duplicate identification using a rather simplistic lexical analysis of requirements. Although simplistic, a surprising accuracy can be reported. The evaluation scheme in the paper and the baseline can be used in evaluations of future improved approaches.

2. Research methodology

The research presented in this thesis has mainly been conducted using an engineering approach, where situations have been observed and better solutions have been proposed and evaluated. It is based on previous findings in the study of a market-driven software development company, which was struggling with process bottlenecks (Regnell et al., 1998) and usability competition (Natt och Dag & Madsen, 2000). A number of research issues, based on those findings, were formulated with one fundamental vision in mind: to support the market-driven software development organisation to timely deliver products that satisfy end users. The purpose was thus to explore, describe and explain the market-driven software development strategy, discover if improvements may be necessary and, if they are, decide on which improvements that may be rewarding.

The vision and the research focus was used to formulate relevant research questions, of which the ones addressed in this thesis are found in the next section. With the research questions as the guide, research projects were designed using both *fixed* and *flexible* design strategies. In a fixed strategy, which is also referred to as the *quantitative* approach, the

design is finished before data collection begins and the data collected is usually in the form of numbers. In contrast, the flexible design, also referred to as the *qualitative* approach, evolves during data collection and usually involves collection of non-numerical data (Robson, 2002[1]).

The fixed and flexible design strategies may be further classified. It is virtually impossible to cover for all possible forms of enquiry, but the following traditional research strategies are widely recognized (ibid.):

Traditional fixed design research strategies

1. *Experimental strategy*
   A small number of variables are measured and others are controlled. The researcher actively and deliberately introduces some form of change in the situation, circumstances or experience of participants with the view to producing a resultant change in their behaviour.

2. *Non-experimental strategy*
   A small number of variables are measured while others are controlled. The research does not try to change the situation, circumstances or experience.

Traditional flexible design research strategies

1. *Case study*
   A single 'case' or a small number of related 'cases' are studied to develop detailed, intensive knowledge. The study is made in the context of the case.

2. *Ethnographic study*
   How a group, organisation or community live, experience and make sense of their lives and their world is captured, interpreted and explained.

3. *Grounded theory*
   A theory is generated from data collected during the study.

Research in software engineering is young and the subject is cross-disciplinary. Therefore several research approaches and methods have been adopted from other fields. Attempts have been made to characterize

1. The 2002 edition of *Real World Research* by Colin Robson has been thoroughly revised.

research in software engineering but the picture is still not as clear as in many other, more mature research fields (Shaw, 2002). Thus, software engineering research methodology consensus is still to be reached. A part of the problem lies in defining the boundaries of the field, which differ from typical engineering research that builds on clearly defined scientific principles. In a 1989 workshop, four methodologies were identified to address this problem (Adrion, 1993; Glass, 1994; Wohlin et al., 2000):

- *Scientific method*
  The world is observed and a model or theory of behaviour is proposed based on the observations. The proposition is measured and analysed, and hypotheses of the proposition are validated. If possible, the procedure is repeated.
- *Engineering method*
  Existing solutions are observed and better ones are proposed. The propositions are realised and then measured and analysed. Until no further refinements can be made the procedure is repeated.
- *Empirical method*
  A model is proposed and measured and then analysed using statistical methods. The model is validated and the procedure is repeated.
- *Analytical method*
  A formal theory, or a set of axioms, is proposed from which a method is developed. Results are derived and, if possible, compared with empirical observations.

The software engineering researcher typically seeks better (e.g. more efficient, faster, less cumbersome, etc.) ways to develop and evaluate software of acceptable quality, and motivated by real world problems, solutions are sought that are also applicable to the real world. The research presented in this thesis has mainly been conducted using the engineering method. However, one singular label does not fully cover for the approach taken. In addition to the engineering method, both the scientific method and the empirical method have contributed to the research in this thesis. A further classification of the research is found in Section 2.3.

2.1 Research questions

The research questions that have impelled the research presented in this thesis are listed below in the order they are addressed in the thesis. The research and accompanying research questions focus on requirements elicitation and management in the market-driven software development situation.

RQ1. Can discrete-event simulation be used to identify potential, future bottlenecks in the requirements process?

RQ2. Which treatments are possible to avoid requirements process overload due to high inflow of requirements?

RQ3. May an in-house usability evaluation be used, in a company inexperienced in usability, as a reliable sources for competitive software requirements?

RQ4. How may differences and similarities of the needs and opportunities from different market segments distributed around the world, be collected, measured and visualized?

RQ5. Can lexical constituents of natural language requirements be used to automatically identify duplicates among requirements as a means to reduce process overload?

2.2 Research methods

As mentioned, depending on the specific research methods chosen, research may fit more or less accurately to a specific methodology. There are a vast number of research methods available and the ones to choose is dependent on the type of information that is sought (Robson, 2002). The following list include the research methods most used:

- *Surveys and questionnaires*
  A relatively small amount of data in standardized form is collected from a relatively large number of individuals. The sample of representative individuals is collected from a known population. The standardize form is usually a written questionnaire.

- *Interviews*
  An interviewer asks questions and the response is noted. Interviews

may be fully structured, semi-structured or unstructured. The fully structured interview has predetermined questions with fixed wording, usually in a pre-set order. The semi-structured interview has predetermined questions, but the order may be modified based upon interviewers perception of what seems most appropriate. Also, question wording may be changed and questions may be omitted or added. Finally, the unstructured interview has a general area of interest and may be completely informal.

- *Observation*
  People are watched and what they do is recorded. Observation may be *direct* or *non-participatory*. In the former the researcher takes part in the activity while observing. In the latter the researcher does not take part in the activity but rather tries to be quiet, watch and understand. Ethically questionable in this stance is to conceal being a observer.

- *Simulation*
  Used to generate artificial historical data, which is observed to investigate the plausible behaviour of a real system. This method was further described in Section 1.3. To give good results, simulation studies require a sound and thorough realization (Banks et al., 1996).

- *Content analysis*
  The content of existing documentation, which can be virtually any written information, is analysed and conclusions based on the content is reported.

In the next section a declaration of the methods used in this thesis is found.

2.3 Research classification

The results in this thesis have been reached through the use of several of the presented strategies and research methods. Table 3 provides a mapping between the presented papers and the research questions, strategies, methodologies and methods used. As mentioned before, one methodology may not be a complete and accurate representation of the presented research. The methodologies listed should rather be regarded as an indication of the long-term approach taken.

Table 3. A mapping between the papers in this thesis and the research questions, strategies, methodologies and methods used. The listed methodologies do not make up an exact representation of the presented research, but are rather indications of the long-term approach taken.

Paper	Question	Strategy	Methodology	Method
I	RQ1, RQ2	Fixed & Flexible, Case study	Empirical	Simulation, Interview
II	RQ3	Fixed & Flexible, Case study	Engineering	Questionnaire, Interview
III	RQ4	Fixed & Flexible, Case study	Empirical, Engineering	Survey, Questionnaire, Interview
IV	RQ1, RQ5	Fixed & Flexible, Case study	Engineering, Empirical	Content analysis, Interview

2.4 Validity

Although well-known strategies, methodologies, and methods have been used to conduct the research and arrive at the conclusions presented in this thesis, the results may be questioned for a number of reasons. The validity of the results is a property that always should be addressed in good research. Validity of research results is subdivided into four different types, each addressing a specific methodological question (Trochim, 2000). They are explained below in the context of a causal study, where a potential relationship between a cause and effect is sought.

- *Conclusion validity*
  Is there a relationship between the cause and the effect? It may be concluded that there is a relationship, that there is a positive relationship, that there is no relationship, etc. In each of these cases, the conclusion validity may be assessed.
- *Internal validity*
  Assuming that there is a relationship, is the relationship a causal one? A correlation between the cause and the effect in a study does not necessarily mean that the construct is causing the effect.

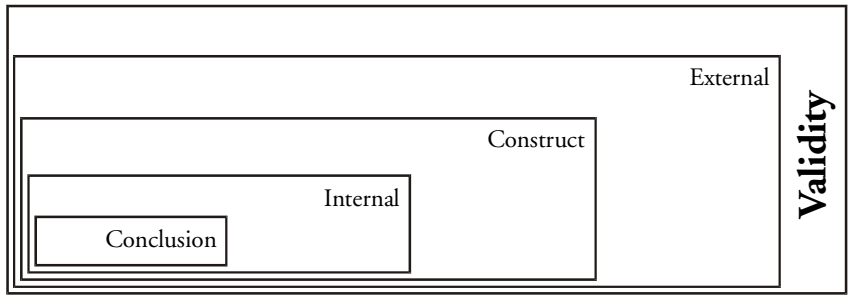

Figure 8. *Illustration of the relationship between the different types of validity.*

- *Construct validity*
  Assuming that there is a causal relationship, can it be claimed that the treatment reflects the *construct* of the treatment and that the measure well reflects the idea of the *construct* of the measure? I.e., was the intended treatment really implemented and was the intended measure really what was measured?

- *External validity*
  Assuming that there is a causal relationship in the study between the constructs of the cause and the effect, can the effect be generalized to other places, times, or people? Claims may be made that the research findings have implications for other similar settings.

As the methodological questions above point out, the validity types build on each other. In Figure 8 this relationship is illustrated, i.e. each type assumes that the previous validity type is ensured.

Attempts should be made to reduce the threats to validity, i.e. the reasons that a conclusion is wrong. For example, there may be insufficient statistical power to detect a relationship, a sample size may be to small, a measure may be unreliable, variability in the data may be caused by random heterogeneity, etc. By showing that the possible alternative explanations are not credible, the most plausible conclusion may correctly and reliably be reached (Trochim, 2000).

Threats to validity of the research results in this thesis are discussed in conjunction with the main contributions in Section 3.2.

3. Research results

This chapter summarizes the main contributions and gives the reader an introduction to the contents of each of the papers included in this thesis.

3.1 Main contributions

This thesis addresses a line of issues concerning requirements elicitation and management in the market-driven software development organisation. The main contributions from each paper presented in this thesis are summarized below and related to the research questions RQ1–RQ5 in Section 2.1.

C1: *Technique to early identify forthcoming bottlenecks in the requirements process. (RQ1, RQ2)*
Discrete-event modelling and simulation techniques are showed to correctly predict overload conditions in a requirements management process. The techniques may provide validated decision support for process improvements.

C2: *Methods for eliciting and validating reliable usability requirements within the company. (RQ3)*
Usability breaks are added to improve a well-known informal usability inspection technique, the heuristic evaluation. The technique is used to show how employees, unskilled in usability, may successfully and rapidly find usability problems in the software product that may be used as input to the requirements process. A complementary usability evaluation, using questionnaires, is conducted and evaluated. The method is suggested be used to cost-effectively validate the appropriateness of requirements elicited in the heuristic evaluation.

C3: *Methods and visualization techniques for distributed prioritization of requirements in order to provide decision support when regarding the needs of several different markets. (RQ4)*
A distributed prioritization process is tested and evaluated. The process is used to capture the different prioritizations of requirements made by stakeholders in order to allow the next software release to satisfy both the developing organisation and the intended users. Two charts are proposed that help interpret prior-

itization data collected in distributed prioritization. The charts visualize differences and agreements among the different stakeholders' prioritizations.

C4: *A baseline for evaluating supportive techniques that may assist the requirements analyst through automated analysis of requirements written in informal natural language. (RQ1, RQ5)*
The feasibility of using natural language processing techniques to identify duplicates is demonstrated through empirical investigation of using information retrieval techniques on real industrial requirements. The results are evaluated and an evaluation baseline is presented. An evaluation scheme is demonstrated and suggested to be used to assess possible improvements.

The contributions above should be accredited to all the corresponding authors who have been involved in the research. As far as my own contributions are concerned, these are further elaborated in conjunction with each of the listed papers in Section 3.3.

3.2 Threats to validity

The contributions listed in the previous section rely on conclusions drawn from the results obtained. In the following paragraphs, the threats to validity of these conclusions are discussed.

In the simulation study the major threat to validity concerns the construct, i.e. the model and the degree to which it faithfully represents its system counterpart (Zeigler et al, 2000). In the presented study, validation has been achieved through an iterative process of running the model and analyse the output behaviour. This process was terminated when the model was considered to capture the system behaviour to the extent demanded by our objectives. Furthermore, the company under study validated the simulation output to be accurate, thus further assuring construct validity. There may be threats to internal validity of the conclusions on how to avoid bottlenecks, but although the results may be somewhat surprising, they are considered plausible. Improvements to the model were then identified and thus also other potential threats to validity.

In the usability evaluation study, the heuristic evaluation was an already tested and approved method for evaluation of usability. Nevertheless, it was noted that the company found the outcome from the

evaluation to be very valuable in the development process. Exactly how simple the heuristic evaluation was perceived to be has not been properly investigated, but the time invested by each evaluator in relation to the outcome of the evaluation indicates that the effort anyhow was well spent. The SUMI questionnaire used to obtain end users' opinions is well documented and has been thoroughly validated (Kirakowski & Corbett, 1996). These threats to construct and internal validity were parts of the earlier concerns when choosing evaluation methods.

In the distributed prioritization study, two identified threats to internal and construct validity are presented in the paper. The first concerned the quality of prioritizations. It was recognized that stakeholders may not know how particular requirements should be interpreted or how important they are for the potential customers on their market. The other threat concerned 'shrewd tactics', implying that stakeholders could give an extra-low priority to requirements they knew other stakeholders would give high priorities, just in order to influence the total result to fit their aims. With the presented technique the actual events are impossible to determine. In both cases, another prioritization method is suggested that may address these threats. The threats to the validity of the results from questionnaire lie in the testing method itself, i.e. how questions are formulated and ordered and which answers were available in the closed questions.

In the study of automated duplicate identification, there may be several threats to internal and construct validity of the implementations of the programs that parsed the requirements. Several questions may be raised: if the stop word list is appropriate, if the stemming scheme is appropriate, the results are credible, etc. Fortunately, errors *were* detected in the analysis when they were repeated. By using different tools to analyse the results, those threats were minimized. However, for the purpose of the study, to investigate the feasibility of the techniques through duplicate identification and to provide a baseline for further evaluations, the methods were carefully evaluated.

The case study strategy, which is used throughout the presented research, entails specific external validity threats. The external validity, i.e. the generalizability, is the most problematic and results from further studies are the only cure to this threat. However, the presented work has been conducted with previously stated objectives in mind (see start of Section 2 on page 36) and it has been of ethical concern and aim to only

report on results that have also been questioned by the authors themselves.

3.3 Summary of papers

The following summary lists the title, author, conference or publication, and abstract of each individual paper in this thesis. In conjuction to each abstract, the indiviual authors' contributions are reported.

PAPER I: Exploring bottlenecks in market-driven requirements management processes with discrete event simulation

Martin Höst, Björn Regnell, Johan Natt och Dag, Josef Nedstam, & Christian Nyberg.

Journal of Systems and Software, 59, 323–332, 2001.

This paper presents a study where a market-driven requirements management process is simulated. In market-driven software development, software packages are released to a market and not developed specifically for a single customer. New requirements are continuously issued, and the objective of the requirements management process is to sort, manage, and prioritize the requirements. In the presented study, a specific requirements management process is modelled using discrete event simulation, and the parameters of the model are estimated based on interviews with people from the specific organisation where the process is used. Based on the results from simulations, conditions that result in an overload situation are identified. Simulations are also used to find process change proposals that can result in a non-overloaded process. The risk of overload can be avoided if the capacity of the requirements management process is increased, or if the number of incoming requirements is decreased, for example, through early rejection of low-priority requirements.

The research in this paper was mainly initiated by Dr. Höst and Dr. Regnell. In terms of writing the authors contributed to an extent corresponding to the order of the authors' names. All the authors contributed in the discussions, implementation and simulations of the queuing network model.

PAPER II: An industrial case study of usability engineering in market-driven packaged software development

Johan Natt och Dag, Björn Regnell, Ofelia S. Madsen, & Aybüke Aurum.

M. J. Smith, G. Salvendy, D. Harris & R. J. Koubek (Eds.), *Proceedings of HCI International: Vol. 1. Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents and Virtual Reality* (pp. 425–429). Mahwah, NJ: Erlbaum. 2001.

In market-driven software development it is crucial to produce the best product as quickly as possible in order to reach customer satisfaction. Requirements arrive at a high rate and the main focus tends to be on the functional requirements. The functional requirements are important, but their usefulness relies on their usability, which may be a rewarding competitive means on its own. Existing methods help software development companies to improve the usability of their product. However, companies that have little experience in usability still find them to be difficult to use, unreliable, and expensive. In this study we present results and experiences on conducting two known usability evaluations, using a questionnaire and a heuristic evaluation, at a large software development company. We have found that the two methods complement each other very well, the first giving scientific measures of usability attributes, and the second revealing actual usability deficiencies in the software. Although we did not use any usability experts, evaluations performed by company employees produced valuable results. The company, which had no prior experience in usability evaluation, found the results both useful and meaningful. We can conclude that the evaluators need a brief introduction on usability to receive even better results from the heuristic evaluation, but this may not be required in the initial stages. Much more essential is the support from every level of management. Usability engineering is cost effective and does not require many resources. However, without direct management support, usability engineering efforts will most likely be fruitless.

The research presented in this paper is based on previous work by Ms. Madsen and Mr. Natt och Dag. The first author pursued, together with Dr. Regnell and Dr. Aurum, with further conclusions on the context. The paper was mainly written by Mr. Natt och Dag.

PAPER III: An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software

Björn Regnell, Martin Höst, Johan Natt och Dag, Per Beremark, & Thomas Hjelm.

Requirements Engineering, 6, 51–62, 2001.

When developing packaged software, which is sold 'off-the-shelf' on a worldwide marketplace, it is essential to collect needs and opportunities from different market segments and use this information in the prioritisation of requirements for the next software release. This paper presents an industrial case study where a distributed prioritisation process is proposed, observed and evaluated. The stakeholders in the requirements prioritisation process include marketing of offices distributed around the world. A major objective of the distributed prioritisation is to gather and highlight the differences and similarities in the requirement priorities of the different market segments. The evaluation through questionnaires shows that the stakeholders found the process useful. The paper also presents novel approaches to visualise the priority distribution among stakeholders, together with measures on disagreement and satisfaction. Product management found the proposed charts valuable as decision support when selecting requirements for the next release, as they revealed unforeseen differences among stakeholder priorities. Conclusions on stakeholder tactics are provided and issues of further research are identified, including ways of addressing identified challenges.

The approach to use fictitous money in the prioritization of requirements was proposed by Mr. Hjelm. Dr. Regnell and Dr. Höst developed the distributed prioritization method together with Mr. Beremark and Mr. Hjelm. The visualization charts and the measures were mainly developed by Dr. Regnell. The questionnaire was designed by Dr. Regnell and Dr. Höst and validated by Mr. Beremark and Mr. Hjelm. The results from the evaluation were analysed and presented by Mr. Natt och Dag.

PAPER IV: A feasibility study of automated natural language requirements analysis in market-driven development

Johan Natt och Dag, Björn Regnell, Pär Carlshamre, Michael Andersson, & Joachim Karlsson.

Requirements Engineering, 7, 20–33, 2002.

In market-driven software development there is a strong need for support to handle congestion in the requirements engineering process, which may occur as the demand for short time-to-market is combined with a rapid arrival of new requirements from many different sources. Automated analysis of the continuous flow of incoming requirements provides an opportunity to increase the efficiency of the requirements engineering process. This paper presents empirical evaluations of the benefit of automated similarity analysis of textual requirements, where existing Information Retrieval techniques are used to statistically measure requirements similarity. The results show that automated analysis of similarity among textual requirements is a promising technique that may provide effective support in identifying relationships between requirements.

The techniques used in this paper were selected, implemented and evaluated by Mr. Natt och Dag, with assitance from Dr. Regnell. The evaluation scheme was suggested by Dr. Regnell. Dr Carlshamre contributed with the data for the interrelationsship analysis and also to conclusions on the presented work. Mr. Beremark provided the underlying data and Mr. Andersson provided additional information and also conducted a manual analysis of a subset of the requirements. Dr. Karlsson contributed with ideas for further applications. Mr. Natt och Dag wrote most of the paper with assistance from Dr. Regnell and Dr. Carlshamre.

4. Further research and future plan

Several issues for further research have been identified during and in connection with the research presented in this thesis. Examples of these issues are listed and described in the following section and a more focused plan and accompanying research strategy for the following two years are presented in Section 4.2.

4.1 Further research

There are many possible improvements to be made to the methods and techniques presented in this thesis. Furthermore, the research strategy used, i.e. the case study, entails threats to external validity. Therefore, a general improvement would be to replicate the studies in various, similar settings or with different data. Before that, no certain conclusions on the generalizability of the results can be made. Other, specific suggestions for improvements are grouped below according to the related paper.

The simulation study may be superseded by investigating different ways of improving the simulation model and by looking into different ways of reducing congestion in the requirements process:

FR1: The simulation model simplifies the implementation of the must- and wish-lists. The simulation model may be improved to more realistically represent the actual process of how requirements are selected for the lists. This may enable investigation of the quality of the process outcome.

FR2: The simulation model simplifies the use of employees to be dedicated to a particular phase. Modelling servers as a single pool of resources, in which each resource has certain competencies, may improve accuracy.

FR3: The simulation study suggest two ways of reducing congestion in the process. Exactly how this may be achieved could be subject for further research.

FR4: The discrete-event nature of requirements may be motivated, but there may also be continuous elements in the process that is better modelled with a continuous simulation model. Advantages and drawbacks of using discrete-event, continuous, or hybrid simulation models may be investigated.

The market-driven development organisation demands quick and accurate results. Thus, the usability evaluation study may be superseded by further evaluations and improvements of the heuristic evaluation as an in-house usability evaluation technique:

FR5: The presented study proposes *usability breaks* in the scenarios to keep inexperienced evaluators focused on usability issues. To which extent this improvement actually helps may be an issue for further research.

FR6: Employees may be short of time and have diverse competencies. Exploring the quickest and best ways to select usability evaluators within a software development company, may provide means to keep obtaining good results from subsequent evaluations.

FR7: Results from the presented study suggest that a short introduction or education in usability could give a usability evaluation outcome of higher quality. Measurements of the improvements achieved by first educating potential usability evaluators in usability may verify this.

FR8: How the outcome of the usability evaluation is utilized in the software development process has not been studied. To better motivate usability evaluations of the kind presented, investigations could be made of to what extent the evaluation results in improved usability of the product. It could also be studied if the techniques cause any other benefits, e.g. any kind of improvements in software or development but not specifically in usability.

Prioritization is an important issue for the market-driven software development company (see 'Specific RE issues' in Table 1 on page 18). Prioritization methods may be further investigated and situations how distributed prioritization is best utilized may be studied:

FR9: The presented paper use fictitious money to prioritize requirements. As pair-wise comparison, based on the analytical hierarchy process, have been shown to be accurate and efficient, this method may be of interest to evaluate in a distributed setting.

FR10: The study concludes that stakeholders and product and quality management find the distributed prioritization useful and the visualization charts valuable decision support. A follow-up study may investigate the actual impact of the decided consolidated prioritization from distributed prioritization.

FR11: The acceptance among stakeholders of the presented technique was somewhat evaluated. Further studies may investigate if stakeholders whose ratings in prioritization differ significantly from the final decision are more inclined to accept the decision thanks to the increased transparency of the prioritization process.

One way to resolve congestion in the requirements process was found to be to relieve the burden on people working in the process (see Paper I). Automated requirements relationship analysis may be one step to towards this goal. The simplistic techniques presented in Paper IV may obviously be further investigated, improved and evaluated:

FR12: The presented study found automated requirements analysis to be feasible for duplicates identification and removal. For a better understanding of when these techniques may be supportive, process issues, such as when requirements analysis should be performed, who should perform the analysis and how the analysis is cost-efficient to perform, may be further studied.

FR13: The content in natural language is one relevant source for analysis. Additional information carrying requirements attributes and fields may improve precision. It may be interesting to investigate how different ways of representing requirements affect the results. Which representation is best suited for high precision in automated similarity analysis?

FR14: Several NLP-related improvements may be considered to increase method accuracy. Examples include: the use of a domain-specific stop list, a thesaurus with general synonym words, spelling correction prior to the automated similarity analysis and by not discriminating between words with a short editing distance.

FR15: In the borderland between natural language processing and linguistics, smart algorithms may improve method accuracy. Some words may be over-represented in the set of false positives. Removing these words may improve the precision. This is also an example of where it may be possible to make the algorithm self-adjustable based on human corrections.

FR16: Pure linguistic methods and techniques may provide means for improving method accuracy. For example, linguistic methods may provide more precise analysis of natural language requirements on a semantic level. This may include the use of ontologies or word nets.

FR17: Automated analysis techniques may work at a technical and engineering level. For the ideas to be useful it is valuable to investigate ways of visualising the results from automated similarity analysis and supporting the requirements engineer in the navigation among related requirements.

4.2 Future plan

A possible plan for further research for the next two years, incited by the results presented in this thesis, is here hypothesized, motivated and presented.

According to a study of software technologies it takes in between 15 to 20 years for a technology to evolve from concept formulation to the propagation throughout the community of users (Redwine & Riddle, 1985; Shaw, 2002). Considering the age of requirements engineering and the attempts made so far to automatically analyse textual representations using natural language processing techniques, a breakthrough may still be some years away. Nevertheless, the issue is interesting and several industrial partners are positive.

Preliminary results from an ongoing, not yet published, industrial survey (Related Papers, [XII], page 13) indicate that tools for basic needs that are simple to use are highly desirable. The results also indicate that natural language is used extensivly when specifying and managing requirements. One of the basic needs, avoiding requirements process bottlenecks, is presented in this thesis and linguistic methods are suggested to partially fulfil this need. Thus, it is of high interest to develop

supportive tools that incorporates several techniques aimed for the market-driven software development organisation.

The following research issues are planned to be addressed:

1. *Explore and understand the current state of affairs in market-driven requirements engineering.*
   An important basis for drawing conclusions about situations when automated analysis may be of interest, is the understanding of the current situation in requirements engineering. Descriptive, qualitative, and quantitative studies provide means to explore and understand how requirements are managed today and what the problematic issues may be.
   Research question (FRQ1): *To what extent is natural language used to specify requirements in current market-driven software development companies?*

2. *Further evaluate automated relationship analysis of requirements.*
   To further validate automated relationship analysis techniques it is of vital importance to replicate the evaluation using different sets of requirements. With a substantial set of requirements databases it may be possible to find means of refinement and improvements of the techniques in order to increase their accuracy. In a forthcoming study, it will be investigated if the techniques may be used to relate market requirements and business requirements.
   Research question (FRQ2): *Which techniques may improve the accuracy of automated relationship analysis of requirements?*

3. *Investigate the limits of real-word applications of automated analysis*
   It is desirable to find techniques suitable for supporting the requirements analyst in her daily work. The intent is not in any way to replace human judgement. Therefore, it is of interest to investigate to what extent the requirements manager could be assisted by the proposed techniques. How reliable are the techniques and when and how are they best used?
   Research question (FRQ3): *What are the possible applications of automated relationship analysis techniques to support the requirements analyst?*

4. *Develop prototype tools to test utilization and visualization of automated analysis techniques.*
   Putting the analysis techniques in the context they belong enables

further assessment of their feasibility. In developing tools to support the techniques, further improvements issues may be revealed. In order for the techniques to be usable, there must be ways to easily adopt them and incorporate them into the present development process. Proper visualization of the results from automated analysis may be one way to aid the requirements manager.

Research question (FRQ4): *How may the result from automated relationship analysis techniques best be supported by and visualized in CASE tools?*

The research questions will be tried to be answered by conducting case studies on as many sets of real industrial requirements as possible. By using empirical methods to validate each step, proper improvement will hopefully be possible to be made. Experiments in a real-world setting would be a desirable strategy to investigate the usefulness of automated analysis techniques. A mapping, corresponding to the one found in Table 3 on page 41, between the research questions and the planned strategies, methodologies and methods to be used, is found in Table 4.

Table 4 concludes the introduction of this thesis. In the subsequent pages, following the references in the next section, research papers I through IV are found.

Table 4. A mapping between future research questions and planned strategies, models, methods. The methodologies may not make up an exact representation of the planned research, but are rather indications of the approach to be taken.

Question	Strategy	Methodology	Method
FRQ1	Fixed & Flexible, Etnographic study	Empirical	Interview, Survey, Questionnaire
FRQ2	Fixed & Flexible, Case study	Engineering, Empirical	Content analysis
FRQ3	Flexible, Etnographic study	Empirical	Survey, Questionnaire, Simulation
FRQ4	Fixed & Flexible, Experimental	Engineering, Empirical	Controlled experiment, Questionnaire, Interview

5. References

Adrion, W. R. (1993). Research methodologies in Software Engineering. In W. F. Tichy, N. Habermann & L. Prechelt (Eds.), Summary of the Dagstuhl Workshop on Future Directions on Software Engineering. *ACM Software Engineering Notes, 18*(1), 35–48.

Ahlbom, H. (2002, May 2). Ericsson hårdbantar forskningen. *Ny Teknik*, p. 2:5.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Harlow, England: Addison-Wesley.

Banks, J., Carson, J. S., & Nelson, B. (1996). *Discrete-Event System Simulation* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Beck, K. (2000). *Extreme Programming Explained*. Reading, MA: Addison-Wesley.

Bias, R. G., & Mayhew, D. J. (1994). *Cost-justifying usability.* Boston: Academic Press.

Bratley, P., Fox, B. L., & Schrage, L. E. (1987). *A Guide to Simulation* (2nd ed.). New York: Springer-Verlag.

Carlshamre, P., & Regnell, B. (2000). Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes. In A. M. Tjoa, R. R. Wagner, A. & Al-Zobaidie (Eds.), *Proceedings of the 11th International Workshop on Database and Expert Systems Applications Process* (pp. 961–965). Los Alamitos, CA: IEEE Computer Society Press.

Carlshamre, P. (2002). *A Usability Perspective on Requirements Engineering – From Methodology to Product Development* (Dissertation No. 726). Linköping: Linköping University, Linköping Studies in Science and Technology.

Carmone, Jr., F. J., Kara, A., & Zanakis, S. H. (1997). A Monte Carlo investigation of incomplete pairwise comparison matrices in AHP. *European Journal of Operational Research, 102*, 538–553.

Concejero, P., Clarke, A., Carter, C., Muehlbach, L., Ruschin, D., Kaasinen, E., et al. (1996). *Currently available HF guidelines and standards* (Tech. Rep. No. AC224/TID/2270/DR/P/002/b1). USINACT Project AC224.

Davis, A. M. (1993). *Software Requirements – Objects, Functions, & States* (Rev. ed.). Upper Saddle River, NJ: Prentice Hall.

Glass, R. L. (1994). The Software–Research Crisis. *IEEE Software, 11*(6), 42–47.

Goguen, J. A. (1996). Formality and Informality in Requirements Engineering. In *Proceedings of the Fourth International Conference on Requirements Engineering* (pp. 102–108). Los Alamitos, CA: IEEE Computer Society Press.

Goodwin, N. C. (1987, March). Functionality and usability. *Communications of the ACM, 30*, 229–233.

Gordon, G. (1969). *System Simulation*. Engelwood Cliffs, NJ: Prentice-Hall.

IEEE 610.12–1990. Standard Glossary of Software Engineering Terminology.

ISO 9241–11:1998. Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability.

Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing*. Upper Saddle River, NJ: Prentice Hall.

Kamsties, E., Hörmann, K., & Schlich, M. (1998). Requirements Engineering in Small and Medium Enterprises. *Requirements Engineering, 3*, 84–90.

Karlsson, J. (1998). *A systematic approach for prioritizing software requirements* (Dissertation No. 526). Linköping: Linköping University: Department of Computer and Information Science.

Keil, M., & Carmel, E. (1995). Customer–Developer Links in software Development. *Communications of the ACM, 38*(5), 33–44.

Kirakowski, J., & Corbett M. (1996). The software usability measurement inventory: Background and usage. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability Evaluation in Industry* (pp. 169–177). London: Taylor & Francis.

Kotonya, G., & Sommerville, I. (1997). *Requirements engineering: processes and techniques*. New York: John Wiley & Sons.

Lauesen, S. (2002). *Software Requirements – Styles and Techniques*. London: Addison-Wesley.

Law, A. M., & Kelton, W. D. (1991). *Simulation Modeling and Analysis* (2nd ed.). Engelwood Cliffs, NJ: Prentice-Hall.

Lubars, M., Potts, C., & Richter, C. (1993). A review of the state of the practice in requirements modelling. In *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 2–14). Los Alamitos, CA: IEEE Computer Society Press.

Manning, C. D., & Schütze, H. (2000). *Foundations of Statistical Natural Language Proccessing*. Cambridge, MA: MIT Press.

Natt och Dag, J., & Madsen, O. S. (2000). *An Industrial Case Study of Usability Evaluation* (Master's Thesis. Report No. CODEN:LUTEDEX (TETS–5390)/1–190/ (2000)&local 8). Lund: Lund University: Department of Communication Systems.

Naylor, T. H., Balintfy, J. L., Burdick, D. S., & Chu, K. (1966). *Computer Simulation Techniques*. New York: John Wiley & Sons.

Nielsen, J. (1993). *Usability Engineering*. San Diego, CA: Morgan Kaufmann.

Nielsen, J. (1994). Heuristic Evaluation. In J. Nielsen, & R. L. Mack (Eds.), *Usability Inspection Methods* (pp. 25–61). New York: John Wiley & Sons.

Novorita, R. J., & Grube, G. (1996). Benefits of structured requirements methods for market-based enterprises. In *Proceedings of Sixth Annual International INCOSE Symposium*. Seattle, WA: INCOSE.

Pegden, C. D., Shannon, R. E., & Sadowski, R. P. (1995). *Introduction to simulation using SIMAN* (2nd ed.). New York: McGraw-Hill.

Potts, C. (1995). Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering* (pp. 128–130). Los Alamitos, CA: IEEE Computer Society Press.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., & Carey, T. (1994). *Human-Computer Interaction*. Wokingham, England: Addison-Wesley.

Redwine, Jr., S. T., & Riddle, W. E. (1985). Software technology maturation. In *Proceedings of the Eighth International Conference on Software Engineering*, (pp. 189–200). Los Alamitos, CA: IEEE Computer Society Press.

Regnell, B. Beremark, P., & Eklund, O. (1998). A Market-Driven Requirements Engineering Process – Results from an Industrial Process Improvement Programme. *Requirements Engineering, 3*, 121–129.

Robertson, S., & Robertson, J. (1999). *Mastering the Requirements Process*. Harlow, England: Addison-Wesley.

Robson, C. (2002). *Real World Research* (2nd ed.). Oxford, UK: Blackwell.

Royce, W. W. (1970). Managing the development of large software systems: concepts and techniques. *Proceedings of IEEE WESTCON* (pp. 1–9). Los Alamitos, CA: IEEE Computer Society Press.

Saaty, T. L. (1980). *The Analytical Hierarchy Process*. New York: McGraw Hill.

Sawyer, P. (2000). Packaged Software: Challenges for RE. In *Proceedings of Sixth International Workshop on Requirements Engineering: Foundation for Software Quality* (pp. 137–142). Essen, Germany: Essener Informatik Beiträge.

Sawyer, P., Sommerville, I., & Kotonya, G. (1999). Improving Market-Driven RE Processes. In M. Oivo, & P. Kuvaja (Eds.), *Proceedings of the International Conference on Product Focused Software Process Improvement* (pp. 222–236). Oulu, Finland: Technical Research Centre of Finlad (VTT).

Shaw, M. (1990). Prospects for an Engineering Discipline of Software. *IEEE Software, 7*(6), 15–24.

Shaw, M. (2002, April). *What makes good research in Software Engineering?* Paper presented at the Fifth European Joint Conferences on Theory and Practice of Software, Grenoble, France.

Siddiqi, J., & Shekaran, C. (1996). Requirements Engineering: The Emerging Wisdom. *IEEE Software, 13*(2), 15–19.

Somerville, I. (2001). *Software Engineering*. Harlow, England: Addison-Wesley.

Sommerville, I., & Sawyer, P. (1997). *Requirements Engineering – A Good Practice Guide*. Chichester, England: John Wiley & Sons.

Trochim, W. (2000). *The Research Methods Knowledge Base* (2nd ed.). Cincinnati, OH: Atomic Dog Publishing.

Wixon, D., & Wilson, C. (1997). The usability engineering framework for product design and evaluation. In *Handbook of human-computer interaction* (pp. 653–689). Amsterdam: Elsevier Science.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). *Experimentation in Software Engineering – An Introduction*. Norwell, MA: Kluwer.

Yeh, A. (1992). Requirements Engineering Support Technique (REQUEST) – A Market Driven Requirements Management Process, In *Proceedings of the Second Symposium on Assessment of Quality Software Development Tools* (pp. 211–223), Los Alamitos, CA: IEEE Computer Society Press.

Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). *Theory of Modeling and Simulation – Integrating Discrete Event and Continuous Complex Dynamic Systems* (2nd ed.). San Diego, CA: Academic Press.

PAPER I

Exploring bottlenecks in market-driven requirements management processes with discrete event simulations

I

Martin Höst, Björn Regnell, Johan Natt och Dag, Josef Nedstam, Christian Nyberg

Journal of Systems and Software, 59, 323–332, 2001.

Abstract

This paper presents a study where a market-driven requirements management process is simulated. In market-driven software development, generic software packages are released to a market with many customers. New requirements are continuously issued, and the objective of the requirements management process is to elicit, manage, and prioritize the requirements. In the presented study, a specific requirements management process is modelled using discrete event simulation, and the parameters of the model are estimated based on interviews with people from the specific organisation where the process is used. Based on the results from simulations, conditions that result in an overload situation are identified. Simulations are also used to find process change proposals that can result in a non-overloaded process. The risk of overload can be avoided if the capacity of the requirements management process is increased, or if the number of incoming requirements is decreased, for example, through early rejection of low-priority requirements.

1. Introduction

Requirements Engineering (RE) in a market-driven context, where succeeding versions of a software package are released to a market, is getting increased attention (Lubars, Potts & Richter, 1993; Potts, 1995; Yeh, 1992). When developing software for a market, rather than for a single customer, the pressure on short time-to-market is evident. An effective engineering of software requirements is an important success factor for meeting market demands. RE involves activities such as analysing and prioritizing requirements, and maintaining a database of requirements that may be implemented in the future. This part of the software process requires resources, and the allocation of resources to activities related to requirements selection and release planning is crucial to the continuous delivery of competitive software releases. Traditional RE has mainly been focused on the bespoke situation where a specific system is developed based on a contract with a specific customer. The market-driven situation, however, has special challenges regarding scheduling constraints and stakeholding (Sawyer, 2000), and there is an industrial need for process improvement in this area (Sawyer, Sommerville & Kotonya, 1999).

One way of analysing process improvement proposals is to carry out pilot studies or controlled experiments within the specific organisation (Wohlin et al., 2000). However, this requires much resources and an alternative approach is to carry out simulations of the organisation instead (Kellner, Madachy, & Raffo, 1999; Pfahl & Lebsanft, 2000). This is an engineering approach that is chosen in many other areas, and it can, of course, be applied in evaluation of software development processes too. After the new processes have been analysed through simulation they can be analysed in experiments and case studies. In this way simulations can be a natural part of technology transfer and evaluation. Simulations may reduce the risk of implementing process changes that are not resulting in improvements. Since many people in the organisation often are involved in experiments and pilot-studies, it may be a large problem if the wrong changes are introduced and evaluated. This can very well damage the continued process improvement work in the organisation for a long time. Thus, there is a clear opportunity for simulation as a first step in the evaluation of new software process technology.

The objective of the presented study is to investigate if simulation can help in exploring bottlenecks and overload situations in RE processes.

The object of simulation is a specific process called REPEAT (Regnell, Beremark, & Eklundh, 1998), which is used by a leading CASE-tool developer for real-time systems development (Telelogic AB). The REPEAT process is a result of an improvement programme that started in 1995, as Telelogic considered efficient RE a key success factor. After the introduction of REPEAT, a significant improvement in delivery precision and product quality was gained. However, after a number of releases with REPEAT, it was realized that market pressure resulted in a number of further challenges regarding through-put and congestion (Regnell et al.). This led to a research project with the objective of further understanding and improving market-driven RE. The presented work is a part of this effort.

All figures and data in this paper refer to the period 1998–1999. Since then, Telelogic has grown considerably, and Telelogic has continuously introduced improvements in order to meet the challenges of the market. The principal results presented in this paper are thus relevant for understanding market-driven requirements management in general, rather than characterizing the current and future situation at Telelogic. In the following, all references to the "current" or "actual" situation relates to the time-frame from 1998–1999.

The simulation study presented in this paper, applies discrete event simulation (Banks, Carson, & Nelson, 1996) using a queuing network model (King, 1990). A major objective of the simulation study is to explore the conditions under which the process becomes overloaded. It is also investigated which resources are needed in order to handle a certain frequency of new requirements. Simulations are carried out in order to explore the conditions that result in an overloaded process, and to find changes to the process that may remove bottlenecks.

The paper is structured as follows. In Section 2, the REPEAT process is presented, and Section 3 presents the simulation model. The results of the performed simulations are presented in Section 4. In Section 5, conclusions and suggestions for further research in the field are presented.

2. The REPEAT process

REPEAT manages requirements continuously by controlling a product pipeline in which three releases are developed in parallel. The product pipeline delivers two new product releases per year. REPEAT covers

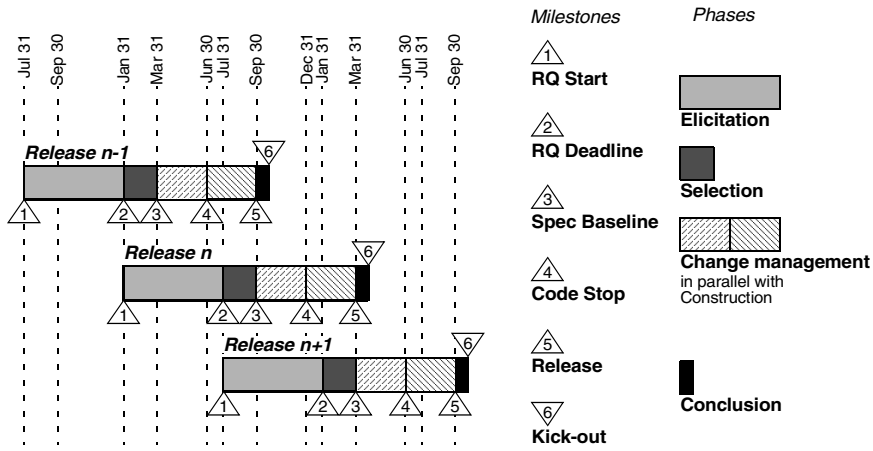

Figure 1. *The milestones and phases of the REPEAT process, aligned with a fixed release schedule.*

typical RE activities, such as elicitation, documentation, and validation, and the process has a strong focus on requirements selection and release planning. A schematic picture of the process is shown in Figure 1.

REPEAT is instantiated for each release, and each process instance has a fixed duration of 14 months. Each REPEAT instance consists of five different phases separated by milestones at pre-defined dates. The Elicitation phase deals with the collection and initial classification of requirements. The Selection phase includes detailed specification of each requirement and release planning. The Change Management phase is active in parallel with construction (design, implementation, and testing of requirements for the coming release) and manages changes in requirements priorities due to events such as emergence of high-priority requirements and delays. The Conclusion phase includes post-mortem documentation. Each of these phases are further described below.

2.1 Elicitation.

The elicitation phase includes two activities: collection and classification. Collection of requirements is made by an issuer that fills out a web-form and submits the requirement for storage in an in-house-built database. Requirements are described using natural language and given a summary name by the issuer. An explanation of why the requirement is needed is also given. The issuer gives the requirement an initial priority P, which suggests in which release it may be implemented. P is a subjective measure

reflecting the view of the issuer, and is measured on an ordinal scale with three levels, as shown in Table 1.

2.2 Selection.

The goals of this phase are:

1. to select which requirements to implement in the current release
2. to specify the selected requirements in more detail
3. to validate the requirements.

The output of this phase is a requirements document which includes a selected-list with a detailed specification and effort estimation in hours of all selected requirements, and a not-selected-list including the requirements that are postponed to the next release. The selected requirements are divided into a must-list and a wish-list. The must-list comprises requirements that are estimated to take 70% of the available effort, while the wish-list comprises requirements that are estimated to take 60% of the available effort. This implies that if the effort estimations are correct, half of the wish-list will be implemented, and the rest will be reconsidered for implementation in the next release. However, all the requirements on the wish-list are specified, so if the estimations are not correct there will still be a number of specified requirements to implement in the release.

2.3 Change management during construction.

This phase of the REPEAT process is carried out in parallel with the design, implementation, and testing of the requirements, and handles changes in the priorities of the requirements. There are two sub-phases of this phase, one before code-stop (3–4 in Figure 1) and one after code-stop (4–5 in Figure 1). After code-stop no implementation is carried out. Instead the focus is on testing. If new priority-1-requirements are issued, these may be allowed to affect ongoing construction, and in the change management phase the requirements on the must- and wish-list may be rearranged so that new and more important requirements can be incorporated. The 70%–60% rule for the must- and wish-lists must, however, still hold, implying that some less important requirements

should be postponed in order to incorporate the new, more important, requirements.

2.4 Conclusion.

In this phase metrics are collected and a final report is written that summarises the lessons learnt from this REPEAT enactment.

During 1998 and 1999, the number of unimplemented requirements in the requirements database has increased, and the REPEAT process has at times been in a state of congestion. Process simulation gives the opportunity of investigating the behaviour of the process under different circumstances. Results from simulations may provide quantitative measures, which can act as decision support when allocating resources to different activities in REPEAT.

3. The simulation model

Based on the REPEAT process model (Regnell et al., 1998), an initial simulation model was created, including some major simplifications. The model was then iteratively refined and specialized until it provided an adequate degree of abstraction. As a last step an interview with personnel at Telelogic gave the actual values for the model parameters, along with a confirmation that the model was sufficiently accurate.

3.1 Structure of the model

The REPEAT process simulator is a queuing network model and is implemented using discrete event simulation (Banks et al., 1996). The simulated model is depicted in Figure 2. Requirements enter the simulator from the environment. A requirement must pass the three phases elicitation, selection and construction in order to be included in a release. (The conclusion phase found in Figure 1 was not included in the simulation model as it is independent from the rest of the phases and does not affect congestion and throughput).

In the elicitation phase, incoming requirements are entered into the system and given an initial priority. In the selection phase, the requirements that are to be included in the release are selected, and in the last phase the requirements are constructed. The phases are modelled as

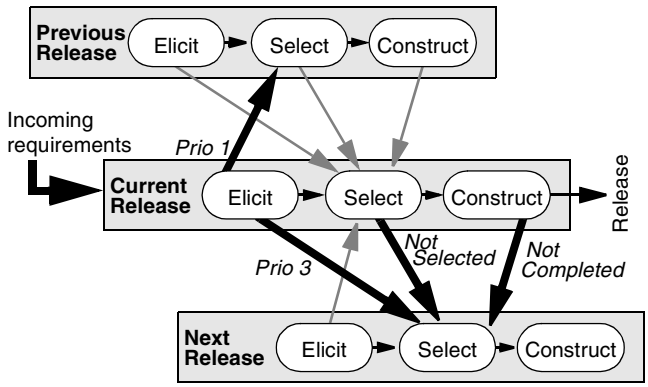

Figure 2. *Simulation model.*

processes with a queue of incoming requirements, and a pool of servers which represent the employees. Each phase is thus modelled as a FIFO queue with m servers. Requirements enter the system according to a Poisson-process, and the elicitation phase is therefore an M/G/m queue, while the two other phases are G/G/m queues.

In the elicitation phase, every requirement receives a priority. All requirements having normal priority, i.e. priority 2, are transferred to the selection phase within the current release. Priority 3 requirements are postponed to the selection phase of the next release. Priority 1 requirements are moved to the selection phase of the previous release.

In the simulation model all releases have their own resources. That is, when a release is instantiated in the model, a number of servers in each phase are created. The servers in the selection phase are idle during elicitation, waiting for the selection phase to start. The servers in the construction phase are idle during elicitation and selection, while waiting for the construction phase of the previous release to finish. The personnel that are represented in the simulation model by the servers, are in reality the same persons represented by the servers in the previous release. This is a simple way of modelling that the same persons divide their time between different activities.

During selection, the time each requirement will spend in construction is estimated. A must-list and a wish-list is constructed according to the description of the REPEAT process given above. Requirements that enter either of these lists are transferred to the construction phase of the current release. Requirements that are not selected for either of these lists are sent

to the selection phase of the next release, where they may or may not be selected for construction.

When the servers start working they check if it is possible to perform the job next-in-line within the deadline of the release. When the deadline for the release approaches, some requirements may be left uncompleted and sent to the selection phase of the next release. This occurs because there is a parameter-controlled error in the estimation of the required work, and because the wish-list includes more requirements than is possible to construct during one release in order to get a better utilization of the available resources.

All this means that a requirement may pass the selection phase several times during its lifetime. A requirement requires serving time in every phase it passes, which imply a certain amount of overhead when re-routing a requirement to another release.

3.2 Parameter estimation

The simulator accepts a set of input parameters which specify the simulated situation. These input parameters include the number of requirements entering the process each day, the number of available servers (employees) for each phase, and the average time spent on a requirement in each phase (see further Table 2).

The actual values for the parameters are based on data from interviews with an expert from Telelogic. The requirements are modelled to have an exponentially distributed intensity of arrival, i.e. they arrive according to a Poisson process. The distributions of the serving times in the various phases can be modelled in a number of ways. In (Höst & Wohlin, 1997; Höst & Wohlin, 1998) it is shown that a suitable way to model serving times based on subjective estimates given by domain experts is to use triangular distributions. Based on a triangular distribution, the interviewed expert estimated the smallest possible value, the most likely value, and largest possible value of the serving time for each phase. Data from interviews also provided estimations of parameters regarding the number of employees in each phase, the number of requirements of different priorities, and the average estimation error that is made when estimating the serving time in the construction phase.

The interviews also exhibited that if a requirement has been in one selection phase, and later is sent to a selection phase in another release, it requires only about a fifth of the time spent in the original phase.

Table 1. Simulation parameters[a].

Parameter	Case 1	Case 2	Case 3	Case 4
Time between two consecutive release start-ups	126	126	126	126
Time from start of release to start of selection phase	126	126	126	126
Time from start of release to start of construction phase	168	168	168	168
Length of construction phase	126	126	126	126
Mean time between two consecutive requirements	0.33	0.33	0.33	1.8
Number of servers in the elicitation phase	30	30	1	30
Elicitation time per requirement[b]	(0.010, 0.031, 0.062)	(0.010, 0.031, 0.062)	(0.010, 0.031, 0.062)	(0.010, 0.031, 0.062)
Number of servers in the selection phase	30	30	16	30
Selection time per requirement[b]	(1, 2, 10)	(1, 2, 10)	(1, 2, 10)	(1, 2, 10)
Number of servers in the construction phase	30	30	165	30
Construction time per requirement[b]	(1, 45, 91)	(1, 45, 91)	(1, 45, 91)	(1, 45, 91)
Fraction of requirements of priority 1	0%	10%	10%	10%
Fraction of requirements of priority 3	0%	25%	25%	25%

a. The unit of parameters representing time is working days.

b. These parameters are defined according to a triangular distribution (lowest possible value, most likely value, highest possible value) as described in Section 3.2.

The interviews with the expert also concerned general experiences with the REPEAT process. After five releases the requirements database contained almost 2,000 requirements. For each of the five releases, about 75 requirements were implemented, which imply that the requirements database contained about 1625 unimplemented requirements.

3.3 Discrepancies between simulator and process

There are two significant simplifications in the model. First of all, the server model does not completely match the actual use of employees. In reality there is a single pool of employees containing a number developers working on a number of modules. The single pool of developers work in all three phases (elicitation, selection and construction) for all releases. The simulation model, however, has one pool of servers for every phase of every release. To adjust this, each phase has a parameter indicating when it starts, and the construction phase has a parameter indicating when it finishes, i.e. the release deadline. The servers of each phase are idle when the phase is inactive. This solution gives an adequate accuracy validated by the interview results.

The second simplification is the way the must- and wish-lists are constructed. The model just takes the first incoming requirements and puts them into the must-list until it is full. Thereafter the wish-list is filled, and late jobs are not selected. This means that priority 1 requirements rarely get selected, as they are sent to the previous release, and arrive there late, when the lists are already full. This simplification can be addressed by changing the simulation model so that requirements are inserted into the lists in a way more similar to the real situation, e.g. by inserting a new job into a random position in the list, whereby the last jobs are pushed off the list. The random distribution can in turn be dependent on the priority of the requirement, or other factors, such as how old the requirement is. These possible enhancements are, however, not implemented in the simulation model as the simulator is not used for investigating the quality of the outcome, but for exploring timing, capacity and throughput.

Another simplification regards the estimated construction effort. Data from the interviews specify the distribution of the total time spent on construction for each requirement. However, large requirements can in reality be divided into several smaller requirements, which in turn can be scheduled over many releases. Therefore, the triangular distribution of the

serving times in the construction phase was modified. The maximum serving time was changed from the original 170 days to 91 days, and the most common serving time was changed from 19 days to 45 days in order to produce the same workload. Otherwise the largest jobs would never be implemented in the simulation model.

Other simplifications include the fact that the simulator model never rejects requirements, which is done to a small extent in the actual process. However, requirements are removed so rarely in reality that we do not believe that it affects the validity of the results very much.

In general, we believe that the identified simplifications have insignificant impact on the principal results of the simulations.

3.4 Model implementation

The model was implemented as a discrete event simulation model in SDL (ITU–T, 1999). A discrete event simulation model was chosen, because it is a straightforward way of implementing models that represent networks of queues. SDL was chosen because it is based on real-time processes and it supports the creation of discrete event simulators. Another advantage of choosing SDL is that the case tool that the modelled company develops can be used to develop systems in SDL. In fact, the model is implemented with the case tool developed by Telelogic. This means that almost all people at the company understand the notation of the model. The presented research represents the first attempt to model the REPEAT process in a simulation model. In the future it would be possible to carry out not only discrete event simulations, but also systems dynamics simulations of the process.

After the model was implemented, a simple version was created by making all serving times exponentially distributed. Special-case parameters were used which enabled analytical validation of the simulator through e.g. Little's theorem (King, 1990). This proved that the simulator was implemented according to the queuing network model. The complete simulator was then validated by comparing throughput and congestion against data from the expert interviews. The simulator matched the real number of requirements implemented per release, and therefore also the real number of requirements waiting to be implemented after five enactments of the REPEAT process.

4. Results

The results from simulations of the REPEAT process are based on a number of executions of the simulator with various input parameter settings in order to verify the simulation model and draw conclusions from it. The simulation model can be used to analyse many different characteristics of requirements management processes, and the simulation results may act as a valuable decision support.

Four simulation cases are presented in order to illustrate the usage of the simulator and to show the impact of process changes.

The first case is a baseline situation where no prioritization of requirements are made, i.e. all requirements have priority 2. This case is primarily used to verify the model and is presented and analysed to facilitate comparison with the following cases. The second case introduces prioritization and is based on the actual situation as determined by the data from interviews. In the third and fourth case, changes to the simulation parameters are introduced in order to investigate what amount of increase in capacity or decrease in work load is needed to make the process stable. These values have been found by observing the result for a number of different values. The parameter values that are used in the simulations are summarized in Table 2.

4.1 Case 1: No priorities

A baseline situation, to which other cases can be compared, is when every elicited requirement is selected and subsequently constructed within the same release, and all requirements have priority 2. In this case, no requirements are re-routed between releases based on priority (requirements are only re-routed when time constraints forces requirements to be forwarded to the selection phase of the next release).

Figure 3 shows the total number of requirements in the selection phases for a number of releases. The selection phases are shown for up to five full releases, with the left-most curve corresponding to release 1 and the subsequent curves corresponding to the following releases. Release 6 and 7 can thus be viewed in part.

The y–axis shows, for each release, the sum of requirements in the queue and requirements currently being processed by the servers. From the time when selection begins, there are always requirements to handle and the servers are never idle. For example, in release 1 there are

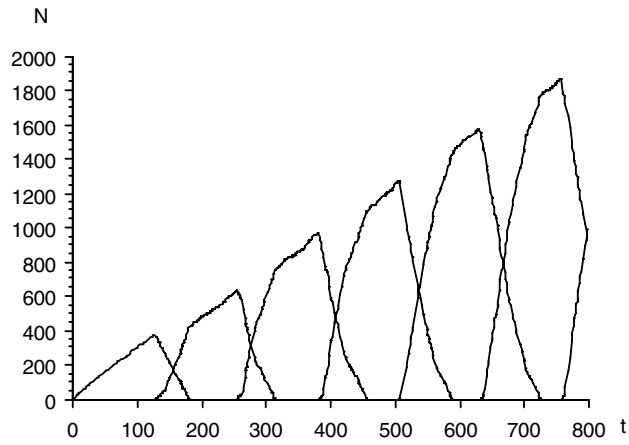

Figure 3. *Case 1: No priorities. Number of requirements in the selection phases of releases 1–5 and parts of release 6 and 7.*

approximately 400 requirements waiting in the queue ready to be analysed. Thus, from the time when selection begins the servers are constantly busy until all requirements have been handled.

An important conclusion that can be made from the figure is that this process is overloaded. In the fifth release there are approximately 1600 requirements waiting to be handled in the next selection phase and the amount increases to the next release. There are approximately an additional 250 requirements for each release.

There is no rejection of requirements in the simulation model. All requirements entering the elicitation phase are kept in the system until released. The requirements that the construction servers do not manage to implement are forwarded to the next release. As the figure shows, the number of requirements in the queue is constantly increasing. Rejecting some requirements, such as duplicates or obsolete requirements, would reduce the queue build up. From the interviews it is found that in reality only about 5–10% of the total number of requirements are rejected. This low rejection rate is not enough to make the process stable.

In release 2, a slight deflection at day 180 from the start of the simulation can be noticed. This happens when requirements no longer are forwarded from the previous release.

In release 1, the slope of the first half of the curve corresponds to that of the arrival intensity. In the subsequent releases the slope of the curves, after the deflections, also corresponds to the arrival intensity. We can therefore draw the conclusion that the elicitation phase does not get

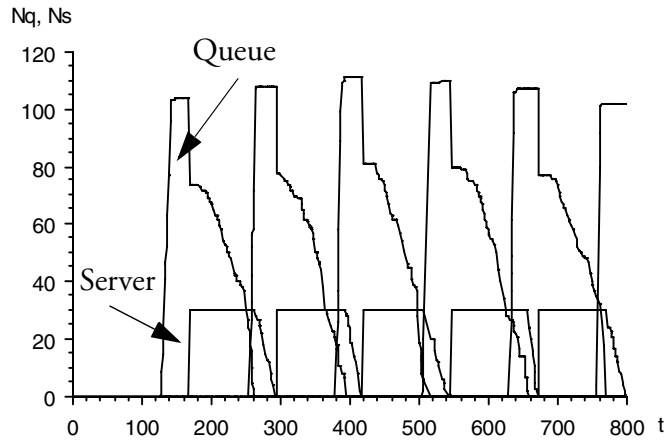

Figure 4. *Case 1: No priorities. Number of requirements in the construction phases of releases 1–5.*

overloaded; it is selection and construction that is in a state of congestion. This conclusion can be verified by a separate analysis of the elicitation phase.

Figure 4 shows the results from the simulation of the construction phases of subsequent releases. As before, the graph shows releases 1 through 5 from left to right. The construction phase is situated later in time and consequently we can only see a part of release 6.

This graph shows both the number of requirements in the queue overlaid with the number of requirements currently being handled in the servers. Focusing on one release, as shown magnified in Figure 5, it can be seen that the queue builds up rather fast. The fast build-up of the queue (105 requirements in about 30 days) can be derived from the selection phase graph (Figure 3). By observing how many requirements that are handled from the point when the selection phase begins and 30 days ahead, it is clear that the decrease corresponds to the number of requirements arriving to the construction phase during the same time period.

The increase of the number of requirements in the construction queue of release 1 abruptly stops at 105 requirements. The reason for this is that when the total effort of the requirements transferred to the construction phase equals the available time in the construction phase, no more requirements arrive from the selection phase.

When construction begins there is a large drop from 105 requirements in the queue to only 75 requirements. This is when all 30 available servers

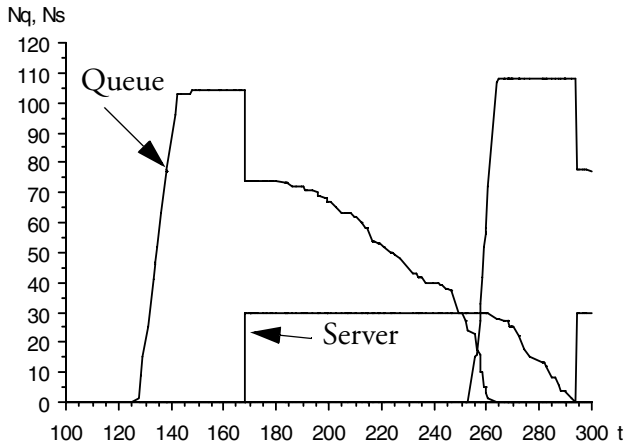

Figure 5. *Case 1: No priorities. Number of requirements in the construction phase of release 1.*

are occupied at once. Then, the servers are constantly busy until there are no more work to do, continuously taking care of the requirements in the construction queue.

It may be tempting to read off the graph that 105 requirements are implemented in the first release. This is unfortunately not true. As it is stated in Section 2, only about half of the wish-list will be implemented. Since the needed effort is not equal for every requirement in the construction phase, it is not possible to use the graph to calculate the number of requirements that is actually implemented. For each release, a certain percentage of the requirements arriving to the construction phase will actually be implemented. The remaining requirements are sent to the selection phase of the next release.

In Figures 4 and 5 it can be seen that when a succeeding release receives requirements to the selection phase, the construction phase of the current release has not yet finished and implementation is still undertaken. When there are no more requirements to implement the servers representing employees enter an idle state and are ready to get to work in the next release. This can be seen in release 1 approximately after day 260.

4.2 Case 2: Actual situation

Using the parameters determined from expert interviews (see Section 3.2) the actual situation can be simulated. Prioritization is now introduced, as

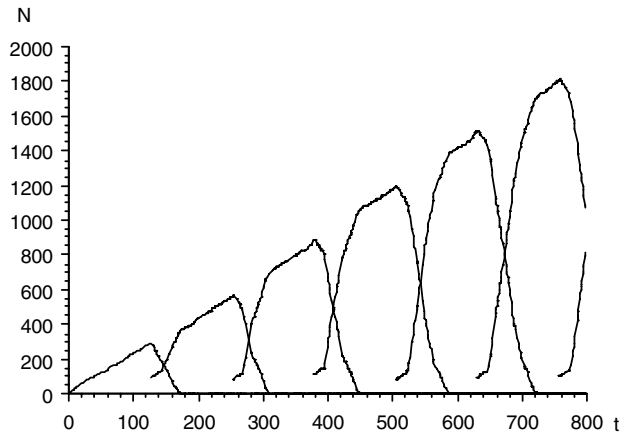

Figure 6. *Case 2: Actual Situation. Number of requirements in the selection phases of releases 1–5.*

explained in Section 2. This will result in some requirements being transferred to the next release and a few requirements being sent to a previous release still under development.

Figure 6 shows the selection phases of release 1 through 5 as before. We can see that there is not much difference from the previous case. As before, the selection phase is overloaded. This is because of the time constraints in the construction phase, and because there is no rejection of requirements. If the capacity is not enough to take care of all requirements, priorities will not completely solve the situation. Priorities will however make the organization focus on the most important requirements.

The reason that no curve, from release 2 and onwards, start at zero requirements requires an explanation. In the simulation model a subsequent release and its queue is not started until it actually should be, as shown in Figure 1. Thus, when the simulation of a release is started, there are some requirements already waiting from a previous release. Here the first part of the curve corresponding to the initial build-up of the queue is not shown.

Since no conditions for the construction phase has been changed compared to case 1, Figure 7, showing the construction phase, does not differ much. About the same number of requirements, on average, are implemented when we add prioritization. The analysis of the construction phase made for case 1 is appropriate for this case as well.

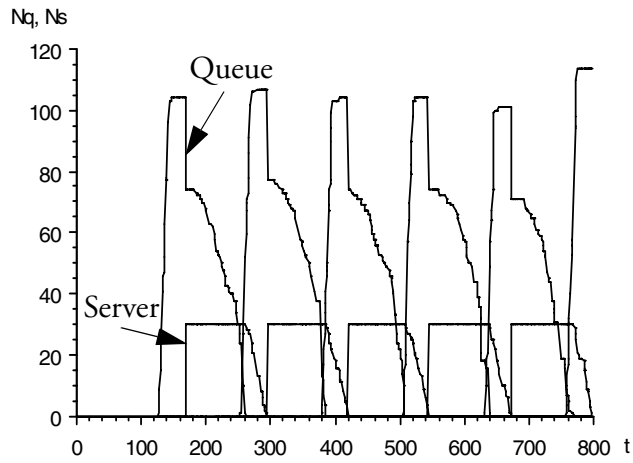

Figure 7. *Case 2: Actual Situation. Number of requirements in the construction phases of releases 1–5.*

4.3 Case 3: Increased capacity

A first obvious change of the process in order to remove bottlenecks would be just to reallocate resources or adding more people. This alternative is simulated in order to ascertain what the impact would be and to find the amount of resources needed (or the needed increase in productivity) to reduce the bottlenecks.

Figure 8 shows the selection phases of 13 releases and Figure 9 shows the construction phase of release 1 after increasing the number of servers in these phases in order to make the process stable. Many releases are shown in Figure 8 to support the assumption of stability. To make it stable, 16 employees are required in the selection phase and as many as 165 persons in the construction phase. This means an increase in construction capacity by a factor 165/30=5.5. Elicitation is not a bottleneck, as it is enough with only one elicitation server (see Table 2).

It can be seen in Figure 9 that when the construction starts (day 168) there is enough server capacity available to take care of every requirement in the queue and additional ones arriving in the following 10 days. However, the utilization of the resources in the construction phase is lower than before.

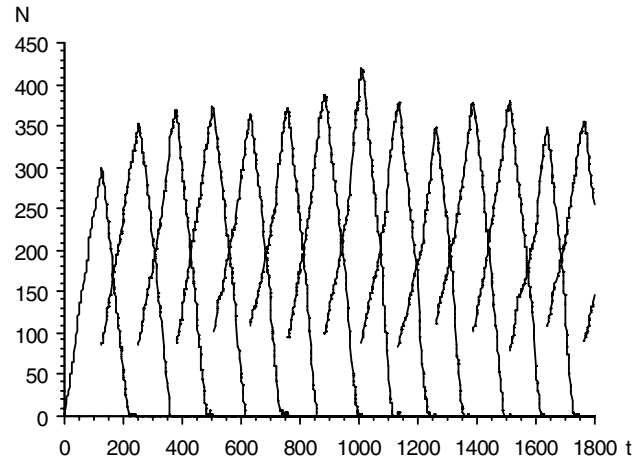

Figure 8. *Case 3: Number of requirements in the selection phases of releases 1–13.*

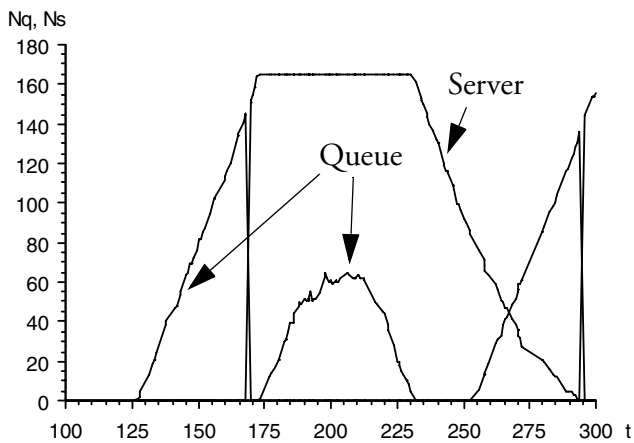

Figure 9. *Case 3: Number of requirements in the construction phase of release 1.*

4.4 Case 4: Decreased work load

Another approach to remove the state of congestion is to reduce the number of requirements that arrive to the elicitation phase and thereby reduce the number of requirements that are forwarded to the selection phase.

In Figure 10, the impact of reducing the arrival intensity is shown. Here the arrival intensity has been decreased from the value from reality of 3 requirement a day to only 0.55 requirements a day. This means a decrease in the rate of incoming requirements from 3 to 1/1.8=0.55,

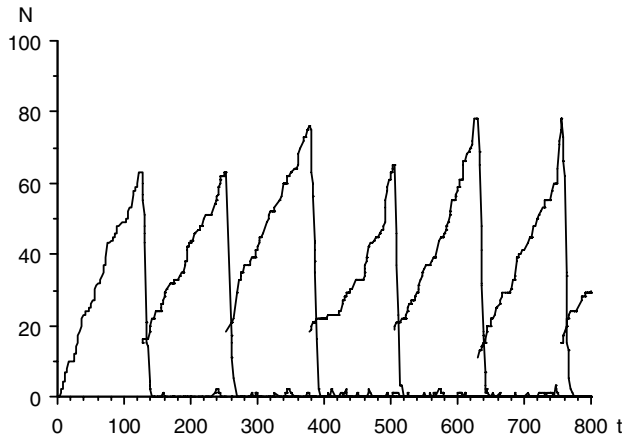

Figure 10. *Case 4: Number of requirements in the selection phases of releases 1–5.*

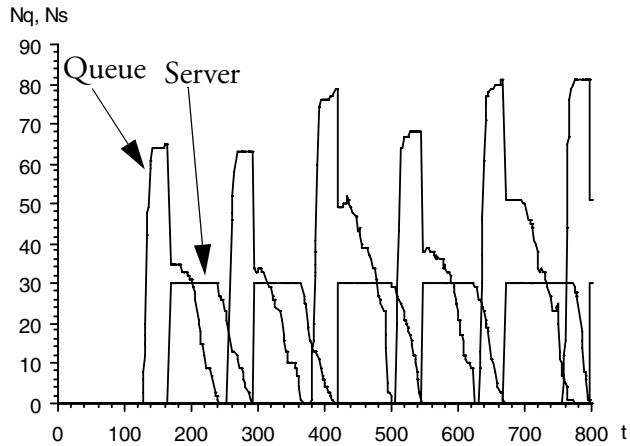

Figure 11. *Case 4: Number of requirements in the construction phases of releases 1–5.*

which corresponds to 0.55/3=18% of the actual value. This reduces the maximum number of requirements in the selection phase to about 80 requirements, which is a notable decrease.

As Figure 11 shows, only about 80 requirements are received to the construction phase for each release. However, now there are resources enough to implement every requirement. The requirements in the selection phase can all be forwarded to the construction phase and implemented.

5. Conclusions

This paper presents a study that shows how discrete event simulation can be used in order to explore overload conditions of an industrial software requirements management process for packaged software. The simulation model is created based on a previous study of the process (Regnell et al., 1998) and the simulation parameters are estimated based on interviews with a process expert with in-depth knowledge of how the process performed during the studied period of 1998–1999. The situation of overload that has been observed in reality can also be observed when executing the simulation model.

It can be concluded from the simulations that there are at least two different ways of changing the process in order to avoid congestion:

- The capacity of the requirements management process can be improved, either by increasing the number of employees or by improving productivity. The simulations show that it is necessary to increase the capacity of the construction phase by a factor 5.5 in order to completely remove the bottlenecks.

- The workload on the requirements management process can be decreased. The simulations show that it is necessary to decrease the rate of issuing new requirements to 18% of the initial value. One way of lowering the workload is through early prioritization (Karlsson, Wohlin, & Regnell, 1998) and thereby rejecting requirements that will not be implemented at an early stage in the process. Of course, this is an area that needs further investigation, as it is always difficult to remove issues early in a process. If the objective is to remove requirements in order to lower the effort required in analysis, then an analysis effort is required in order to know which requirements to remove. Prioritization of requirements and release planning in packaged software development is acknowledged to be an important research area (Regnell, Höst, Natt och Dag, Beremark, & Hjelm, 2000).

In conclusion, the presented simulations represent a feasible way of analysing the investigated process. However, the simulation model can be improved in a number of ways. One improvement regards a more realistic modelling of the must- and wish-lists (see Sections 2-3). Another area of further work is to make a more thorough analysis of the real process by

interviewing more people representing more roles in the organization about their opinions concerning both the parameter values of the model and the validity of the simulation results.

It is also interesting to extend the simulation model by modelling the servers as a single pool of resources where each resource (employee) has certain competencies. This may lead to a more realistic simulation model, where the same persons are involved in many tasks, and, as in reality, the lack of a certain competency may be a bottleneck.

A promising benefit of the presented approach is the potential of achieving validated decision support that can facilitate informed decisions on improvements of software processes in general, and market-driven requirements engineering processes in particular.

Acknowledgements

The authors would like to thank all people involved in the development and investigation of REPEAT, in particular Per Beremark (Group Quality Manager at Telelogic) without whom this work would not have been possible. We would also like to thank Carina Andersson and Lena Karlsson, both with the Dept. of Communication Systems, for providing suggestions for improvements of the simulation model. The presented research is partly funded by the National Board of Industrial and Technical Development (NUTEK).

6. References

Banks, J., Carson, J. S., & Nelson, B. (1996). *Discrete-Event System Simulation* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Höst, M. & Wohlin, C. (1997). A Subjective Effort Estimation Experiment. *Information and Software Technology, 39*, 755–762.

Höst, M., & Wohlin, C. (1998). An Experimental Study of Individual Subjective Effort Estimations and Combinations of the Estimates. In B. Werner (Ed.), *Proceedings of 20th International Conference on Software Engineering* (pp. 332–339). Los Alamitos, CA: IEEE Computer Society Press.

ITU–T Z.100–11/99. *Specification and Description Language (SDL)*.

Karlsson, J., Wohlin, C., & Regnell, B. (1998, February). An Evaluation of Methods for Prioritizing Software Requirements, *Information and Software Technology, 39*, 939–947.

Kellner, M.I., Madachy, R.J., Raffo, D.M. (1999, April 15). Software Process Simulation Modeling: Why? What? How? *Journal of Systems and Software, 46*, 91–105.

King, P. J. B. (1990). *Computer and Communication Systems Performance Modelling*. London: Prentice-Hall.

Lubars, M., Potts, C., & Richter, C. (1993). A Review of the State of the Practice in Requirements Modeling. In *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 2–14). Los Alamitos, CA: IEEE Computer Society Press.

Potts, C. (1995). Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software In *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 128–130). Los Alamitos, CA: IEEE Computer Society Press.

Pfahl, D., & Lebsanft, K. (2000). Using Simulation to Analyse the Impact of Software Requirement Volatility on Project Performance. In K. D. Maxwell, R. J. Kusters, E. P. W. M. van Veenendaal, & A. J. C. Cowderoy (Eds.) *Project Control: The Human Factur, Proceedings of the combined 11th European Software Control and Metrics Conference and the 3rd SCOPE conference on Software Product Quality* (pp. 267–275). Maastricht, Holland: Shaker Publishing.

Regnell, B., Beremark, P., & Eklundh, O. (1998). A Market-Driven Requirements Engineering Process – Results from an Industrial Process Improvement Programme. *Requirements Engineering, 3*, 121–129.

Regnell, B., Höst, M., Natt och Dag, J., Beremark, P. & Hjelm, T. (2000). Visualization of Agreement and Satisfaction in Distributed Prioritization of Market Requirements, In A. L. Opdahl, K. Pohl, & M. Rossi (Eds.), *Proceedings of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality.* Essen, Germany: Essener Informatik Beiträge.

Sawyer, P. (2000). Packaged Software: Challenges for RE. In *Proceedings of Sixth International Workshop on Requirements Engineering: Foundation for Software Quality* (pp. 137–142). Essen, Germany: Essener Informatik Beiträge.

Sawyer, P., Sommerville, I., & Kotonya, G. (1999). Improving Market-Driven RE Processes. In M. Oivo, & P. Kuvaja (Eds.), *Proceedings of the International Conference on Product Focused Software Process Improvement* (pp. 222–236). Oulu, Finland: Technical Research Centre of Finlad (VTT).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). *Experimentation in Software Engineering – An Introduction*. Norwell, MA: Kluwer.

Yeh, A. (1992). Requirements Engineering Support Technique (REQUEST) – A Market Driven Requirements Management Process, In *Proceedings of the Second Symposium on Assessment of Quality Software Development Tools* (pp. 211–223), Los Alamitos, CA: IEEE Computer Society Press.

PAPER II

An industrial case study of usability engineering in market-driven packaged software development

Johan Natt och Dag, Björn Regnell, Ofelia S. Madsen, Aybüke Aurum

M. J. Smith, G. Salvendy, D. Harris & R. J. Koubek (Eds.), *Proceedings of HCI International: Vol. 1. Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents and Virtual Reality* (pp. 425-429). Mahwah, NJ: Erlbaum. 2001.

II

Abstract

In market-driven software development it is crucial to produce the best product as quickly as possible in order to reach customer satisfaction. Requirements arrive at a high rate and the main focus tends to be on the functional requirements. The functional requirements are important, but their usefulness relies on their usability, which may be a rewarding competitive means on its own. Existing methods help software development companies to improve the usability of their product. However, companies that have little experience in usability still find them to be difficult to use, unreliable, and expensive. In this study we present results and experiences on conducting two known usability evaluations, using a questionnaire and a heuristic evaluation, at a large software development company. We have found that the two methods complement each other very well, the first giving scientific measures of usability attributes, and the second revealing actual usability deficiencies in the software. Although we did not use any usability experts, evaluations performed by company employees produced valuable results. The company, who had no prior experience in usability evaluation, found the results both useful and meaningful. We can conclude that the evaluators need a brief introduction on usability to receive even better results from the heuristic evaluation, but this may not be required in the initial stages.

Much more essential is the support from every level of management. Usability engineering is cost effective and does not require many resources. However, without direct management support, usability engineering efforts will most likely be fruitless.

1. Introduction

When developing packaged software for a market place rather than bespoke software for a specific customer, short time-to-market is very important (Potts, 1995). Packaged software products are delivered in a succession of releases and there is a strong focus on user satisfaction and market share (Regnell, Beremark, & Eklundh, 1998). Thus, companies tend to put their primary effort into inventing and implementing new functional features that are expected to improve the product. Although developers rely heavily on the number and the existence of new features, usability is recognized as a competitive advantage on its own. Still after many years of usability engineering research, there are many companies that do not approach and explicitly improve usability. Although several methods and techniques exist (Nielsen, 1993) and studies show their cost-effectiveness (Bias & Mayhew, 1994), the seeming difficulty of approaching usability prevents the success of companies. Since usability evaluations of software products are necessary in order to increase user-friendliness (Nielsen, 1993), there is a need to put even more focus on usability evaluation methods that are easy to use and adopt and that give fast and appropriate results.

In this paper we present an industrial case study that employs two known usability evaluation methods (Natt och Dag & Madsen, 2000). The study was conducted at Telelogic AB, a large software developing company in Sweden, and the methods were used to evaluate their main product, Telelogic Tau, a graphical software development tool aimed for the telecommunications industry. It is shown that the two methods may be used for continuous evaluation of usability without much effort or resources and without any particular experience in usability engineering.

2. Research methodology

Several factors have been taken into consideration when choosing evaluation methods. The methods must be easy to perform and give understandable results that can be utilized in daily work without extraordinary analysis. Furthermore, the methods need to be appropriate for use over and over again, and it must be possible to extend the proficiency in using these methods when experience in usability evaluation within the company increases. If usability experience is lacking, the methods must not be too sensitive to the evaluators' performances.

To obtain satisfactory results that fulfil these requirements, we carefully selected two known usability evaluation methods, a questionnaire and a heuristic evaluation, which give quantitative and qualitative results respectively.

2.1 The SUMI questionnaire

In order to obtain end users' opinions about the software we used a commercially available questionnaire, 'The Software Usability Measurement Inventory' (SUMI) (Kirakowski & Corbett, 1996). SUMI is a standard questionnaire specifically developed and validated to give an accurate indication of which areas of usability should be improved. It is tested in industry and mentioned in ISO–9241 as a method of measuring user satisfaction. The questionnaire consists of 50 statements to which each end user answers whether he or she agrees, disagrees or is undecided. Only 12 end users are necessary to get adequate precision in the analysis, but it is possible to use fewer users and still obtain useful results.

The questionnaire was sent to 90 selected end users in Europe. Returned answers were sent to the questionnaire designers who statistically analysed the answers and compared them with the results in a continuously updated standardization database. The database contains the answers from over 1000 SUMI questionnaires used in the evaluations of a wide range of software products. The comparison results in the measurement of five usability aspects:

- Efficiency – the degree to which users feel that the software assists them in their work.
- Affect – the user's general emotional reaction to the software.

- Helpfulness – the degree to which the software is self-explanatory. Adequacy of documentation.
- Control – the extent to which the users feel in control of the software.
- Learnability – the ease with which the users feel that they have been able to master the system.

Each aspect is represented by 10 statements in the questionnaire and the raw scores for each of the aspects are converted into scales with a mean value of 50 and a standard deviation of 10, with respect to the standardization database. Also, a global scale is calculated that is represented by the answers to 25 of the 50 statements that best reveal global usability.

To identify the statements to which the answers differ significantly from the average in the standardization database, *item consensual analysis*, a feature developed by the questionnaire designers, is used. Through comparison between the observed and expected answer patterns, the individual usability problems may be more accurately determined.

2.2 The heuristic evaluation

To find usability problems specific to the evaluated software, we used a slightly extended version of a standard heuristic evaluation (Nielsen, 1994). In the heuristic evaluation, usability experts go through the interface and inspect the behaviour of the software. The behaviour is compared to the meaning and purpose of a set of ten guidelines called heuristics that focus on the central and most important aspects of usability, such as 'user control and freedom' and 'flexibility and efficiency in use'. This enables the evaluator to systematically check the software for usability problems against actual requirements in the specification and given features in the product. The result is a list of identified usability problems that may be used to improve the software.

In the market-driven development organization, there may be little experience in usability and usability experts may not be available. As this was the situation at Telelogic, we used experts on the software from within the company. The number of evaluators needed may then increase, as less usability problems may be found. Experts specializing only in usability tend to find problems mainly related to how easy the system is to

operate, whereas domain experts rather find problems related to how well the system responds to its intended behaviour. In this sense the usability and the domain experts complement each other, as domain experts, according to Muller, Matheson, Page and Gallup (1998), bring perspectives and knowledge not otherwise available. However, an evaluation is more likely to be conducted in the first place if we initially do not require the involvement of usability experts.

Twelve employees from within the company participated in the study. They were presented with a set of scenarios comprised of different tasks to perform. The method was extended in order to add increased structure to the evaluation and to help the evaluators to stay focused on usability issues. This was accomplished through the introduction of *usability breaks* (UB). A UB is a position in the detailed scenario where the evaluator is supposed to (1) stop the execution of the scenario and write down any problems found up to that point together with the associated heuristics, and (2) go through the ten heuristics to identify additional problems with the tasks just performed.

The evaluators were advised to spend 1 uninterrupted hour on finding relevant usability problems. In addition to generating a list of problems, the evaluators were asked to write down during which scenario identified problem were encountered, at what specific UB they were found, the heuristics that apply, the severity of each problem (high, medium or low), and a suggestion of a solution.

3. Results

3.1 The SUMI questionnaire

Of the 90 questionnaires sent to end users, 62 were properly filled out and returned. The analysis of the returned answers revealed that the evaluated software did not meet the appropriate standards on several aspects of usability. In Figure 1 the medians for each of the six SUMI scales are shown. The median corresponds to the middle value in a list where all the individual scores given by each evaluator have been numerically sorted. The figure also shows error bars for each scale, representing the upper and lower 95% confidence limits. As seen in the figure, all but two of the six SUMI scales are below average. The sub-scale affect is the only one that lies above average, indicating that users feel slightly better about this

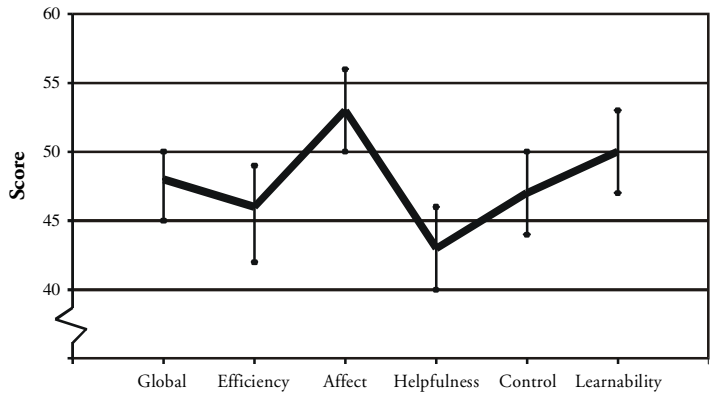

Figure 1. *The SUMI satisfaction profile.*

product than they feel in general about software products. The learnability sub-scale indicates that the software may be regarded to be as easy or hard to learn as software products are in general.

The item consensual analysis (see Section 2.1) revealed 9 specific statements that differed significantly from the standardization database (99.99% certain). In Figure 2, the statement that was most likely to differ from the expected is shown. This particular result reveals that the software may not be very helpful. Of the remaining 8 statements that most certainly differed from the standardization database, only 1 statement generated a more positive response than what was expected. As many as 74% of the end users disagreed with the statement 'if the software stops it is not easy to restart it', indicating that the software is easy to restart. Further analysis of the statements revealed several reasons to the low scores in Figure 1.

Statement 28: The software has helped me overcome any problems I have had using it.

	Agree	**Undecided**	**Disagree**
Profile	12	19	31
Expected	17.10	30.97	13.93
Chi Square	1.52	4.63	20.93

Figure 2. *Item consensual analysis of the answers to statement 28.*

Table 1. Sample usability problem found by an evaluator.

Problem	How to change a unidirectional channel to a bidirectional channel
Scenario	A
UB	1.3
Heuristic	Flexibility and efficiency in use
Severity	High
Solution	Add a channel symbol to the symbol menu and then add symbols for unidirectional (one in each direction) and bidirectional.

3.2 The heuristic evaluation

The heuristic evaluation revealed 72 unique usability problems directly related to the software application. A sample usability problem identified by an evaluator is shown in Table 1. About 20% of the identified problems were considered highly severe, about 65% somewhat severe, and no more than 14% less severe. This indicates that usability needs attention in order to increase the usefulness of the software, which is confirmed by the results from the more reliable SUMI questionnaire evaluation.

Solutions were given to most of the problems but for the 18 that had none a solution may be inferred through the problem descriptions and through the particular UBs in the scenario. The problem descriptions had high enough quality to be used as input into the requirements process.

Figure 3 shows the number of times each of the ten heuristics were used to classify the identified problems. The high use of flexibility and efficiency in use indicates that users may get frustrated when using the software. Further analysis of the particular problems related to this heuristic and at which UBs the problems were encountered, reveals that there are many tasks that are bothersome to complete due to non-intuitive functionality and a non-supportive graphical interface.

The results in Figure 3 are confirmed by the results from the SUMI questionnaire evaluation. Efficiency is identified as a problematic area by both methods and helpfulness is related to recognition rather than recall and user control and freedom. This and the fact that experts on the software were used as evaluators indicate that the heuristic evaluation pinpoints relevant usability problems.

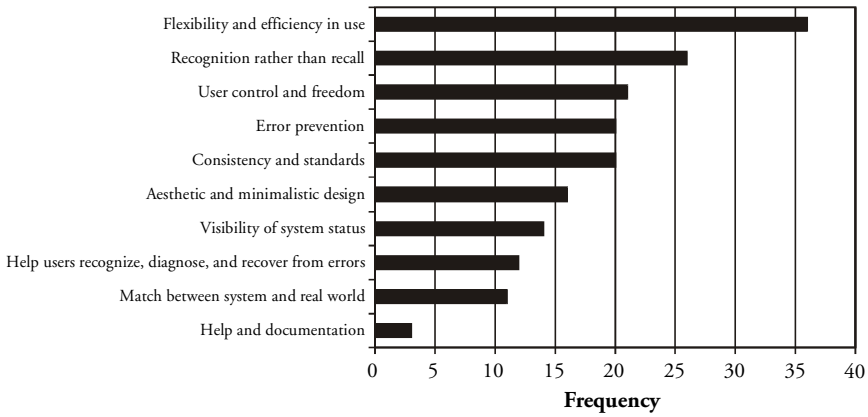

Figure 3. *The number of heuristics used to classify the 72 identified usability problems.*

4. Conclusions

In this paper we have presented the results of using two known usability evaluation methods at a market-driven software development company inexperienced in usability. We have found that although experience on usability is lacking, the two methods are easy to use and do not require many resources. The questionnaire is available at a low cost and system experts can easily develop the heuristic evaluation scenarios. Furthermore, the two methods complement each other very well. Both kinds of result were found to be usable and meaningful by the developing company and the generated problem list was particularly welcomed. The results gave them insight into the specific areas that needed improvement and helped them to appreciate the issues to put their effort into.

The selection of evaluators was the most time-consuming task. Mainly, this was because there was little support for usability issues. There was a noteworthy interest from the development department, but we have found that management support on every level in the organization is crucial to effectively get results. Without management support it is not very likely that the results will be used in further development at all. Also, a short 30-minute introduction to the concept of usability will most likely motivate the evaluators to perform even better and be more focused on usability issues in particular.

The initial drawbacks were nevertheless highly compensated by the low cost and the quick and useful results. The estimated costs of applying

Table 2. Estimation of cost of applying the two usability evaluation methods.

	Method	Heuristic Evaluation	SUMI Questionnaire
Cost (in man-days)	**Small**	2	1
	Medium	4	3
	Extensive	4	3
	Training	1	2
	Material	None	US$500
	Reliability	Medium	High

the methods are shown in Table 2 (Melchior, Bösser, Meder, Koch, & Schnitzler, 1995).

Currently, we are applying a follow-up study to investigate precisely how the generated problem lists have been used in succeeding releases, what impact on software usability they have had, and to what extent the usability has increased.

Acknowledgements

This work is partly funded by the National Board of Industrial and Technical Development (NUTEK), Sweden, within the REMARKS project (Requirements Engineering for Market-Driven Software Development) grant 1K1P–97–09690.

5. References

Bias, R. G., & Mayhew, D. J. (1994). *Cost-justifying usability.* Boston: Academic Press.

Kirakowski, J., & Corbett M. (1996). The software usability measurement inventory: Background and usage. In P. W. Jordan, B. Thomas, B. A. Weerdmeester, & I. L. McClelland (Eds.), *Usability Evaluation in Industry* (pp. 169–177). London: Taylor & Francis.

Melchior, E.-M., Bösser, T., Meder, S., Koch, A., & Schnitzler, F. (1995). Usability Study – Handbook for practical usability engineering in IE projects (Report. No. ELPUB 105 10107). Cork, Ireland: University College Cork, The BASELINE Consortium.

Muller, M. J., Matheson, L. Page, C., & Gallup, R. (1998). Participatory Heuristic Evaluation. *Interactions, 5*(5), 13–18.

Natt och Dag, J., & Madsen, O. S. (2000). *An Industrial Case Study of Usability Evaluation* (Master's Thesis. Report No. CODEN:LUTEDEX (TETS–5390)/1–190/ (2000)&local 8). Lund: Lund University: Department of Communication Systems.

Nielsen, J. (1993). *Usability Engineering*. San Diego, CA: Morgan Kaufmann.

Nielsen, J. (1994). Heuristic Evaluation. J. Nielsen, & R. L. Mack (Eds.), *Usability Inspection Methods* (pp. 25–61). New York: John Wiley & Sons.

Potts, C. (1995). Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering* (pp. 128–130). Los Alamitos, CA: IEEE Computer Society Press.

Regnell, B. Beremark, P., & Eklund, O. (1998). A Market-driven Requirements Engineering Process – Results from an Industrial Process Improvement Programme. *Requirements Engineering, 3*, 121–129.

PAPER III

An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software

III

Björn Regnell, Martin Höst, Johan Natt och Dag, Per Beremark, Thomas Hjelm

Requirements Engineering, 6, 51–62. Springer-Verlag, 2001.

Abstract

When developing packaged software which is sold "off-the-shelf" on a world-wide market place, it is essential to collect needs and opportunities from different market segments and use this information in the prioritization of requirements for the next software release. This paper presents an industrial case study where a distributed prioritization process is proposed, observed and evaluated. The stakeholders in the requirements prioritization process include marketing offices distributed around the world. A major objective of the distributed prioritization is to gather and highlight the differences and similarities in the requirement priorities of the different market segments. The evaluation through questionnaires shows that the stakeholders found the process useful. The paper also presents novel approaches to visualize the priority distribution among stakeholders, together with measures on disagreement and satisfaction. Product management found the proposed charts valuable as decision support when selecting requirements for the next release, as they revealed unforeseen differences among stakeholder priorities. Conclusions on stakeholder tactics are provided and issues of further research are identified including ways of addressing identified challenges.

1. Introduction

Requirements Engineering (RE) research is beginning to address the specific issues in market-driven RE, where requirements are invented for packaged software offered "off-the-shelf" to a mass market (Potts, 1995; Lubars, Potts, & Richter, 1993). This situation is in many respects different from the contract situation, where a developer interacts with a specific customer to elicit requirements for a bespoke system. A description of the specific challenges of market-driven RE can be found in (Potts, 1995; Lubars, Potts, & Richter, 1993; Sawyer, Sommerville, & Kotonya, 1999; Kamsties, Hörmann, & Schlich, 1998). As pointed out by Sawyer (2000), two major differences between bespoke software development and market-driven software development regards the characteristics of stakeholding and schedule constraints. The developer decides about the requirements and selects the stakeholder representatives. The developer bears all financial risks and there is no single customer who is the principal stakeholder as with bespoke software development. There is a major pressure on time-to-market and the software product is often offered to a market through recurrent releases, requiring careful release planning and requirements prioritization. Such challenges pose special demands on the RE process. Examples of market-driven RE processes, which try to address these special demands, can be found in (Regnell, Beremark, & Eklund, 1998; Deifel, 1999; Yeh, 1992; Carmel & Becker, 1995).

This paper focuses on the important challenge within market-driven RE of how to combine information from different market segments and make a trade-off between their priorities. In particular, the paper focuses on the visualization of disagreement between stakeholders and differences in the satisfaction with a certain priority decision. A number of charts are proposed, which are intended to be used as decision support when determining what to implement in the coming release of a software package.

The presented work is conducted in the context of an industrial case study, which is a follow-up on the study reported in (Regnell, Beremark, & Eklund, 1998). One of the main challenges identified in (Regnell, Beremark, & Eklund, 1998) was to relate the continuous prioritization of incoming requirements to a long-term product strategy for a range of market segments. An important issue here is how to incorporate the expertise from marketing departments and the visions of top-level

management in the prioritization process. This paper focuses on how to elevate a prioritization strategy, such as (Karlsson & Ryan, 1997) or (Büyükekici, Deifel, Jacobi, & Sandner, 1999), by making differences in the priorities of the various market segments explicit.

The presented case study is conducted as part of an improvement program for a specific industrial RE process for packaged software, called REPEAT (Requirements Engineering Process at Telelogic), which is enacted by the Swedish CASE-tool vendor Telelogic AB; a company with 450 employees (January 2000), more than 7000 users world-wide, and a revenue for 1999 of 318 million SEK (increase from 178 million SEK, 1998). REPEAT is used in-house at Telelogic for eliciting, selecting and managing requirements on a product family called Telelogic Tau; a software development tool package for real-time systems, used by many of the world's largest telecom systems providers in their software development. Telelogic Tau supports standardized graphical languages, such as SDL, MSC, TTCN, and UML, and provides code generators for integration with several real-time operating systems (for further information, see www.telelogic.com).

The paper is structured as follows. Section 2 describes the process used within the presented case study for making distributed requirements prioritization in a world-wide organization. The planning and operation of the case study is reported in Section 3. The main findings from the questionnaire-based evaluation of the distributed prioritization case study is summarized in Section 4, together with a description of the visualization charts for supporting the selection of requirements. Section 5 provides a discussion on the findings and identifies issues of further research.

2. A distributed prioritization process

When selling packaged software, the potential market segments may be spread worldwide. This calls for a distributed marketing organization with close relations to targeted customers. The organization, in which the presented case study has been conducted, is depicted in Figure 1. The Product Strategy Team (PST) is responsible for making strategic decisions and communicates with a number of Market Operations (MOs) for gathering information and promoting strategies. The PST also communicates with Customer Support and Product Development.

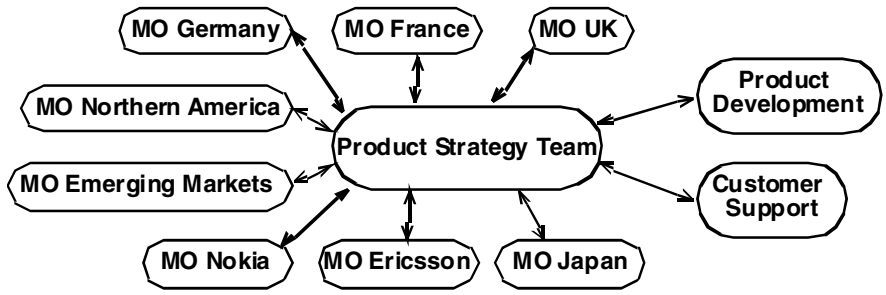

Figure 1. *The Product Strategy Team communicates with a number of stakeholders representing valuable information sources on market opportunities, user expectations, and technology trends.*

At the time of writing there are 8 Market Operations representing 6 geographically segmented markets in Northern America, Germany, France, United Kingdom, Japan, Emerging Markets, and also 2 markets segmented by key corporations in the telecom industry. The centralized Customer Support department has valuable information regarding what customers desire based on support issues. Similarly, the centralized Product Development department has valuable information on where technology is heading. Hence, there are in total 10 stakeholders from which the PST gathers information on requirements priorities, before making strategic decisions regarding products. These strategies are applied in every-day requirements decisions, made by the Requirements Management Group (More information on the role of this group can be found in (Regnell, Beremark, & Eklund, 1998)).

In the efforts of further improving RE at Telelogic, a baseline sub-process for strategic prioritization of high-level requirements was formulated where each stakeholder can contribute with their particular priorities. This process is the object of the presented pilot case study which started in October 1999 (see further Section 3). The sequencing of tasks in the process is illustrated in Figure 2.

Step 1. The PST starts the process by making a candidate list of strategic high-level requirements. The requirements on the candidate list are divided in two abstraction levels. At the high level, the items are feature groups, and at the next level the items are individual features.

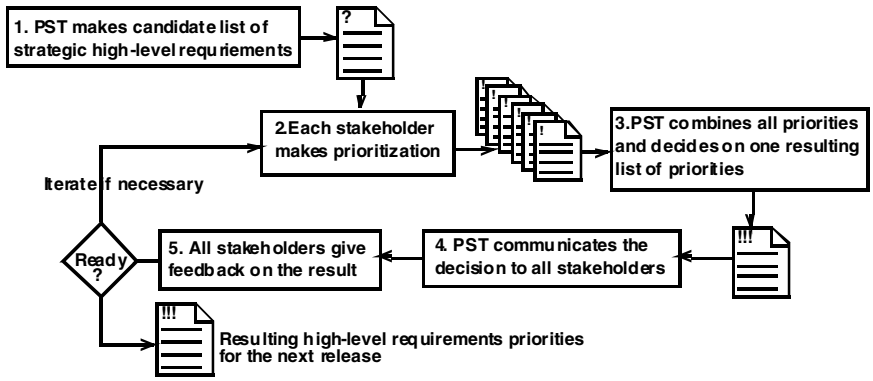

Figure 2. *The Product Strategy Team communicates with a number of stakeholders representing valuable information sources on market opportunities, user expectations, and technology trends.*

Step 2. The candidate list is distributed to the different stakeholder around the world. Each stakeholder in parallel makes a prioritization of the requirements on the candidate list. It is also possible for each stakeholder to add new features or feature groups to the list.

Step 3. Based on the separate priorities of each stakeholder, the PST decides on a combined list including an aggregation of the individual priorities.

Step 4. The decision is communicated to all stakeholders.

Step 5. The stakeholders give feedback on the impact of the decision to the PST. If the PST finds it necessary, another iteration may be started. Otherwise, the process results in a decision on high-level requirements priorities for the next release. This list is then used as a guidance when making trade-offs between value and cost of more detailed requirements (see Regnell, Beremark, & Eklund, 1998) for further description on the use of must- and wish-lists).

When the PST aggregates the priorities of the individual stakeholders into a combined decision, there are a number of criteria for adjusting the influence of each stakeholder. This aggregation can be carried out by weighting the market segments based on one or several *weighting criteria*, for example:

- revenue last release
- profit last release
- number of sold licenses last release
- predictions of the above criteria for the coming release
- number of contracts lost to competitors
- number of potential customers with nil licenses to date
- size of total market segment.

Which weighting criteria to use depend on which general strategy that the top-level management perceives best fit with the current market phase (Moore, 1991).

3. Case study planning and operation

During the summer of 1999, the case study was planned with the objective of obtaining feedback on the distributed prioritization process shown in Figure 2. The case study was conducted during the fall of 1999, and comprised one iteration of the process.

First, the PST decided on a list of requirements. The requirements were represented on two levels of abstraction. The lower level consisted of features, which in turn were grouped into high-level feature groups. This grouping was natural, according to the product manager, as there were related feature requirements which easily formed coherent groups. In this study there were 17 feature groups (denoted A-Q in the analysis). The feature groups A-P each consist of up to 10 closely related features as shown in Table 1. Feature group Q consists of remaining features with no natural relation to other features. Feature groups B and G contained no specified features, and were only prioritized on the group level. In total there were 58 features. The total number of objects given to stakeholders for prioritization was thus 17+58=75, as the stakeholders were asked to prioritize both the feature groups and the features.

Each feature group and each feature were represented by a short but informative name together with a natural language description comprising a couple of sentences. Examples of requirements at feature group level were ‘External tool integration’ and ‘Support for deployment and partitioning on target level’. The former feature group included

Table 1. Number of features per feature group.

Feature group	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	**Total**
Number of features	10	0	4	2	3	4	0	2	2	3	2	6	4	7	2	4	3	58

features such as 'Integrate SDL Suite with DOORS' and 'Improved integration with Clearcase'[2].

All requirements were described in a spreadsheet and sent out via e-mail to the 10 stakeholders. Each stakeholder made two prioritizations: one for the feature groups and one for the individual features. Each prioritization was carried out through the distribution of a pre-defined amount of fictitious money ($100,000) over the items to be prioritized. During the prioritization, the stakeholders were given the possibility to add new items, both feature groups (denoted group Z in the analysis) and features.

Each stakeholder was represented by a number of persons, ranging from one to six. Each group of representatives delivered one set of priorities for the 17+1 feature groups A-Z and one set of priorities for the 58 (plus added) features. These priorities represent the stakeholder representatives' agreed upon views on the importance of the requirements.

After the individual stakeholder prioritization, a questionnaire was sent out to each stakeholder, with the objective of receiving feedback on the distributed prioritization process. The results from the analysis of the questionnaire answers are summarized in Section 4. The outcome of the prioritization is presented in Section 5, together with a number of charts for visualizing the data from distributed prioritization.

4. Results from questionnaires

We received responses from 8 participants (1 response each from 6 stakeholders and 2 responses from 1 stakeholder). The questions and answers can be found in Table 2. Some stakeholders gave additional information to questions number 10, 14, and 16. This additional

2. The requirements in this study are strictly confidential, but the few carefully selected examples above were allowed to be revealed in order to give a flavour of their abstraction level, as long as they cannot be explicitly related to the anonymous data given in the paper.

information, together with an interpretation of the results, is presented in the following.

The first questions were aimed at relating the time spent on the prioritization to the other answers. Unfortunately, there seem to have been different interpretations of the questions, which makes the answers to questions 2 and 4 unreliable. As a result we can not draw any conclusions from questions 3 and 5 either.

As indicated by the answers to question 15 and question 17, the stakeholders seem to have good confidence in the usefulness of distributed prioritization of requirements using the selected method. One quarter of the stakeholders thought the distributed prioritization was very useful. The rest of the stakeholders though it was rather useful.

According to the answers to questions 6 and 7, the stakeholders also thought that it was easy or very easy to understand what to do and not difficult at all to carry out the prioritization. The answers to question 12 possibly indicate that there were some difficulties interpreting the requirements description, although most stakeholders did not come up against any problems. Also, the answers to question 13 show that the stakeholders were not very confident that the other stakeholders would have the same interpretation of the requirements. The quality of requirements descriptions may influence the participants' opinion about the method. The requirements descriptions can, however, be improved regardless of the method used and possibly improve confidence in the interpretation.

The answers to question 8 also support the usefulness and simplicity of the method. The stakeholders found that the result closely reflected their "gut feeling" of what was important. This positive outcome naturally influences the opinion of the method but it is also important in order to avoid further iteration in the prioritization process.

All the stakeholders used the same order of prioritization, starting with the high-level requirements and then proceeding with the low-level requirements. This is not very surprising, as the method description indirectly suggested this order. The majority of the stakeholders also used the same strategy when distributing the money and started with the requirements they found most important. Only one stakeholder started from the top of the list. The rest of the stakeholders used variants of the "most important" strategy. Stakeholder number 8 emphasized that they started with the requirements that were important for the whole company, not only for them. Another stakeholder used a simple algorithm

Table 2. The results of the questionnaire used to evaluate the prioritization method.

	Question	Answer format	1	2	3	4	5	6	7	8
1	Number of people involved in prioritization	Count	1	1	4	6	1	5	2	6
2	Time spent on high-level requirements prioritization	Person minutes	10	60	110	300	30	60	180	180
3	Was the time spent enough?	1–More than enough 2–Enough 3–Too short	2	2	2	3	3	2	2	2
4	Time spent on low-level requirements prioritization	Person minutes	20	60	200	30	30	20	80	540
5	Was the time spent enough?	1–More than enough 2–Enough 3–Too short	2	2	3	3	3	-	2	2
6	How easy was it to understand what to do based on the method description?	1–Very Easy 2–Easy 3–Neither easy nor difficult 4–Difficult 5–Very difficult	2	1	2	1	2	1	2	2
7	How easy was it to carry out the prioritization of requirements?	1–Very Easy 2–Easy 3–Neither easy nor difficult 4–Difficult 5–Very difficult	3	2	3	3	2	2	2	2
8	When finished with prioritization, did invested money reflect the “gut feeling” of what was important?	1–The result was exactly correct 2–The result was almost correct 3–The result was not very correct	2	2	2	2	1	2	2	-

* Additional information on marked numbers can be found in the text.

- The dash indicates that no answer was given.

Table 2. (contd.) The results of the questionnaire used to evaluate the prioritization method.

	Question	Answer format	1	2	3	4	5	6	7	8
9	Order of prioritization	1–High-level, then low-level 2–Low-level, then high-level 3–Iterated between levels	1	1	1	1	1	1	1	1
10	Strategy when distributing the money	1–Started with the important 2–Started from top of list 3–Other	1	3*	2	1	1	1	1	3*
11	Best way to distribute the money?	1–Consider all low-level requirements as a whole and distribute one sum of money on all of them 2–Have one sum of money for each group, so that we can consider them separately 3–Equally good to distribute money on all or per group 4–Don't know	1	4	1	4	2	2	2	2
12	How easy was it to interpret the requirements descriptions?	1–Very Easy 2–Easy 3–Neither easy nor difficult 4–Difficult 5–Very difficult	2	2	3	2	2	2	4	2
13	Do you think you have the same interpretation of requirements as others?	1–Exactly 2–Almost exactly 3–Partly the same 4–Not very similar 5–Completely different	2	4	3	2	3	2	3	2

* Additional information on marked numbers can be found in the text.

- The dash indicates that no answer was given.

Table 2. (contd.) The results of the questionnaire used to evaluate the prioritization method.

	Question	Answer format	1	2	3	4	5	6	7	8
14	Most influential criterion/criteria when deciding what was important	1–Increasing sales 2–Increasing profit 3–Finding new customers 4–Reducing support efforts 5–Beating competitors 6–Other	1 3 4	5 6* 3	4 6*	1	1	1 5 4	3 1 6*	1 2 4
15	Usefulness of the method with monopoly money	1–Very useful 2–Rather useful 3–Not useful 4–Waste of time	1	2	2	2	1	1	2	2
16	Is the priority information you provide enough, or should you have had opportunity to provide more?	1–The method collects necessary information from us 2–It is not enough. We want to provide additional information	2*	1	2*	1	1	1	2*	2*
17	Usefulness of prioritization in general, by collecting information from different sources	1–Very useful 2–Rather useful 3–Not useful 4–Waste of time	1	2	2	2	1	2	2	2
18	Agreement with the resulting list of priorities made by the product manager	1–Agreed completely 2–Agreed to most of it 3–Agreed to some of it 4–Disagreed in most cases 5–Did not agree at all	1	2	2	3	2	2	3	3

* Additional information on marked numbers can be found in the text.

– The dash indicates that no answer was given.

to distribute the money and started investing *$N* on the most important requirements. Then, the stakeholder invested *$N/2* on the next important ones, *$N/4* on the next important ones after that, and so on until all wanted investments were made. *N* was then adjusted so that the sum became $100,000.

The answers to question 14 reveal the most influential criteria used by the stakeholders when deciding what was important. We can see that increasing sales was a highly influential criterion for most of the stakeholders. Other important criteria were to find new customers, to reduce support effort, and to beat competitors. Stakeholder 2 provided other criteria and stated that using company skills to the best and technology push was rather important. Stakeholder 3 thought it was important to create complete functions and a truly usable product. Stakeholder 7 wanted to satisfy a key customer.

In this study, the stakeholders were asked to distribute a total of $100,000 on all requirements across groups. As question 11 suggests, another way would be to give $100,000 for each group of low-level requirements and distribute them only within the group. This second suggested way to distribute the money was supported by half of the stakeholders, although they had not tried it. Only two, as the answers would indicate, were convinced that the used distribution method was the best. The rest had difficulties to decide which was the best approach.

The priority information that the method collected was enough for half of the stakeholders. The other half wanted to provide additional priority information and brought up several interesting and important aspects. Stakeholder 1 though it was necessary to discuss the impact of new features: if they open new markets, increase sales to existing customers, and keep existing customers pleased. Stakeholder 3 found that "after the collection of requirements it was clear that some top requirements were not specified in detail and therefore rejected or questioned". Stakeholder 3 also though that a requirement capturing process is needed for requirements that have not been addressed in the past.

Stakeholder 7 was concerned about that "even if much money was spent on new requirements [...] it is not known if they will be part of the product". The stakeholders, which provided new requirements, is probably the only ones prioritizing them, and thus it is not very likely that new requirements would get a high priority in the resulting list of

priorities. Therefore the added requirements were placed in separate groups (see further Section 5).

Stakeholder 8 thought that the prioritization procedure itself needed a prioritization, since "all people involved in the requirement definition have too much of the view from the sales. The people how work with the tool in real and big projects are not asked".

Although the answers to question 12 and 13 indicate some difficulties with the requirement interpretations, the stakeholders' response to question 18 shows that they largely agreed with the resulting list of priorities made by the product manager.

5. Visualization of prioritization data

The raw data provided by each stakeholder is given in Appendix A, where Table 4 shows the data for the prioritization of the high-level feature groups, while Table 5 shows the prioritization of each individual feature. The stakeholders were, as explained in Section 3, given the opportunity to add new feature groups and new features to the lists. Feature group Z in Table 4 and features numbered 99 in Table 5 represent added feature groups and added features respectively.

The resulting priorities show that there are large differences in stakeholder opinions on which requirements are most important. It seems also to be the case, that the stakeholders had different strategies for how they distributed their priorities. Some stakeholders have put all their money on a few requirements, while others have distributed their money more evenly on a larger number of requirements. Stakeholder M2, M5, M6, M7, and M10 added their own new groups of features (requirement Z) and gave them a certain priority. Hence, it should be noted that the priority of Z is special, since the stakeholders voted on different requirements.

In order to compare the priorities of each stakeholder the data is preferably normalized by dividing each data point by the total of each column, resulting in that the sum of priorities for each stakeholder equals 1. The normalized data from the prioritization of the feature groups is shown in Table 3. Each cell shows the normalized priority, subsequently denoted p_{ij} for requirement i and stakeholder j, where there are n=18 requirements and m=10 stakeholders.

Table 3. Normalized priorities for each stakeholder and high-level requirement.

Req.	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	Sum
A	0...000	0.000	0.000	0.200	0.000	0.073	0.000	0.000	0.300	0.108	0.681
B	0.200	0.000	0.050	0.010	0.000	0.073	0.000	0.000	0.150	0.054	0.537
C	0.200	0.100	0.080	0.050	0.000	0.080	0.000	0.000	0.000	0.216	0.726
D	0.050	0.000	0.200	0.030	0.000	0.150	0.070	0.000	0.000	0.054	0.554
E	0.050	0.300	0.070	0.000	0.000	0.000	0.070	0.000	0.000	0.054	0.544
F	0.000	0.000	0.000	0.010	0.000	0.025	0.000	0.000	0.000	0.054	0.089
G	0.050	0.000	0.000	0.010	0.000	0.000	0.000	0.000	0.000	0.027	0.087
H	0.050	0.200	0.000	0.050	0.000	0.040	0.030	0.200	0.150	0.108	0.828
I	0.000	0.000	0.000	0.200	0.000	0.010	0.000	0.000	0.000	0.000	0.210
J	0.000	0.000	0.000	0.150	0.200	0.020	0.070	0.000	0.000	0.027	0.467
K	0.000	0.000	0.150	0.050	0.000	0.125	0.070	0.200	0.100	0.054	0.749
L	0.000	0.000	0.150	0.050	0.000	0.070	0.070	0.200	0.000	0.054	0.594
M	0.200	0.200	0.150	0.000	0.000	0.040	0.030	0.000	0.200	0.054	0.874
N	0.000	0.000	0.000	0.000	0.000	0.010	0.000	0.200	0.000	0.000	0.210
O	0.000	0.000	0.000	0.020	0.000	0.025	0.030	0.200	0.000	0.027	0.302
P	0.200	0.000	0.150	0.170	0.300	0.230	0.140	0.000	0.100	0.027	1.317
Q	0.000	0.100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.100
Z	0.000	0.100	0.000	0.000	0.500	0.030	0.420	0.000	0.000	0.081	1.131
Total:	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	10.000

When the data in Table 1 was received from the stakeholders it became obvious that its interpretation would be easier if it was visualized using some type of diagrams. The following diagrams were developed by the researchers in order to support the visualisation and interpretation of the data:

- *Distribution Chart* for showing how the different stakeholders voted and the resulting priority ranking.
- *Disagreement Chart* for showing the size of variation between stakeholder priorities for each requirement.
- *Satisfaction Chart* for showing how well the resulting priority ranking corresponds to the priorities of each individual stakeholder.
- *Influence Chart* for showing how much influence each market is given on the resulting priority.

These diagrams are explained in the following, and exemplified with real data from the industrial case study.[3]

The resulting priority denoted p_i for each requirement i is calculated as the weighted average of the priorities over stakeholders using a weight denoted w_j where the sum of all w_j equals 1. The weights w_j are used to adjust the influence on the resulting priorities for each stakeholder. If all stakeholders have equal influence, then all $w_j = 1/m$, and the unweighted resulting priorities correspond to the ordinary average.

Figure 3 shows an unweighted Distribution Chart, where the priority distribution is visualized using a stacked Pareto bar chart. The requirements are sorted based on the resulting priorities, starting with the requirement with the highest total priority. Each stakeholder is distinguished with a specific colour. We can see that many stakeholders have given high priority to requirement P, whereas e.g. requirement N, although important for stakeholder M8, is given a low total priority.

As a disagreement measure denoted d_i for each requirement i, the variation coefficient was used. The variation coefficient is calculated for each requirement as the standard deviation of the weighted priorities divided by the average of the priorities over the m stakeholders. The

3. Unfortunately, the timing of the decision-making in the PST hindered the use of these diagrams in the actual release plan decision meetings, but they were examined by the PST afterwards. Hence, the PST did not build their decision on the diagrams, although the PST felt that the charts would have been valuable as an integral part of the prioritization process.

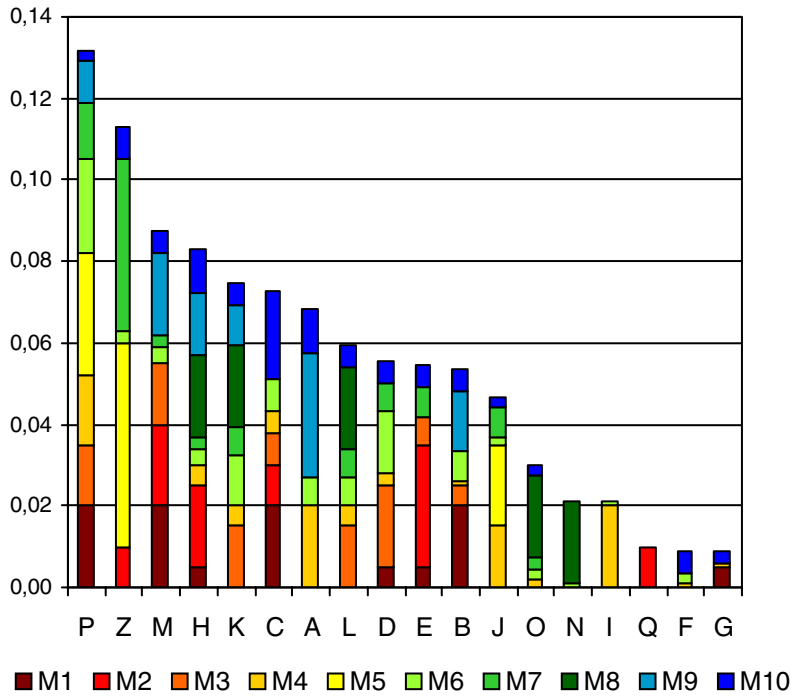

Figure 3. *Priority Distribution Chart for equal market influence.*

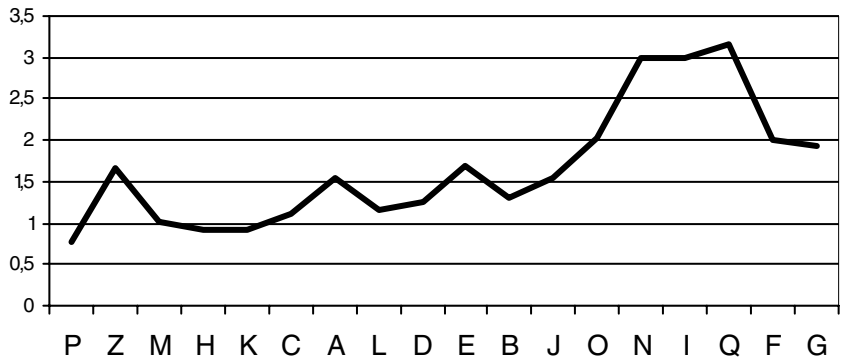

Figure 4. *Disagreement Chart visualizing the dispersion of priorities among stakeholders.*

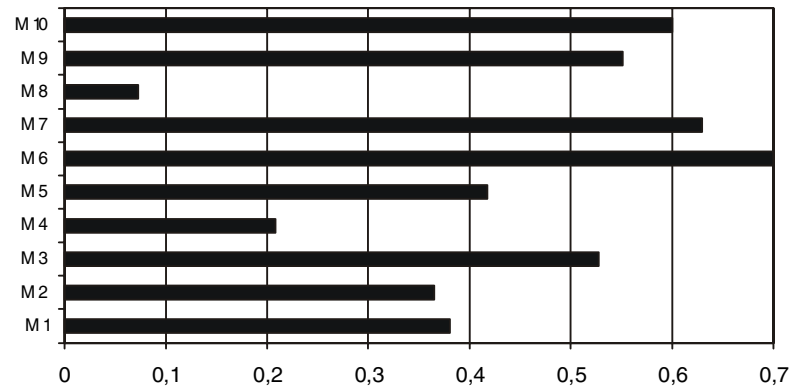

Figure 5. *Satisfaction Chart visualizing the correlation of each stakeholder's priorities with the resulting priorities.*

variation coefficient is a commonly used statistical measure of dispersion, and it can be argued that large disagreement means precisely that there is a large variation between the priorities of the stakeholders.

Figure 4 shows an unweighted (i.e. all w_j equals $1/m$) Disagreement Chart that displays the disagreement measure d_i for each requirement. The chart makes it clear that there is more consensus on e.g. the priority of P compared to N. We also see that, although C and A have almost the same total priority in the Distribution Chart, there is more disagreement over A than over C.

Another result from the case study is the visualization of stakeholder satisfaction with the resulting priority order decided by the PST. This is accomplished by correlating the ranks of the total priorities with each stakeholder's original priority ranks using the Spearman rank-order correlation coefficient (Siegel & Castellan, 1988), denoted r_j for each stakeholder j. The use of the Spearman rank-order correlation coefficient is motivated by the intuitive notion of satisfaction as how well the resulting priority order corresponds to the original priority order of each stakeholder. The Satisfaction Chart, shown in Figure 5, reveals that stakeholder M6 is most satisfied and that stakeholder M8 is least satisfied with the resulting unweighted priorities.

In the case study, the different stakeholders were prioritized by the Product Manager according to a subset of the criteria mentioned in Section 5. This revealed, e.g., that two stakeholders representing special markets were considered more important than the others (M1 and M5) and that four stakeholders were to be given low priority (M6, M10, M8,

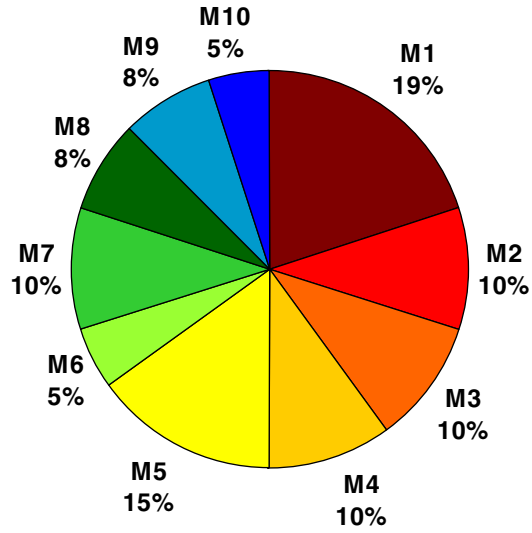

Figure 6. *Influence Chart visualizing the weighting of stakeholder influence.*

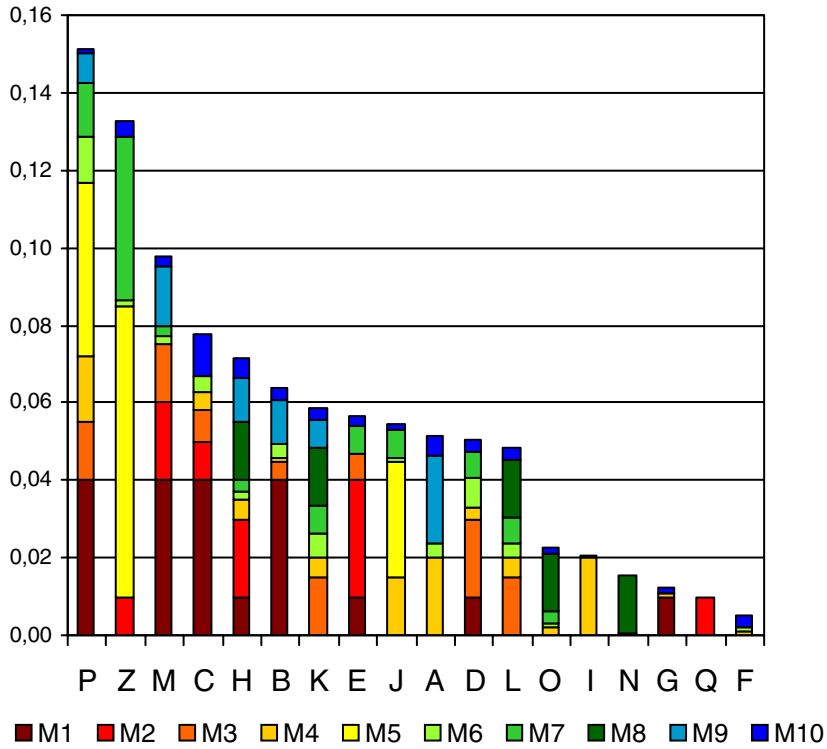

Figure 7. *Priority Distribution Chart for weighted market influence.*

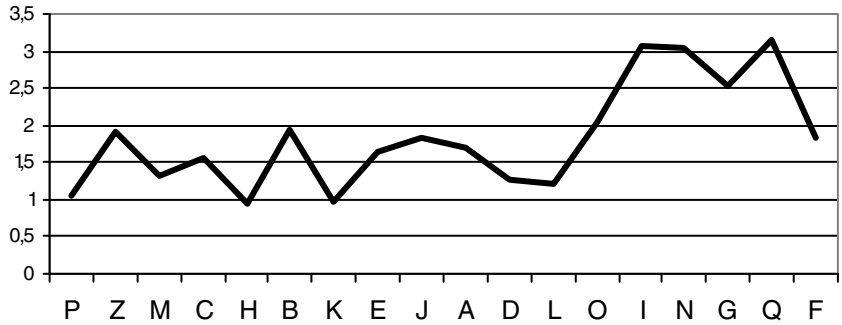

Figure 8. *Disagreement Chart with weighted market influence.*

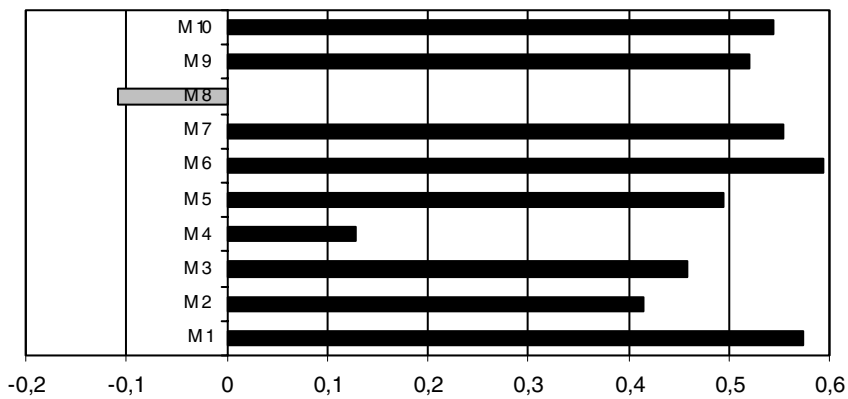

Figure 9. *Satisfaction Chart visualizing the correlation of each stakeholder's priorities with the total priorities with weighted market influence.*

and M9). Figure 6 shows an Influence Chart, which displays the decision by the Product Manager on the market influence, in terms of the weights w_j for each stakeholder. Given these weights, recalculation of p_i, d_i, and r_j are made, and the charts are redrawn.

Figure 7 displays the Distribution Chart after weighting by market influence using the weights of Figure 6. It can be seen that the priorities of M1 and M5 are boosted. This in turn have altered the priority order, making, for example, C and B to move up, and A and L to move down.

The Disagreement Chart in Figure 8 reveals an increase in disagreement for requirement C and B, while the disagreement over

requirement A and L is only slightly affected. In the Satisfaction Chart in Figure 9 we can see that stakeholder M8 is least satisfied with the weighted priority order. The negative value on the satisfaction means that the resulting priority order have a tendency to be the opposite of the individual priority order, meaning that high priorities of M8 are given low priorities in the result and vice versa. Also, both stakeholder M1 and M5 are more satisfied compared to the unweighted case.

6. Conclusions and further work

The presented work is aimed at making similarities and differences in priorities among market segments explicit. This is valuable as a decision support in market-driven software development, when strategic high-level requirements are to be selected for the coming release of a software package. The presented charts for visualization of requirements priority distribution, disagreement, and satisfaction have been developed based on data from an industrial case study. The main objective of the case study was to study how distributed prioritization can be used in the connection between every day requirements management and a high-level product strategy. The case study included real requirements and real stakeholders.

The presented case study was evaluated qualitatively through questionnaires answered by stakeholders and by interviews with product and quality management. The general conclusions from the case study are (1) that distributed prioritization is useful, and (2) that the visualization charts are valuable decision support.

When comparing the sum of priorities from all stakeholders to the management's original view of the priorities from before the case study, the management was able to identify three interesting groups of requirements:

- *Confirmed Priorities.* Some requirements were prioritized in a way that corresponded to the views before the case study, and thus confirmed the strategic value of these requirements. This information was valuable, as it strengthened the PST in the belief that many of the requirements previously in focus were still right.

- *Elevated Priorities.* Some requirements were given surprisingly high priorities, compared to what was anticipated. This information was valuable, as it gave the opportunity to adjust the strategies to focus on new opportunities which might otherwise have been missed.
- *Cancelled Priorities.* Some requirements were given surprisingly low priorities. This information was valuable, as it helped to avoid implementing requirements that may be of little success on the market.

Based on the analysis and visualisation of the data from the case study, the following major findings were made:

- *Differences among stakeholders.* The colour-coded Priority Distribution Chart shows that there were large differences among the stakeholders regarding requirements priorities. The Disagreement Chart shows that for some of the requirements the stakeholders agreed more on their priorities compared to other requirements. The Satisfaction Chart shows that there were large differences in the stakeholders' satisfaction with the resulting total.
- *Impact of influence weights.* When some stakeholders were given a higher influence than others on the resulting total, the Satisfaction Chart shows that some stakeholder's priorities were very far from the resulting total, while others were relatively close.
- *Stakeholder tactics.* The stakeholders had different tactics when assigning absolute priority values to the requirements. Some stakeholders concentrated on a few requirements and gave them high priorities while other requirements were given zero. Some stakeholders made a more even distribution of their priorities.

The main challenges with the proposed approach include the following:

- *Difficulties with absolute assessment.* The individual prioritization was carried out by placing an absolute amount of fictitious money on each requirement in accordance to its perceived importance. The assessment of priorities through assignment of absolute numerical values is inherently difficult, and there are strong indications that relative judgements through pairwise comparison rather than assigning absolute values is easier for humans and also more accurate (Karlsson & Ryan, 1997).

- *Assessment of prioritization quality.* The quality of the priorities given by stakeholders is a major issue. The stakeholders are representatives from marketing and may not know how a particular requirement should be interpreted or how important it is for the potential customers of their market. Furthermore, all stakeholders gave the same priority to many requirements, especially the ones with low priorities. This makes it more difficult to discriminate requirements. If pairwise comparison (Karlsson & Ryan, 1997) is introduced, the stakeholders are forced to take a stand for every requirement. When using pairwise comparison it is possible to calculate a consistency index (ibid.), which may indicate if requirements are interpreted inconsistently, e.g. due to poor knowledge or ambiguous descriptions.

- *Sensitivity to 'shrewd tactics'.* It is difficult to know if some stakeholders, e.g., gave an extra low priority to requirements they knew other stakeholders would give high priorities, just in order to influence the total result to fit their aims. Such tactics would compromise the quality of the result in that it would not truly correspond to the importance to the different market segments represented by the stakeholders. It is difficult to avoid this type of obstruction in general. However, if pairwise comparison (Karlsson & Ryan, 1997) is introduced, obstructive tactics may be more difficult to carry through in a consistent way.

There are several candidate techniques involving relative judgement through pairwise comparison (Karlsson, Wohlin, & Regnell, 1998), and one of the most promising technique is the Analytical Hierarchy Process (AHP) (Saaty, 1980). This technique is, however, not in its original form adapted to distributed prioritization with multiple stakeholders. Furthermore, this technique may require specialized tools for the matrix calculations required. An interesting question for further research is how to incorporate relative judgement through pairwise comparison in the presented distributed prioritization process, including the usage of an Internet-based tool for gathering pairwise comparison data in a global organization. It is also interesting to evaluate the utility of the charts for visualization of distribution, disagreement and satisfaction, when applied to prioritization data from pairwise comparison.

Other work in the area have focused on arriving at a consensus between stakeholders. Boehm and Ross (1989) proposes Theory-W,

which is concerned with how to negotiate between many stakeholders with conflicting win-conditions. Kotonya and Sommerville (1996) proposes a Viewpoint-oriented approach with explicit planning for conflict resolution. The approach presented in this paper is not focused on creating consensus among stakeholders, but rather on revealing the differences between stakeholder priorities in order to make an informed decision about which stakeholders to prioritize in relation to foreseen market opportunities. The business strategy governing the selection of requirements to implement in the next release of a packaged software product may very well result in that some stakeholders representing a market segment with a lower priority will not get what is considered important for those particular stakeholders.

The process proposed in Section 2 is general and Step 2 and 3 can be carried out using many different methods. Based on the above conclusions, one of the most promising methods to investigate in further studies is pairwise comparison. Another industrial case study, similar to the one presented here, but with pairwise comparison instead of absolute priority assignment, would give the interesting opportunity of comparing methods and further improve distributed requirements prioritization in global software development organizations. It would also be interesting to investigate if the stakeholders whose ratings differ significantly from the final decision are more inclined to accept this decision due to the increased transparency of the prioritization process.

Acknowledgements

The authors would like to thank everyone at Telelogic who have participated in the case study. This work is partly funded by the National Board of Industrial and Technical Development (NUTEK), Sweden, within the REMARKS project (Requirements Engineering for Market-Driven Software Development) grant no. 1K1P-99-06123. This paper is an extended version of a paper presented at the REFSQ workshop (Regnell, Höst, Natt och Dag, Beremark, & Hjelm, 2000). The authors would like to thank the workshop participants for fruitful discussions, and the REFSQ organizers and the anonymous reviewers for valuable improvement suggestions.

7. References

Boehm, B. W. &, Ross, R. (1989, July). Theory-W Software Project Management: Principles and Examples. *IEEE Transactions on Software Engineering, 15*, 902–916.

Büyükekici, B., Deifel, B., Jacobi, C., & Sandner, R. (1999). Prioritization of Complex COTS. In A. L. Opdahl, K. Pohl, & E. Dubois (Eds.) *Proceedings of the Fifth International Workshop on Requirements Engineering: Foundations for Software Quality* (pp. 161–168). Namur, Belgium: Presses Universitaires De Namur.

Carmel, E., & Becker, S. (1995, February). A Process Model for Packaged Software Development. *IEEE Transactions on Engineering Management, 42*, 50–61.

Deifel, B. (1999). A Process Model for Requirements Engineering of CCOTS. In *Proceedings of Tenth International Workshop on Database and Expert Systems Applications* (pp. 316–320). Los Alamitos, CA: IEEE Computer Society Press.

Kamsties, E., Hörmann, K., & Schlich, M. (1998). Requirements Engineering in Small and Medium Enterprises. *Requirements Engineering, 3*, 84–90.

Karlsson, J. & Ryan, K. (1997). A Cost-Value Approach for Prioritizing Requirements. *IEEE Software, 14*(5). 67–74.

Karlsson, J., Wohlin, C., & Regnell, B. (1998, February). An Evaluation of Methods for Prioritizing Software Requirements, *Information and Software Technology, 39*, 939–947.

Kotonya, G., & Sommerville, I. (1996, January). Requirements Engineering with Viewpoints. *Software Engineering Journal, 11*, 5–18.

Lubars, M., Potts, C., & Richter, C. (1993). A Review of the State of the Practice in Requirements Modeling. In *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 2–14). Los Alamitos, CA: IEEE Computer Society Press.

Moore, G. (1991). *Crossing the Chasm*. New York: Harper-Collins.

Potts, C. (1995). Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. In *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 128–130). Los Alamitos, CA: IEEE Computer Society Press.

Regnell, B., Beremark, P., & Eklundh, O. (1998). A Market-Driven Requirements Engineering Process – Results from an Industrial Process Improvement Programme. *Requirements Engineering, 3*, 121–129.

Regnell, B., Höst, M., Natt och Dag, J., Beremark, P. & Hjelm, T. (2000). Visualization of Agreement and Satisfaction in Distributed Prioritization of Market Requirements, In A. L. Opdahl, K. Pohl, & M. Rossi (Eds.), *Proceedings of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality.* Essen, Germany: Essener Informatik Beiträge.

Saaty, T. L. (1980). *The Analytical Hierarchy Process*. New York: McGraw Hill.

Sawyer, P. (2000). Packaged Software: Challenges for RE. In *Proceedings of Sixth International Workshop on Requirements Engineering: Foundation for Software Quality* (pp. 137–142). Essen, Germany: Essener Informatik Beiträge.

Sawyer, P., Sommerville, I., & Kotonya, G. (1999). Improving Market-Driven RE Processes. In M. Oivo, & P. Kuvaja (Eds.), *Proceedings of the International Conference on Product Focused Software Process Improvement* (pp. 222–236). Oulu, Finland: Technical Research Centre of Finlad (VTT).

Siegel, S., & Castellan, N. J (1988). *Nonparametric Statistics for the Behavioral Sciences* (2nd Ed.). New York: McGraw-Hill.

Yeh, A. (1992). Requirements Engineering Support Technique (REQUEST) – A Market Driven Requirements Management Process, In *Proceedings of the Second Symposium on Assessment of Quality Software Development Tools* (pp. 211–223), Los Alamitos, CA: IEEE Computer Society Press.

Appendix A: Raw data from prioritization

Table 4. Raw data from prioritization of high-level feature groups.

Req.	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
A	0	0	0	20,000	0	7,250	0	0	30	10,000
B	20,000	0	5,000	1,000	0	7,250	0	0	15	5,000
C	20,000	10,000	8,000	5,000	0	8,000	0	0	0	20,000
D	5,000	0	20,000	3,000	0	15,000	7,000	0	0	5,000
E	5,000	30,000	7,000	0	0	0	7,000	0	0	5,000
F	0	0	0	1,000	0	2,500	0	0	0	5,000
G	5,000	0	0	1,000	0	0	0	0	0	2,500
H	5,000	20,000	0	5,000	0	4,000	3,000	20	15	10,000
I	0	0	0	20,000	0	1,000	0	0	0	0
J	0	0	0	15,000	20,000	2,000	7,000	0	0	2,500
K	0	0	15,000	5,000	0	12,500	7,000	20	10	5,000
L	0	0	15,000	5,000	0	7,000	7,000	20	0	5,000
M	20,000	20,000	15,000	0	0	4,000	3,000	0	20	5,000
N	0	0	0	0	0	1,000	0	20	0	0
O	0	0	0	2,000	0	2,500	3,000	20	0	2,500
P	20,000	0	15,000	17,000	30,000	23,000	14,000	0	10	2,500
Q	0	10,000	0	0	0	0	0	0	0	0
Z	0	10,000	0	0	50,000	3,000	42,000	0	0	7,500
Total	100,000	100,000	100,000	100,000	100,000	100,000	100,000	100	100	92,500

Table 5. Raw data from prioritization of individual features.

Req.	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
A1	0	10,,000	3,000	3,000	0	800	0	200	0	1,250
A2	0	10,000	0	25,000	0	800	0	400	0	7,500
A3	0	0	0	20,000	0	400	0	0	0	1,250
A4	5,000	10,000	0	2,000	0	400	0	400	20	2,500
A5	0	0	0	1,000	0	0	0	0	0	0
A6	0	0	0	10,000	0	0	0	0	0	0
A7	0	10,000	0	14,000	0	1,400	1,200	200	0	2,500
A8	5,000	0	0	14,000	0	400	0	200	0	0
A9	0	0	4,000	11,000	0	400	0	0	0	2,500

Table 5. Raw data from prioritization of individual features.

Req.	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
A10	0	0	0	0	0	1,400	0	100	0	2,500
A99	54,000	5,000	0	0	0	0	0	0	0	5,000
C1	10,000	0	6,000	0	0	1,800	0	0	10	10,000
C2	0	0	1,000	0	0	4,200	0	400	0	10,000
C3	0	0	4,000	0	0	400	0	500	0	0
C4	0	0	0	0	0	0	8,400	300	0	0
C99	0	0	16,000	0	0	0	0	0	0	12,500
D1	0	0	0	0	0	6,600	0	0	0	0
D2	5,000	0	8,000	0	0	5,400	0	400	20	2,500
D99	0	0	0	0	0	10,800	18,000	0	0	0
E1	2,000	0	0	0	0	0	0	0	0	0
E2	2,000	0	0	0	0	800	0	0	0	0
E3	2,000	10,000	3,000	0	0	1,800	0	300	0	0
E99	0	0	0	0	0	0	8,400	0	0	0
F1	0	0	1,000	0	0	0	0	200	0	2,500
F2	3,000	0	0	0	0	0	0	200	0	0
F3	0	0	1,000	0	0	0	0	0	0	1,250
F4	0	0	0	0	0	0	1,200	0	0	0
F99	0	0	4,000	0	20,000	2,600	1,200	0	0	0
H1	2,000	0	5,000	0	0	800	0	400	10	2,500
H2	0	0	3,000	0	0	400	0	400	0	2,500
H99	0	0	0	0	0	0	0	0	0	0
I1	0	0	0	0	0	0	0	0	0	0
I2	0	0	0	0	0	0	0	0	0	0
I99	0	0	0	0	0	0	0	0	0	0
J1	0	10,000	0	0	0	2,000	0	200	0	0
J2	2,000	0	0	0	0	2,800	0	200	0	5,000
J3	0	0	0	0	0	0	0	400	0	0
J99	0	0	0	0	30,000	0	19,200	0	0	0
K1	0	0	0	0	0	800	1,200	300	20	2,500
K2	0	0	0	0	0	1,400	1,200	300	0	2,500
K99	0	0	0	0	0	9,000	0	0	0	0
L1	4,000	0	3,000	0	5,000	1,800	1,200	300	0	5,000
L2	0	0	3,000	0	5,000	1,000	1,200	300	0	0
L3	0	0	0	0	0	0	0	300	0	2,500
L4	0	0	0	0	0	0	0	0	0	1,250
L5	0	0	5,000	0	0	1,800	0	200	0	0

Table 5. Raw data from prioritization of individual features.

Req.	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
L6	0	5,000	0	0	0	0	0	100	0	1,250
L99	0	0	0	0	0	0	0	0	0	0
M1	0	0	0	0	0	0	0	50	10	0
M2	0	0	0	0	0	0	0	0	0	0
M3	0	5,000	2,000	0	0	400	0	100	0	0
M4	0	5,000	0	0	0	2,800	0	0	10	1,250
M99	0	5,000	8,000	0	5,000	600	0	0	0	0
N1	4,000	0	0	0	0	800	0	0	0	2,500
N2	0	0	2,000	0	0	0	0	200	0	0
N3	0	0	0	0	0	800	0	200	0	1,250
N4	0	0	0	0	0	1,400	0	300	0	0
N5	0	0	3,000	0	0	0	0	200	0	0
N6	0	0	0	0	0	0	4,000	0	0	0
N7	0	0	0	0	0	0	0	0	0	0
N99	0	0	5,000	0	0	0	18,000	0	0	0
O1	0	0	0	0	5,000	0	0	500	0	0
O2	0	5,000	0	0	0	1,000	0	0	0	0
O99	0	0	0	0	0	1,000	1,200	0	0	1,250
P1	0	0	0	0	0	6,000	4,000	300	0	0
P2	0	0	8,000	0	10,000	12,000	4,000	200	0	2,500
P3	0	0	0	0	10,000	2,400	4,000	50	0	0
P4	0	0	0	0	10,000	1,800	0	400	0	0
P99	0	0	0	0	0	6,000	2,400	0	0	0
Q1	0	10,000	2,000	0	0	400	0	300	0	0
Q2	0	0	0	0	0	0	0	0	0	0
Q3	0	0	0	0	0	400	0	0	0	2,500
Q99	0	0	0	0	0	0	0	0	0	0
Total	100,000	100,000	100,000	100,000	100,000	100,000	100,000	10,000	100	100,000

PAPER IV

A feasibility study of automated natural language requirements analysis in market-driven development

Johan Natt och Dag, Björn Regnell, Pär Carlshamre, Michael Andersson, Joachim Karlsson

Requirements Engineering, 7, 20–33, 2002.

IV

Abstract

In market-driven software development there is a strong need for support to handle congestion in the requirements engineering process, which may occur as the demand for short time-to-market is combined with a rapid arrival of new requirements from many different sources. Automated analysis of the continuous flow of incoming requirements provides an opportunity to increase the efficiency of the requirements engineering process. This paper presents empirical evaluations of the benefit of automated similarity analysis of textual requirements, where existing information retrieval techniques are used to statistically measure requirements similarity. The results show that automated analysis of similarity among textual requirements is a promising technique that may provide effective support in identifying relationships between requirements.

1. Introduction

1.1 Background

The market-driven development organisation faces many challenges that differ from those found in organisations developing bespoke software. Software is developed for a large market, rather than for a specific customer, new versions are developed in a succession of releases, and there is a high pressure on short time-to-market (Sawyer, Sommerville, & Kotonya, 1999; Lubars, Potts, & Richer, 1993; Deifel, 1999).

To meet market demands it is important to have an effective and efficient requirements engineering process. Special demands are set as requirements arrive continuously at a high rate from many different sources during the whole development process (Regnell, Beremark, & Eklundh, 1998). As there is no single specific customer to negotiate with, requirements must be invented within the developing organisation based on foreseen end user needs (Potts, 1995). These invented requirements may come from sources such as marketing, support, development, testing, usability evaluations and technology forecasting, and are often collected for storage in a database. The requirements engineering activities are then focused on analysing and prioritizing the requirements in the database and on maintaining the database for the future. In this study we have focused on a large software developing company, Telelogic AB, that develops a CASE tool for the worldwide telecommunications market. Their development process is described in Regnell et al., and its main properties are: 1. Releases are pipe-lined to enable a new release every sixth month while each release takes 14 months to complete. 2. Elicitation is continuously active and a requirement may be issued at any time by an issuer that foresees a market need. 3. Each requirement is stored in a database as an entity described in natural language. 4. Each requirement has a life cycle progression through specific states. The Telelogic development process has shown to have high resemblance to another market-driven development process independently developed and used at an Ericsson company (Carlshamre & Regnell, 2000).

Requirements are continuously collected through a web form and are stored in a database for further analysis (Regnell et al., 1998). The requirements are described in natural language and are of varying quality and nature. Some requirements are brief ideas while others are detailed

descriptions of new features with accompanying code. Many requirements are short-worded and poorly written.

During the development of a release the requirements engineer (or analyst) must handle the diverse and large set of requirements that is available in the database and resolve ambiguities, find relationships, eliminate duplicates, etc. As shown in a study of the Telelogic requirements process (Höst, Regnell, Natt och Dag, Nedstam, & Nyberg, 2000), these activities are causing a congestion that may be avoided by cutting down heavily on the number of elicited requirements or making early and strict prioritization.

The trade-off between analysing only a subset of all the collected requirements and not collecting that many requirements to give time for proper analysis may be difficult to make. Extra information could be extracted if all requirements are collected (for example, duplicates may indicate that certain issues are more important than others). However, trying to handle all incoming requirements may increase the risk of relationships between requirements being overlooked or discovered too late, which may cause problems in prioritization (Karlsson & Ryan, 1997) and release planning (Carlshamre & Regnell, 2000).

Consequently, there is a wish to find requirements relations early, without spending too much time on in-depth analysis. These relationships should preferably be found even when specification quality is low and even if requirements are short, poorly worded or misspelled. One possible approach, investigated in this paper, is to assist the requirements engineer through automated analysis of the textual information in the requirements. This approach may help the requirements engineer to handle the large set of requirements by automatically finding and make suggestions on relationships between requirements.

Two different automatic text-processing approaches may be used to aid the requirements engineer in the situation described above: the statistical approach or the linguistic approach. In this paper we focus on the statistical approach, which originates from the work by H. P. Luhn (Luhn, 1957). There are several reasons that we choose to explore this approach:

1. The ideas have not, as we far as we know, been applied to analyse the type of requirements that is collected in the situation we describe (see further Section 1.2).

2. The statistical approach has been thoroughly tried and examined and has been found fairly successful for automatic text analysis (van Rijsbergen, 1979).

3. The linguistic approach is still regarded as expensive to implement and not always more effective than well-executed statistical approaches (Mitra, Buckley, Singhal, & Cardie, 1997).

4. Before proceeding with more advanced methods, the statistical approach may help reveal the nature of the requirements in a market-driven organisation.

5. A baseline produced from empirical investigation using real industry requirements is needed to compare against further improvements.

The results of the presented work show that, for a particular set of requirements, a simple similarity analyser that uses the statistical text-processing approach identifies a large fraction of the requirements duplicate pairs found by experts. The duplicates are important to find to avoid doing the same job twice, assigning the same requirement to different developers, or getting two solutions to the same problem. The portion of requirement pairs incorrectly identified as duplicates is shown to have little negative impact on the value of the method. Further effort may thus be fruitful to assist the requirements engineer in handling the large set of requirements found in a market-driven development organisation.

1.2 Related work

The role of natural language processing in requirements engineering is discussed in Ryan (1993), where the conclusion is drawn that natural language-processing techniques must be realistic and effort has to be made to identify where such techniques may be useful. It is argued that the validation of requirements still has to be an informal, social process. Thus, an automated system could or should not replace the human requirements engineer. Such systems are still not feasible or cost-effective to construct.

Various attempts have been made to use automated techniques to assist the analysis of requirements written in natural language:

1. Gervasi and Nuseibeh (2000) use lightweight formal methods (low cost, partial analysis) to partially validate a syntactically correct NASA Software Requirements Specification (SRS) document. A glossary was manually produced from the SRS to aid the method.

2. Ambriola and Gervasi (1997) present a web-based environment where Model–Action–Substitution rules and a domain- and system-specific glossary are used to extract abstractions and build models.

3. Rayson, Emmet, Garside, and Sawyer (2000) report on a project called REVERE, where statistical and probabilistic natural language-processing methods are used to assist the analysis of complex and voluminous texts.

4. Park, Kim, Ko, and Seo (2000) present a system that uses a sliding window model and a parser to support the analysis of requirements using a similarity measuring technique.

5. Rolland and Proix (1992) present an environment that generates conceptual specifications from problem space descriptions written as sentences in natural language.

6. Osborne and MacNish (1996) describe an approach to resolve ambiguities where only a controlled language is allowed when writing requirements in order to facilitate for a lexicon and grammar-enabled parser.

7. Cybulski and Reed (1998) describe an elicitation method and a supporting management tool that help in analysing and refining requirements by using a parser, semantic networks, a domain-mapping thesaurus, and faceted classification schemes to allow proper formalisation of requirements written in natural language.

8. Chen et al. (1994) present ideas where concepts in texts from electronic meetings are automatically classified by using automatic indexing cluster analysis and hopfield net classification.

9. Landauer and Dumais (1997) present the Latent Semantic Analysis (LSA) computational model for generation of a representation from large corpora. The representation captures the similarity of meanings of words and sets of words.

Although relevant and promising for several areas and approaches in requirements engineering, the above attempts do not address the situation described in the previous section. The main concerns in the context of this work are the following:

- Requirements are considered to be found in a separate document that is to be analysed, quality assured and produced before implementation begins. This is not the situation in the market-driven organisation where requirements arrive continuously and may, at any time, affect previous, current and coming releases of the software.
- The initial quality of the requirements is often considered to be adequate for semantic parsing. This may not be the case when requirements are collected from many different sources and stored in a database.
- Real industrial requirements are not always used to validate the methods or techniques presented. Accuracy and efficiency are not always reported.
- The semantic nature of invented requirements may not share the properties of regular corpora used in many linguistic approaches.
- Simple, robust methods can act as a baseline for better understanding and further improvements and comparisons of techniques.

Several approaches seem promising but we believe that more effort needs to be put into this field to reach consensus on which methods, techniques, approaches and tools may be appropriate for different types of developing organisations. In this paper we focus on the market-driven organisation and do not present a new model or a full-featured approach. Rather, the feasibility of using automated similarity analysis is empirically investigated using real industrial requirements and a benchmark is provided to which further effort may be compared.

1.3 Paper structure

The paper is structured as follows. In Section 2 the situation of requirements similarity analysis in market-driven development is described. Section 3 explains how automated similarity analysis of natural language requirements may be performed. Section 4 presents a case study

where actual requirements collected from industry have been analysed. The case study explores the quality of a simple automatic similarity analyser. In Section 5, further applications of automated support are presented together with a small study using the analyser from Section 3 to investigate if similar requirements also are interdependent. Section 6 identifies possible further work and improvements. In the final section the results are discussed and conclusions presented.

2. Requirements similarity analysis

Requirements carry information on which decisions are based. This information can be either explicit or implicit. The explicit information constitutes all the written text, drawn charts and other artefacts that are used as the basis for communicating requirements. The implicit information consists of all the assumptions, rules, standards and the domain knowledge possessed by the requirement issuers and the requirements analyst. When natural language requirements arrive at a rapid flow from many different issuers, a quick analysis is required to guarantee requirements' quality before they are used as a basis for further decisions. Although the linguistic quality of the requirements may be low it is often left unattended as the requirements make sense. Rather, the information explicitly stated may not give sufficient decision support. For this reason the requirements engineer uses implicit and explicit information accompanied by personal skills to analyse the requirements for completeness, ambiguity, similarity, etc. Completeness analysis is performed to ensure that enough information is included in the requirements to enable further refinement, such as setting priority, estimating effort and deriving new requirements (see example requirements in Figure 4). Ambiguity analysis is performed to identify the risks of multiple interpretations among requirements. Similarity analysis is discussed below.

If supplementary information is needed to accept the requirement, the analyst may have to consult the issuer to make sure that the issuer and the analyst share the same interpretation. Thus, the requirements engineer acts to assure the quality of each requirement before allowing it to be further refined in the continuous requirements engineering process (Carlshamre & Regnell, 2000). The situation is illustrated in Figure 1, where example activities have been identified in the *quality gateway*.

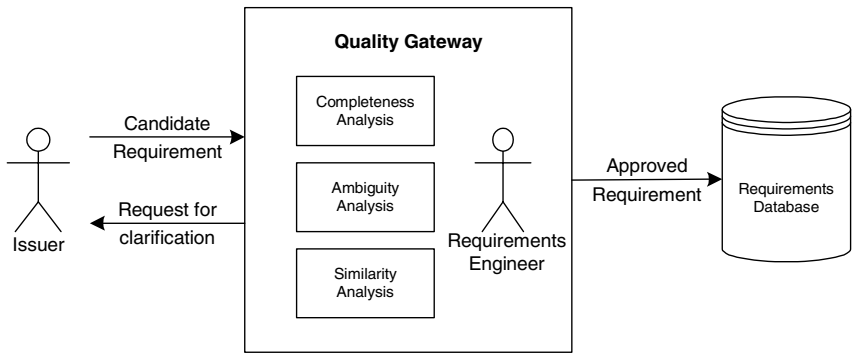

Figure 1. *Requirements Quality Gateway with three examples of quality assuring activities.*

The activities in the quality gateway are typically performed manually as there are few supportive tools available. The activities are tedious and time-consuming, but necessary in order to assure software quality and to satisfy market needs. It would therefore be highly beneficial if some of these tedious activities could be partly automated.

This paper focuses on similarity analysis, which is performed in order to find requirements that may be merged, grouped, eliminated or linked. For example, two similar requirements may be merged into one or may simply be grouped together to make sure they are handled simultaneously during development. A requirement may be similar to another to the extent that it is regarded as a duplicate and thus eliminated. Furthermore, two requirements may be similar in a certain aspect that establishes some kind of interrelationship (such as dependencies between requirements and requirement decompositions). The requirements engineer may also find it desirable to split large requirements into two or more requirements, which may become similar or related to each other and other requirements in the database.

When the requirements engineer decides whether two requirements are similar or not, it is with regard to the implications for further development. Of course these decisions are made by humans, but computer analysis of explicit information expressed in natural language may supply the requirements engineer with information regarding similarity to support these decisions.

3. Automated similarity measurement

Statistical approaches to automated similarity measurement are widely used in information retrieval (IR), which is a well established discipline concerned with automated storage and retrieval of documents written in natural language (Frakes & Baeza-Yates, 1992). The presented work is based on existing IR techniques applicable in the analysis of natural language requirements. Figure 3 provides an overview of the steps in similarity measurement, where a similarity metric $S_{A,B}$ is calculated for a pair of textual requirements *(A,B)*. The calculation of a similarity measure (further described in Section 3.1) is made subsequent to a number of pre-processing steps (elaborated in Section 3.2). The assessment of similarity metrics is described in Section 3.3.

3.1 Similarity measures

In order to find relationships between requirements that may be merged, grouped or eliminated, a quantification of the degree of association between the requirements is needed. Several similarity measures are available, but no comparative studies exist that give a definite answer to which one to choose in this particular situation. In this paper we have therefore used three simple and well-known similarity measures to calculate the similarity between sentences: the Dice, Jaccard and cosine coefficients (Salton, 1989). These measures all take the words in two sentences and calculate the similarity based on how many words they have in common. The coefficients are defined as follows, where A and B are requirements:

$$S^{D}_{A,B} = \frac{2\left|\{words_A \cap words_B\}\right|}{\left|\{words_A\}\right| + \left|\{words_B\}\right|}$$

$$S^{J}_{A,B} = \frac{\left|\{words_A \cap words_B\}\right|}{\left|\{words_A\}\right| + \left|\{words_B\}\right| - \left|\{words_A \cap words_B\}\right|}$$

$$S^{C}_{A,B} = \frac{\left|\{words_A \cap words_B\}\right|}{\sqrt{\left|\{words_A\}\right|\left|\{words_B\}\right|}}$$

All three measures have the desired property of normalisation, which imply that they give a value between 0 and 1 to indicate how similar a pair of sentences are, where 0 means that the sentences have no words in common and 1 means that the sentences are identical. The empirical investigation reported in Section 4 applies these measures to textual software requirements.

3.2 Preparing the source data

Before the similarity measure can be calculated the words of each sentence have to be extracted. This is achieved through *lexical analysis*, which takes an input stream of characters and converts it into a stream of words or tokens. This immediately raises the question of what should count as a word or token. For example, digits, hyphens, punctuation and letter case bring some problems that have to be considered. It is not technically difficult to solve these problems, but the chosen lexical analysis policy will affect the similarity measure. For example, preserving letter case will distinguish the words like 'System' and 'SYSTEM' and thus produce lower similarity measures. How to choose the policy thus depends on what type of data is to be analysed and the expected outcome.

Frequently occurring words like 'a', 'the', 'of', etc., will inadequately boost the similarity measures. These words, known as *stop words*, are therefore filtered out before similarity calculation. Which words to eliminate again depends on the type of data. It is reasonable to start out with a known stop word list that has been derived from general text.

Another issue is the *morphological variants* of words, i.e. the word forms. Words that are written in different forms usually carry the same information and should thus be considered equal. Therefore, words should be reduced to their ground form so that an automated word matcher would report a positive match. The technique used to reduce words to their ground form is called stemming and produces a stem from a word. For example, both the words 'replace' and 'replacement' may result in the stem 'replac' and consequently the words would be considered equal. There are several ways to stem words, such as affix removal, successor variety, table lock-up, and *n*-gram (Frakes & Baeza-Yates, 1992). In this paper we have used an affix removal stemmer, the Porter algorithm (Porter, 1980), which consists of a set of condition/action rules. It is a compact algorithm that has been shown to give good results in IR (Frakes & Baeza-Yates). The similarity measure may be

calculated by counting the number of stems produced from each requirement and the number of stems the requirements has in common. The common stems may be found using *exact match* or *inexact match*. Exact match requires the stem to be exactly equal, whereas inexact match calculates the similarity between the stems. Spelling errors may call for inexact match but brings the difficulty of choosing a good algorithm and a threshold level for match. The analyser used in this paper is designed to require an exact match between stems. The low linguistic quality of the requirements will of course affect the similarity measure. However, we have chosen not to include spelling correction, as we are interested in the performance of using a simple technique. It is also questionable if there is time for manual pre-processing in industrial settings.

3.3 Assessing the quality of similarity measures

In order to evaluate the technique used to suggest similar requirements, a notion of quality is needed. We have chosen to use a contingency table, which defines a number of quality aspects in similarity measurement. Assume that $S(r_i, r_j)$ is a function that takes a pair of requirements and gives a similarity measure between 0 and 1. In addition we select a threshold value t, which acts as a selection criterion. If $S(r_i, r_j) \geq t$ then (r_i, r_j) is considered to be a suspected duplicate pair. Assume also that there exists a set of pairs of requirements that are identified as actual duplicate pairs. The similarity measure hence provides an approximation of this set of actual duplicate pairs, and the quality of the estimation may be defined according to Figure 2 (Salton, 1989).

The resulting pairs that have a similarity value above or equal to the threshold level are regarded as duplicate pairs suggested by the analyser. Matches between actual duplicate pairs and those suggested by the analyser are denoted true positives. The actual duplicate pairs not identified by the analyser are consequently denoted false negatives, i.e. they were wrongly suggested as non-duplicate pairs. The analyser may also suggest duplicate pairs that actually were non-duplicate pairs. These are denoted false positives. The rest are denoted true negatives and constitute all the requirement pairs that fell below the threshold level and were correctly suggested as non-duplicate pairs. The accuracy of the analyser is defined as the sum of the true negatives and the true positives divided by the total number of possible requirement pairs and indicates how well the actual duplicate pairs and non-duplicate pairs are identified.

	Below similarity threshold	**Above or equal to similarity threshold**	***Total***
Actual non-duplicates	A True negatives	B False positives	*A+B*
Actual duplicates	C False negatives	D True positives	*C+D*
Total	*A+C*	*B+D*	*A+B+C+D*

True positives rate = D/(C+D)
False positives rate = B/(A+B)
Accuracy = (A+D)/(A+B+C+D)

Figure 2. *Assessment scheme with contingency table*

The total number of requirement pairs is calculated as $A + B + C + D$, which is equal to $(n \cdot (n - 1))/2$, when n is the number of requirements.

The contingency table will help reveal the performance of the method. In order to evaluate the feasibility of the analyser, a deeper investigation of the requirement pairs is needed. Taking any two identified pairs, they may or may not involve the same particular requirements. For example, the requirement pairs *(A, F)* and *(C, F)* share the requirement *F*. If the analyser assigns similarity values above zero to each of these pairs and a similarity value equal to zero to the pair *(A, C)* it would nevertheless be interesting to look at the three involved requirements together. We denote these preferred groupings of requirements as *n*-clusters, where n is the number of requirements in the cluster. The two single pairs in the previous example will thus form a *3*-cluster. The cluster distribution can be derived by calculating the transitive closure of a graph in which the nodes correspond to requirements and edges correspond to pairs of requirements (r_i, r_j) with $S(r_i, r_j) \geq t$. The sizes of the clusters and the number of clusters reveal the usefulness of the automated similarity analysis. It may be desirable to have many requirements grouped into *n*-clusters where n is the greatest number of requirements that the requirements analyst is capable of handling simultaneously. Example cluster distributions are presented in Figure 6a-b.

4. Empirical investigation

In order to investigate the potential benefits of automated similarity analysis, we have applied the similarity measures described in Section 3.1

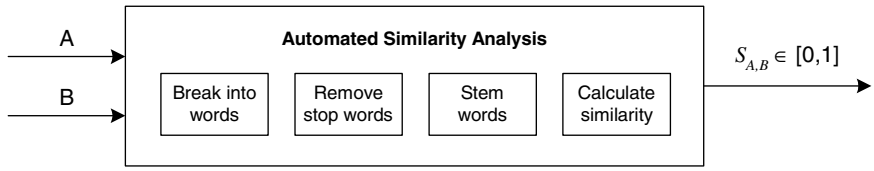

Figure 3. *A functional view of automated similarity analysis between requirement A and B, producing a measure $S_{A,B}$ ranging from 0 to 1.*

to real industrial requirements. The measures were used to see if automated analysis can correctly determine if a certain requirement is a duplicate of an already existing requirement.

For the investigation we have developed a computer program to perform the tasks specified in Figure 3. The pre-processing steps are handled by a lexical analyser, a stop word remover and a stemmer (explained in Section 3). The stop list remover excludes words with low discrimination value, and consists of 425 words derived from the Brown corpus (Francis, & Kucera, 1982). For the stemming of words, the Porter algorithm is applied (Porter, 1980). The similarity calculation produces a list of requirement pairs along with a value for each pair representing the similarity measure.

Telelogic, a large software developer, has allowed us restricted access to a requirements database of 1,920 confidential requirements. Telelogic develops software development tools for a wide market and handles requirements arriving at a high rate from several different stakeholders (about three requirements a day (Höst et al, 2000). The requirements are submitted through a web interface and thereafter managed by requirements engineers (Regnell et al., 1998).

In Figure 4, two examples of requirements from the database are shown. Many of the attributes are set at different stages in the requirements process, reflecting the refinement of the requirement from submitted to implemented or rejected (Regnell et al., 1998). The stage is represented by the 'Status' and the possible stages are shown in the leftmost column in Table 1. The table also shows, in the second column, the distribution of the 1,920 requirements over the different stages.

4.1 Preparations

When a requirements engineer analyses a requirement, the requirement is checked on many different properties. Three related properties are

RqId	RQ96-270
Date	
Summary	Storing multiple diagrams on one file
Why	It must be possible to store many diagrams on one file. SDT forces to have 1 diagram per file. It's like forcing a C programmer to have not more than one function per file... The problem becomes nasty when you work in larger projects, since adding a procedure changes the system file (.sdt) and you end up in a mess having to "Compare systems".
Description	Allow the user to specify if a diagram should be appended to a file, rather than forcing him to store each diagram on a file of its own.
Dependency	4
Effort	4
Comment	This requirement has also been raised within the multiuser prestudy work,but no deeper penetration has been made. To see all implications of it we should have at least a one-day gathering with people from the Organizer, Editor and InfoServer area, maybe ITEX? Här behövs en mindre utredning, en "konferensdag" med förberedelser och uppföljning. Deltagare behövs från editor- och organizergrupperna, backend behövs ej så länge vi har kvar PR-gränssnittet till dessa.
Reference	
Customer	All
Tool	Don't Know
Level	Slogan
Area	Editors
Submitter	*x*
Priority	3
Keywords	storage, diagrams, files, multi-user
Status	Classified

RqId	RQ97-059
Date	Wed Apr 2 11:40:20 1997
Summary	A file should support storing multiple diagrams
Why	ObjectGeode has it. It's a powerful feature. It simplifies the dayly work with SDT. Easier configuration management. Forcing one file for each procedure is silly.
Description	The SDT “Data model'' should support storing multiple diagram on one file.
Dependency	4
Effort	1-2
Comment	Prestudy needed
Reference	http://info/develop/anti_og_package.htm
Customer	All
Tool	SDT SDL Editor
Level	Slogan
Area	Ergonomy
Submitter	*x*
Priority	3: next release (3.3)
Keywords	diagrams files multiple
Status	Classified

Figure 4. *Two example requirements denoted duplicates in the database. These two requirements were also suggested as duplicates by the similarity calculator at the 0.75 threshold level using the cosine similarity measure.*

(1) whether or not it is regarded as a duplicate of another requirement already in the database, (2) if it is possible to merge it with another requirement and (3) if it should be split into two or more requirements before further analysis. If a requirement has one of these properties, it is assigned the 'Duplicate' status and an appropriate action is taken. When a requirement is merged, all the information is added to the requirement it is merged with. When a requirement is split, the information is distributed over two or more new requirements. When a requirement is a pure duplicate (property 1 above), no further action is taken with the information.

As shown in Table 1, 130 of the 1,920 requirements were either duplicates, merges or splits. In the analysis, only those that are 'true' duplicates are considered, since we know beforehand that merges and splits will match partially and thus bias the result. When these were removed, 101 requirements remained. The resulting set is shown in column 3 of Table 1 (set A_{full}).

Some of the 101 duplicates involved more than one requirement. This means that a requirement may be denoted a duplicate of two other requirements. To resolve this we parsed every identified duplicate and constructed a set of unique duplicate pairs. However, doing this creates a set of duplicate pairs that may be related (which addresses the discussion about clusters at the end of Section 3.3). Therefore, we calculated all these relations and created new duplicate pairs to denote the relation. For example, if requirement *A* initially was denoted a duplicate of requirements *B* and *C*, and requirement *D* was denoted a duplicate of requirement C, we would first create the duplicates pairs *(A, B)*, *(A, C)* and *(D, C)*. Then we would add the pairs *(B, C)*, *(A, D)* and *(B, D)* to fully reflect all possible relations. This is acceptable since the duplicate relation is transitive. That is, if both *A* and *D* are duplicates of *C*, then *A* would also be a duplicate of *D*.

According to the requirements database manager, not all the requirements having status New or Assigned had been analysed for duplicates, and it was only certain that those having priority 1 had been analysed. Therefore, we considered removing all requirements with status 'New' or 'Assigned', not having priority 1. After doing this we noticed that some duplicate pairs referred to the removed requirements. Thus, we decided to analyse two sets: one with all requirements and one with the 'New' and 'Assigned' requirements with priority not equal to 1 removed. As the second set does not include all the requirements addressed in the

Table 1. Number of requirements in the database and in the different sets prepared for analysis.

Status	Original	A_{full}	$A_{reduced}$
New	406	406	12
Assigned	428	428	31
Classified	601	601	601
Implemented	252	252	252
Rejected	103	103	103
Duplicates	130	101	90
Total	1,920	1,891	1,089
Duplicate pairs	-	142	124

duplicate pairs, those pairs were removed from the duplicates pair set. The resulting number of requirements and duplicate pairs are shown in column 4 in Table 1 (set $A_{reduced}$).

The textual information used to represent each requirement was collected from the 'Summary' field, which corresponds to a short requirement title, and the 'Description' field, which corresponds to a further explanation (see the examples in Figure 4). As these fields were empty for a subset of the requirements, three different requirement sets were prepared from each of sets A_{full} and $A_{reduced}$. The first set comprised all the requirements that had a non-empty 'Summary' field. The second set comprised all the requirements that had a non-empty 'Description' field. The third set comprised all the requirements that had a non-empty 'Summary' field or a non-empty 'Description' field (NB. Not *exclusive or*. Requirements having a non-empty 'Summary' field and a non-empty 'Description' field were included in the last set). In the analysis of the sets using both fields, the two fields were treated as one. Table 2 shows the number of requirements in each of the sets after the requirements with the empty fields had been removed.

Table 2. Final sets prepared for the analysis.

	B_{full}		$B_{reduced}$	
Non-empty field	Require-ments	Duplicate pairs	Require-ments	Duplicate pairs
Summary	1,830	142	1,085	124
Description	1,570	99	915	86
Summary or Description	1,887	142	1,088	124

Table 3. Contingency table data for the summary field in set B_{full} using the cosine similarity measurement.

	0+	0.125	0.25	0.375	0.5	0.625	0.75	0.875	1
True positives (D)	114	114	105	80	62	47	42	31	30
True negatives (A)	1,578,213	1,578,581	1,628,049	1,666,093	1,670,881	1,672,945	1,673,247	1,673,341	1,673,349
False positives (B)	95,180	94,849	46,555	8,111	2,864	499	146	52	44
False negatives (C)	28	28	35	61	80	93	100	111	112

4.2 Results

The similarity calculator was run once for each of the prepared requirements sets to calculate the three similarity coefficients described in Section 3.1. The quality was assessed by producing contingency tables for nine different threshold levels as explained in Section 3.3. The threshold levels ranged from 0 to 1 with a 0.125 interval. All the possible combinations resulted in 162 contingency tables (3 measurements · 2 sets · 3 fields · 9 thresholds = 162 tables).

In Table 3, nine contingency tables are shown for the analysis on the 'Summary' field of set B_{full} using the cosine similarity measure. The number of possible unique pair-wise comparisons, which is the same as the total number of possible unique requirement pairs, is denoted $A + B + C + D$ in the contingency table in Figure 2, and corresponds to the sum of each column in Table 3. The first row shows the number of correctly identified duplicate pairs and decreases as the threshold increases. Most requirement pairs are, correctly, considered as non-duplicate as shown in the second row. Their number increases with the threshold level. The third row shows how many duplicate pairs the analyser identified that actually were not identified as duplicate pairs by the experts. Finally, in the fourth row are all the actual duplicate pairs that the analyser did not find.

The number of false positives and negatives at threshold level 1 may raise some questions. There may be false negatives because requirements concerning exactly the

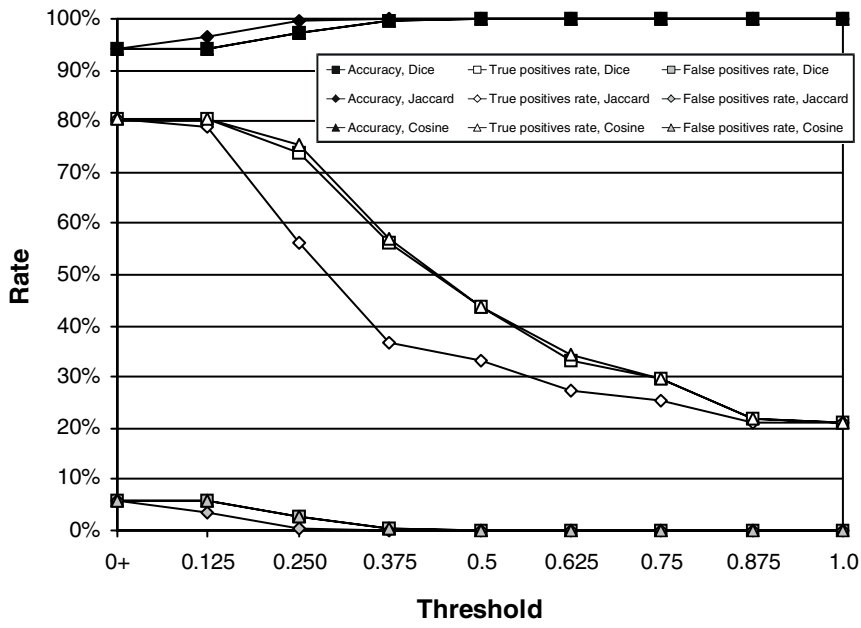

Figure 5a. *Similarity analysis performance using the summary field in set* B_{full}.

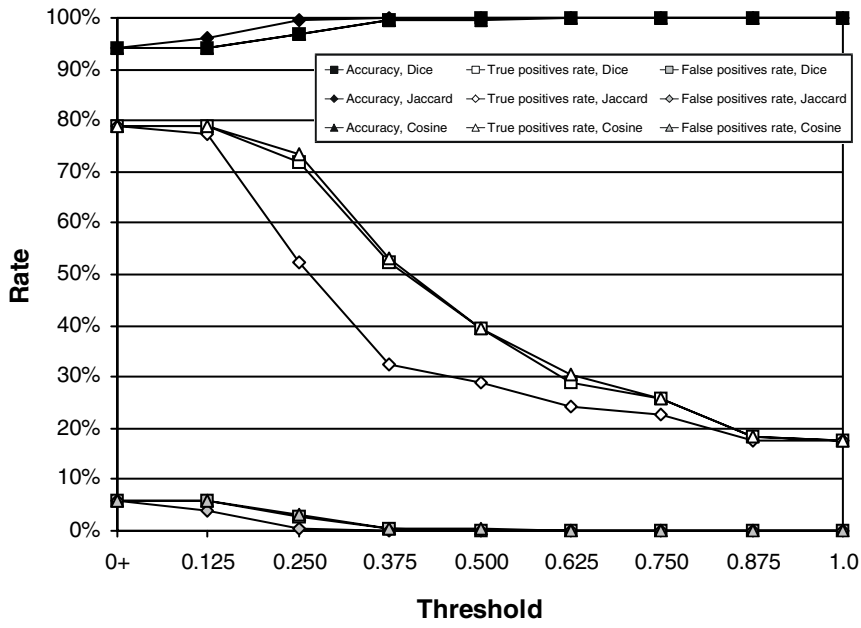

Figure 5b. *Similarity analysis performance using the summary field in set* $B_{reduced}$.

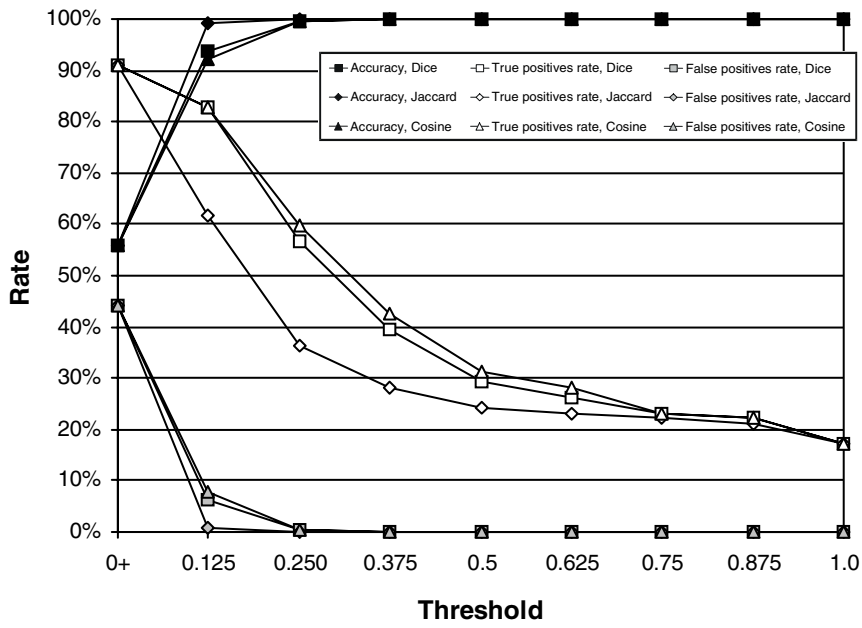

Figure 5c. *Similarity analysis performance using the description field in set* B_{full}.

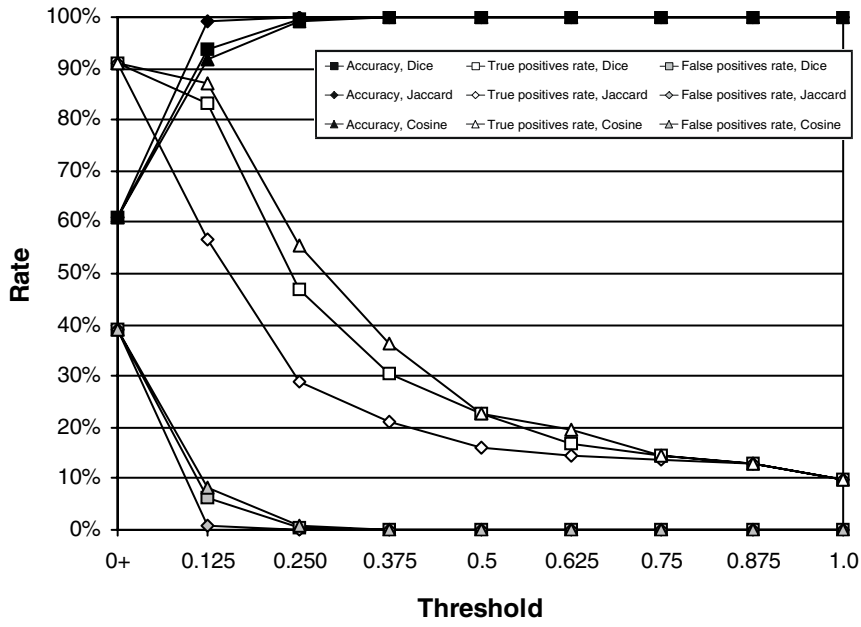

Figure 5d. *Similarity analysis performance using the summary and description fields in set* $B_{reduced}$.

same issue may be worded differently. The reasons that there may be false positives are several:

1. A requirement may be partially implemented and result in new requirements. The implemented requirement and the new requirements may then have the same information in some textual attributes. Since none of these requirements are marked as duplicates in the database the automatic analyser may produce a false positive.

2. The compared textual attributes may be wrong and misleading, not reflecting the actual meaning of the requirement.

3. Two requirements may be highly related and concern the same issue and have the same information in one textual attribute. Nevertheless, they do not have to be duplicates.

4. If all non-matching words in two requirements happen to be stop words, and thus eliminated before the similarity calculation, the reduced requirements may give a similarity measure of 1 but actually have different wordings.

The rate of true positives, the rate of false positives and the accuracy (see Section 3.3) were plotted to compare the measurements and to see which would generate the best result. In Figure 5a-b, four graphs are shown to support the conclusions on:

- which measurements may be considered the best
- whether or not the requirements with status 'New' or 'Assigned' and not priority 1 should be ignored
- which fields or combination of fields give the best results.

The graphs show that the rate of correctly identified duplicate pairs (the true positives rate) decreases from 80% or 90% at threshold level 0+ to about 20% at threshold level 1. The lowest degree of similarity is found when there is only one single word matching. Each measure will then give a similarity value just above 0 and thus, using threshold level 0+, suggest exactly the same set of duplicate pairs (every similarity measure but zero between two requirements results in a suggested duplicate pair). Correspondingly, the highest degree of similarity is found when all words match. Each measure will then give a similarity measure of 1 and produce

exactly the same set of duplicate pairs. Between these threshold levels the curves differ slightly, which shows that the similarity measures perform differently. The Dice and cosine similarity coefficients show no significant difference, but the Jaccard coefficient performs slightly worse. Thus, for this particular set of requirements, the Dice or cosine coefficient is preferable.

The false positive rate is very low, decreasing from 5.69% down to 0.01%. The accuracy of the similarity analyser is as high as 94.3% at the lowest threshold level and increases to near 100% at threshold level 1. This curve suggests that the Jaccard coefficient is a better choice, contradicting the choice based on the positives rate.

Looking at the two topmost graphs, which show the results from using only the 'Summary' field, we can see that there is no considerable difference between the results for set B_{full} and $B_{reduced}$. This implies that either (1) there are 'New' and 'Assigned' requirements with lower priorities that have been analysed and found to be duplicates, of which some are identified by the program, or (2) the requirements have not been analysed and few matches were found by the program. Alternative 1 seems more plausible and is also confirmed by the contingency table – more duplicates are identified which must be related to the 'New' and 'Assigned' requirements with lower priorities.

The two leftmost graphs, showing the results from using the 'Summary' or the 'Description' fields respectively (from set B_{full}), differ on the low and high threshold levels. At threshold level 0+, the true positives rates is as high as above 90% using a combination of the 'Summary' and the 'Description' fields. However, the false positives rate is substantially higher and the true negatives rate has also dropped significantly. The conclusion from this comparison is that using only the 'Summary' field gives more accurate answers. The reason for this is that the 'Description' field contains too much noise that incorrectly boosts the similarity measures.

Finally, the top left and the two bottommost graphs support the rather evident: a combination of the 'Summary' and the 'Description' field results in a combination of the results from using the 'Summary' and the 'Description' fields separately.

The high number of requirement pairs identified at threshold level 0+ in Table 3 may at first seem very discouraging. However, calculating the cluster distribution of all the positives (true and false) as explained in

Section 3.3 gives support to the following conclusions and the usefulness of the result.

The cluster distributions for the $B_{reduced}$ set are shown in Figure 6a-b. Each figure shows four graphs. The first three show the cluster distribution using the cosine measure on the 'Summary' and the 'Summary' + 'Description' fields respectively. The last graph in each row shows the cluster distribution for the actual duplicates found by the experts.

The graphs show that with increasing threshold the number of clusters of larger size decreases. For example, in Figure 6a at threshold level 0.375 there is one very large cluster involving 123 different requirements.

What is noteworthy about this is that the presented study is made on a very large set of requirements but that in reality the requirements arrive continuously, a few at a time. The similarity analysis can thus be made incrementally on a smaller set of requirements, avoiding the need for

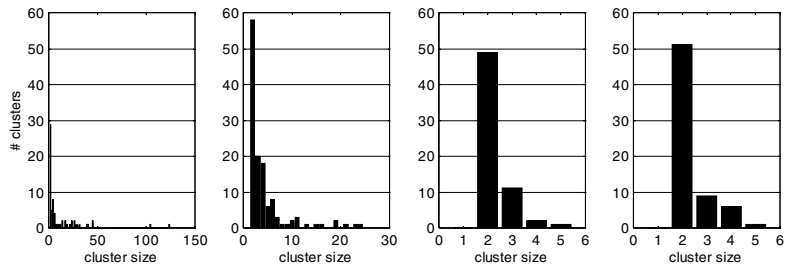

Figure 6a. *Requirements cluster distribution for the $B_{reduced}$ set using the cosine measure on the 'Summary' field. The three leftmost graphs show the number of clusters of different sizes for various thresholds compared to the actual cluster distribution on the right.*

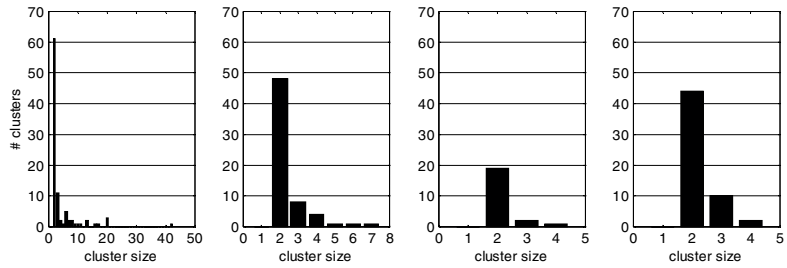

Figure 6b. *Requirements cluster distribution for the $B_{reduced}$ set using the cosine measure on the 'Summary' and the 'Description' fields. The three leftmost graphs show the number of clusters of different sizes for various thresholds compared to the actual cluster distribution on the right.*

interpreting the results of similarity analysis of the entire set of requirements at one time. The cluster distribution shows that if we analyse one randomly selected requirement from the database (which may represent a newly submitted requirement), the worst case would be that the analyser suggests a cluster of 123 requirements to be identical (Figure 6a, leftmost graph). This is thus the maximum number of requirements the requirement analyst must handle simultaneously. As the number may seem too high for the lower thresholds, it is reasonable to suggest that too large clusters may be ignored, as they are probably irrelevant.

Considering both performance and cluster distribution, we may also conclude that the Dice and cosine measures are superior. The true positives rate has already been shown to be higher, and the higher false positives rate is compensated by the suggestion of analysing a group of related requirements simultaneously, instead of checking each of the several thousand possible duplicate pairs.

Another interesting issue is whether the automated analyser reveals duplicate pairs that the experts missed. To explore this we let an expert analyse the 75 false positives suggested when using the cosine measurement on the 'Summary' field for set B_{full} at threshold level 0.75. Table 4 shows the surprising result from the analysis. It turned out that 37% of the suggested duplicate pairs were actually missed by the experts! For that threshold level, the true positives rate would then increase from 26% (Figure 5b) to almost 40%, the already low false positives rate would decrease, and the already high accuracy would increase. The analyst did not regard two requirements in a pair as duplicate or similar if they were to be implemented in different parts of the software. The table also shows the additional relationships identified, which thus imply that only 21 of the 75 pairs identified would be completely wrong.

Table 4. Result of expert analysis of the false positives for the set Bfull at the threshold 0.75 using the cosine measure on the summary field.

Relationship	Count
Duplicate	28
Similar	13
Related	8
Part of	5
Not related	21

The manual analysis also indicated that the analyser might have a problem when there are too few words in the fields. One suggestion would then be to use the 'Description' field only when the 'Summary' field has too few words.

Furthermore, the threshold value can be tuned based on the requirements engineer's

consideration of the best trade-off between few false positives and many true positives.

In summary, it may be concluded that:

1. The similarity analysis technique gives reasonably high accuracy considering its simplicity.
2. For incremental analysis of requirements, given that related requirements are grouped into clusters, the Dice and cosine may be considered the superior measures.
3. A large explanatory field tends to give a worse result, as the discrimination between requirements declines. However, if one field has too few words it may be worth using other lengthy fields.
4. The grouping of suggested duplicate requirements into clusters reduces the analysis burden considerably.

5. Further applications

There are numerous conceivable applications of automated similarity analysis beyond identifying duplicates. The following briefly describes some of these application areas, of which we have only evaluated one so far.

5.1 Requirements interdependencies

Requirements interdependencies are important to identify and keep track of for requirements prioritization and release planning purposes, as interdependencies may govern what partitions of a particular set of requirements are allowed from a functional perspective, or eligible from a cost/value perspective. Carlshamre and Regnell (2000) describe a number of salient interdependencies found in a study of empirical data The relationship between similarity and interdependency is evident in the case where we have two requirements R1 and R2, with the exact same 'Summary' field. This would be a true duplicate pair in the previous sense, but it would also represent an OR interdependency, which imply that either one of the requirements could be implemented. The existence of common keywords may indicate other types of interdependencies as well. For example, if there are several requirements that include the word

'sorting', it may be wise to consider implementing these together to save development resources, which would represent an interdependency regarding cost of implementation.

To investigate whether the similarity measurement technique could be used to support the identification of interdependencies in a set of requirements, we applied the same analysis technique as described in Section 3.1 to five different sets of 20 high-priority requirements, previously studied manually by experts (for further information on the results of the manual study, see Carlshamre, Sandahl, Lindvall, Regnell, and Natt och Dag, 2001). Among the total of 100 requirements, there were in total 155 pair-wise interdependencies manually identified by experts from each of the five organisations.

Results. Each set of 20 requirement slogans were relieved of stop words and reduced to stems, before being separately fed to the similarity calculator using the cosine coefficient. The automatic analyser reported 70 similar pairs on a 0+ threshold (9, 18, 21, 10 and 12 pairs in each set respectively), of which 25 were true positives. Table 5 shows the frequencies of actual dependencies in relation to the similarity measure using the assessment scheme presented in Table 1.

Table 5. Contingency table for dependencies and similarities.

	Similarity=0	**Similarity>0**	**Total**
Actual non-dependencies	750	45	795
Actual dependencies	130	25	155
Total	880	70	950

A chi-square test (Siegel & Catellan, 1988) gives a *p*-value less than 0.0001, which shows that the similarity measure varies significantly with actual dependencies.

Thus, by checking for lexical similarity, this particular case demonstrates that it is a promising technique to support the interdependency identification process by automatic analysis. Although the accuracy may not suffice for this technique to be used on its own,

automatic lexical analysis may be used in conjunction with other techniques to reduce the effort of identifying interdependencies.

5.2 Requirements gathering

When a stakeholder is proposing a new requirement, it may be valuable to know if a similar requirement has already been implemented and, if so, in what release. If a similar requirement has not been implemented, it may be desirable to know if a similar requirement has been proposed.

5.3 Strategic fit

A company may define key areas that are of specific importance for the requirements work (e.g., usability, decision-making features or invoicing capabilities). When such requirements are proposed, they can be identified by a similarity analysis approach and thus more easily be given the appropriate management attention.

5.4 Defect tracking

Companies with mature software products that have gone through series of releases often have many defects to track and analyse. As new defects are reported, a similarity analysis approach can aid testers to identify if similar defects have been reported earlier.

5.5 Support Issues

Some companies allow their customers to get feedback on support issues through their web sites. Similarity analysis approaches can help the customer to enter questions in natural language and more easily analyse the questions and find suitable answers.

6. Further improvements

There are a number of potential improvements that can be made to the presented requirements similarity measurement method, including the following suggestions to be evaluated in further research:

- Process issues such as when similarity analysis should be used, who should perform the analysis and how the analysis is cost-efficient to perform.
- How different ways of representing requirements affect the results. Which representation is best suited for high precision in automatic similarity analysis?
- Different attributes' impact on similarities. Use of other attributes may increase precision.
- Improve method accuracy. Examples include: the use of a domain-specific stop list, a thesaurus with general synonym words, spelling correction prior to the automated similarity analysis and by not discriminating between words with a short editing distance.
- Smart algorithms: some words may be over-represented in the set of false positives. Removing these words may improve the precision. This is an example of where it may be possible to make the algorithm self-adjustable based on human corrections.
- Evaluate linguistic methods that may provide more precise analysis of natural language requirements on a semantic level. This may include the use of ontologies or word nets.
- Ways of visualising the results from automated similarity analysis and supporting the requirements engineer in the navigation among related requirements.

In order to make these improvements and to make the methods more general it is of course desirable to apply the methods to other requirement sets from industry. Also, it is of great interest to compare different approaches and combinations of approaches. The implementation cost and computational effort needed for statistical methods, linguistic methods and other computational models (such as the LSA approach in Landauer & Dumas, 1997) are of much interest for applications aimed at market-driven organisations.

7. Conclusions

Automated similarity analysis is a promising technique for supporting requirements engineers to identify requirements duplicates and

interdependencies. This conclusion is drawn on the basis of empirical studies on industrial requirements. Automated analysis is, in the particular cases of the presented investigations, able to identify as many as 80% of the actual duplicates and still only incorrectly classify about 6% of all the possible requirement pairs.

When using automated similarity analysis for interdependency identification, a significant correlation was found between similarity and interdependency. The results show a correct classification of 16% of the actual interdependencies.

We do not believe that the presented technique can replace human judgement, but our results suggest that automated similarity analysis on a syntactic level using information retrieval techniques may be effective in pinpointing true duplicates and interdependencies. Further studies are needed in order to increase the understanding of the benefits and limits of automated analysis of natural language requirements (Ryan, 1993). It is especially important to conduct further research in real situations, where new requirements are continuously arriving from multiple sources, and where requirements are analysed incrementally by a requirements engineer with domain expertise. In these investigations it is also of importance to consider the relationship between effort needed to put a method to work in a market-driven company and the efficiency of the method. Conducting real-world studies is a necessary means for valid assessments of the benefits and costs of decision support systems in a market-driven requirements engineering context.

Acknowledgements

This work is partly funded by the National Board of Industrial and Technical Development (NUTEK), Sweden, within the REMARKS project (Requirements Engineering for Market-Driven Software Development) grant 1K1P-97-09690. A previous version of this paper was published at the Seventh International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ'2001). We would like to direct warm thanks to Per Runeson, Martin Höst and Thomas Olsson, all at the Department of Communication Systems, Lund, for their valuable input and enthusiastic suggestions.

8. References

Ambriola, V., & Gervasi, V. (1997). Processing Natural Language Requirements. In *Proceedings of the Twelfth International Conference on Automated Software Engineering* (pp. 36–45), Los Alamitos, CA: IEEE Computer Society Press.

Carlshamre, P., & Regnell, B. (2000). Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes. In A. M. Tjoa, R. R. Wagner, A. & Al-Zobaidie (Eds.)*, Proceedings of the 11th International Workshop on Database and Expert Systems Applications Process* (pp. 961–965). Los Alamitos, CA: IEEE Computer Society Press.

Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., & Natt och Dag, J. (2001). An Industrial Survey of Requirements Interdependencies in Software Product Release Planning. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering* (pp. 84–91). Los Alamitos, CA: IEEE Computer Society Press.

Chen, H., Hsu, P., Orwig, R., Hoopes, L., & Nunamaker, J. F. (1994). Automatic Concept Classification of Text from Electronic Meetings. *Communications of the ACM, 37*(10), 56–73.

Cybulski, J. L., & Reed, K. (1998). Computer Assisted Analysis and Refinement of Informal Software Requirements Documents. *Proceedings of the Fifth Asia-Pacific Software Engineering Conference* (pp. 128–135). Los Alamitos, CA: IEEE Computer Society Press.

Deifel, B. (1999). A Process Model for Requirements Engineering of CCOTS. In *Proceedings of Tenth International Workshop on Database and Expert Systems Applications* (pp. 316–320). Los Alamitos, CA: IEEE Computer Society Press.

Frakes, W. B., & Baeza-Yates, R. (1992). *Information Retrieval : Data Structures & Algorithms*. Engelwood Cliffs, NJ: Prentice-Hall.

Francis, W. N., & Kucera, H. (1982). *Frequency Analysis of English Usage*. New York: Hougton Mifflin.

Gervasi, V., & Nuseibeh, B. (2000). Lightweight Validation of Natural Language Requirements. In *Proceedings of the Fourth IEEE International Conference on Requirements Engineering* (pp. 140–148). Los Alamitos, CA: IEEE Computer Society Press.

Höst, M., Regnell, B., Natt och Dag, J., Nedstam, J., & Nyberg, C. (2001). Exploring Bottlenecks in Market-Driven Requirements Management Processes with Discrete Event Simulations. *Journal of Systems and Software, 59*, 323–332.

Karlsson, J. & Ryan, K. (1997). A Cost–Value Approach for Prioritizing Requirements. *IEEE Software, 14*(5). 67–74.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review, 104*, 211–240.

Lubars, M., Potts, C., & Richter, C. (1993). A Review of the State of the Practice in Requirements Modeling. In *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 2–14). Los Alamitos, CA: IEEE Computer Society Press.

Luhn, H. P. (1957). A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development, 1*, 309–317.

Mitra, M., Buckley, C., Singhal, A., & Cardie, C. (1997). An Analysis of Statistical and Syntactic Phrases. In *Proceedings of 5th International Conference on Computer-Assisted Information Searching on Internet* (pp. 200–214). Paris: Centre de Hautes Etudes Internationales d'Informatique Documentaires.

Moreno, A. M. (1997). Object-Oriented Analysis from Textual Specifications. In *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering* (pp. 48–55). Skoki, IL: Knowledge Systems Institute.

Osborne, M., & MacNish, C. K. (1996). Processing Natural Language Software Requirement Specifications. In *Proceedings of the Second IEEE International Conference on Requirements Engineering* (pp. 229–236). Los Alamitos, CA: IEEE Computer Society Press.

Park, S., Kim, H., Ko, Y., & Seo, J. (2000). Implementation of an Efficient Requirements-Analysis Supporting System Using Similarity Measure Techniques. *Information and Software Technology, 42*, 429–438.

Porter, M. F. (1980). An Algorithm for Suffix Stripping. *Program, 14*, 130–137.

Potts, C. (1995). Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software. In *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 128–130). Los Alamitos, CA: IEEE Computer Society Press.

Rayson, P., Emmet, L., Garside, R., & Sawyer, P. (2000). The REVERE Project: Experiments with the application of probabilistic NLP to Systems Engineering. In *Proceedings of the Fifth International Conference on Applications of Natural Language to Information Systems* (pp. 288–300). New York: Springer-Verlag.

Regnell, B., Beremark, P., & Eklundh, O. (1998). A Market-Driven Requirements Engineering Process – Results from an Industrial Process Improvement Programme. *Requirements Engineering, 3*, 121–129.

Rolland, C., & Proix, C. (1992). A natural language approach for requirements engineering. In P. Loucopoulos (Ed.)*, Lecture Notes in Computer Science, Vol. 653: Advanced Information Systems Engineering, 10th International Conference* (pp. 257–277). Berlin: Springer-Verlag.

Ryan, K. (1993). The Role of Natural Language in Requirements Engineering. In Proceedings of IEEE International Symposium on Requirements Engineering (pp. 240–242). Los Alamitos, CA: IEEE Computer Society Press.

Salton, G. (1989). *Automatic Text Processing : The transformation, Analysis, and Retrieval of Information by Computer*. Reading, MA: Addison-Wesley.

Sawyer, P., Sommerville, I., & Kotonya, G. (1999). Improving Market-Driven RE Processes. In M. Oivo, & P. Kuvaja (Eds.), *Proceedings of the International Conference on Product Focused Software Process Improvement* (pp. 222–236). Oulu, Finland: Technical Research Centre of Finlad (VTT).

Siegel, S., & Castellan, N. J (1988). *Nonparametric Statistics for the Behavioral Sciences* (2nd ed.). New York: McGraw-Hill.

van Rijsbergen, C. J. (1979). *Information Retrieval* (2nd ed.). London: Butterworths.

Reports on Communication Systems

101 **On Overload Control of SPC-systems**
Ulf Körner, Bengt Wallström, and Christian Nyberg, 1989.
CODEN: LUTEDX/TETS- -7133- -SE+80P

102 **Two Short Papers on Overload Control of Switching Nodes**
Christian Nyberg, Ulf Körner, and Bengt Wallström, 1990.
ISRN LUTEDX/TETS- -1010- -SE+32P

103 **Priorities in Circuit Switched Networks**
Åke Arvidsson, *Ph.D. thesis*, 1990.
ISRN LUTEDX/TETS- -1011- -SE+282P

104 **Estimations of Software Fault Content for Telecommunication Systems**
Bo Lennselius, *Lic. thesis*, 1990.
ISRN LUTEDX/TETS- -1012- -SE+76P

105 **Reusability of Software in Telecommunication Systems**
Anders Sixtensson, *Lic. thesis*, 1990.
ISRN LUTEDX/TETS- -1013- -SE+90P

106 **Software Reliability and Performance Modelling for Telecommunication Systems**
Claes Wohlin, *Ph.D. thesis*, 1991.
ISRN LUTEDX/TETS- -1014- -SE+288P

107 **Service Protection and Overflow in Circuit Switched Networks**
Lars Reneby, *Ph.D. thesis*, 1991.
ISRN LUTEDX/TETS- -1015- -SE+200P

108 **Queueing Models of the Window Flow Control Mechanism**
Lars Falk, *Lic. thesis*, 1991.
ISRN LUTEDX/TETS- -1016- -SE+78P

109 **On Efficiency and Optimality in Overload Control of SPC Systems**
Tobias Rydén, *Lic. thesis,* 1991.
ISRN LUTEDX/TETS- -1017- -SE+48P

110 **Enhancements of Communication Resources**
Johan M. Karlsson, *Ph.D. thesis*, 1992.
ISRN LUTEDX/TETS- -1018- -SE+132P

111 **On Overload Control in Telecommunication Systems**
Christian Nyberg, *Ph.D. thesis*, 1992.
ISRN LUTEDX/TETS- -1019- -SE+140P

112 **Black Box Specification Language for Software Systems**
Henrik Cosmo, *Lic. thesis*, 1994.
ISRN LUTEDX/TETS- -1020- -SE+104P

113 **Queueing Models of Window Flow Control and DQDB Analysis**
Lars Falk, *Ph.D. thesis*, 1995.
ISRN LUTEDX/TETS- -1021- -SE+145P

114 **End to End Transport Protocols over ATM**
Thomas Holmström, *Lic. thesis*, 1995.
ISRN LUTEDX/TETS- -1022- -SE+76P

115 **An Efficient Analysis of Service Interactions in Telecommunications**
Kristoffer Kimbler, *Lic. thesis*, 1995.
ISRN LUTEDX/TETS- -1023- -SE+90P

116 **Usage Specifications for Certification of Software Reliability**
Per Runeson, *Lic. thesis,* May 1996.
ISRN LUTEDX/TETS- -1024- -SE+136P

117 **Achieving an Early Software Reliability Estimate**
Anders Wesslén, *Lic. thesis*, May 1996.
ISRN LUTEDX/TETS- -1025- -SE+142P

118 **On Overload Control in Intelligent Networks**
Maria Kihl, *Lic. thesis,* June 1996.
ISRN LUTEDX/TETS- -1026- -SE+80P

119 **Overload Control in Distributed-Memory Systems**
Ulf Ahlfors, *Lic. thesis,* June 1996.
ISRN LUTEDX/TETS- -1027- -SE+120P

120 **Hierarchical Use Case Modelling for Requirements Engineering**
Björn Regnell, *Lic. thesis,* September 1996.
ISRN LUTEDX/TETS- -1028- -SE+178P

121 **Performance Analysis and Optimization via Simulation**
Anders Svensson, *Ph.D. thesis*, September 1996.
ISRN LUTEDX/TETS- -1029- -SE+96P

122 **On Network Oriented Overload Control in Intelligent Networks**
Lars Angelin, *Lic. thesis,* October 1996.
ISRN LUTEDX/TETS- -1030- -SE+130P

123 **Network Oriented Load Control in Intelligent Networks Based on Optimal Decisions**
Stefan Pettersson, *Lic. thesis,* October 1996.
ISRN LUTEDX/TETS- -1031- -SE+128P

124 **Impact Analysis in Software Process Improvement**
Martin Höst, *Lic. thesis,* December 1996.
ISRN LUTEDX/TETS- -1032- -SE+140P

125 **Towards Local Certifiability in Software Design**
Peter Molin, *Lic. thesis,* February 1997.
ISRN LUTEDX/TETS- -1033- -SE+132P

126 **Models for Estimation of Software Faults and Failures in Inspection and Test**
Per Runeson, *Ph.D. thesis*, January 1998.
ISRN LUTEDX/TETS- -1034- -SE+222P

127 **Reactive Congestion Control in ATM Networks**
Per Johansson, *Lic. thesis,* January 1998.
ISRN LUTEDX/TETS- -1035- -SE+138P

128 **Switch Performance and Mobility Aspects in ATM Networks**
Daniel Sobirk, *Lic. thesis,* June 1998.
ISRN LUTEDX/TETS- -1036- -SE+91P

129 **VPC Management in ATM Networks**
Sven-Olof Larsson, *Lic. thesis,* June 1998.
ISRN LUTEDX/TETS- -1037- -SE+65P

130 **On TCP/IP Traffic Modeling**
Pär Karlsson, *Lic. thesis,* February 1999.
ISRN LUTEDX/TETS- -1038- -SE+94P

131 **Overload Control Strategies for Distributed Communication Networks**
Maria Kihl, *Ph.D. thesis*, March 1999.
ISRN LUTEDX/TETS- -1039- -SE+158P

132 **Requirements Engineering with Use Cases – a Basis for Software Development**
Björn Regnell, *Ph.D. thesis*, April 1999.
ISRN LUTEDX/TETS- -1040- -SE+225P

133 **Utilisation of Historical Data for Controlling and Improving Software Development**
Magnus C. Ohlsson, *Lic. thesis,* May 1999.
ISRN LUTEDX/TETS- -1041- -SE+146P

134 **Early Evaluation of Software Process Change Proposals**
Martin Höst, *Ph.D. thesis*, June 1999.
ISRN LUTEDX/TETS- -1042- -SE+193P

135 **Improving Software Quality through Understanding and Early Estimations**
Anders Wesslén, *Ph.D. thesis*, June 1999.
ISRN LUTEDX/TETS- -1043- -SE+242P

136 **Performance Analysis of Bluetooth**
Niklas Johansson, *Lic. thesis,* March 2000.
ISRN LUTEDX/TETS- -1044- -SE+76P

137 **Controlling Software Quality through Inspections and Fault Content Estimations**
Thomas Thelin, *Lic. thesis,* May 2000
ISRN LUTEDX/TETS- -1045- -SE+146P

138 **On Fault Content Estimations Applied to Software Inspections and Testing**
Håkan Petersson, *Lic. thesis,* May 2000.
ISRN LUTEDX/TETS- -1046- -SE+144P

139 **Modeling and Evaluation of Internet Applications**
Ajit K. Jena, *Lic. thesis,* June 2000.
ISRN LUTEDX/TETS- -1047- -SE+121P

140 **Dynamic traffic Control in Multiservice Networks – Applications of Decision Models**
Ulf Ahlfors, *Ph.D. thesis*, October 2000.
ISRN LUTEDX/TETS- -1048- -SE+183P

141 **ATM Networks Performance – Charging and Wireless Protocols**
Torgny Holmberg, *Lic. thesis,* October 2000.
ISRN LUTEDX/TETS- -1049- -SE+104P

142 **Improving Product Quality through Effective Validation Methods**
Tomas Berling, *Lic. thesis,* December 2000.
ISRN LUTEDX/TETS- -1050- -SE+136P

143 **Controlling Fault-Prone Components for Software Evalution**
Magnus C. Ohlsson, *Ph.D. thesis*, June 2001.
ISRN LUTEDX/TETS- -1051- -SE+218P

144 **Performance of Distributed Information Systems**
Niklas Widell, *Lic. thesis,* February 2002.
ISRN LUTEDX/TETS- -1052- -SE+78P

145 **Quality Improvement in Software Platform Development**
Enrico Johansson, *Lic. thesis,* April 2002.
ISRN LUTEDX/TETS- -1053- -SE+112P

146 **Elicitation and Management of User Requirements in Market-Driven Software Development**
Johan Natt och Dag, *Lic. thesis,* June 2002.
ISRN LUTEDX/TETS- -1054- -SE+158P