Improving Requirements Selection Quality in Market-Driven Software Development

Karlsson, Lena

2003

Improving Requirements Selection Quality in Market-Driven Software Development

Lena Karlsson

LUND UNIVERSITY
Department of Communication Systems
Lund Institute of Technology

ISSN 1101-3931
ISRN LUTEDX/TETS--1063--SE+132P

Printed in Lund, Sweden by E-kop

November 2003

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:

Lena Karlsson
Department of Communication Systems
Lund University
P.O. Box 118
SE-221 00 LUND
Sweden

Tel: +46 46 222 03 65
Fax: +46 46 14 58 23
E-mail: lena.karlsson@telecom.lth.se

Abstract

The thesis aims at finding means for assessing and improving the requirements engineering (RE) process in order to enhance software product quality and increase the competitive edge of software organisations. Quality can be defined as the ability to satisfy the customers' expectations. Thus, from an RE perspective, improved quality implies improved requirements elicitation and prioritisation efforts.

The thesis focuses on software-developing organisations that release their product in several releases on an open market. These *market-driven* organisations do not have one single customer to negotiate a contract with, but rather a set of potential customers with various wishes that need to be prioritised.

The thesis is based on empirical research strategies, including both quantitative and qualitative approaches. The research results include a survey of current RE challenges in market-driven organisations in which issues such as lack of communication, turbulent markets and problems with release planning emerged. Process behaviour was explored in an analytical model using queuing theory to model three process phases: screening, evaluation and implementation. The model was validated in a survey. Furthermore, a case study using model building and simulation visualises relations between product quality and lead-time. A post-release analysis method, called PARSEQ, was introduced in a case study, with the intent to evaluate the requirements selection quality in prior releases and find improvement suggestions. Finally, a controlled experiment comparing two requirements prioritisation techniques was conducted, with the result that the simple and intuitive technique was superior regarding time-consumption, ease of use and accuracy.

Acknowledgements

This work was partly funded by the Swedish agency for Innovation Systems (VINNOVA), under grant for the Center for Applied Software Research at Lund University (LUCAS).

First and foremost, I would like to thank my supervisor Dr. Björn Regnell for his great knowledge and support, and for always having time for a chat. Thanks also to Dr. Per Runesson and Dr. Martin Höst for giving me this opportunity and to Dr. Joachim Karlsson for valuable comments on the thesis.

Furthermore, all my other current and former colleagues who have contributed to a great place to work: Lic. Carina Andersson, Lic. Daniel Karlström, Lic. Johan Natt och Dag, Lic. Thomas Olsson, Lic. Enrico Johansson, Dr. Magnus C. Ohlsson, Dr. Thomas Thelin, Dr. Håkan Petersson and Lic. Tomas Berling.

Great research collaborators and co-authors include Åsa G. Dahlstedt and Dr. Anne Persson (Skövde University), Patrik Berander and Prof. Claes Wohlin (Blekinge Institute of Technology), Stefan Olsson and Magnus Höglund (Focal Point AB), Torbjörn Söderberg, Stefan Bjurberg and Peter Friberg (Axis Communications AB), Dr. Per Carlshamre (Ericsson), and Lic. Bertil I. Nilsson (Lund Institute of Technology).

Finally, thanks to Mum and Dad for encouraging me to always strive for more and to my brothers for teaching me their homework. And last, but not least, Josef Nedstam - my colleague, friend and love. You make me complete.

Contents

Introduction

The quality of a product can be defined as its ability to satisfy the needs and expectations of the customers (Bergman & Klefsjö, 1994). These needs may be hidden or difficult to communicate and different customers may have different needs. Thus, the quality of a software product is to a large extent determined by the quality of the requirements elicitation and selection decisions, i.e. what requirements that are identified and chosen for a product.

The goal of the research presented in this thesis is to enhance software product quality and increase the competitive edge of software organisations by exploring and discovering means for improved Requirements Engineering (RE). By developing and applying methods and techniques for assessing and improving the requirements process, the software product quality is expected to improve. The thesis describes several approaches to understand, represent and solve software problems, with focus on RE challenges. All of the included papers have an empirical element, and most of the issues that are examined involves industry partners.

Software continually becomes part of more and more products. Still, the area of software engineering is young and immature and software organisations need support in order to be successful (Sommerville, 2001). The intangible and flexible nature of software causes software projects to be overrepresented in project failure statistics. Typical project problems

include lack of functionality, poor quality, budget overruns and missed deadlines (CHAOS, 1994). Even when the product is delivered on time and on budget, a project may fail if the product does not meet customer expectations. Therefore, we see a large range of potential improvements in the area of software engineering.

Many software developing organisations release their products on an open market with numerous potential customers and users. The thesis focuses on such *market-driven* organisations, which experience special challenges not recognised by organisations that develop customer-specific products. Satisfying customers' expectations is much more difficult when the customers are less well-defined and have diverse opinions.

The thesis includes five papers. The first one investigates the special challenges experienced by market-driven organisations and provides a basis for the research questions which were explored in the other four studies.

1. A survey of RE challenges in market-driven organisations
2. A simulation study visualising the relations between product quality and lead-time
3. An analytical model illustrating the relation between product quality, process capacity and lead time
4. An introduction of a method for evaluating how well the requirements are selected in market-driven organisations
5. An experiment that describes a comparison between requirements prioritisation techniques

The thesis is structured as follows. Section 1 presents the research focus and research questions examined in the thesis. Section 2 continues with a description of the research area of market-driven RE and requirements selection quality. Research methodology and validation issues are described in Section 3, before the research focus is classified regarding the methodology in Section 4. Finally, the research results and contributions are presented and future plans are described in Section 5.

List of Papers

The five papers presented in this thesis are listed below.

PAPER I: CHALLENGES IN MARKET-DRIVEN REQUIREMENTS ENGINEERING - AN INDUSTRIAL INTERVIEW STUDY.
Lena Karlsson, Åsa G. Dahlstedt, Johan Natt och Dag, Björn Regnell, Anne Persson
Proceedings of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), Essen, Germany, September 2002.

PAPER II: UNDERSTANDING SOFTWARE PROCESSES THROUGH SYSTEM DYNAMICS SIMULATION: A CASE STUDY.
Carina Andersson, Lena Karlsson, Josef Nedstam, Martin Höst, Bertil I Nilsson
Proceedings of the 9th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS'02), Lund, Sweden, April 2002.

PAPER III: AN ANALYTICAL MODEL FOR REQUIREMENTS SELECTION QUALITY EVALUATION IN PRODUCT SOFTWARE DEVELOPMENT.
Björn Regnell, Lena Karlsson, Martin Höst
Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03), Monterey Bay, California, USA, September 2003.

PAPER IV: POST-RELEASE ANALYSIS OF REQUIREMENTS SELECTION QUALITY - AN INDUSTRIAL CASE STUDY.
Lena Karlsson, Björn Regnell, Joachim Karlsson, Stefan Olsson
Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Velden, Austria, June 2003.

PAPER V: SIMPLE IS BETTER? - AN EXPERIMENT ON REQUIREMENTS PRIORITISATION.
Lena Karlsson, Patrik Berander, Björn Regnell, Claes Wohlin
Proceedings of the 3rd Conference on Software Engineering Research and Practise in Sweden (SERPS'03), Lund, Sweden, October 2003.

Related Publications

UNDERSTANDING SOFTWARE PROCESSES THROUGH SYSTEM DYNAMICS SIMULATION: A CASE STUDY.
(same as Paper II)
Carina Andersson, Lena Karlsson, Josef Nedstam, Martin Höst, Bertil I Nilsson
Proceedings of the 1st Conference on Software Engineering Research and Practise (SERP'01), Ronneby, Sweden, October 2001.

AN ANALYTICAL MODEL OF REQUIREMENTS SELECTION QUALITY IN SOFTWARE PRODUCT DEVELOPMENT.
(shorter version of Paper III)
Björn Regnell, Lena Karlsson, Martin Höst
Proceedings of the 2nd Conference on Software Engineering Research and Practise in Sweden (SERPS'02), Ronneby, Sweden, October 2002.

MARKET-DRIVEN REQUIREMENTS ENGINEERING PROCESSES FOR SOFTWARE PRODUCTS - A REPORT ON CURRENT PRACTICES.
(based on the same interview material as Paper I)
Åsa G. Dahlstedt, Lena Karlsson, Johan Natt och Dag, Björn Regnell, Anne Persson
Proceedings of the International Workshop on COTS and Product Software (RECOTS'03), Monterey Bay, California, USA, September 2003.

1. Research Focus

The goal of this research is to find means for improving the requirements process and activities in order to enhance software product quality and increase the competitive edge of software organisations. The main research questions that have been investigated are:

RQ1. Which are the RE challenges that need further investigation?

- Which are the current RE practices, concerning release planning and requirements identification and specification?
- How can the increased understanding of the nature of RE help us to identify new research questions?

RQ2. Can process modelling be used to visualise and communicate RE process behaviour?

- How can a process model aid in resource allocation and project planning?
- How can a process model help when reasoning about decision quality and the relations between product quality, capacity and project lead-time?

RQ3. How can requirements selection quality and release planning be assessed and improved?

- Can retrospective analysis and prioritisation techniques be used to improve requirements selection quality?
- Which are the differences between an elementary and an elaborate prioritisation technique regarding requirements selection quality?

The three research questions represent three main parts of the research, as shown in Figure 1: Problem understanding, Modelling and visualisation and Solution proposals. The first research question was investigated in Paper 1 by conducting a survey of Swedish software-developing organisations. The survey laid the foundation for the research questions and the papers that followed in the two other parts of the research. The second question is examined in Papers II and III, both including model

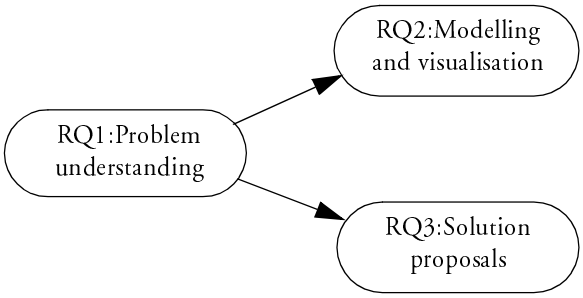

Figure 1. *The main parts of this research*

building, and focus on visualising different process relations. The third research question regards comparing and developing techniques for improved requirements selection quality and resulted in Papers IV and V.

2. Research Area

This section provides some theoretical background of requirements engineering (RE) and describes the context of the five papers.

Software engineering is an engineering discipline whose goal is to cost-effectively develop software systems. This includes all aspects of software development, from the early stages of system specification through to maintaining the system after it is put into use (Sommerville, 2001). *Software requirements* are in SWEBOK (2003) defined as "properties that must be exhibited in order to solve some problem of the real world". In other words, requirements define what the system should do, i.e. what functionality the system shall include. Thus, *requirements engineering* regards the process of identifying, analysing, prioritising, documenting, validating and managing these software properties (Lauesen, 2002).

Software systems are products that consist of both hardware and software, such as embedded products, or products that are pure software applications (Thayer, 2002). Regarding RE, the issues that are dealt with are mainly the same for pure software products and embedded products. When developing embedded products it may not be decided at the RE stage whether to implement functionality as hardware or as software. Therefore, we may benefit from taking a system perspective. However,

from here on we use the term *products* and refer to systems that partly or completely consist of software.

2.1 Requirements Engineering

Traditionally, RE takes place in the beginning of every project, and results in a specification that defines the product to be developed. This view is based on the *Waterfall model* (Royce, 1970), where requirements engineering is followed by design, implementation, testing and maintenance activities. However, this cascade process may not be the most appropriate in practice, since the flexible nature of software enables the development process to be more iterative and evolutionary. The possibility to adjust the product to new wishes and needs is often exploited, which results in a volatile and unpredictable development environment that calls for continuous RE efforts.

The six main activities in RE (Sommerville & Sawyer, 1997) are illustrated in Figure 2. Elicitation, Analysis, Prioritisation, Specification and Validation are often conducted in sequence, while Management is a continual activity and is performed throughout the product life-cycle. However, all these activities are in practice performed iteratively since it is necessary to continually perform elicitation in order to keep up with the customer expectations.

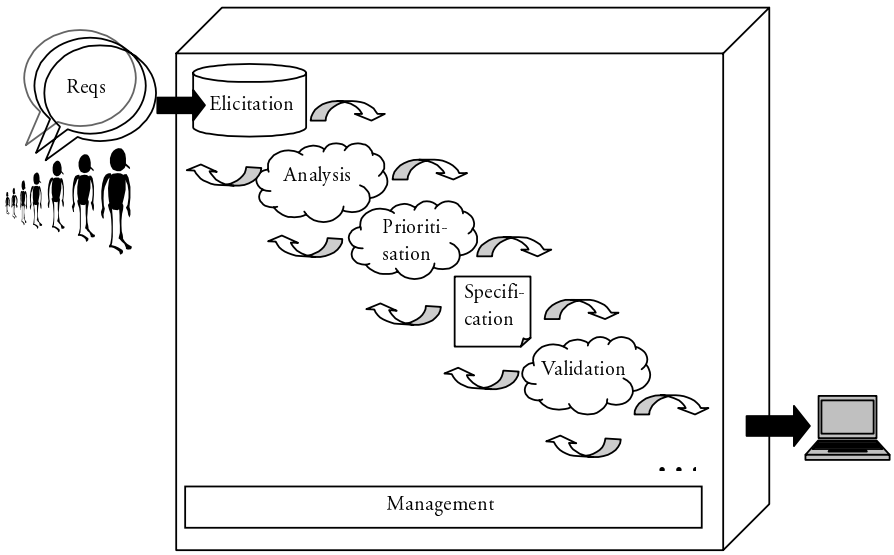

Figure 2. *The requirements engineering process*

- *Requirements elicitation* or identification is the process of finding and formulating the requirements for a software product. There are different techniques applicable to different situations.
- *Requirements analysis* is concerned with investigating the requirements for conflicts, overlaps, inconsistencies etc. It also includes deciding system boundaries and assessing risks.
- *Requirements prioritisation* involves value and effort estimation of the requirements and is the foundation for release planning.
- *Requirements specification* or documentation involves the production of a product specification with requirements and system models.
- *Requirements validation* includes the activities performed to ensure that the requirements follow certain quality criteria, and corresponds to the real user needs.
- *Requirements management* is the process of taking care of all requirements changes and keeping track of interactions between requirements and product documentation.

This thesis focuses on the earlier activities of RE: the identification, analysis and prioritisation of requirements, with the goal of improving these activities and thereby enhance product quality.

2.2 Market-Driven Requirements Engineering

Products can be divided into different categories depending on the type of market where the product is vended. In the *customer-specific* case (also called *bespoke* or *contract-driven*) the product is ordered by a specific customer and the supplier develops and maintains the product for that customer. The customer often represents a large organisation such as a military, governmental or financial institution. A product contract is negotiated with the customer, describing what the product shall include, when it shall be delivered, and how much it will cost.

The case where the product is developed for an open market is usually called *market-driven* development (or *packaged software* development). The customer may be another organisation or a consumer and the products may be, for example, computer packages or mobile phones. In

both cases there is a large range of *potential* customers on a mass market and suppliers need to take diverse needs into account.

Many organisations do in fact deal with both market-driven and customer-specific projects. Depending on the product and situation, an organisation may negotiate contracts with customers for one product at a certain time, while another product is sold on a mass market. In this thesis, we focus on the market-driven aspects of these organisations.

Among the early work that use the term market-driven are Lubars et al. (1993) in their field study of requirements modelling. They investigated differences between customer-specific and market-driven projects in the areas of requirements definition, specification and validation. Later on, Potts (1995) wrote about "invented requirements and imagined customers" as key issues in market-driven RE. Some of the characteristics expressed in those studies and in Carlshamre (2001) and Sawyer (2000) are summarised below.

The characteristics of market-driven RE include, for example, that the mass market product often has a life cycle with several consecutive releases and lasts as long as there is a market for it. Therefore, release planning is an important activity. A highly important issue is to have shorter time-to-market than the competitors, in order to yield high market shares and be successful on the market.

Requirements are often invented by developers or elicited from a set of potential customers since there is no single, almighty customer to ask. This may yield too many requirements with respect to the available resources for one release, and it is necessary to make estimations of implementation effort and market value in order to prioritise and select a set that will fit the market and corporate strategy. Requirements are prioritised within the developing organisation before release planning, while in the customer-specific situation the requirements are negotiated and contracted.

2.3 Requirements Prioritisation

The objective of requirements prioritisation is to aid in the requirements selection and release planning process when a set of requirements is selected based on effort and value estimates. Issues such as requirements interdependencies and product scope are also taken into account and the activity results in a sorted list of requirements.

The prioritisation requires accurate input estimates of cost and value, and may be based on a certain prioritisation technique. There are several prioritisation techniques to choose from, although many organisations still use ad hoc techniques and just pick a set that seems reasonable (Lubars et al., 1993). More elaborate techniques involve structured sorting algorithms or pair-wise comparisons to obtain a sorted list of requirements to use for release planning. This requires the product requirements to be fairly well analysed and complete so that the implications of each requirement are understood.

A common prioritisation technique in industry may be numeral assignment, where each requirement receives a value, for example in the range of 1 to 10 or High, Medium and Low. These values are compiled and point out the most important requirements. However, this kind of absolute judgements are less reliable than relative judgements such as pair-wise comparisons (Karlsson et al., 1996). Another study by Karlsson et al. (1998) examined different sorting algorithms involving pair-wise comparisons. These techniques are rather time-consuming but tend to be more accurate than techniques involving numeral assignment. The study concluded that the Analytical Hierarchy Process (AHP) (Saaty, 1980) was superior but also time-consuming.

In Extreme Programming (Beck, 2000), a technique called Planning Game is used, which is based on categorisation. The requirements are divided into three groups corresponding to different degrees of necessity and risk. Then, a set of requirements is selected based on these groups and the estimated implementation effort. This technique is fast and simple but does not have the benefits of relative judgements.

Thus, there are several techniques that may support decision-makers in the requirements selection process, some more elaborate than others. A comparison of the AHP and the Planning Game is provided in Paper V.

2.4 Requirements Selection Quality

The quality and success of a software product often depends on the ability to make correct decisions regarding requirements selections. Requirements selection quality can be defined as the ability to provide a high-quality set of selected items, i.e. to make correct choices. Requirements selection decisions are a part of the release planning process, which aims at deciding the content of the next release

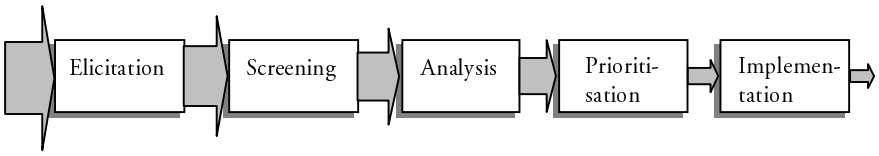

Figure 3. *Requirements selection decisions are made at several process stages*

(Carlshamre & Regnell, 2000). Release plans are often revised and changed through out development as more knowledge is gained about the market and development progress.

The requirements selection decisions are often made in several stages of the RE process. Starting out with a large set of potential requirements, the selection brings the number down for each activity in the requirements process, as illustrated in Figure 3. The discarded requirements are typically stored in a database for investigation in future releases. During elicitation, decisions concern which stakeholder representatives to consult to elicit ideas for new features. Then there is often a screening activity performed to make a first quick assessment to decide whether a requirement is worth spending more time on. Requirements that are clearly out of the scope of the next release are rejected in order to avoid an overload in the requirements repository (Regnell et al., 1998). The analysis activity regards identifying requirements that are in conflict or requirements that are duplicates and can be discarded from the process. During prioritisation, requirements are compared and judged based on, for example, their estimated customer value and implementation cost (Karlsson & Ryan, 1997). Finally, the requirements can be re-assessed during implementation and decisions regarding postponing or removing requirements can be made based on the information gained during the development.

2.5 Project Evaluation

It is often not possible to know whether or not a product is a success, until after it is released to the market. The implementation may have required more effort than expected or the included functionality may not correspond to the customer needs. After a release it is valuable to conduct some kind of *project evaluation* in order to learn from mistakes and improve the process before the next release. The evaluation can be performed in different ways, for example through a *post-mortem project*

evaluation (Smith, 1996). This evaluation or review is usually an open-ended discussion of the strengths and weaknesses of the project plan and execution. Sometimes it is facilitated by an outside consultant or someone with an objective view of the project. At the end of the session, a postmortem report is prepared as a formal closing of the project, which is then used in the project planning stage of future projects. The post-mortem analysis is an excellent method for Knowledge Management, since it captures experience and improvement suggestions from completed projects (Birk et al., 2002; Rus & Lindvall, 2002).

Another type of project evaluation is the *project retrospective* (Kerth, 2000). Here, the emphasis is on constant reflection and improvement, not just after a project is finished, but rather *during* the project. The emphasis is on sharing knowledge and one way is to select, for example, three improvements and focus on them in order to share a common goal throughout the organisation.

3. Research Methodology

This section gives an overview of the different research strategies and approaches that are used in the thesis. It also describes certain validity issues that need to be considered for each strategy.

3.1 Research Strategies

There are two main approaches to research: the *fixed* and the *flexible* design (Robson, 2002). The fixed design, also called *quantitative*, deals with designs that are highly pre-specified and prepared. A conceptual framework or theory is required in order to know in advance what to look for. It is often concerned with quantifying a relationship or comparing two or more groups and the results are often *prescriptive*, i.e. it suggests a solution, method or tool that is more appropriate than others.

The flexible design, also called *qualitative*, relies on qualitative data and require less pre-specification. The design is intended to evolve and develop during the research process as the researchers gain more knowledge. The flexible design is concerned with studying objects in their natural setting and is often *descriptive*, i.e. it describes some issue of the real world. Qualitative data may also include numbers, but are to a large extent focused on words. Many times, however, a design may include

both fixed and flexible methods and yield both quantitative and qualitative data.

This section describes three major types of strategies: Survey, Case study and Experiment. Surveys and Case studies can be both fixed and flexible, while experiments are typically fixed (Wohlin et al., 2000). Pure qualitative research designs include strategies such as Grounded theory and Ethnography, which are not included in this thesis but are described by Robson (2002).

Surveys

Surveys can be both flexible and fixed, i.e. include different degrees of pre-specification. Surveys is a wide term that includes everything from open-ended interviews to questionnaires with closed questions. While questionnaires can reach a large set of people and provide data that is easy to analyse, there is a risk of low response rates and questionnaires can be prone to misunderstandings. Interviews have higher response rates and the interviewer may explain and clarify questions during the session to avoid misunderstandings. However, there are disadvantages such as high time consumption and that the interviewer may impose a bias (Robson, 2002).

Surveys are often used within social sciences for opinion polls or market research. A representative sample is taken from the population under study, so that the results can be generalised to that particular population. The purpose of surveys is to understand, describe, explain or explore the population (Wohlin et al., 2000).

Case studies

A case study is conducted to investigate a single case within a specific time space and can be either fixed or flexible. The researcher typically collects detailed information on one single project, and different data collection procedures may be applied. Case studies differ from experiments in that the variables are not being manipulated, i.e. the case study samples from variables representing the typical situation. A case study is easier to plan than an experiment, but it is also more difficult to interpret and generalise to other situations (Wohlin et al., 2000).

Experiments

Experiments are used when we want control over the situation and want to manipulate the behaviour. Results are often reported as averages and proportions, thus it is a quantitative design. Experiments involve more than one treatment to compare the outcomes and enable statistical analysis. As other fixed designs, the experiment is theory-driven and requires a substantial amount of conceptual understanding from the start (Robson, 2002).

The design of an experiment should be made so that the objects involved represent all the methods or tools we are interested in. The strength of an experiment is that we can investigate in which situation the claims are true and they provide a context in which certain methods or tools are recommended for use (Wohlin et al., 2000).

3.2 Validity

All research designs and strategies have certain validity issues that need to be considered. In fixed designs there are mainly four types of validity threats that need to be considered: conclusion, internal, construct and external validity. In flexible designs there is a different set of validity issues, which regards description, interpretation and theory (Robson, 2002). In addition to these, there are matters of respondent and researcher bias, in which people involved in the research, deliberately or not, affect the results.

Validity in fixed designs

Some of the validity problems encountered in fixed designs are briefly described here, while more thorough presentations are available in (Robson, 2002; Wohlin et al., 2000).

1. *Conclusion validity* is concerned with the relationship between the treatment and the outcome. We want to make sure that there is a statistical relationship, i.e with a given significance.

2. *Internal validity* is needed to make sure that the relationship between the treatment and outcome is a causal relationship, i.e. that the treatment actually caused the result.

3. *Construct validity* is concerned with the relation between the theory and the observation and refers to the extent to which the experiment setting actually reflects the construct under study. Using multiple strategies to measure the same thing may improve the construct validity and ensure that the result is an effect of the treatment.
4. *External validity* regards generalisation. If there is a causal relationship between the cause and effect, can the result be generalised outside the context of our study?

Validity in flexible designs

The three main validity issues in flexible designs are described here. A more detailed presentation is available in Robson (2002).

1. *Description* is regarding the accuracy and completeness of the data. Thus, if interviews occur they should be audio-taped and possibly transcribed in order to keep all data for reference and analysis.
2. *Interpretation* implies that frameworks and theories shall emerge from the knowledge gained during the research, instead of being predetermined and thus biased. This is prevented by analysing and demonstrating how the interpretation was reached.
3. *Theory* is closely related to interpretation, but regards the threat of not considering alternative explanations of the phenomena under study. This is confronted by continuously revising and refining the theory until it accounts for all known cases.

4. Research classification

This section presents how the research questions and the research designs described earlier are represented in the different papers, see Table 1. It also depicts the specific validity measures taken in each study.

Table 1. *Research classification*

Research Question	Research Strategy	Research Design	Paper
RQ1	Survey	Flexible	Paper I
RQ2	Case study	Fixed	Paper II
	Survey	Fixed	Paper III
RQ3	Case study	Flexible	Paper IV
	Experiment	Fixed	Paper V

4.1 RQ 1: Which are the RE challenges that need further investigation?

The first research question was investigated by conducting a flexible survey at five different Swedish companies involved in market-driven development. Seven respondents participated in interviews, which were recorded and later transcribed and analysed. The interviews were open-ended and had a flexible structure, and took different shape depending on the respondents.

The data was stored both on tape and as printed transcriptions in order to keep data complete and accurate. Thus, the *description validity* is taken into consideration. In most cases two to three researchers conducted the interviews and notes were taken during the interviews. During analysis, three researchers with different research interests dug into the data and made different conclusions that were later discussed. Such discussions from different angles help to ensure that conclusions were not emerged through prejudices, but through the knowledge gained during the study. Thereby, we believe that the *interpretation validity* is regarded.

The five companies were of different size and age, and from different business areas, and at two of the companies, interviews were held with two people in different organisational positions. In that manner, we believe that the gained knowledge is multifaceted and the results reflect a broad image of the reality. The selection of companies from different categories of age, size and business were made with the intent to regard *theory validity*.

Since the study is based on a flexible design we do not intend to generalise the results to a larger population. The intention with the survey

was to gain understanding of RE in order to formulate different research hypotheses, which were then further investigated in other research studies.

4.2 RQ 2: Can we use process modelling to visualise and communicate RE process behaviour?

The second question regards process issues such as product quality, process capacity and project lead-time, and was first investigated in a case study where we used simulation as a means for visualisation. The modelling and simulation was rather fixed in its nature since it required a lot of background knowledge and preparation.

The main validity matter in case studies is *external validity*. In this case, the simulation model and the factors included in the model are rather general, and can probably be used by other organisations. However, the model may need some adaptation in order to be calibrated to other specific cases. The study did not involve any kind of treatment (for example introduction of a method or tool) and therefore there are no threats to internal validity.

Secondly, an analytical model was built, illustrating the requirements selection process. The model takes different issues into account, such as selection quality in different phases, work capacity, time-to-market and final product quality. The analytical model was validated in a survey, in which a questionnaire was filled out by several practitioners, such as product managers and requirements engineers.

The main validity issues in fixed surveys is *construct validity* and *external validity*. If the questions in the questionnaire are misinterpreted or not understood by the respondents, the construct validity is challenged. In this case, the questionnaire included more questions than parameters, and therefore it was possible to check if the parameters were consistently estimated. Thus, we believe that the answers of the consistent respondents reflect the questions that were asked. *External validity* is problematic since it is impossible to ask the whole population, and thus the sample might not be representative. In this case, we used a *convenient sample* (Robson 2002) hence the result may not be possible to generalise to a larger population. However, the intention was to validate the model and not to make generalised conclusions.

4.3 RQ 3: How can requirements selection quality and release planning be assessed and improved?

The third research question was investigated in two different studies. First a case study was conducted, which was rather flexible and took place during a one day session. A post-release analysis method, called PARSEQ, was applied and it was elaborated in cooperation with the participating organisation. The results were compiled and analysed afterwards.

In flexible case studies, there are mainly the same validity issues as in flexible surveys. However, since it regards one single case the threat to *theory validity* is handled by creating alternative explanations within that case, for example by asking different persons within one organisation. In this case, three staff members from the small organisation participated, one marketing person and two developers. Audio-taping the one-day conversation would have been impractical as a means for data collection, and instead extensive notes were taken to regard *description validity*.

Then the question was examined in a controlled experiment where 16 academics participated in a comparison of requirements prioritisation techniques. The design was fixed, i.e. prepared and well-defined and the experiment itself was conducted during a half-day session and was followed by several days of analysis.

Experiments are fixed in nature and applies to all four validity issues earlier. *Internal validity* is considered by isolating the treatment from other influencing factors to ensure that the outcome is actually caused by the treatment. A typical example of threats to internal validity is that the groups given different treatments already differ from each other in one way or another. This was regarded by sampling the subjects based on a pre-test so that the groups' characteristics were as similar as possible and additionally the subjects were given treatment in different orders.

Conclusion validity was regarded by conducting hypothesis tests when possible and indicates significance of the results. The experiment was performed with a rather small and specific set of subjects, hence there is a threat to *external validity*. However, we believe that the subjects' (in this case mainly Ph D students) views are rather similar to the practitioners' who are intended to use the techniques examined in the experiment (Höst et al., 2000).

5. Research Results

This thesis uses several research strategies to investigate and solve questions regarding requirements selection quality: a survey, an analytical model, a simulation study, a proposed evaluation method and an experiment. In this section, the main research results and contributions are summarised and plans for further work are described.

5.1 Main Contributions

The research presented here has lead to the conclusion that many organisations find RE difficult. Among the issues that were discovered in the survey (Paper I) the most interesting ones regard communication problems, difficulties with decision-making, and lack of reflection due to time pressure. However, these are not specific to RE but the contributions that are presented here aim at reducing these problems with an RE perspective.

Communication problems may be decreased by using models to reason about RE issues and try out different possible solutions in order to come to an agreement. Such models are presented in Papers II and III. Making appropriate decisions regarding, for example, release plans, requires good estimation skills and domain knowledge but also techniques to aid in the selection of requirements. Two such techniques are compared in Paper V. Project management literature often stresses the importance of reflection of prior work in order to learn for the future. However, few organisations actually perform such post-mortem analyses. Paper IV proposes such an analysis aiming at reflecting on the requirements decisions made during prior releases.

This research can hopefully help organisations improve their communication and decision-making with the goal to increase product quality.

Report on current RE challenges (RQ 1)

The first research contribution is the industrial survey conducted at five Swedish software-developing companies, which is described in Paper I. The survey gave insights into how requirements engineering is handled in the studied organisations, and the challenges they face. It generated research questions and laid the foundation for the studies that followed.

The survey presents typical challenges managed by market-driven software organisations and can be used by other organisations to review, and avoid, the traps that the surveyed organisations have fallen into. It also provides the foundation for conducting a larger survey of requirements challenges in software organisations.

Some of the challenges that were encountered are special to market-driven organisations, for example that the requirements change constantly during development due to market fluctuations. A turbulent market also forces new demands on the requirements specification. It can no longer be a pre-defined document, but has to be flexible to cope with constant changes. Another challenge appears when more requirements than can be managed are proposed to the organisation, since it creates an overload in the requirements repository. These challenges probably originates from the flexible and complex nature of software.

Other challenges are more ubiquitous and also faced by organisations that develop physical products. These challenges include for example lacking communication between departments, which may cause requirements to be misunderstood. There is also a constant trade-off between cost and value and between requirements that correspond to perceived user needs and requirements that are inventive and technology-driven (Ulrich & Eppinger, 2000).

Modelling and visualisation of factors that affect quality, capacity and lead-time (RQ 2)

The second contribution is described in two papers. Paper II describes a simulation model that was built in a case study at a large Swedish company. It illustrates the relationship between different factors that affects the project lead time and product quality. It also describes a relationship between the requirements phase and the test phase. It can help project managers in project planning and allocation of resources to different development activities and act as a foundation for discussion. It would be possible to adapt the simulation model to other organisations using organisational-specific parameters.

Paper III describes an analytical model of a requirements selection process which can be used to reason about decision-quality and investigate hypothetical changes in process parameters. Given certain organisation-specific parameters, it is possible to calculate for example the minimal capacity in different phases, mean time-to-market and average

share of all project effort put on RE. The model is intended to be used for making a diagnosis of the organisation's current state of RE and to provide a platform for testing improvement proposals.

Methods for assessing and improving the requirements selection quality (RQ 3)

The third contribution is also investigated in two papers. Paper IV presents a novel post-release analysis method (PARSEQ), which was applied in a case study, aiming at investigating the requirements selection quality and finding process improvement proposals. The method can help organisations gain knowledge of how to improve decision-making by looking at decision outcomes in retrospect. The improvement proposals that emerged included improving estimations of development effort and competing products' market-value as well as enhancing the overall picture of related requirements and trimming the division of large requirements into smaller increments. However, the benefit is not only the resulting improvement proposals, but also the opportunity to reflect, discuss and learn from other members of the organisation before entering another release cycle.

Paper V regards requirements prioritisation, which is an important means for improving requirements selection. Two prioritisation techniques, the elaborate Analytical Hierarchy Process and the elementary Planning Game, were compared in an experiment regarding time-consumption, ease of use and accuracy. It resulted in the elementary Planning Game technique appearing as superior regarding all three aspects. However, the conclusion was that a combination of the two techniques may be even better in order to make use of the techniques' different benefits.

5.2 Further Research

This section describes how the research can be continued and evolved in the future. All five papers have possibilities of deeper investigation, which is further detailed in the different papers. The research is intended to continue with basically the same focus as in this thesis. The main goal is to provide methods and techniques that may improve product quality and therefore the plan is to continue and intensify the studies performed so

far. Below is a description of four research studies of interest. They are presented in the order of precedence, although they will probably be conducted in parallel to some extent.

FR 1: Post-release analysis in industry

We intend to apply the PARSEQ method in additional organisations to examine its possibilities and lay the ground for finding more general improvement areas. Each organisation's decision quality will be assessed and possibly compared. A set of general root cause categories may be compiled and, if possible, some common improvement proposals may emerge. The method can be extended to also include rejected requirements to investigate if some of them are rejected based on incorrect decisions. It would also be interesting to investigate the limitations of the method in order to develop and expand it to a more general method that can easily be applied on organisations in different areas and with different processes.

FR 2: Deeper analysis of survey results

RQ 1 is difficult to investigate in full, and Paper I only takes a first step towards creating an understanding of requirements challenges in industry. We intend to do further investigation in an enlarged interview study with, 12-15 interviews (including the interviews already conducted). This would result in a broader spectrum of interviewees and organisations and thereby a wider range of RE activities and challenges.

All interviews will be transcribed and thoroughly analysed in a qualitative analysis tool. Since Paper I was written, a focus group meeting have taken place which aimed at discussing the issues that emerged during the first set of interviews. The results from the focus group meeting revealed a set of additional challenges, which may be taken into consideration in the next stage of the study.

FR 3: Simulating the analytical model

The analytical model described in Paper III would benefit from a more dynamic approach and therefore we intend to create a simulation model and investigate it in a case study. The simulation will be used to try out and evaluate possible changes to the process. The analytical model is

limited by its static characteristics and there are several aspects of reality which a simulation model could take into account, such as requirements dependencies, and requirements decomposition and bundling.

The model structure will be used as the basis for a discrete-event simulation model where the three process activities screening, evaluation and implementation are illustrated as three queues. Model parameters can be derived from empirical data within an organisation using questionnaires and interviews to capture subjective assessments and, when available, archive analysis to obtain quantitative data. Paper II and Höst et al. (2001) will act as valuable input to the model building and simulation.

FR 4: Further experimentation on prioritisation techniques

The experiment presented in Paper V will be performed on a larger number of subjects in order to increase the data set. It can be conducted within an RE course, using students as subjects. Since Paper V indicates that the results do not differ depending on the number of requirements in the prioritisation, it may be sufficient to use a rather small set of requirements in order to ease the subjects' burden. Furthermore, we would like to try a combination of the two techniques in practice by, for example, conducting a case study.

Time frame and priorities for continued research studies

The main focus in the PhD thesis will be on the post-release analysis method and its refinement and adaptation. It can be tried out in several case studies, each one during one-day sessions and a number of weeks of analysis. The survey expansion will be rather extensive and time-consuming and therefore it will be conducted over a large period of time and in cooperation with other researchers. The simulation of the analytical model will be performed in master's project and therefore we expect to have a running simulation model within a year. However, the project may not include performing the case studies and adapting the model to specific organisations. The experiment on prioritisation techniques may be performed in a few months, although the analysis of the experiment is expected to be rather time-consuming. However, a case study with a combination of techniques may be performed at a later stage.

6. References

CHAOS report (1994) http://www.standishgroup.com/sample_researchchaos_1994_1.php (visited 2003-09-20)

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

Bergman, B., Klefsjö, B. (1994) *Quality from Customer Needs to Customer Satisfaction*, Studentlitteratur.

Birk, A., Dingsoyr, T., Stålhane, T. (2002) "Postmortem: Never Leave a Project without It", *IEEE Software*, pp.43-45, May/June.

Carlshamre, P., Regnell, B. (2000) "Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes", *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, pp. 961–965.

Carlshamre, P. (2001) *A Usability Perspective on Requirements Engineering – From Methodology to Product Development* (Dissertation No. 726), Linköping University.

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*, Vol 59, pp. 323-332.

Höst, M. Regnell, B., Wohlin, C. (2000) "Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment", *Empirical Software Engineering*, vol. 5, pp. 201-214.

Karlsson, J., Ryan, K. (1997) "A Cost-Value Approach for Prioritising Requirements", *IEEE Software,* pp. 67-74, Sept/Oct.

Karlsson, J. (1996) "Software Requirements Prioritising", *Proceedings of 2nd International Conference on Requirements Engineering (ICRE)*, pp. 110-116.

Karlsson, J., Wohlin, C., Regnell, B. (1998) "An Evaluation of Methods for Prioritising Software Requirements", *Information and Software Technology,* Vol 39, pp. 939-947.

Kerth, N.L. (2000) *Project Retrospectives: A Handbook for Team Reviews,* Dorset House Publishing.

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

Lubars, M., Potts, C., Richter, C. (1993) "A review of the state of the practice in requirements modelling", *Proceedings of IEEE International Symposium on Requirements Engineering,* pp. 2–14.

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

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

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

Royce, W. W. (1970) "Managing the development of large software systems: concepts and techniques", *Proceedings of IEEE WESTCON*, pp. 1–9.

Rus, I., Lindvall, M. (2002) "Knowledge Management in Software Engineering", *IEEE Software*, pp.26-38, May/June.

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

Sawyer, P. (2000) "Packaged Software: Challenges for RE", *Proceedings of Sixth International Workshop on Requirements Engineering: Foundation for Software Quality* (REFSQ 2000) pp. 137–142.

Smith, P. G. (1996) "Your Product Development Process Demands Ongoing Improvement", *Research Technology Management*, Vol. 39, March/April.

Sommerville, I. (2001) *Software Engineering*, Addison-Wesley.

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

SWEBOK, http://www.swebok.org (visited 2003-09-25)

Ulrich K.T., Eppinger, S.D. (2000) *Product Design and Development*, McGraw-Hill.

Thayer, H. (2002) "Software System Engineering: a Tutorial", *IEEE Computer*, pp. 68-73, April.

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

PAPER I

Challenges in Market-Driven Requirements Engineering - An Industrial Interview Study

Lena Karlsson, Åsa G. Dahlstedt, Johan Natt och Dag, Björn Regnell, Anne Persson

Proceedings of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), Essen, Germany, September 2002.

I

Abstract

Requirements engineering for commercial off-the-shelf software packages entails special challenges. This paper presents preliminary results from an empirical study investigating these challenges through a qualitative approach using semi-structured interviews. The survey is exploratory with the objective of eliciting relevant topics for further research. Seven employees at five software companies with a market-driven development focus were interviewed. The areas of interest include process-related issues on release planning, requirements quality and decision support, as well as artefact-related issues regarding requirements as discrete entities and their representation. The paper also contains a characterization of each company, regarding aspects such as products, processes and customers. A number of challenging issues were elicited, including communication gaps between marketing and development, the problem of balancing the influence between marketing and development on requirements decisions, as well as the limited value of monolithic requirements specifications and the problem requirements overloading.

1. Introduction

This paper reports on preliminary results from the first stage of an industrial survey, focusing on current practice and challenges concerning requirements engineering (RE) in Swedish software industry. Before problems related to inefficient RE can be properly addressed, more research is needed to better understand the challenges that the software industry is currently facing. This survey aims at discovering hypotheses for future research and complementing existing surveys in the area of RE.

The study focuses on market-driven development, which is currently gaining increased interest compared to development of customer-specific systems. This is due to the emergence of the market for off-the-shelf or packaged software [17, 4]. Market-driven RE differs from customer-specific in several ways. Sawyer [17] concludes that the major differences between market-driven and customer-specific RE are the characteristics of stakeholding and schedule constraints. In market-driven projects, the main stakeholder is the developing organization and therefore requirements are *invented* by the developers, since there are no discrete set of users who can articulate their requirements. There is also a major pressure on time-to-market for these software products. Other characteristics of market-driven RE are release planning and prioritization, and the use of requirements specifications [4, 10, 15].

There are several surveys that concern or include RE issues [5, 6, 7, 8, 9, 13, 14]. However, none of these have a primary focus on market-driven development. Furthermore, in most of these surveys the participating projects and organizations are fairly large, in terms of number of persons and requirements involved, as well as in terms of the duration of the projects. This survey provides characteristics from a number of small, market-driven development companies.

The survey presented in this paper aims at understanding the conditions and practices that characterize modern software industry, particularly with regard to requirements identification, requirements specification, and product management. In this initial survey, seven persons at five different companies participated. The forthcoming stages of the study include workshops with RE experts and an extended interview survey based on these initial results.

The paper includes a characterization of the companies involved, regarding company facts, typical projects and development processes. The result of the study is a set of interesting issues that may indicate the

direction of further research. However, the small number of interviews implies that a larger survey needs to be performed in order to generalize the results and to be able to formulate relevant research hypotheses.

The remainder of the paper is organized as follows. In Section 2 the research method is described. Section 3 presents the results uncovered through the study. Section 4 concludes the paper and discusses future work and further research issues.

2. Research Method

The study was carried out using a qualitative interviewing approach has been used. This approach is useful to explore an area of interest, to obtain an overview of a complex area, and to discover diversities rather than similarities [16]. As the purpose of the study is to gain an improved understanding of the nature of requirements engineering within market-driven software companies, the qualitative approach is considered suitable. Furthermore, interviewees in different roles from both large and small companies were interviewed in order to collect different viewpoints and perspectives on the nature of RE.

The research procedure is illustrated in Figure 1. The initial stage of the study involved a brainstorming and planning meeting to identify different areas of interest and plan the study.

In order to gain an insight into the area of market-driven requirements engineering, aim at interviewing a large number of software-developing companies. However, it was concluded that this initial stage of the study would benefit from selecting a handful of companies before adjusting the interview instrument and carry out the full study. Therefore seven interviewees within five Swedish software companies were asked to participate. The interviewees were selected among our industrial partners.

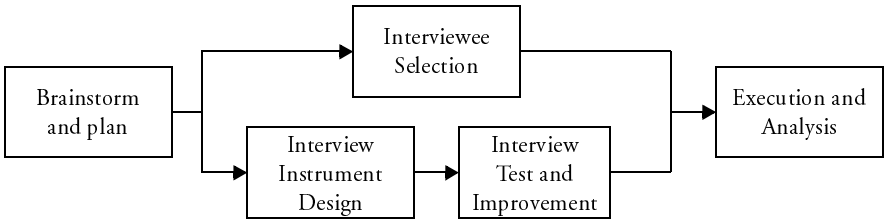

Figure 1. *The research procedure used in the presented interview study*

The companies all have a market-driven development focus. They have had at least one market release of a software product or are just about to release their first.

The interview instrument was designed with respect to the different areas of interest and with inspiration from other requirements engineering surveys [13, 7]. The questions in the interview instrument were divided into three groups in order to give a structure to the interview:

- *Characterization* – facts about the company, its products, and the interviewee.
- *Process issues* – questions on the development procedure, requirements activities, and decision support.
- *Artefact issues* – questions on requirements as entities, how requirements are documented, tool usage, and requirements interdependencies.

To test the interview instrument, two pilot interviews were carried out. Some questions were clarified and the structure was improved before proceeding. A summary of the interview instrument is available in Appendix A.

The study has a semi-structured interviewing strategy, where the set of questions were the same for all interviewees [16]. However, the order of the questions varied depending on the interviewees' knowledge and role in the company. This meant e.g. that some issues were discussed more in-depth with certain interviewees. In order not to steer the interviewee, additional questions were asked depending on the interviewee's answers until all areas of interest were covered.

All interviews involved one interviewee and three interviewers. One of the three interviewers ran the interview process while the other two posed additional questions, in order to cover all areas of interest. The duration of the interviews was 90 to 150 minutes. All interviews were recorded on tape and extensive notes were taken in order not to lose information.

Afterwards, the interviews were transcribed in order to facilitate and improve the analysis. The size of the transcripts was 7 to 23 pages of text depending on the length of the interview.

The analysis was performed mainly through marking and discussing interesting sections in the transcripts [16]. The interviewers examined the transcripts from different perspectives and searched for explicitly stated or concealed RE challenges. Another researcher, who did not attend the

interviews, also analysed the transcripts to ensure the quality of the study and to enhance confirmability [12].

2.1 Validity

It is almost impossible to achieve an unbiased interview analysis; all interviewers have expectations and interpretations affecting the analysis of the interviewees' answers. Furthermore, notes and transcripts may be insufficient, since accentuations and gestures are difficult to reproduce. The analysis hence relies on the researchers' perceptive skills and good memory. Recording the interviews on video-tape would cover these details and probably improve the analysis. In our study, using three interviewers with different background and experience has reduced the risk of bias and misinterpretations.

The results presented in the paper are first and foremost issues and suggestions for further research, which have surfaced during the analysis of the interviews. However, there may be several other important issues in the transcripts, not yet discovered.

Evidently the interviews only capture the subjective opinion of each interviewee. A larger number of employees should be interviewed to capture the general view of each company. We have chosen companies of different size and maturity to manage the issue of transferability [12]. However, we do not attempt to generalize to a larger population, but merely to discuss some interesting issues discovered during the interviews, and present some hypotheses for future research.

3. Analysis and Results

This section presents some interesting issues discovered during the analysis of the interviews. Each issue is concluded by a conjecture to summarize our thoughts on the subject. These conjectures may be topics of further research.

Section 3.1 corresponds to the characterization questions in the interview, and includes other facts to widen the description of the companies. Section 3.2 corresponds to the process issues and discusses some issues concerning the development process. Section 3.3, finally, deals with the artefact issues in the interview instrument.

3.1 Company characterization

Table 1, which summarizes the characteristics of each company, is inspired by [18]. The companies' age range from 1 to 20 years. The companies also vary in size, from 12 employees to 1200, which means that the results include challenges for both large and small companies.

Two managing directors and one product leader were interviewed to gain the managers' view, and three project leaders and one head of developers were interviewed as development representatives. The responsibilities of the project leaders differ. In the small companies, project leaders tend to have more overall responsibilities such as process introduction and business analysis. In contrast, the large company uses a project leader to co-ordinate resources and work with requirements. All companies use natural language on a high level of abstraction to specify requirements. Two companies use UML at a later stage for modelling the system, one of which also creates state charts. In three of the five companies, databases are used to collect suggestions from users and developers.

Only two of the companies have a well-defined, elaborate process with defined phases and regular evaluation and improvement efforts.

3.2 Process issues

This section addresses questions on the development process and development decisions. It also focuses on the requirements engineering procedures and activities. These questions revealed some interesting issues concerning the process and the challenges companies face in requirements engineering.

Living with changing requirements. Market-driven projects do not have a defined customer. However, customer representatives and retailers may be employed to validate the product. Challenges these companies face are for example that the market changes, competitors improve and customers are not certain of their requirements. Company C and D use a strategy to develop functions to only 60-90 % and then release a beta-test, because, as the project leader at Company D states, "customers will always change their minds". In this manner, it may be possible to receive customer opinions earlier and with less effort spent on the project.

Conjecture: *Requirements are volatile and it is necessary to find methods to cope with changes. Although these market-driven companies do not have a single well-defined customer, it is important to obtain feedback from potential customers. Beta-test releases make it possible to receive customer feedback at an early stage.*

Market-driven vs. technology-driven requirements. The interviewees at Company B and C estimate their requirements sources to be 50 % users and market department and 50 % internal developers. The project leader at Company B concludes that some developers see themselves as "artists" who enjoy coding but do not reflect on whether the code solves customer problems or not. The same interviewee identifies a problem in the company focus; since the company is new and inventive the focus has been on development and coding instead of marketing and sales. Therefore there have been difficulties when introducing the products to an open market, since the products do not live up to the users' expectations. On the other hand, too much marketing may result in unrealistic requirements, which are not possible to implement with the resources available, and hence new and creative requirements may be ignored.

Conjecture: It is necessary to find good trade-off between requirements corresponding to perceived user needs and new, inventive ones that may provide a competitive advantage. The decision regarding which aspect to focus on may depend on the maturity of the market. To succeed on a stable market it may be necessary to create technical inventions, while on an immature market it is important to satisfy customer needs.

Gap between marketing staff and developers. When interviews were held with two persons at the same company, some different views were discovered. For example, in Company C, the Managing Director claims that it is not difficult to understand requirements written by different people, while the head of developers regards that particular issue to be the greatest challenge in his daily work.

There are also different views with regard to what is a "good requirement". From the company perspective, represented by the Managing Director or marketing, a good requirement is something that makes money for the company. In contrast, project leaders and developers focus on the way requirements are described, for example completeness and understandability.

	Company A	Company B	Company C	Company D	Company E
Total number of employees	1200	65	13	15	12
Age	20	5	5	2	1
Product in focus	Software development tool.	Embedded software with focus on image processing. 3 product lines.	Software development support tool based on requirements management.	Communication tool for distributed groups working with software development.	Software development and visualization tool for integration of electronic business processes.
Number of employees involved in the product in focus	80	65 (in all 3 product lines)	13	7	12
Customer	Software developing companies, mostly in the telecommunication industry.	Consumer electronics retailers and distributors.	Software developing companies.	Safety critical software developing companies.	Large companies that need an improved business process.
End user	Developers at software-developing companies.	Bankers, students, office personnel.	Developers and managers at software developing companies.	Managers and developers at safety critical software developing companies.	Software developers that integrate electronic business processes
Interviewee(s)	1. Requirements engineer and project leader	1. Project leader 2. Product manager	1. Founder and managing director 2. Head of developers	1. Project leader	1. Founder and managing director
Role/responsibility of the interviewee(s)	1. Co-ordinate resources and clarify, define and prioritize requirements.	1. Introduce process and break down the requirements specification. 2. Coordinate between marketing, development and production. Suggest and write requirements.	1. Release planning, marketing and sales. 2. Allocate work between developers. Suggest, document and time-estimate requirements.	1. Process responsibility and business analysis.	1. Marketing, customer contacts, and release planning.

	Company A	Company B	Company C	Company D	Company E
Projects	New release every 6 months to include new requirements and correct detected errors. Involves all 80 employees.	Includes 25-30 employees for a new product line and 5-15 employees for a new version of existing product. Initiated because of new requirements or when many errors are detected.	New release when enough requirements or errors have been added in the database, approximately every 6 months. Divided into functions involving 1-3 employees for 1-10 weeks.	3 parallel projects involving all 7 employees at 2 locations. At least one release every month.	No regular releases, puts together a project including all employees when many bugs or requirements have been identified.
Requirements documentation	Natural language. Support system database where customers and developers add requirements. A web site with the ones for the current release written on a high abstraction level.	Natural language, state charts and UML in the requirements specification. Changes to the specification are rarely documented.	Natural language. Database included in the tool where customers and employees can add requirements.	Natural language. Customers' requirements suggestions on virtual story cards, which can follow through the process.	Natural language in the requirements specifications and UML at a later stage.
Process	Defined but not documented, followed thanks to the experienced staff.	Introducing process. The ad hoc process works thanks to the experienced staff.	Elaborate, documented and integrated in the tool they develop and use. Based on requirements status.	Elaborate. Extreme Programming, pair programming excluded.	Not documented, but the experienced staff knows the activities by heart.
Project or process evaluation	Not regularly. Have increased requirements awareness. Earlier prioritization in categories Must, Wish, and Postpone. Decreased database size.	Not yet in use. Have increased systems thinking and awareness of requirements and testing.	Included in the process. Occurs after each release. Have adjusted the status range.	Weekly meetings to sum up the week's problems. Individual time surveillance.	Strong process focus in the company thanks to the process focus in the tool.

Furthermore, the project leader at Company A describes a gap between marketing staff and developers. The marketing department's task is to write requirements, which in their opinion means writing down ideas for the next release. Concurrently, developers expect the requirements to be written down clearly enough to start coding.

Conjecture: When marketing and developers have different views on requirements engineering, better communication and more collaboration may decrease the gap. A reduced gap may increase the requirement quality and thereby the quality of the final product.

Elaborate vs. elementary processes. Two of the new and small companies express no need for an elaborate process. In fact, the product leader at Company B expressed a wish for "a simple tool", which may be interpreted as a wish for "a simple process". The project leader at the same company mentioned that many of the developers are reluctant to introduce a process, because they believe that it would limit their freedom and creativity. Others, working with an elaborate process, for example the head of developers at Company C, find the process valuable since it structures the work and all employees are aware of their respective responsibilities.

Conjecture: It is difficult to balance between elaborate and elementary development processes. Opinions differ on whether the process limits or simplifies development. The necessary degree of elaboration depends on the maturity of the company. Immature and small companies may succeed with an elementary process while more mature companies need an elaborated process to accomplish their goals.

Organizational stability and market turbulence. Companies that use an ad hoc process still manage to get sufficient results, to a large extent due to the skills of their staff. A low staff turnover and skilful developers are needed to survive. "The project's success depends on the individuals" was mentioned in Company B, which had recently gone through a large downsizing process. Companies without a defined process take a significant risk if key persons leave the organisation, since they lack the necessary documentation and structure. "Some projects would die if certain people leave" is another quotation from this interviewee.

Conjecture: A low staff turnover will help companies succeed even in times of instability and business environment turbulence, since the knowledge remains with the skilful developers in the company.

Relation between time estimates and release plans. Release planning decisions are mostly taken by marketing or management departments. The project leader at Company A claims that the release plan is very much dependent on accurate time estimates, since the estimates affect how many of the requirements that are selected. This is especially important in companies with regular releases, since under-estimation may result in an exceeded deadline while over-estimation may exclude valuable requirements. Time estimates are mostly performed by developers, since the requirements have to be carefully examined and partitioned in order to make a reasonable estimate.

Conjecture: The better time estimates, the better requirements and products. Improved communication between marketing and developers may increase release planning quality.

3.3 Artefact issues

Artefact issues deal with requirements as entities and include questions on the documentation of requirements, requirements interdependencies and tools for requirements engineering. This section discusses some artefact issues that emerged during the interview.

Traditional requirements specifications. Market-driven development companies have to deal with a steady stream of new requirements as well as select an optimal part of these requirements to implement in the next version [2]. Among the interviewed companies, two use a traditional requirements specification to structure and store the requirements prior to every new project. It is interesting to note that these two companies are also the ones with the least elaborate and structured RE process.

The companies with more elaborate RE processes manage their requirements as single units, one by using Extreme Programming story cards [1] and the other by using their own requirements management tool. In both cases, the requirements are continuously managed, time estimated and prioritized.

Conjecture: The classical, monolithic requirements specification is best suited in cases where the requirements are reasonably few and sufficiently stable. In the market-driven situation, the requirements specification is of limited value when managing a steady stream of incoming requirements of varying quality.

Requirements overload. Company A has had substantial problems with an overload in the requirements database, since more requirements were added than could be managed [11]. This resulted in difficulties when prioritizing requirements for the next release, since there were thousands of requirements to consider. The problem has been temporarily solved by the requirements manager through creating a "TOP 10-list" with the ten most important requirements for the next release. There is an apparent risk that the selected requirements are not the most important ones, and that some of the requirements left in the database should have been included instead.

There is also a risk that customers and developers, using the database, feel neglected since they are not given any feedback on their requirement suggestions. The project leader at Company D stresses the importance of customer feedback, because "if they are neglected they never come back".

Conjecture: Requirements suggestions from developers and customers are essential. However, too many requirements suggestions complicate release planning. Furthermore, the important feedback to those who suggest requirements may be overlooked. There is a need for a method to prevent databases from being flooded with requirements in order to facilitate release planning.

Simple techniques for basic needs. All the interviewed companies use natural language to define their requirements. Two of the companies use UML at a later stage and one of these creates state charts. The smaller companies have fewer requirements to handle, and therefore do not express the need for CASE tools to manage the requirements. The number of requirements is small enough to be handled manually, typically a few hundred. The main challenges at the small and immature companies are at the moment to understand the requirements and to make all requirements complete, easy to understand as well as to follow a certain standard – not to manage a massive set of requirements. The product leader at Company B requests a simple tool, since the ones considered would have a too large introduction overhead and a too steep learning curve. The head of developers at Company C wants a checklist to control that certain items or attributes are considered when different stakeholders write requirements in different ways.

Conjecture: There is a lack of tools and techniques simple enough to be introduced in small and immature companies. Such simple techniques and tools could increase requirements completeness and understandability.

Requirements bundling. Requirements relate to and affect each other in various ways, which affects for instance release planning [3]. All companies acknowledge this, but few have routines to document and take these relations into account. Instead, they tend to group related requirements, i.e. by bundling requirements which should be implemented together. Bundling is usually made based on the fact that the requirements deal with the same part of the system, e.g. the user interface or the same part of the software code. Requirements bundling is used to facilitate release planning and implementation. Interdependencies, such as conflicting requirements or duplicates, are managed manually when the requirements are planned for a new release.

Conjecture: Dependencies between requirements are managed through bundling. This could imply that the complexity of requirements interdependencies is not well understood. On the other hand it could imply that bundling is sufficient enough to make the decisions needed and that a deeper understanding of requirements interdependencies is not needed.

Living with design in requirements. The nature of requirements differs; some concern the user, some concern the developer. Requirements are expressed at many levels of abstraction, ranging from abstract ones such as "usable" to more detailed ones concerning, for example, memory capacity. Since requirements specifications also often concern the design, it is difficult to draw a clear line between the phases.

Conjecture: Requirements specifications often partly concern design issues and there is a risk that the specification is not flexible enough to create a good design. However, the line between requirements and design is thin and design issues will always concern requirements. The key is to make requirements management flexible enough to handle it.

4. Conclusions and Further Research

This paper reports on interesting issues discovered in the interview study on market-driven requirements engineering. These are divided into three sections: Characterization, Process issues and Artefact issues complying with the interview structure. The issues discussed in this paper are preliminary results of a continuing study, and will be used to find hypotheses for our future research.

Although Curtis et al. [6] performed their survey on large systems development and that it was performed 14 years ago, the presented study indicates that some problems remain. Communication is still a corner stone in software development and the project's success depends on the staff. Another recent survey [8] also confirms that organizational problems, for example lack of skilled personnel and high staff turnover, have a larger impact than technical problems on requirements engineering.

In agreement with Lubars et al. [13], Company C and D manage requirements changes by tracking them through the development to locate the effect of each requirements change. Furthermore, the same companies have a strategy to develop functions to less than 100% in order to get customer feedback earlier.

As a complement to other surveys the presented study describes a communication gap between marketing and developers, resulting in insufficient time estimates and requirements quality. The balance between marketing and developers' requirements is also recognized as a dilemma. The use of a requirements database rather than a traditional, monolithic requirements specification is also salient, as well as the urge to group requirements into bundles to ease requirements structuring and work partitioning.

This paper concludes the first stage of a research project, which is planned in four stages. The next stage is one or more workshops where the results from the interview study are discussed and further validated with a selection of RE experts. Then an extended interview study will be carried out, involving additional companies. And, finally, in order to even further validate, quantify and generalize the results, a quantitative survey is planned, involving a larger number of companies.

Acknowledgement

This work is partly funded by the Swedish Knowledge Foundation (KK-stiftelsen) and the Swedish Agency for Innovation Systems (VINNOVA) under grant for The Centre for Applied Software Research at Lund University (LUCAS), Sweden. We would like to direct a thank to participating interviewees, without whom this paper would not have existed. We would also like to thank Pär Carlshamre for his support during the first stages of this study and Carina Andersson for reviewing the paper.

References

[1] Beck, K., *Extreme Programming Explained,* Addison-Wesley, 2000

[2] Carlshamre, P., *Usability Perspective on Requirements Engineering - From Methodology to Product Development*, Ph D Dissertation No. 726, Department of Computer and Information Science, Linköping University, Sweden, 2001.

[3] Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Natt och Dag, J., "An Industrial Survey of Requirements Interdependencies in Software Product Release Planning", *Fifth International Symposium on Requirements Engineering*, pp. 27-31, Toronto, Canada, August 2001.

[4] Carmel, E., Becker, S., "A Process Model for Packaged Software Development", *IEEE Transactions on Engineering Management*, Vol. 42, No. 1, pp 50-60, 1995.

[5] Chatzoglou, P. D., "Factors Affecting Completion of the Requirements Capture Stage of Projects with Different Characteristics", *Information and Software Technology,* Vol. 39, No. 9, pp 627-640, 1997.

[6] Curtis, B., Krasner, H., Iscoe, N., "A Field Study of the Software Design Process for Large Systems", *Communications of the ACM,* 31(11) :1268-1287, Nov. 1988.

[7] El Emam, K., Madhavji, N.H., "A Field Study of Requirements Engineering Practices in Information Systems Development, *Second IEEE Int. Symposium on Requirements Engineering* (RE'95), pp 68-80, 1995.

[8] Hall, T., Beecham, S., Rainer, A., "Requirements Problems in Twelve Software Companies: An Empirical Analysis", *Proceedings of the Conference on Empirical Assessment in Software Engineering* (EASE), 2002

[9] Hofmann, H.F., Lehner, F., "Requirements Engineering as a Success Factor in Software Projects", *IEEE Software*, pp 58-66, July/August 2001.

[10] Honour, E., "Principles of Commercial Systems Engineering", Proceedings from Fifth Annual International Symposium of the National Council on Systems Engineering, St Louis, MO, July 24-27, 1995.

[11] Höst, M., Regnell, B., Natt och Dag, J., Nedstam, J., Nyberg, C., "Exploring Bottlenecks in Market-driven Requirements Management Processes with Discrete Event Simulation", *The Journal of Systems and Software,* Vol. 59, pp 323-332, 2001.

[12] Lincoln, Y., S, Guba, E., G., *Naturalistic Enquiry,* Newbury Park and London, 1985

[13] Lubars, M., Potts, C., Richter, C., "A Review of the State of the Practice in Requirements Modelling", *First IEEE Int. Symposium on Requirements Engineering* (RE'93), pp2-14, 1993.

[14] Nikula, U., Sajaniemi, J., Kälviäinen, H., "A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises", TBRC Research Report 1, Telecom Business Research Center Lappeenranta, Lappeenranta University of Technology, 2000.

[15] Potts, C., "Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software", *Int. Proceedings: 2nd IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press*, pp. 128-130, 1995.

[16] Robson, C., *Real World Research,* Blackwell USA, 1997.

[17] Sawyer, P., "Packaged Software: Challenges for RE", *Proceedings of the sixth Int. Workshop on Requirements Engineering: Foundations of Software Quality* (REFSQ'00), pp 137-142, 2000.

[18] Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P., "Scenarios in System Development: Current Practice", *IEEE Software*, pp 34-45, March/April 1998.

Appendix: The interview instrument

	Characterization
1.1	Tell us about the company (number of employees, age, business area, etc.)
1.2	Tell us about the company's product/products (time on the market, typical customer/end-user, size of product projects, etc.)
1.3	Tell us about your position in the company (role, daily tasks, responsibility, etc.)
	Process issues
2.1	What is the procedure when developing a product? (kind of process, activities performed, documentation developed, evaluations performed, etc.)
2.2	What is a "requirement" to you?
2.3	In what way are requirements handled? (requirements process, activities, etc.)
2.4	What challenges do you face when working with requirements? What has been successful regarding requirements engineering?
2.5	How much resources are spent on requirements engineering? How much would be optimal?
2.6	What is a "good requirement" to you? And to the company? Is the quality of the requirements assessed? How?
2.7	What kinds of decisions are taken during the development of a product? What kind of support is needed in those decisions?
2.8	Is it possible to make decisions too late? What can be the effect in that case?
2.9	How is it decided what to include in the product? How are the requirements prioritized? What is difficult when deciding what to include in the product?
	Artefact issues
3.1	How are requirements documented? What information and attributes are documented about the requirements?
3.2	What support and what tools do you use to document your requirements? What pros and cons does these tools have?
3.3	How many requirements are handled in a typical project? Who suggests the requirements?
3.4	What kinds of dependencies between the requirements have you come across? Are dependencies documented? Are dependencies actively looked for?
3.5	How do dependencies affect product development? How is it handled?

PAPER II

Understanding Software Processes through System Dynamics Simulation: A Case Study

Carina Andersson, Lena Karlsson, Josef Nedstam, Martin Höst, Bertil I Nilsson

Proceedings of the 9th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS'02), Lund, Sweden, April 2002.

II

Abstract

This paper presents a study with the intent to examine the opportunities provided by creating and using simulation models of software development processes. A model of one software development project was created through means of system dynamics, with data collected from documents, interviews and observations. The model was simulated in a commercial simulation tool. The simulation runs indicate that increasing the effort spent on the requirements phase, to a certain extent, will decrease the lead-time and increase the quality in similar projects. The simulation model visualizes relations in the software process, and can be used by project managers when planning future projects. The study indicates that this type of simulation is a feasible way of modelling the process dynamically although the study calls for further investigations as to how project or process managers can benefit the most from using system dynamics simulations.

1. Introduction

This study was performed in cooperation with Ericsson Mobile Communications AB and is based on a development project carried out in 1999.

As a step in the constantly ongoing work with quality improvements at Ericsson this study was made to show if simulation can be used for visualizing how different factors affect the lead-time and product quality, i.e. number of faults. One of the most important factors that affect the lead-time of the projects and the product quality is the allocation of human resources to the different process phases. Thus, the focus of this simulation study is on resource allocation.

Simulation is commonly used in many research fields, such as engineering, social science and economics. That is, simulation is a general research methodology that may be applied in many different areas. Software process modelling and improvement is, of course, no exception and simulation has started to gain interest also in this area. For example, in [4] a high-maturity organization is simulated with system dynamics models, and in [6] a requirements management process is simulated with a discrete event simulation model. In [8] an overview of simulation approaches is given.

There are several advantages of building and simulating models of software processes. By simulation new knowledge can be gained that can help to improve current processes. Simulation can also be used for training and to enforce motivation for changes.

The objectives of the study that is presented here are to investigate if it is possible to develop a simulation model that can be used to visualize the behaviour of selected parts of a software process, and to evaluate the usefulness of this type of models in this area. The objective of the model is to identify relationships and mechanisms within a project. The study is focused on the tendencies of the simulation results and not the quantitative aspects.

The outline of the paper is as follows: In Section 2 the method used in this study is described. Section 3 describes the execution of the simulation study. Section 4 presents the results of the simulation and Section 5 discusses and summarizes the results of the study.

2. Method

This project was designed as a case study. Case studies are most suitable when data is collected for a specific purpose and when a subgoal of the study is establishing relationships between different attributes. A main activity in case studies is observational efforts.

With support from existing results in literature [3, 16], the research approach was created in three consecutive steps: problem definition, simulation planning and simulation operation. This methodology is based on the process chain concept, but due to lack of enough available, reliable data, the process in practice went into an interactive pattern.

In the first phase, problem definition, the problem was mapped. Then through deeper definition and delimitation, an agreement was created around the study's purpose.

The main part in the second phase of the study, simulation planning, was to identify factors influencing the product quality. The assigner of this study wished to test the idea of using simulation models and this was governing in the details of the study. This was natural as most of the ideas to the quality factors were picked up from the organization's project, through interviewing the project staff and through documents. To add a broader perspective, results and ideas were taken from software literature. Influence diagrams were built including the different quality factors' relation to each other, but primary their effects on lead-time and product quality.

The third phase, operating the simulation model, started with translating a small part of the theoretical model into the simulation tool. A short test showed that the simulation tool worked properly. More features were added from the theoretical model into the simulation tool and more test runs were performed. The verification and validation of the model was made stepwise through the input of the whole model into the simulation tool, and the yardstick to compare with was given by documents and discussions with the assigner.

3. Developing the simulation model

In the simulation domain there are two main strategies: continuous and discrete modelling. The continuous simulation technique is based on system dynamics [1], and is mostly used to model the project

environment. This is useful when controlling systems containing dynamic variables that change over time.

The continuous model represents the interactions between key project factors as a set of differential equations, where time is increased step by step. In the standard system dynamics tools, these interconnected differential equations are built up graphically. A system of interconnected tanks filled with fluid is used as a metaphor. Between these tanks or levels there are pipes or flows through which the variables under study are transported. The flows are limited by valves that can be controlled by virtually any other variable in the model. Both this mechanism and the level-and-flow mechanism can be used to create feedback loops. This layout makes it possible to study continuous changes in process variables such as productivity and quality over the course of one or several projects. It is however more problematic to model discrete events such as deadlines and milestones within a project [9, 10].

In the discrete model, time advances when a discrete event occurs. Discrete event modelling is for example preferred when modelling queuing networks. In its simplest form, one queue receives time-stamped events. The event with the lowest time-stamp is selected for execution, and that time-stamp indicates the current system time. When an event occurs an associated action will take place, which most often will involve placing a new event in the queue. Since time always is advanced to the next event, it is difficult to integrate continually changing variables. This might result in instability in any continuous feedback loops [9, 10].

To suit the purpose of this study, which is to visualize process mechanisms, continuous modelling was used. The continuous model was chosen in order to include systems thinking [13] and because it is better than the discrete event model at showing qualitative relationships.

3.1 Problem definition

The study is based on a process that is similar to the waterfall model [14]. The whole process is shown in [1], but the simulation model was focused on the requirements phase and the test phase. The other phases, with broken lines in [1], were excluded to get a less complex model. The requirements phase includes the pre-study phase and the feasibility study phase. The test phase involves the unit, system and acceptance tests. All these types of tests are included, since the data available did not separate between test types and they overlapped in terms of time.

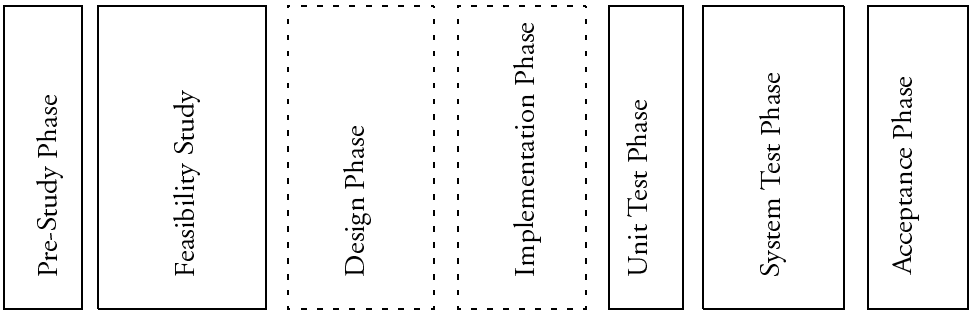

Figure 1. *Process description*

3.2 Simulation planning

This step included identifying factors that affect the quality of the developed software and the lead-time of the project. This was made through interviews with project staff and based on information in literature [5, 7].

The relevant project staff consisted in three persons with whom discussions were held continually during the entire study. Among the factors discovered during interviews, only those considered relevant to software development processes were selected. The identified factors are listed in Table 1. Discussions with concerned personnel pointed out the

Table 1. Factors that affect quality and lead-time

Number of personnel in the project	Amount of new market requirements
Level of personnel education	Amount of requirements changes
Level of personnel experience	Level of inadequate requirements
Level of personnel salary	Amount of review
Level of personnel turnover	Amount of rework
Communication complexity	Level of structure in the project organization
Geographical separation of the project	Standards that will be adhered to e.g. ISO and IEEE
Software and hardware resources	Amount of software functionality
Environment, e.g. temperature, light, ergonomics	Testing and correcting environment and tools
Amount of overtime and workload	Productivity
Level of schedule pressure	Amount of program documentation
Level of budget pressure	Level of reusable artefacts, e.g. code and documentation

most important factors in respect to both quality and lead-time. The factors considered to affect quality and lead-time the most were chosen to be included in the influence diagrams, see [2].

Influence diagrams [12] for the requirements and the test phase were built to show how the chosen factors affect the lead-time and the software quality. Each factor's importance for each phase was considered together with the relationships between the factors. The influence diagram for the requirements phase is shown in [2]. The factors in the influence diagram are further explained below.

- *Amount of functionality* is estimated software funtionality to be developed.
- *Amount of new market requirements* is a measure of the change in market expectations.
- *Amount of requirements changes* is a measure of the changes made in the requirements specifications.
- *Amount of review* involves reviewing requirements specifications.
- *Amount of rework* is the effort spent on reworking both new and inadequate requirements.
- *Communication complexity* is an effect in large project groups where an increasing number of participants increases the number of communication paths.
- *Level of inadequate requirements* is a measure of the requirements specification quality.
- *Level of personnel experience* is a measure of knowledge of the current domain.
- *Level of schedule pressure* is the effect of the project falling behind the time schedule.
- *Number of personnel* is the number of persons working with requirements specifications in the project.
- *Productivity* is a measure of produced specifications per hour and person.
- *Time in requirements phase* is the lead-time required to produce the requirements specifications in this project.

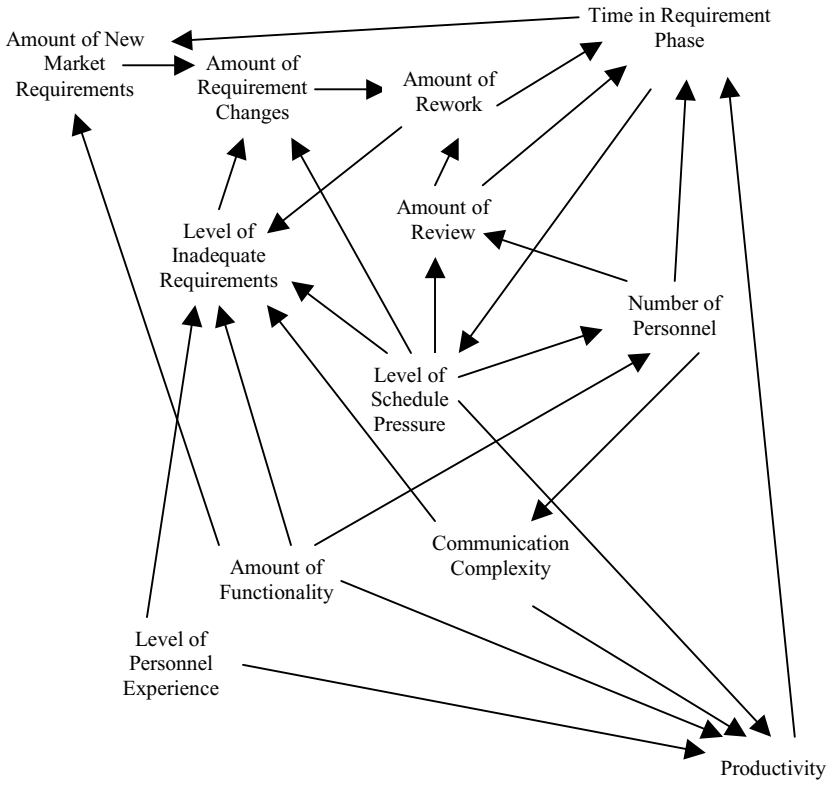

Figure 2. *Influence diagram for the requirements phase*

It is beyond the scope for this paper to present all details of the simulation model. In this paper the simulation model and related models, such as influence diagrams, are presented in some detail for the requirements phase. The requirements phase is by its nature more intuitive and easy to understand than the test phase. For a presentation of details of the complete simulation model with all related models refer to [2]. For example, the influence diagram for the test phase is presented in [2] and not here.

At the same time as the influence diagrams were constructed, causal-loop diagrams were built to get a basic understanding of the feedback concepts. Causal-loop diagrams are often used in system dynamics to illustrate cause and effect relationships [1]. When examining these

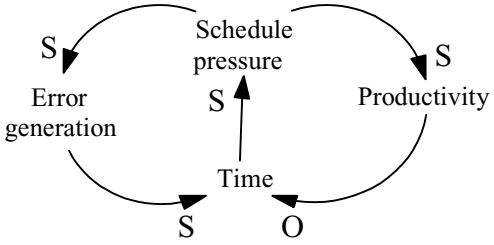

Figure 3. *Causal-loop diagram*

relationships isolated, they are usually very easy to understand. However, when they are combined into long chains of cause and effect, they can become complex. The causal-loop diagrams increase the understanding of these complex relations. [3] illustrates how the schedule pressure affects the time spent in the requirements phase. An increased schedule pressure increases the error generation, due to a higher stress level. A high error density increases the amount of necessary rework and thereby increases the time in the requirements phase, which in turn increases the schedule pressure. At the same time, high schedule pressure increases the productivity because of its motivational role. Increased productivity decreases the time spent in the requirements phase, which in turn decreases the schedule pressure.

Information about the relationships between the factors in the causal-loop diagram is shown by adding an "O" or an "S" to the arrows. An "O" implies a change in the opposite direction, while an "S" implies a change in the same direction.

3.3 Simulation operation

The simulation model was built based on the knowledge gained from creating influence diagrams and causal-loop diagrams. The idea behind the model of the requirements phase is based on a flow of tasks, from customer requirements to finished specifications. In the requirements phase there is a transformation from uncompleted to completed tasks by the production of specifications. A fraction of the specifications are not acceptable and needs to be taken care of in the rework loop, see [4].

The test phase in the model is based on the same idea as the requirements phase and is built in a similar way. A flow of test cases is

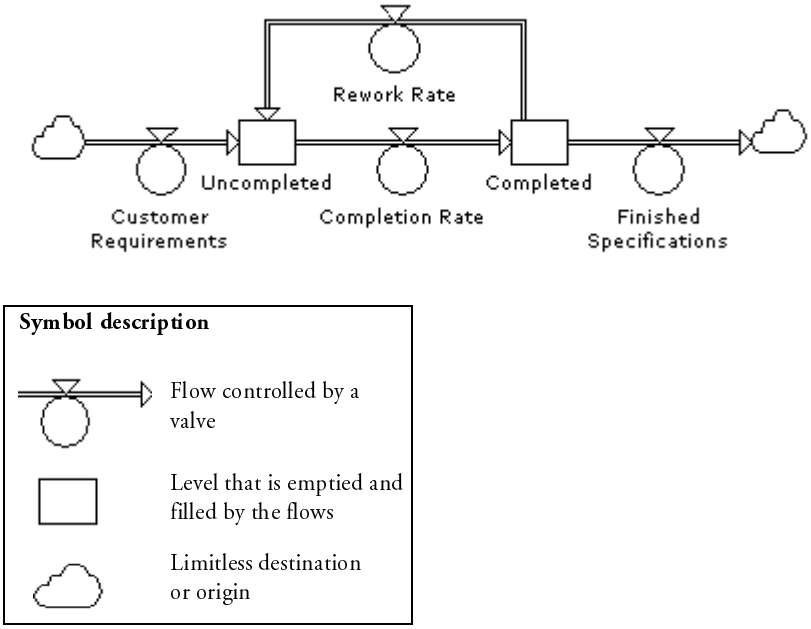

Figure 4. *Basic model of the requirements phase*

performed, a certain percentage of the functionality has to be corrected and retested, and the rest is supposed to be acceptable.

This basic model was built in the Powersim simulation tool [11] and further developed with help from the factors in the influence diagrams. Factors from the influence diagrams were added to the model in order to affect the levels and flows. The causal-loop diagrams were also considered during the development, to ensure that the model was adapted to systems thinking.

To avoid getting a too complex model, all of the factors in the influence diagrams were not included in the simulation model. Some factors were included indirectly in the parameters in the model. These can be extracted from the parameters and are thereby possible to affect from the user interface, for example the communication complexity which is included in the productivity. The construction was made step by step, by adding a few factors at a time and then running the simulation. The values of the parameters were taken from project documentation except one that was taken from [7], *Amount of new market requirements*. This parameter was not available in project documentation but the value from

[7] is an average from several software projects and was considered to be valid also for this project. Some values were estimated by iteration and verified by discussions with concerned personnel at the organization. The verification of the simulation model was made through checking that the amount of code that is used as an input to the model is the same as the output amount of code. The verification also included comparing the time in the simulation to the time according to the project documentation to ensure that the estimations were correct.

The final model for the requirements phase is seen in Appendix A. The flows in [4] is the base of the final model, which is then further developed. To get a measure of the quality of the specifications, another flow was included, which counts the inadequate specifications. This measure affects the amount of defect code that is produced in the design and implementation phases which in turn affects the test phase. The design and implementation phases are in the simulation model modelled as a delay. A second flow is added to the basic model to terminate this phase and start the following phases.

The rest of the additions to the basic model can be described in four groups, where each group originates from the influence diagram.

- The first group, *Lines of code* and *Functionality,* describes the functionality of the code to be developed. This group controls the inflow to the phase.
- The second group is *Percentage, Effort* and *Duration*. The *Percentage* allocates a percentage of the planned total effort to the requirements phase and is controlled from the user interface. This makes it possible to study how the amount of resources in the requirements phase affects the lead-time and quality.
- The third group, *Productivity* and *Duration,* controls the completion rate of the specifications. The *Duration* also affects the amount of inadequate specifications because of the schedule pressure that might increase during the project's duration.
- The fourth group, *Amount of rework* and *Functionality*, decides how much of the specifications that needs to be reworked after the reviews.

Note that some factors are part of more than one group. This is because some factors affect more that one other factor.

4. Results from the simulation

The final model was simulated to show how a relocation of resources to the different process phases affects the quality of the software products and the lead-time of the project. This model included both the requirements phase and the test phase. The model was run several times with different values of the percentage of the planned project effort, spent on the requirements phase. The results are given in precise figures but since there are a number of uncertainties they should be broadly interpreted. For example, the results are uncertain because of the difficulty in measuring the values of the included factors. It is the tendencies in the results that are important and not the exact figures.

The simulation runs indicate that the effort spent on the requirements phase has a noticeable effect on the lead-time of the project. The decrease in days, when increasing the effort in the requirements phase, arises from the increased specification accuracy. A more accurate specification facilitates the implementation and decreases the error generation and will result in a higher product quality from the start. This decreases the amount of necessary correction work and thereby shortens the time spent in the test phase. At a certain point the total lead-time will start to increase again because the time in the test phase stops decreasing while the time in the requirements phase continues to increase. The time in the test phase stops decreasing because there is always a certain amount of functionality that needs to be tested at a predetermined productivity. The number of days in [5] is the total lead-time for the whole project.

In the same manner, the quality increases when increasing the effort in the requirements phase to a certain extent. The simulation runs indicate that the quality optimum appears in the same area as the lead-time

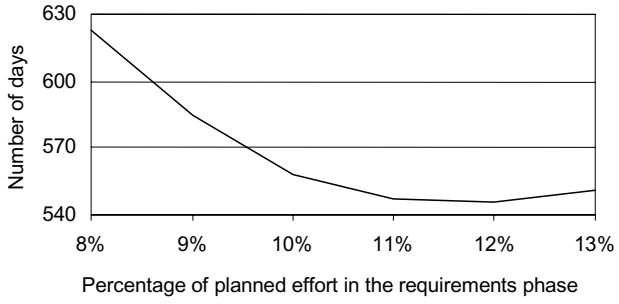

Figure 5. *Simulation results for the total lead-time*

optimum. The increase in quality originates from a higher specification accuracy, which is explained above. However, if too much effort is spent in the requirements phase, the quality will start to decrease again because there is less effort left for design, implementation and test tasks.

As a step in the verification of the results, they were compared to results in the software literature [7, 15]. This literature points at the same magnitude of effort in the requirements phase for a successful project as the simulation results.

To summarize, the simulations indicate that there is an optimum for both the quality and the lead-time. If the effort in the requirements phase is lower than the optimal value, increasing it towards the optimum will result in increased quality of the developed software and decreased lead-time.

5. Discussion

One result of this study is a simulation model that visualizes different relations in a software development process. A simulation of this kind can contribute to enhancing the systems thinking in an organization. Thereby it is easier for the members of the organization to understand the relationships between the quality factors in the process.

The results from this kind of simulation shall not be interpreted precisely since there, of course, are a number of uncertainties. It is the tendencies and the behaviour in the results that are important and by changing the parameters in the model it is possible to get a picture of how the process mechanisms interact. This is a simplified model of the reality and therefore there are a number of sources of uncertainty. The included factors might not be the ones that affect the model the most, the assumed relations between the factors might not be correct and the values of the factors can be incorrectly estimated. However, the results, that there is an optimum for the effort that is spent in the requirements phase, can be intuitively expected for many projects in software organizations.

A simulation of this kind can also be used to increase the motivation of the organization to work with quality issues and to increase the product quality early in the project.

One part of the knowledge gained from simulations is received in the model building process. The procedure to build the model forces the

participants to communicate their mental models and to create a common image of the organization's direction.

To summarize, it seems to be feasible to build and use this kind of model for this kind of process. There are, however, a number of uncertainties which are important to take into account when the results are interpreted. This is a first model, based on one project, that needs to be further elaborated in order to obtain a model that can be applied on other projects. Thus, the model has not been empirically validated in real projects after it was developed. As far as the authors know, the model is not currently in use at Ericsson.

The impression after developing and getting feedback on the model is that it is uncertain whether most knowledge is gained by developing the model or using it. This is one of a number of issues that need to be further investigated in the area of software process simulation.

The model could either be used, for example by a project manager, by only changing the parameters, or it could be used by changing also the structure of the model, for example by adding or deleting factors and adding or deleting relationships between factors. It may be that users of the models need to understand the internal structure of the model and not only the interface to it. This would limit the choice of modelling techniques, and it would for example mean that models with an internal design, that is not easy to understand for the users of the models, would not be suitable in all cases.

Acknowledgement

The authors would like to thank Wladyslaw Bolanowski and Susanne S. Nilsson at Ericsson Mobile Communication AB for all their help with this study. This work is partly funded by the Swedish Agency for Innovation Systems (VINNOVA) under grant for Centre for Applied Software Research at Lund University (LUCAS).

References

[1] Abdel-Hamid, T., Madnick, S.E., *Software Project Dynamics: An Integrated Approach,* Prentice Hall, 1991.

[2] Andersson, C., Karlsson, L., "A System Dynamics Simulation Study of a Software Development Process", CODEN:LUTEDX(TETS-5419)/1-83/(2001)&local 3, Department of Communication Systems, Lund Institute of Technology, 2001

[3] Banks, J., Carson, J.S., Nelson B.L., *Discrete-Event System Simulation,* Prentice Hall, 1996

[4] Burke, S., "Radical Improvements Require Radical Actions: Simulating a High-Maturity Software Organization", Technical Report CMU/SEI-96-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburg, USA, 1996.

[5] Fenton, N.E., Pfleeger, S., *Software Metrics: A Rigorous & Practical Approach,* International Thomson Computer Press, 1996

[6] Höst, M., Regnell, B., Natt och Dag, J., Nedstam, J, Nyberg, C., "Exploring Bottlenecks in Market-Driven Requirements Management Processes with Discrete Event Simulation", Accepted for publication in Journal of Systems and Software, 2001.

[7] Jones, T.C., *Estimating Software Cost,* McGraw-Hill, 1998

[8] Kellner, M.I., Madachy, R.J., Raffo, D.M., "Software Process Simulation Modelling, Why? What? How?", Journal of Systems and Software, Vol. 46, No. 2-3, pp. 91-105, 1999.

[9] Martin, R., Raffo, D., "A Model of the Software Development Process Using both Continuous and Discrete Models", *International Journal of Software Process Improvement and Practice*, 5:2/3, June/September, pp. 147-157, 2000.

[10] Martin, R., Raffo, D., "Application of a Hybrid Process Simulation Model to a Software Development Project", proceedings of *PROSIM 2000,* July 12-14, London, UK

[11] Powersim corporation, www.powersim.com, 010903

[12] Rus, I., Collofello, J.S., "Assessing the Impact of Defect Reduction Practices on Quality, Cost and Schedule", proceedings of *PROSIM 2000*, July 12-14, London, UK

[13] Senge, P.M., *The fifth discipline,* Random House Business Books, 1990

[14] Sommerville, I., *Software Engineering*, Addison-Wesley, 1996

[15] Stewart, R.D., Wyskida, R.M., Johannes, J.D., *Cost Estimator's Reference Manual,* John Wiley & Sons Inc, 1995

[16] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., *Experimentation in Software Engineering: An Introduction,* Kluwer Academic Publisher, 2000

PAPER III

An Analytical Model for Requirements Selection Quality Evaluation in Product Software Development

III

Björn Regnell, Lena Karlsson, Martin Höst

Proceedings of the 11th IEEE International Requirements Engineering Conference (RE'03), Monterey Bay, California, USA, September 2003.

Abstract

In market-driven development of product software, a crucial decision for each candidate requirement is whether or not to select it for implementation in the next release. This paper presents an analytical model of the requirements selection process, which is used for reasoning about decision quality. A network of queues, with two classes of jobs, models the selection of requirements of different quality. The feasibility of model parameter estimation is validated in a survey involving product managers and system engineers. The results show that some of the respondents have made internally consistent parameter estimations, indicating that the model is relevant and its parameters understandable. It is also shown that a majority of the consistent respondents estimate that most of their implemented product requirements are incorrectly selected. The main objective of the model is to provide tools for evaluation of improvement proposals by estimating the impact of process change.

1. Introduction

This paper describes an analytical model of a market-driven Requirements Engineering (RE) process for product software. Market-driven RE differs from customer-specific RE in several ways, for example in the characteristics of stakeholding and the pressure on time-to-market [12, 13]. Requirements are often invented by the developers and elicited from a set of potential customers with scattered wishes [9]. A requirements repository is continuously enlarged with new candidate requirements [4, 11]. Commonly, market-driven software-developing organisations provide successive releases of the software product and release planning is an essential activity [2, 3].

A major challenge in market-driven RE is to select and prioritize the right set of requirements to be implemented in the next release [9], while avoiding congestion in the selection process [11]. A previous study has shown that capacity issues can successfully be investigated using simulation [5]. However, that study concentrated on different work load situations without taking neither requirements quality nor selection decision quality into account.

This paper presents a novel analytical model representing a condensed view of the requirements selection process.

There are three main purposes of the model:

1. An idealistic and simplified view of reality can be used for principal reasoning about both quality and capacity in requirements selection. The general properties of complex processes is modelled through careful choices of appropriate approximations.

2. The analytical power of the model can provide theoretical limits on the highest possible product quality that is achievable given specific parameter settings of input to the process, the quality of the decisions in the process, and the capacities of the different parts of the process.

3. The analytical model can act as a baseline for further empirical research using simulations of more elaborate models of requirements selection that are intractable using analytical methods.

The goal is to be able to answer questions such as: “How good is a specific organisation at selecting the right requirements?”, “What capacity is needed in order to achieve a certain quality?”, “How long does it take to

get good ideas released to the market?" "How long must our customers wait until they get feedback on their proposals?"

The presented model is based on queuing theory [8]. Queuing theory is good for modelling systems with a steady flow of discrete jobs that are waiting in queues to be served by a number of servers and allows for calculations of attributes such as serving time, system load, and stability criteria. The model applies queuing theory for networks with multi-class jobs [7], which allows for modelling that requirements are of different quality classes.

Related work include queuing models for staffing of the software maintenance process [1, 10], but these studies do not take decision quality into account.

The paper is structured as follows. Section 2 presents the context of requirements selection and proposes a simplified, generic model of the selection process. Section 3 introduces a fundamental model of requirements selection quality using a contingency table. Section 4 proposes the novel analytical model based on queuing theory, and also presents analytical results derived from the model. Section 5 reports on an industrial survey where the feasibility of model parameter estimation using expert judgement is validated. Section 6 explains how the presented model can be used by practitioners. Section 7 concludes the paper and provides directions for further research.

2. Requirements Selection

When developing product software for a large, open market with many customers and users, there is a potential of eliciting numerous candidate requirements from a plurality of sources. A typical way of handling this situation is to store continuously incoming requirements in a repository and, as candidate requirements arrive, continuously evaluate them based on their estimated construction effort and predicted market value. Release plans are decided based on a cost-benefit trade-off in relation to the specific business strategy. In order to prevent the process from being overloaded [6], it may be a good idea to have an initial screening function. Screening is a quick assessment of whether a new candidate is worth spending more time on, or if it should be rejected before proceeding to a deeper evaluation. The evaluation stage includes further requirements analysis, specification, validation and prioritisation. Previous studies have

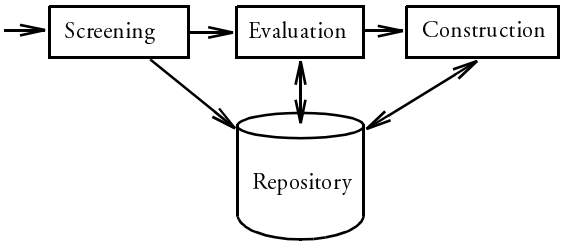

Figure 1. *A simplified requirements selection process*

shown evidence of screening and evaluation stages in industrial RE practice [2, 6, 11].

A simplified requirements selection process is illustrated in Fig. 1. The screening activity prevents a portion of all incoming requirements from being allowed to reach the evaluation activity. The screening is based on the product profile and business strategy. The selected requirements are evaluated with respect to estimated market value and development effort. Those considered fruitful are planned for construction. The construction phase includes designing, implementing, and testing the requirements that are selected based on the result of the evaluation.

A corresponding state model of requirements refinement is shown in Fig. 2. (More state-based models of the requirement life-cycle is described in [2].) Newly issued requirements are in the state *issued*. After being screened, each requirement can either be put aside to the state *early reject*, or propagated to the state *accepted*, implying that the requirement is accepted for evaluation. In order for a requirement to advance to the state *planned*, it has to be considered profitable by the evaluation activity. If the requirement is considered to be of low commercial value, it will continue to the state *late reject*. Finally, when the planned requirements are completed by the construction activity they will enter the state

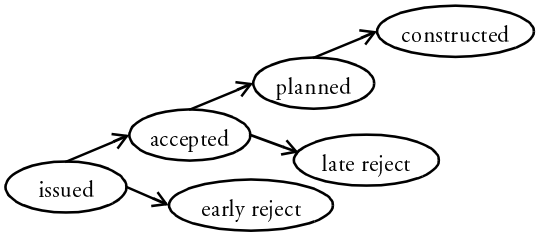

Figure 2. *A state model of requirements refinement.*

constructed. The requirement states are stored in the requirements repository, and changes in requirements states imply an update of the repository.

3. Selection Quality

In principle, if perfect information about a requirement was given, it would be possible to, in advance, tell if it is a good decision to include a requirement for implementation in the software product. This basic assumption can be used to make the following definitions:

- α-requirements are the requirements that under perfect decision-making should be *selected*.
- β-requirements are the requirements that under perfect decision-making should be *rejected*.

The α-requirements represents the "golden grains" of all incoming requirements and all other requirements are termed "poor" β-requirements. In reality, decision-making is imperfect, and the real nature of the requirements can not be estimated with high accuracy until after product release, when the implemented requirement has been on the market for a while. However, this ideal view allows for the definition of ratios that can be used to assess the decision quality of the requirements selection process, as shown in Table 1. The four different ratios of correct and incorrect decisions are denoted A to D. Obviously it is favourable to maximize the correct selection ratios A and D, and minimize the incorrect selection ratios B and C.

Table 1. A contingency table for assessing requirements selection quality

		Decision	
		Selected	Rejected
Requirements Quality	α	A correct selection ratio	B incorrect rejection ratio
	β	C incorrect selection ratio	D correct rejection ratio

In all stages of the selection process it is desirable to select requirements of type α and reject requirements of type β. However, the knowledge about the market, competitors and customers is limited, insight into all possible relations and dependencies among requirements is restricted, and effort estimation is error prone. These facts of reality lead to decision-making that is far from perfect. Therefore, some β-requirements will be, incorrectly, selected whereas some α-requirements will be rejected. This is reflected in Table 1, where an α-requirement is either correctly selected or incorrectly rejected, and a β-requirement is either incorrectly selected or correctly rejected.

4. Analytical Model

In Fig. 3, a queuing network model is presented, that is based on the refinement model in Fig. 2. Each refinement step is modelled by a number of queues with one server per queue. In order to enable analytical calculation, arrival processes are assumed to be Poisson processes, the service rates are exponentially distributed, which gives that each queue is (in queuing theory terms) M/M/1 [8]. The M/M/1 case is favourable from an analytic viewpoint, as its properties are derivable from rather simple formulas. (In general, if the arrival process is resulting from many different sources then Poisson tends to be a good approximation, as the sum of a large number of different arrival distributions converge towards a exponential distribution. However, if the traffic is very bursty in nature with bulk arrivals, this assumption may be inadequate and other distributions can then be treated using simulation.)

Each server in the model represents one employee of the software product provider, for example a requirements engineer or developer, while

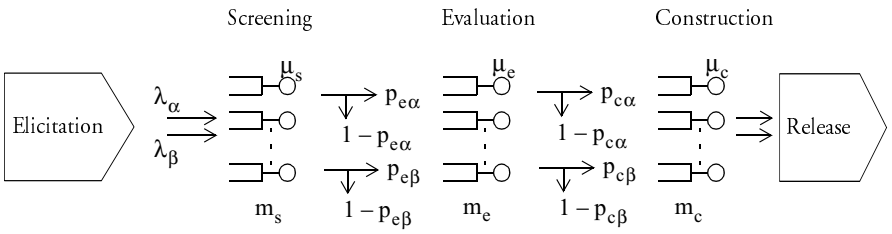

Figure 3. *A queuing network model of the requirements selection process.*

each queue represents a work repository for the server. To simplify calculations, all servers are assumed to have the same service rates within one phase. The service rates of each phase models the average time for one engineer to do the work of that phase. The work is divided randomly among servers, with equal probability. This simplification does not take different competencies and productivities into account, but if an empirical grounded average is used, this may be adequate in a basic model.

After screening, a fraction of the requirements are discarded and the remaining ones are propagated to evaluation. The requirements selected for evaluation includes both α and β requirements. The average evaluation effort is estimated to be the same for all requirements, regardless of quality. Finally, after disposing some of the evaluated requirements, the construction phase is entered.

It is assumed that the requirements repository is only investigated once during a release, i.e. once a requirement is disposed it can not return into the process. This simplification does not take requirements decomposition into account. If there is a large number of new requirements generated within the selection process, this can be modelled by increasing the arrival rate correspondingly.

There are many aspects of a real, complex requirements selection process that are not taken into account here, as we aim for a clear and uncomplicated model that is easy to understand and allows for analytical treatment. Intricate aspects of reality that has been excluded from the model include: deadline and budget restrictions, competition between requirements analysis and construction resources, disposed requirements during construction, dependencies between requirements, requirements decomposition into "sub-requirements", etc. More complex models that take such aspects into account can be further analysed using simulation, which is a matter of further research (see Section 7).

4.1 Model parameters

In order to analytically reason about the presented model, a number of parameters are defined. Common notation from queuing theory is used for arrival and service rates, denoted λ and μ respectively. The number of servers in each stage is denoted m, and the propagation probabilities between stages are denoted p. Using indices of α and β for the two different classes of requirements, as well as indices s, e, c, for screening,

Table 2. Model parameters

λ	Total arrival rate of requirements issued from elicitation.
λ_α , λ_β	Arrival rates of α - and β -requirements.
μ_s , μ_e , μ_c	Service rates in screening, evaluation and construction.
m_s , m_e , m_c	Number of servers in screening, evaluation and construction.
$p_{e\alpha}$, $p_{c\alpha}$	Probabilities for an α -requirement to continue to evaluation and construction.
$p_{e\beta}$, $p_{c\beta}$	Probabilities for a β -requirement to continue to evaluation and construction.
g	Golden-grain-ratio.
δ_{max}	Maximum allowed work load.
C_s , C_e , C_c	The total work capacity in screening, evaluation and construction.

evaluation, and construction, respectively, gives the model parameters summarized in Table 2. The maximum allowed work load is denoted δ_{max}, which should be less than 1 in order to ensure stability. The capacities of a queuing network stage is denoted *C* and correspond to the number of servers *m* multiplied by the service rate μ.

An interesting parameter is the relation between high- and low-quality requirements. This relation is defined by the constant *g*, called the *golden-grain-ratio*, and describes the proportion of the requirements that are of high quality. The golden-grain-ratio is defined as:

$$g = \frac{\lambda_\alpha}{\lambda_\alpha + \lambda_\beta} = \frac{\lambda_\alpha}{\lambda}$$

4.2 Parameter Estimation

A major benefit of an analytical model is the possibility to study different "what if"-scenarios, e.g. to investigate what happens if the early rejection ratio of low-quality requirements is increased or decreased. When studying such scenarios, it is enough to have rough approximates of the parameter values as the directions of changes are of more interest than the actual values. Nevertheless, the accuracy of parameter estimation is an important issue and the more accurate the parameters can be estimated, the more reliable are the analytical results.

One way to estimate the values of the model parameters defined above, is to base them on historical data from development of earlier product releases. Another possibility is to estimate the parameters subjectively, using expert judgement by experienced professionals within the software developing organisation under study.

Measurements from earlier development can be used if a measurements programme is installed where the life-cycle of each requirement is stored in a requirements repository and the actual requirement quality is recorded after being released to a market and its quality in terms of market value versus development cost is clearer. Consequently, it is necessary to determine and record if each implemented requirement actually should have been selected for implementation or not. And, accordingly, it is necessary to determine if every discarded requirement actually should have been discarded or if it in fact was a golden grain. If these metrics are too difficult to obtain in a specific organisation, subjective judgement by experts may be a more tractable approach to the calibration of an analytical model.

Section 5, reports on an industrial survey, in which the parameters of the proposed model are subjectively estimated by product managers and system developers.

4.3 Analytical Results

Using the parameters in Section 4.1, we can describe the rate of requirements entering the different refinement states in Figure 2.

$$\text{Issued} = \lambda_\alpha + \lambda_\beta = \lambda$$

$$\text{Accepted} = \lambda_\alpha p_{e\alpha} + \lambda_\beta p_{e\beta}$$

$$\text{Planned} = \text{Constructed} = \lambda_\alpha p_{e\alpha} p_{c\alpha} + \lambda_\beta p_{e\beta} p_{c\beta}$$

$$\text{EarlyReject} = \lambda_\alpha (1 - p_{e\alpha}) + \lambda_\beta (1 - p_{e\beta})$$

$$\text{LateReject} = \lambda_\alpha p_{e\alpha} (1 - p_{c\alpha}) + \lambda_\beta p_{e\beta} (1 - p_{c\beta})$$

From the rates in different states it is possible to derive rates A to D in Table 1, i.e. the correct and incorrect decision rates.

$$A = \frac{\lambda_\alpha \cdot p_{e\alpha} \cdot p_{c\alpha}}{\lambda_\alpha + \lambda_\beta}$$

$$B = \frac{\lambda_\alpha \cdot (1 - p_{e\alpha}) + \lambda_\alpha \cdot p_{e\alpha} \cdot (1 - p_{c\alpha})}{\lambda_\alpha + \lambda_\beta}$$

$$C = \frac{\lambda_\beta \cdot p_{e\beta} \cdot p_{c\beta}}{\lambda_\alpha + \lambda_\beta}$$

$$D = \frac{\lambda_\beta \cdot (1 - p_{e\beta}) + \lambda_\beta \cdot p_{e\beta} \cdot (1 - p_{c\beta})}{\lambda_\alpha + \lambda_\beta}$$

where $A + B + C + D = 1$

With the definition of g from section 4.1 it is possible to simplify the equations for the selection and rejection ratios.

$$A = g \cdot p_{e\alpha} \cdot p_{c\alpha}$$

$$B = g(1 - p_{e\alpha} \cdot p_{c\alpha})$$

$$C = (1 - g) \cdot p_{e\beta} \cdot p_{c\beta}$$

$$D = (1 - g) \cdot (1 - p_{e\beta} \cdot p_{c\beta})$$

According to the definitions in Table 1, it is desired to maximize A+D, i.e. the ratio of *correct decisions*, and minimize B+C, i.e. the ratio of *incorrect decisions*. The ratio A/(A+C) is the fraction of *correct selections*.

The criteria for stability are based on the fact that the total capacity C of the servers in one stage must be larger than the arrival rate from the previous stage, in order for the queuing network not to be overloaded. The stability criteria for the different stages are:

$$C_s = m_s \cdot \mu_s > \lambda_\alpha + \lambda_\beta$$

$$C_e = m_e \cdot \mu_e > \lambda_\alpha \cdot p_{e\alpha} + \lambda_\beta \cdot p_{e\beta}$$

$$C_c = m_c \cdot \mu_c > \lambda_\alpha \cdot p_{e\alpha} \cdot p_{c\alpha} + \lambda_\beta \cdot p_{e\beta} \cdot p_{c\beta}$$

When considering that it is unrealistic to have a server occupation of 100%, we can adjust the equations with the maximum allowed work load for each phase, δ_{max}. The desired value of δ_{max} depends on the tolerance of a high probability that queues may be very long and hence risking very long waiting times.

The golden-grain-ratio g and maximum allowed work load $\delta_{max} \leq \lambda / C$ can be inserted into the equations for capacity above, which gives the following lower bound of capacities in the different stages of requirements refinement:

$$C_{smin} = \frac{\lambda}{\delta_{max}}$$

$$C_{emin} = \frac{\lambda}{\delta_{max}}(p_{e\alpha} g + p_{e\beta}(1-g))$$

$$C_{cmin} = \frac{\lambda}{\delta_{max}}(p_{e\alpha} p_{c\alpha} g + p_{e\beta} p_{c\beta}(1-g))$$

In a system of *m* parallel M/M/1 queues with a total capacity of $C = m \cdot \mu$, the results from queuing theory allows us to calculate the average number of customers, denoted *N*, and the average time in the system, denoted T (see for example [8]).

Let *x* denote each step *s*, *e*, *c*, for screening, evaluation and construction respectively. The arrival rate of each step is given by:

$$\lambda_s = \lambda_\alpha + \lambda_\beta = \lambda$$

$$\lambda_e = \lambda \cdot (g \cdot p_{e\alpha} + (1-g) \cdot p_{e\beta})$$

$$\lambda_c = \lambda \cdot (g \cdot p_{e\alpha} \cdot p_{c\alpha} + (1-g) \cdot p_{e\beta} \cdot p_{c\beta})$$

By using the results from queuing theory [8], it is possible to calculate *N* and *T* for each step as:

$$N_x = m_x \cdot \frac{\frac{\lambda_x}{C_x}}{1 - \frac{\lambda_x}{C_x}} \qquad T_x = m_x \cdot \frac{\frac{1}{C_x}}{1 - \frac{\lambda_x}{C_x}}$$

Based on the *T* and *N* entities, we define the following three interesting metrics:

- Mean Time To Market, *MTTM*, defined as the sum of the average times in screening, evaluation and construction. *MTTM* represents the average time from that a requirement is issued until it is finally constructed. $MTTM = T_s + T_e + T_c$
- Mean Response Time, *MRT*, defined as the time for a requirement to either be rejected in screening or pass through evaluation. *MRT* represents the average time from that a requirement is issued until a decision can be communicated back to the issuer, informing the issuer whether or not the requirement will be implemented. *MRT* can be obtained using Little's result [8] which says that in general $N = \lambda T$, and thus we can say that $MRT = (N_s + N_e)/(\lambda_\alpha + \lambda_\beta)$
- Requirements engineering Effort Share, *RES*, defined as the average share of all service effort that is put on screening and evaluation. Let $\delta_{1x} = \lambda_x/(m_x \cdot \mu_x)$ denote the load in one of the m_x parallel queues in step *x*. *RES* is defined as:

$$RES = \frac{\delta_{1s} m_s + \delta_{1e} m_e}{\delta_{1s} m_s + \delta_{1e} m_e + \delta_{1c} m_c}$$

Table 3. Parameter assumptions of four example cases.

Proba-bility	Case RANDOM	Case LOW	Case HIGH	Case IDEAL
$p_{e\alpha}$	0.5	0.6	0.7	1
$p_{e\beta}$	0.5	0.4	0.3	0
$p_{c\alpha}$	0.5	0.8	0.9	1
$p_{c\beta}$	0.5	0.2	0.1	0

Table 4. Parameter assumption, same for all examples.

λ	g	δ_{max}	m_s	m_e	m_c	μ_s	μ_e	μ_c
3 req/ day	10 %	80 %	1	2	20	4 req/ day	2 req/ day	0.05 req/ day

4.4 Example

To make the presented model and its analytical results more concrete, four examples are given, where the values of the propagation probabilities varies. As shown in Table 3, the LOW case has lower probabilities of selecting α-requirements than the HIGH case. The RANDOM case corresponds to a case where the screening and evaluation work is no better than flipping a coin when deciding to reject or propagate a requirement.

The RANDOM case illustrates an example where it is equally probable to select α- and β-requirements, i.e. half of each requirement type is selected and no discrimination between α- and β-requirements are made. It is also equally probable to select and reject requirements in evaluation, meaning that the decision quality is not improved over time.

In the LOW, HIGH and IDEAL cases it is more probable to select α-requirements than β-requirements, which hopefully comply with a real-world situation. Furthermore, in the LOW and HIGH cases, the probability of selecting high-quality requirements increases over time, i.e. the probability of selecting α-requirements in the evaluation phase is larger than in the screening phase, which is reasonable as more effort is put on evaluation than screening. Furthermore, the increasing probability

models that more knowledge is gained through out the process, and this knowledge is assumed to increase selection quality.

Case HIGH is preferable over case LOW, since the probabilities of selecting α -requirements is higher in Case HIGH. Both cases HIGH and LOW are superior to the RANDOM case, which can be seen as a baseline case. Having probabilities that are lower than 0.5 for propagating α requirements and higher than 0.5 for propagating β requirements, implies that the organisation is systematically favouring bad requirements, and this is hopefully not realistic.

The IDEAL case is special, as no α requirements are rejected, and no β requirements are selected. Screening is hence perfect, thus providing the evaluation step only with α requirements. IDEAL is optimal in the sense that the product only includes high-quality requirements. This case is not likely to occur in practice, but represents an upper limit of what is achievable in terms of selection quality.

Table 4 shows the other parameter assumptions that are kept constant for all four example cases. In the simulation study in [5] the mean arrival rate, λ, was approximately 3 requirements per day, and the same arrival rate is chosen for all cases. The golden-grain-ratio is chosen to be 10%,

Table 5. Results based on parameter assumptions

Entity		Case RANDOM	Case LOW	Case HIGH	Case IDEAL
minimal screening capacity	C_{smin} req/day	3.75	3.75	3.75	3.75
minimal evaluation capacity	C_{emin} req/day	1.88	1.58	1.28	0.375
minimal construction capacity	C_{cmin} req/day	0.94	0.45	0.34	0.375
correct selections	$\frac{A}{A+C}$	10%	40%	70%	100%
mean time to market	*MTTM* days	81.8	33.0	29.1	30.1
mean response time	*MRT* days	1.40	1.31	1.23	1.05
RE effort share	*RES*	9%	16%	19%	13%

i.e. one tenth of the arriving requirements are of high quality and 90% are of low quality. The maximum work load, δ_{max}, is set to 80% to avoid overload and prevent too long queue lengths (a work load closer to 1 gives a much higher risk of very long queues). The number of servers in screening, evaluation, and construction is set to 1, 2, and 20 respectively, meaning that 1 engineer is working with initial requirements screening, 2 engineers work with requirements evaluation including analysis, specification, validation, prioritisation, and 20 engineering deal with construction including design, implementation, and testing. The service rates of each phase is chosen to be 4, 2, and 0.05 requirements per day respectively, which implies that for each requirement on average, screening takes 0.25 days, evaluation takes half a day, and construction takes 20 days.

Given the parameter values in Table 3 and 4, it is possible to calculate the minimum capacities required in the different steps in order to ensure stability. It is also interesting to investigate the decision quality from Table 1. Furthermore, the metrics *MTTM*, *MRT*, and *RES* reveals interesting properties of the different cases. The results are assembled in Table 5.

The minimum capacity in the screening phase is equal in all cases, as this value only depends on the arrival rates and the maximum work load. The minimum capacities in both evaluation and construction is lower in case HIGH than in case LOW, as the proportion of α -requirements is higher. This is because fewer requirements propagate when a higher percentage of α -requirements (which is only a tenth of the total number of requirements) and a lower percentage of β -requirements (which comprises nine tenths of the requirements) is selected. For similar reasons, the minimum capacities in RANDOM are higher than in the other cases.

The ratio of correct selections is defined as A/(A+C), and the LOW case results in a product with 40% correctly selected requirements, while the HIGH case has as much as 70%. The IDEAL case has 100% correct selections, and the RANDOM case has only 10% correctly selected requirements, which is equal to the golden-grain-ratio.

The Mean Time To Market decreases from 81.8 days to 29.1 days, comparing RANDOM, LOW, and HIGH. There is a slight increase of *MTTM* to 30.1 days for the IDEAL case, which is caused by the larger number of α -requirements selected for construction compared to the HIGH case. The Mean-Response-Time ranges from 1.05 days in the IDEAL case to 1.40 days in the RANDOM case.

The share of effort spent on requirements engineering is minimal 9% in the RANDOM case, and maximal 19% in the HIGH case. The IDEAL case has a *RES* of only 13%, as a higher share of the effort is spent on construction of α -requirements as a result of the perfect selection process.

5. Parameter Estimation Survey

In order to validate the feasibility of subjective estimation of model parameters, a questionnaire survey has been conducted. The survey is based on a questionnaire that is constructed to allow for checking if experts can make internally consistent estimations. If the practitioners are able to make consistent parameter estimations, it is an indication that the model is understandable and relevant to these practitioners. The values of the parameters themselves are interesting, but the survey is rather small, which makes generality of the values questionable.

5.1 Survey design and operation

The survey questionnaire is shown in Fig. 4. The questions were selected so that parameters were estimated in different ways. There are more questions than parameters, making it possible to check if the parameters are consistently related. In Fig. 5, the relations between the questions and the model parameters are given. These relations are not shown to the participants in the survey. Question g) and h) are used as "control" questions and can be compared with questions a)-e) using the equations in Fig. 5. (The survey included 4 additional questions not analysed in this study).

The survey was conducted in two steps. Firstly, it was run in a class room session during an industrial course in software architecture for practising chief architects. The course had 12 participants and 11 responded. All respondents were from companies developing software products for open markets except one, leaving 10 relevant responses. Secondly, the questionnaire was handed out during a national industry conference on software product management with around 65 participants, of which 25 responded, with companies developing software-intensive products for open markets. The respondents were from both marketing and development and had positions such as product

Questionnaire

a) How many requirements on "user level" do you get from all different sources?
(number of requirements per week)

Least possible value:	1	5	10	20	50	100	500	1000
Most probable value:	1	5	10	20	50	100	500	1000
Largest possible value:	1	5	10	20	50	100	500	1000

b) Of all incomming requirements, how large is the share of "golden grains", i.e. requirements that should be implemented with regard to market opportunity, product strategy and development resources?
0% 5% 10% 20% 25% 33% 50% 67% 75% 80% 90% 95% 100%

c) How many of all incomming requirements are rejected in an early quick screening?
0% 5% 10% 20% 25% 33% 50% 67% 75% 80% 90% 95% 100%

d) How many of the "golden grains" are *incorrectly* rejected in an early quick screening?
0% 5% 10% 20% 25% 33% 50% 67% 75% 80% 90% 95% 100%

e) How many of the requirements that undergo a deeper analysis are rejected?
0% 5% 10% 20% 25% 33% 50% 67% 75% 80% 90% 95% 100%

f) How many of the "golden grains" that undergo a deeper analysis are *incorrectly* rejected?
0% 5% 10% 20% 25% 33% 50% 67% 75% 80% 90% 95% 100%

g) How many of all requirements that are finally implemented in a typical release, should actually *not* have been implemented with regard to the real outcome of development cost and market value?
0% 5% 10% 20% 25% 33% 50% 67% 75% 80% 90% 95% 100%

h) How many of the requirements that were *not* implemented, should actually have been implemented, if knowing the facts given after the product has been out on the market for a while?
0% 5% 10% 20% 25% 33% 50% 67% 75% 80% 90% 95% 100%

Figure 4. *Questionnaire for parameter estimation.*

Questionnaire expressions

a) λ

b) g

c) $g(1-p_{e\alpha}) + (1-g)(1-p_{e\beta})$

d) $1-p_{e\alpha}$

e) $\dfrac{g \cdot p_{e\alpha} \cdot (1-p_{c\alpha}) + (1-g) \cdot p_{e\beta} \cdot (1-p_{c\beta})}{g p_{e\alpha} + (1-g) p_{e\beta}}$

f) $1-p_{c\alpha}$

g) C/(A+C)

h) B/(B+D)

Figure 5. *Equivalent parameter expression for each survey question (not shown to respondents).*

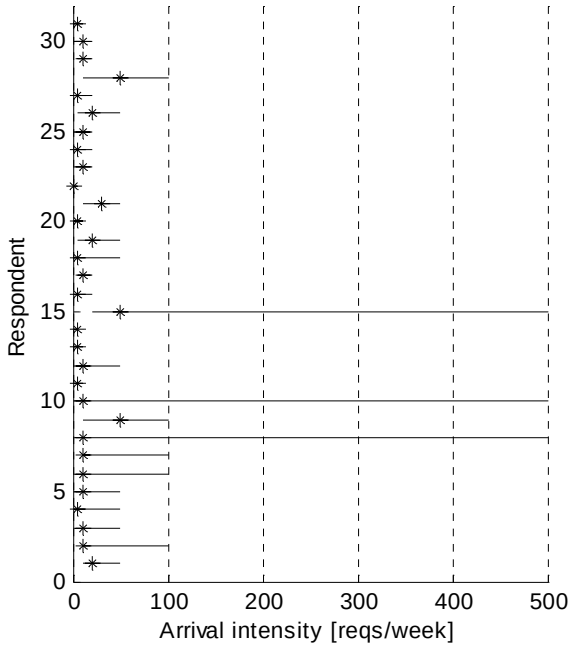

Figure 6. *Number of issued requirements per week.*

managers, system managers, sales managers, and requirements engineers. In total, from both survey occasions, we thus obtained 36 responses, of which 3 were directly discarded due to incomplete responses to many questions, leaving 33 responses as input to the survey analysis.

5.2 Survey Results

Fig. 6 shows a scatter plot of the answers to question a) regarding the rate of arriving requirements. In order to give a view of the range of the arrival rates, three values were requested: minimum, most probable, and maximum. This is illustrated in Fig. 6 by a solid line between minimum and maximum, and with an asterisk at the most probable value. Two responses were discarded as they had misunderstood the range questions and gave a maximum that was less than minimum. The minimal values range from 0 to 20 and the maximum values range from 5 to 500. The most probable values range from 1 to 50 with a mean of 13.6 reqs/week, which corresponds to an average λ of 2.7 reqs/day.

Golden-grain-ratio. The responses on the golden-grain-ratio in question b) ranges from 5% to 67% with a mean value of 21%, as shown in the box-plot in Fig. 7. The box has lines at the lower quartile, median, and upper quartile values. This result is based on 30 responses, discarding incomplete responses and one that responded a *g* of zero.

Internal estimation consistency. By using the equations of Fig. 5, it is possible to derive the propagation probabilities of each step. It is also possible to use the propagation probabilities to calculate C/(C+A) and B/(B+D) and compare with the control questions g) and h). The internal consistency of each respondent is calculated as the absolute difference between the calculated values and the responses to question g) and h). These two differences are denoted d_1 and d_2 respectively. A respondent with both d_1 and d_2 below 0.5 is considered reasonably consistent, and assumed to have understood the questions. Among the 33 survey responses, as many as 12 were internally consistent by this definition. The calculated values of the propagation probabilities and the differences between calculated and responded values are shown in Table 6, together with a calculated value of the ratio of correct selections for each of the 12 consistent responses.

Propagation probabilities. The average propagation probabilities are $p_{e\alpha}$=0.84, $p_{c\alpha}$=0.87, $p_{e\beta}$=0.61 and $p_{c\beta}$=0.56. This is in line with what can be expected, as it is reasonable for the α-probabilities to increase over time and for the β-probabilities to decrease over time in order to reflect the fact that a greater effort put on evaluation compared to screening will yield a better ability to discriminate good requirements from bad. By

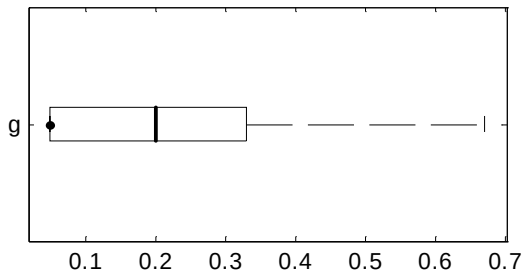

Figure 7. *A box plot of the golden-grain-ratio of 30 responses.*

Table 6. Calculated model parameter values based on 12 respondents with reasonable internal consistency.

Id	g	$p_{e\alpha}$	$p_{e\beta}$	$p_{c\alpha}$	$p_{c\beta}$	d_1	d_2	$\frac{A}{A+C}$
4	0,50	0,90	0,77	0,95	0,52	0,22	0,09	0,68
6	0,25	0,80	0,32	0,95	0,86	0,19	0,10	0,48
7	0,33	0,95	0,52	0,95	0,67	0,24	0,13	0,56
10	0,33	0,90	0,72	0,95	0,87	0,35	0,06	0,40
11	0,05	0,67	0,12	0,80	0,41	0,12	0,08	0,37
13	0,50	0,90	0,77	0,95	0,52	0,12	0,09	0,68
27	0,20	0,50	0,46	0,67	0,24	0,23	0,09	0,44
28	0,33	0,90	0,72	0,95	0,50	0,21	0,05	0,54
29	0,20	0,90	0,83	0,67	0,45	0,38	0,11	0,29
30	0,20	0,90	0,83	1,00	0,75	0,40	0,01	0,27
34	0,50	1,00	0,90	0,75	0,22	0,11	0,14	0,79
35	0,20	0,75	0,39	0,80	0,73	0,46	0,21	0,34
mean	0,30	0,84	0,61	0,87	0,56	0,25	0,10	0,49

sorting the propagation probability values in the order of $p_{c\alpha}$, $p_{e\alpha}$, $p_{e\beta}$, and $p_{c\beta}$, we should ideally get a monotonic decrease. The plot in Fig. 8 shows the four propagation probabilities in this order for all 12 consistent respondents. Solid lines connect the corresponding values for each response. Only 5 of 12 respondents estimate that the propagation probabilities are monotonously decreasing, indicating that the effort spent in evaluation does not pay off for the other 7 respondents.

Correct selection distribution. Fig. 9 shows the distribution of the calculated values of A/(A+C), i. e. the ratio of correct selections for the 12 consistent responses. The figure shows that a majority of respondents indirectly estimate that their organisations have less than half of the product content that is based on correctly selected requirements. Only one respondent provides answers that give a calculated ratio of correct selections above 75%. This is rather surprising and the results of this survey suggest that there is an opportunity of significantly improving requirements decision quality in industrial practice.

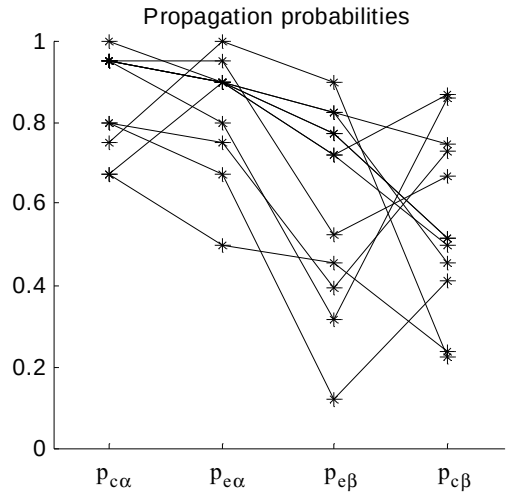

Figure 8. *Propagation probabilities.*

6. Using the Model in Practice

Practitioners can use the presented model to investigate process improvement scenarios, by calculating the consequences of hypothetical changes in productivity, staffing capacity, requirements elicitation yield and selection quality. By estimating the model parameters, a specific organisation can get valuable decision support in requirements planning, resource allocation, and training.

To illustrate the practical implications of the model, two improvement scenarios for a hypothetical organisation are given. The scenarios are

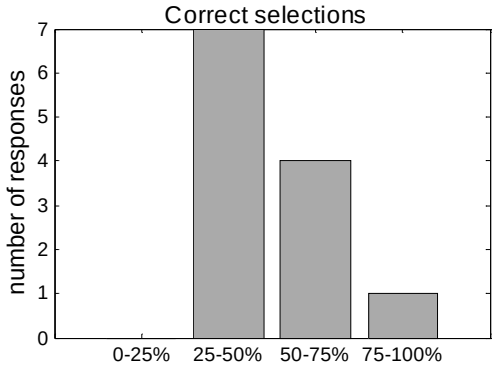

Figure 9. *Distribution of correct selections.*

based on the mean parameter estimations from the industrial survey in Section 5.

A decision quality improvement target. The ratio of correct selections, A/(A+C) in Table 5, is a measure of the decision quality in the organisation. This ratio can, for example, be increased by improving the elicitation process and thereby increasing the golden-grain-ratio so that less low-quality requirements enter the requirements repository in the first place. In practice this can be done by early eliminating duplicates and discussing the product strategy so that less irrelevant requirements are entered. An organisation with characteristics based on the mean values of the survey results from Table 6 would, e.g., need an increase in the golden-grain-ratio from 30% to 65% if demanding an increase in the ratio of correct selections from 48% to 80%. The selection quality would also benefit from improved propagation probabilities. For example, a target of 80% correct selections with a golden-grain-ratio of 30% will be satisfied by the propagation probabilities $p_{e\alpha}$=0.9, $p_{c\alpha}$=0.95, $p_{e\beta}$=0.35 and $p_{c\beta}$=0.26. Better decisions will hopefully result in product releases with a higher share of profitable features.

A Mean-Time-To-Market improvement target. Assume that the number of employees working with screening, evaluation and construction are m_s= 1 , m_e= 3 , and m_c= 20 respectively. Also, set the productivity of each employee to be μ_s= 25 , μ_e= 5 , μ_c = 0.5 requirements per week. Then the *MTTM* for this particular organisation can be calculated using the presented model. The time in each phase is 0.09, 0.52 and 5.31 weeks respectively, which sums up to 5.92 weeks, i.e. the *MTTM* is approximately 30 working days. There are two possibilities to decrease *MTTM*: increasing the number of employees or increasing each employee's productivity. For example, increasing the number of employees to 2, 4 and 30 respectively will result in a *MTTM* of 3,85 weeks, i.e. 19 days. A shorter time to market will hopefully increase the chance of reaching a higher market share.

7. Conclusions

The modelling of creative and innovative processes with a high degree of variation based on human judgement such as requirements engineering is

difficult. However, any relevant model that can help practitioners in abstracting the general properties of such a process is valuable. On the basis of a set of carefully chosen approximations, the presented work introduces an analytical model of requirements selection in product software development, which considers not only the capacity of the selection process, but also the quality of the decisions. The model is represented by a queuing network, comprised of three phases corresponding to general stages of a continuous process for product software development.

The requirements in the model are of either high or low quality and decisions about selecting or rejecting requirements are made in two steps: firstly a quick screening, and secondly a deeper evaluation. Some decisions may be incorrect and therefore some high-quality requirements may be incorrectly rejected and some low-quality requirements may be incorrectly selected. The quality of these decisions is critical to the success of any software product. The analytical model gives the possibility of expressing the following interesting properties of the process that are linked to the success of its products:

- The ratio of correct selections, giving the product quality as its share of rightly chosen requirements.
- The mean time to market, measured as the time taken from requirements issue to release of implementation.
- The mean response time, giving the time from requirements issue until feedback can be given to stakeholders.
- The lowest level of productivity and staffing needed to avoid an unstable, overloaded process.

The presented model has been validated in an industrial survey, where experts were asked to make a series of non-trivial parameter estimations. The fact that 12 out of 36 respondents were able to make internally consistent estimations indicate that the model is understandable and relevant. Surprisingly, the survey also revealed that a majority of respondents had products with less than 50% rightly selected requirements.

The presented model includes several assumptions and approximations that are needed to make the model tractable analytically. Future research is needed to investigate how these approximations affect the accuracy of

the model, in particular application domains and specific product software development organisations.

Simulation can be used when the analytical model is limited by approximations. For example, simulation can make it possible to let disposed requirements be reconsidered and re-entered into the process. It is also possible to allow for requirements decomposition into sub-requirements, as well as to take requirements bundling and dependencies into account. Industrial case studies using simulation of models, with some of the approximations removed, is thus an interesting area of further research.

Acknowledgements

This work is partly funded by the Swedish Agency for Innovation Systems (VINNOVA) under grant for The Centre for Applied Software Research at Lund University (LUCAS), Sweden. We would like to thank Dr. Christian Nyberg for his theoretical expertise and outstanding ability to explain complex queuing theory in an understandable way.

References

[1] Antoniol, G., Casazza, G., Di Lucca, G. A., Di Penta, M., Rago, F., "A Queue Theory-Based Approach to Staff Software Maintenance Centers", *IEEE International Conference on Software Maintenance* (ICSM2001), pp. 510-519, 2001.

[2] Carlshamre, P., Regnell, B., "Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes", *IEEE International. Workshop on the Requirements Engineering Process* (REP'2000), Greenwich, UK, September 2000.

[3] Carshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Natt och Dag, J. "An Industrial Survey of Requirements Interdependencies in Software Release Planning", *IEEE International Conference on Requirements Engineering* (RE'01), pp. 84-91, 2001.

[4] Higgins, S. A., de Laat, M., Gieles, P. M. C., Guerts, E. M., "Managing Product Requirements for Medical IT Products", *IEEE International Conference on Requirements Engineering* (RE'02), pp. 341-349, 2002.

[5] Höst, M., Regnell, B., Natt och Dag, J., Nedstam, J., Nyberg, C., "Exploring Bottlenecks in Market-Driven Requirements Management Processes with Discrete Event Simulation", *Journal of Systems and Software,* Vol. 59, pp 323-332, 2001.

[6] Karlsson, L., Dahlstedt, Å.G., Natt och Dag, J., Regnell, B., Persson, A., "Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study", *International Workshop on Requirements Engineering: Foundations of Software Quality* (REFSQ'02), Essen, Germany, September 2002.

[7] King, P., J., B., *Computer and Communication Systems Performance Modelling,* Prentice Hall, 1990.

[8] Kleinrock, L., *Queueing Systems*, John Wiley & Sons, 1975.

[9] Potts, C., "Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software", *Proceedings of the Second IEEE International Symposium on Requirements Engineering* (RE'95), pp. 128-30, 1995.

[10] Ramaswamy, R., "How to Staff Business-Critical Maintenance Projects", *IEEE Software*, 17(3):90-94, May/June 2000.

[11] Regnell, B., Beremark, P., Eklundh, O., "A Market-Driven Requirements Engineering Process - Results from an Industrial Process Improvement Programme", *Requirements Engineering*, 3:121-129, 1998.

[12] Sawyer, P., "Packaged Software: Challenges for RE", *Proc. 6th Int. Workshop on Requirements Engineering: Foundations of Software Quality* (REFSQ'00), Stockholm, Sweden, pp 137-142, June 2000.

[13] Yeh, A., "Requirements Engineering Support Technique (REQUEST): A Market Driven Requirements Management Process", *IEEE Second Symposium of Quality Software Development Tools*, pp. 211-223, New Orleans USA, May 1992.

PAPER IV

Post-Release Analysis of Requirements Selection Quality - An Industrial Case Study

Lena Karlsson, Björn Regnell, Joachim Karlsson, Stefan Olsson

Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Velden, Austria, June 2003.

IV

Abstract

The process of selecting requirements for a release of a software product is challenging as the decision-making is based on uncertain predictions of issues such as market value and development cost. This paper presents a method aimed at supporting software product development organisations in the identification of process improvement proposals to increase requirements selection quality. The method is based on an in-depth analysis of requirements selection decision outcomes after the release has been launched to the market and is in use by customers. The method is validated in a case study involving real requirements and industrial requirements engineering experts. The case study resulted in a number of process improvement areas relevant to the specific organisation and the method was considered promising by the participating experts.

1. Introduction

This paper presents a method for identifying improvement areas of the requirements selection process in a market-driven software product development context. The method is called PARSEQ (Post-release Analysis of Requirements SElection Quality) and is based on retrospective examination of decision-making in release planning, at a time when the consequences of requirements selection decisions are visible. PARSEQ is applied in a case study where the requirements selection for a particular release of a specific software product is analysed and improvement areas that are relevant to the studied software organisation are identified.

PARSEQ is intended to be used by software organisations that operate in a market-driven context, offering software products to many customers on an open market. Market-driven requirements engineering (RE) differs from customer-specific RE in several ways, for example in the characteristics of stakeholders and schedule constraints [17, 19]. Requirements are often invented by the developers as well as elicited from potential customers with different needs [14], and it is common to use a requirements database that is continuously enlarged with new candidate requirements [7, 15]. Commonly, market-driven software developing organisations provide successive releases of the software product and release planning is an essential activity [3, 4]. A major challenge in market-driven RE is to prioritise and select the right set of requirements to be implemented in the next release [14], while avoiding congestion in the selection process [15]. This decision-making is very challenging as it is based on uncertain predictions of the future, while crucial for the product's success on the market [3, 11].

Given issues such as uncertain estimations of requirements market value and cost of development, it can be assumed that some requirements selection decisions are non-optimal, which in turn may lead to software releases with a set of features that are not competitive or satisfy market expectations. It is only afterwards, when the outcome of the development effort and market value is apparent, it is possible to tell with more certainty which decisions were correct and which decisions were less accurate. But by looking at the decision outcome in retrospect, organisations can gain valuable knowledge of how to improve the requirements selection process and increase the chance of market success.

In [18, 5], post-mortem evaluations are discussed in a project management context. An evaluation of the project's performance after it

has been completed is useful both for personal and organisational improvement and can be conducted as an open discussion of the strengths and weaknesses of the project plan and execution. Furthermore, much can be learned about organisational efficiency and effectiveness by this kind of evaluation, which offers an insight into the success or failure of the project. The lessons learned can be used when planning forthcoming projects to improve project performance and prevent mistakes. Continuous process improvement is important in the maturity of software development and, in particular, requirements engineering is pointed out as a critical improvement area in a maturing organisation [13]. A recent process improvement study based on analysis of defects in present products is reported in [12]. The post-mortem analysis is also an excellent method for Knowledge Management, since it captures experience and improvement suggestions from completed projects [1, 16].

The PARSEQ method is evaluated in a case study, where requirements selection decisions for an already released software product were revisited by the decision-makers of the specific organisation. The market value and development cost of the requirements that were candidates for a previous release that was launched 18 months earlier, were re-estimated based on the knowledge gained during the two following releases. The re-estimation resulted in a new priority order, which in turn suggested that some selected requirements should have been postponed and some deferred requirements should have been selected for that release. Each such suspected inappropriate selection was analysed in order to understand the grounds for each decision, which in turn lead to the identification of several areas of process improvements.

The paper is structured as follows. Section 2 presents the PARSEQ methods and its main steps. In Section 3, the case study operation is described and the main results are reported. Section 4 discusses the validity of the findings and the generality of the approach outside the specific case study context. Conclusions and directions of further research are given in Section 5.

2. The PARSEQ Method

Retrospective evaluation of software release planning may give a valuable input to the identification of process improvement proposals. In particular, post-release analysis of the consequences of previous decision-making may be a valuable source of information when finding ways to improve the requirements selection process.

The PARSEQ method is based on a systematic analysis of candidate requirements from previous releases. By identifying and analysing a set of root causes to suspected incorrect requirements selection decisions, it is hopefully possible to find relevant improvements that are important when trying to increase the specific organisation's ability to plan successful software releases.

In order to perform the PARSEQ method the following foundation practices are required:

- A database with continuously incoming requirements that are dated at arrival and tagged with a refinement state.
- Methods for estimating each requirements' cost and value. The estimations are saved in the database.
- Multiple releases of the product and the requirements from prior releases are saved in the database.
- Employees who have decision-making experience from prior releases are available.

PARSEQ is divided into 4 steps: requirements sampling, re-estimation of cost and value, root cause analysis, and elicitation of improvements, as shown in Fig 1. The method uses a requirements database as input and assumes that information is available in the database regarding when a requirement is issued and in which release a requirement is implemented. The output of the method is a list of process improvement proposals. Each step in PARSEQ is subsequently described in more detail.

Requirements sampling. The main input to the post-release analysis is a list of requirements that were candidates for a previous product release that now has been out on the market for a time period long enough to allow for an assessment of the current market value of its implemented requirements. First, such a relevant previous release is selected

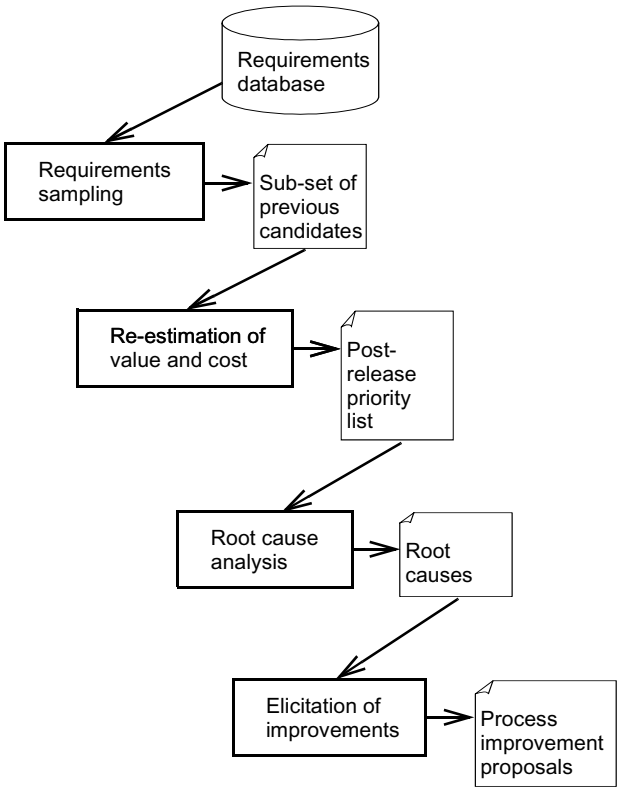

Figure 1. *An outline of the activities and products of the PARSEQ method.*

(subsequently called *reference release*). Secondly, the requirements database is examined and those requirements that were candidates for the reference release are retrieved. The previous candidates are requirements that were suggested and dated prior to the reference release, but were not implemented before the reference release, i.e. the candidate requirements were either implemented in the reference release or in a subsequent release, or they were rejected.

The purpose of the sampling is to compose a reasonably small but representative sub-set of requirements, since the complete database may be too large to investigate in the post-release analysis. The sample should include requirements that were selected for implementation in the reference release as well as postponed or rejected requirements. The requirement set is thereby useful for the analysis as it consists of typical examples of release planning decisions.

The requirements sampling can be performed in a number of ways, such as concentrating on a special market segment or on a difficult part of the product or on particularly difficult decisions. However, if the sample is supposed to represent the whole product and its market, the sample should be as broad as possible. The following types of requirements may then be excluded:

- Very similar requirements, since they do not extend the sample.
- Requirements dated several releases ago, as they may have evolved out of scope.
- Requirements dated recently, since their cost and value are not yet estimated.
- Requirements estimated to have a very long or very short implementation time, as they are atypical and likely to be split or joined.

The output from the requirements sampling is a reasonable amount of requirements, high enough to be representative, yet low enough to allow the following steps of PARSEQ to be completed within reasonable time.

Re-estimation of value and cost. The requirement sample is input to the next step of PARSEQ, where a re-estimation of current market value and actual development cost is made in order to find suspected inappropriate decisions that can be further analysed. As the reference release has been out on the market for a while, a new assessment can be made, which applies the knowledge gained after the reference release was launched, which presumably should result in more accurate priorities. The re-estimation is made to find out how the organisation had decided for the reference release, i.e. which requirements that would have been selected, if they knew then what they know now. With todays knowledge, about market expectations and development costs, a different set of requirements may have been selected for implementation in the reference release. If this is not the case, either the organisation has not learned anything since the planning of the reference release, or the market has not changed at all.

The implemented requirements have a known development cost (assuming that outcome of the actual implementation effort is measured for each requirement), but the postponed or rejected requirements need to be re-estimated based on the eventual architectural decisions and the knowledge gained from the actual design of the subsequent releases.

By using, for example, a cost-value prioritisation approach with pairwise comparisons [9, 10], an ordered priority list can be obtained where the requirements with a higher market value and a lower cost of development are sorted in the priority order list before the requirements with a lower market value combined with a higher development cost.

The purpose of the re-estimation is to apply the knowledge that has been gained since the product was released, to discover decisions that would have been made differently today. The discrepancies between the decisions made in the planning of the reference release and the post-release prioritisation are noted and used in the root cause analysis. The output of this step is thus a list of requirements that was given a high post-release priority but were not implemented in the reference release, as well as requirements with a low post-release priority but still implemented in the reference release.

Root cause analysis. The purpose of the root cause analysis is to understand on what grounds release-planning decisions are made. By discussing the decisions made in prior releases, it may be possible to create a basis for the elicitation of process improvement proposals.

The output of the re-estimation, i.e. the discrepancies between the post-release prioritisation and what was actually selected for implementation in the reference release, is analysed in order to find root causes to the suspected inappropriate decisions. This analysis is based on a discussion with persons involved in the requirements selection process. The following questions can be used to stimulate the discussion and provoke insights into the reasons behind the decisions:

- Why was the decision made?
- Based on what facts was the decision made?
- What has changed since the decision was made?
- When was the decision made?
- Was it a correct or incorrect decision?

Guided by these questions, categories of decision root causes are developed. Each requirement is mapped to one or several of these categories to illustrate the decision disposition. This mapping of requirements to root cause categories is the main output of this step together with the insights gained from the retrospective reflection.

Elicitation of improvements. The outcome of the root cause analysis is used to facilitate the elicitation of improvement proposals. The objective of this last step of PARSEQ is to arrive at a relevant list of high-priority areas of improvement. The intention is to base the discussion on strengths and weaknesses of the requirements selection process and to identify changes to current practice that can be realised. The following questions can assist to keep focus on improvement possibilities:

- How could we have improved the decision-making?
- What would have been needed to make a better decision?
- Which changes to the current practices can be made to improve requirements selection in the future?

The results of PARSEQ can then be used in a situated process improvement programme where process changes are designed, introduced and evaluated. These activities are, however, out of the scope of the presented method.

3. Case Study

PARSEQ was tried out in a case study to investigate its feasibility and gain more knowledge for future research on post-release analysis of requirements selection as a vehicle for process improvement. In the first section of this chapter, the case study site and context is described as well as the tool used in the study. Next, the realisation of the PARSEQ method is described, i.e. how each step of the method was carried out in the case study. Finally, the results from the case study are reported, including a number of improvement proposals.

3.1 Background

The case study site is a small-sized organisation developing stand alone software packages. The organisation stores the requirements for the software package in a database that contains already implemented requirements as well as suggestions for new requirements. Each requirement is tagged with a certain state to describe its level of refinement. Examples of states include New, Accepted for prioritisation,

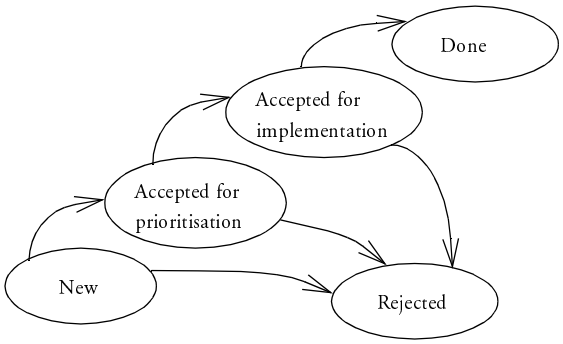

Figure 2. *A simplified version of the requirement state model in the database.*

Accepted for implementation and Done, see Fig. 2. When a requirement for some reason is not appropriate for the package, its state is set to Rejected. Other states include Clarification needed, Insignificant improvement, Badly documented, Duplicate and Draft.

To analyse the requirements in the database a commercial tool for product management and requirements management, Focal Point[1] was applied. Focal Point has capabilities for eliciting, reviewing, structuring, and prioritising requirements as well as for planning optimal releases that maximise the value for the most important customers in relation to development time and available resources. One prioritisation method in Focal Point is pair-wise comparisons [9]. It is helpful for keeping up concentration and objectivity and Focal Point also provides solutions for reducing the number of comparisons and motivating the priorities. This tool also aids in visualising the decision in a number of different chart types. Due to redundancy of the pair-wise comparisons, the tool also includes capabilities such a consistency check that describes the amount of judgement errors that are made during the prioritisation.

3.2 Operation

The participating anonymous organisation was given the task to use PARSEQ to reflect on a set of decisions made during prior releases. The

1. For more information see www.focalpoint.se.

case study was executed during a one-day session, with approximately 5 hours of efficient work.

Requirements sampling. A release that was launched 18 months ago was selected as reference release, and since then another release has been launched and yet another one is planned to be released in the near future.

The requirements database contains more than 1000 requirements that were issued before the reference release and implemented in either that release or postponed to one of the following ones. Of these requirements, 45 was considered a reasonable number to extract. The requirements were equally allocated over the three releases: A, B and C, i.e. 15 were implemented in the reference release A, 15 in release B and another 15 were planned for release C.

Note that the releases were not equally large in terms of number of requirements, i.e. the samples are not representative. The 15 requirements from release A were selected among 137 requirements, while the releases B and C only consisted of 28 and 26 requirements, respectively as shown in Fig. 3.

The requirements were selected randomly from a range where the ones estimated as having a very high, or very low, development effort had been removed, since they are not considered as representative. Very similar requirements had also been excluded to get an as broad sample as possible, as well as very new ones as development costs had not been estimated.

All market changes, architectural decisions and new knowledge gained during the 18 months between the reference release A and release C could

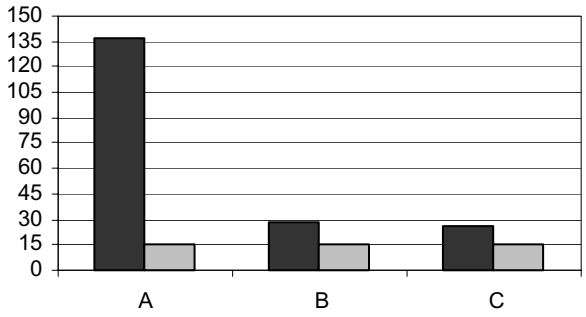

Figure 3. *Number of implemented requirements (dark grey) in each release compared to the sample (light grey).*

be applied. The selected requirements are all in the states Done or Accepted for implementation; no rejected or postponed requirements were considered in the analysis. The requirements sampling took approximately one hour and was performed by a developer before the session.

Re-estimation of cost and value. The re-estimation was performed to find out what requirements the organisation would have selected for release A if they knew then what they know now. With the knowledge gained since the reference release was planned, it is possible that a different set of requirements would have been selected. However, it is important to note that one additional requirement in the release would imply that another one has to be removed, in order to keep the budget and deadline.

The market value was estimated using pair wise comparisons and the cost was estimated in number of hours, based on expert judgement. The following question was used in the pairwise comparison of the candidates to the reference release: "Which of the requirements would, from a market perspective, have been the best choice for release A?". This question was carefully chosen with the objective of enforcing focus on the retrospective nature of the estimation. Thus, the assessment concerned the market value given what is known today, and not whether the decisions made during the reference release were correct or not, given the knowledge available at that time.

The 45 requirements were re-estimated by using the Focal Point tool and pair-wise comparisons to prioritise them based on the selected question. The prioritisation was performed by a marketing person, who has good knowledge of customer demands, guided by a developer, and was attended by the two researchers. Note that both the marketing person and developer had performed the original estimations as well, otherwise the results may be biased by differences in personal opinions rather than a desired effect of changes in priorities over time. When uncertainties or disagreements of a comparison were discovered, the issue was briefly discussed to come to an agreement. The consistency check showed that the prioritisation was carefully performed and only two comparisons had to be revised and changed.

The total time of the prioritisation was just over one hour, in which 70 comparisons were made. The short time is thanks to the algorithms in the tool, which reduces the number of comparisons and points out the

inconsistencies among the comparisons [2, 8]. Otherwise, the number of comparisons would have been n(n-1)/2, which in this case equals 990.

The development cost of the requirements that were actually implemented was known, while the development cost of the requirements that are planned for a coming release had to be re-estimated. However, it was decided to use the available cost estimations, since the estimates recently had been reviewed and updated.

A bar chart was created in the Focal Point tool to visualise and facilitate analysis of the decisions, see Fig. 4. The grey bars illustrate the requirements implemented in release A, and the white bars represent requirements implemented or planned for release B or C. The prioritisations were performed on a ratio scale with nine steps including a neutral value. The pair-wise comparisons were input to matrix calculations implemented in Focal Point and used to calculate eigen-values, averages and normalised relative priorities in the range between 0 and 1 (for more information on the details, see [8, 9]). Thus, it is possible to subtract the cost from the value, getting a *resulting priority*, which is marked by the black arrows in the bar chart [6]. The bars are sorted on their resulting priority from top down. Thus the bar chart shows the ideal order in which requirements should be implemented if only customer value and development costs were to be considered. The bar chart does not take requirements dependencies into account.

Some of the requirements were not identified in release A, but turned out to be important when they later were identified. Furthermore, requirements interdependencies, release themes and architectural choices complicate the situation and thus this ideal order is not the most suitable in reality.

In an ideal case, the requirements at the top of the bar chart would have consisted of requirements from release A. The requirements at the top of the bar chart are estimated as having the highest value and the lowest cost and should therefore be implemented in an as early release as possible. The requirements at the bottom are estimated as having the lowest value and the highest cost and should therefore be implemented in a later release or, in some cases, not at all.

The bar chart illustrates the discrepancies between the two estimation occasions and points out the requirements to discuss.

Root cause analysis. The bar chart is used in the Root cause analysis, to find out the rationale for the release-planning decisions. The discussion

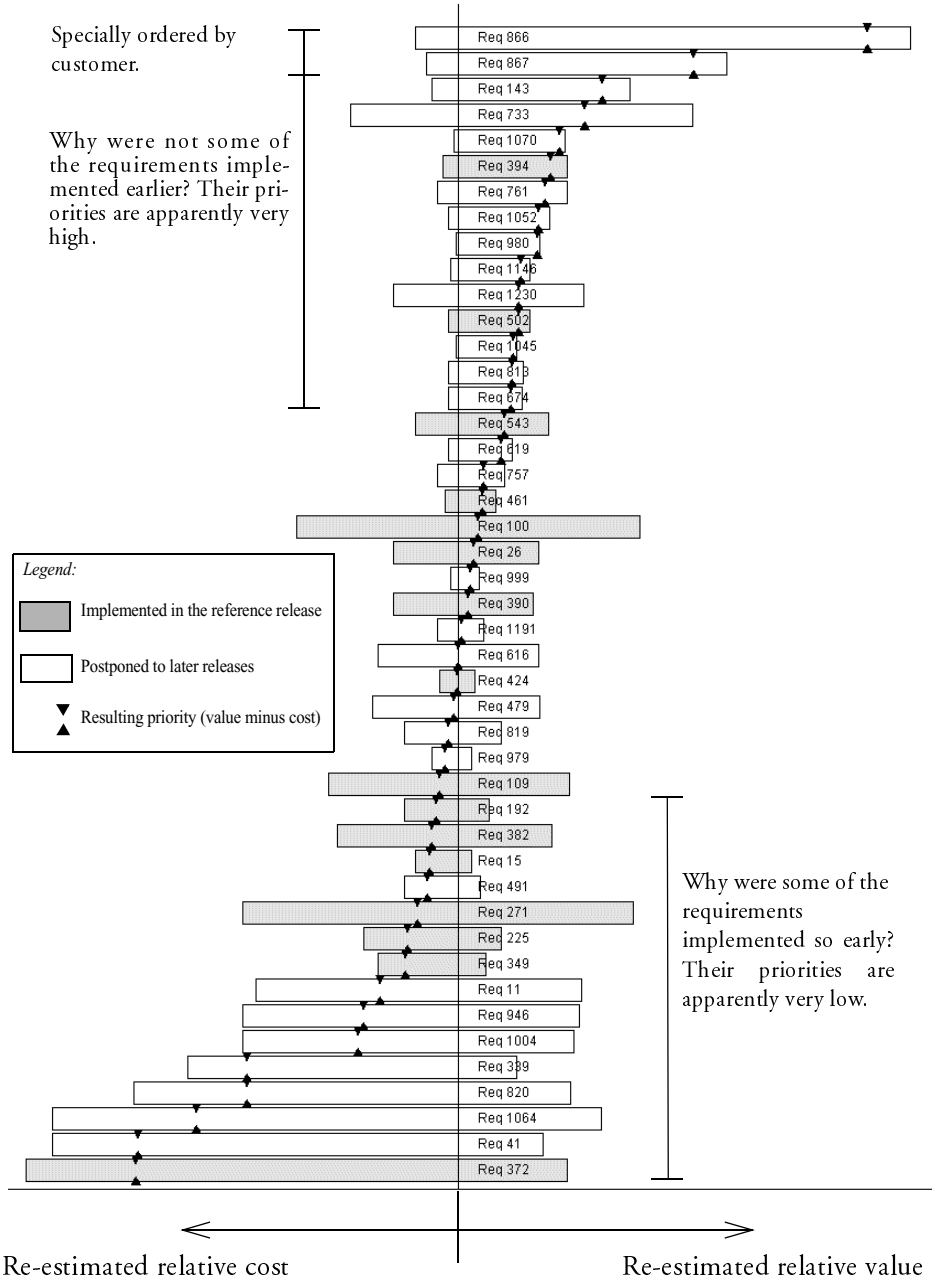

Figure 4. *Bar chart from the post-release analysis of the requirements in the database using the Focal Point tool.*

was attended by three representatives from the organisation: one marketing person and two developers, as well as the two researchers.

The top 15 requirements were scanned to find the ones that were estimated differently in the re-estimation, i.e. the ones that originate from release B or C. These were discussed to answer the main question "Why wasn't this implemented earlier?" and motivations to the decision was stated by the participants. In a similar manner, the 15 requirements at the bottom of the bar chart were investigated, to find the ones that originate from release A and B. These requirements were discussed concerning the question "Why did we implement this so early?". Notes were taken of the stated answers for later categorisation of the release-planning decision root causes.

After the meeting, the researchers classified the stated decision root causes into a total of 19 different categories, inspired by the notes from the meeting. A sheet with the requirements that had been discussed during the root cause analysis was compiled, which the organisation representatives used to classify the requirements. The result from the classification is displayed in Table 1 and Table 2, where 4 categories have been removed as they were not used.

Elicitation of improvements. Another purpose of the case study was to capture improvement proposals by encouraging the participants to, in connection with each requirement, state some weak areas in need of improvement. This also appeared to be difficult since each decision was dependent on the specific context or situation. Therefore, no list of improvement proposals was compiled at this stage. Instead, more generic improvement proposal areas were elicited by investigating Table 1 and Table 2 and the notes taken from the root cause analysis discussion. This is described below.

3.3 Results

The case study showed that it was possible to use the proposed method in practice. The release-planning decisions that were made in prior releases could be categorised and analysed and process improvement areas could be identified. The results indicate that the organisation has gained a lot of knowledge since the planning of the reference release, which is a promising sign of evolution and progress.

Table 1. "Why was this requirement implemented so early?"

	Root Causes	Req 192	Req 382	Req 15	Req 271	Req 225	Req 349	Req 372	Req 41
Implem. issues	RC1: Under-estimation of development effort	■	■		■				
	RC2: Part of release theme		■		■			■	■
	RC3: A quick fix to provide customers opportunity to give feedback		■		■				■
Customer issues	RC4: Requirement ordered by a specific customer								■
	RC5: Requirement specifically important for a key customer				■			■	■
	RC6: Over-estimation of customer value			■		■	■	■	■
	RC7: Impressive on a demo					■		■	■
	RC8: Competitors have it, therefore we must also have it							■	
	RC9: Competitors do not have it; gives competitive advantage			■		■			■

Table 2. "Why was this requirement not implemented earlier?"

	Root Causes	Req 143	Req 733	Req 1070	Req 761	Req 1052	Req 980	Req 1146	Req 1045	Req 813	Req 674	Req 866	Req 867
Implementation issues	RC10: Over-estimation of development effort	■							■	■			
	RC11: Insufficient understanding of scale-up effects			■				■					
	RC12: No good design solution available		■		■								
	RC13: Sub-optimal decision based on requirements partitioning		■										
	RC14: Only partial implementation in a first increment				■	■	■				■		
Cus. issues	RC15: Requirement ordered by a specific customer											■	■

The causes for implementing requirements earlier than necessary are shown in Table 1. Most of the root causes originate from wishing to satisfy customer demands, either one specific customer or the whole market. However, the evaluation showed that the customer value was not as high as expected. On the other hand, it is difficult to measure "good-will" in terms of money, and therefore these decisions may not be essentially wrong. Other root causes of implementing requirements earlier than necessary concern implementation issues, such as incorrect effort estimations, which lead us to believe that estimations ought to be more firmly grounded. Another reason concerns release themes which is a kind of requirements interdependency that is necessary to respect. Developing and releasing small increments of requirements, in order for customers to give feedback early, is a good way of finding out more exactly what customers want, while assigning a low development effort.

As Table 2 shows, the reasons for implementing requirements later than optimal mainly apply to implementation issues. The category complying with the most requirements regards partial implementation in a first increment, which means that it was implemented earlier, but only partially and therefore the requirement remains.

The root cause tables and the material from the discussion were used in the investigation of possible improvement areas. Five areas were found, which could be linked to the root causes, which are described below.

Trim the division of large requirements into smaller increments.

The manner in which large requirements, affecting several components or having a large implementation effort, are divided into smaller increments can be more thoroughly investigated. The division can be done for several reasons: to get customer feedback at an early stage, to investigate alternative design solutions or to make small incremental improvements of the functionality. Root causes number 3 and 14 deal with requirements developed in increments and the discussions resulted in the idea that the organisation would benefit from an improved increment planning.

Enhance the overall picture of related requirements

Some requirements were acknowledged as being related to other requirements due to involving the same feature. These would probably have benefited from creating an overall picture of the release so that all

aspects of the specific feature were accounted for. In some cases a feature involved several requirements and after implementing some of them the developers felt content. The related requirements could instead have been designed concurrently in one larger action to avoid sub-optimal solutions. It would also have helped in identifying the most important requirements for that feature. These requirements relations could be taken into consideration more carefully as root cause number 13 describes.

Additional elicitation effort for usability requirements

It was recognised that the requirements dealing with the user interface did not fulfil some special customer needs, as described by root cause number 11. The problem concerned scale-up effects and could have been discovered through a more thorough requirements elicitation. Actions to take include building prototypes and asking customers with special user interface needs.

Improve estimations of market-value of features in competing products

It seems that many requirements were implemented with the objective of outperforming competitors, as reflected in root cause number 7, 8 and 9. However, looking too much at what competitors have or what may look nice on a prototype or demo may bring less value to the product than expected. The value estimations of the competitors' products may need to be improved.

Improve estimations of development effort

Root causes number 1 and 10 concern over- and underestimations of the development effort. Results from an earlier study indicate that the release plan is very dependent on accurate time estimates, since the estimates affect how many of the requirements that are selected [11]. Under-estimation may result in an exceeded deadline and over-estimation may exclude valuable requirements. Improving this area may enhance release-planning and requirements selection quality.

4. Discussion

The case study participants found the one-day exercise interesting and instructive. They all agreed that it was valuable to reassess previous releases and reflect on the decisions made. It was during the root cause analysis that the most learning occurred since the discussions between the participants were very fruitful. A set of improvement issues to bear in mind during requirements selection was assessed as valuable for future releases.

Despite the fact that 20 out of 45 requirements were assessed as belonging to the wrong release, there were few decisions that were essentially wrong. Keeping in mind the knowledge available at the time of the reference release, most release-planning decisions were correct, i.e. market opportunities and risks have to be taken, incremental development is applied and only a limited amount of time can be assigned to requirements elicitation and evaluation. However, no matter how successful organisation or product, there are always room for improvements.

There are a number of validity issues to consider in the case study. First of all, the data was not extracted from a representative sample because the releases varied in size. Therefore there are probably many more requirements from the largest release that would be interesting to consider. Since the data only included requirements that were implemented or postponed and no rejected requirements, there would be more decisions to consider in a more thorough evaluation.

The criterion that was used to capture the true value of the requirements appeared to be somewhat difficult to use. Since the development cost was known in most cases, it was difficult for the participants to concentrate on the customer value only, without implicitly taking the cost into account. It was also difficult to, in retrospect, consider the reference release and the value at that particular time without regard of the situation today.

The prioritisation itself is also a source of uncertainty; when not performed thoroughly, the bar chart may not show the appropriate requirements priorities. Since the prioritisation is based on subjective assessments, it is highly dependent on the persons involved. Nevertheless, the consistency check proved that the prioritisation was performed carefully and few judgment errors were made [9, 10].

As discussed in [1], it is important to consider the organisation's situation. If there are unfinished activities or underlying conflicts, the atmosphere may not be appropriate for discussing the project's problems. Then the post-release analysis may cause participants to blame each other instead of learning from the mistakes.

Finally, the decision categories that emerged during the root cause analysis may not reflect the typical kinds of decisions. A different set of requirements would probably generate a different set of categories, and therefore these shall not be used by themselves. It is also possible that the categories are formulated vaguely or incorrectly, so that their interpretations differ.

The presented improvement areas are specific to the particular case study organisation and need to be examined in further detail to point out the exact measures to take. However, the participants state that the exercise itself, imposing thought and reflection, may be more fruitful than the particular improvement proposals.

5. Conclusions

The presented method for post-release analysis of requirements selection quality, called PARSEQ, was tested in a case study where candidate requirements for a previous release were evaluated in retrospect. The case study demonstrated the feasibility of the method in the context of the specific case and the results from the case study encourage further studies of the method. This may support the hypothesis that the method is generally applicable in the improvement of industrial processes for market-driven requirements engineering in product software development.

The following areas are interesting in further investigations of PARSEQ:

- *Include rejected requirements.* The case study only included requirements that were planned for implementation in the reference release or postponed to coming releases. It would be interested to go through the set of rejected requirements and see if there exist suspected inappropriate rejections, which may be of valuable input to the elicitation of improvements.

- *Selection quality metrics*. Given that the requirements sample is representative to the distribution of appropriate and inappropriate decisions, it may be possible to use PARSEQ to provide numerical estimations of the selection quality in terms of fractions of "good" and "bad" decisions.
- *Connect improvement proposals and root-causes*. It is fairly easy to extract root-causes from the discussion on misjudged requirements. However, advancing from root-causes to improvement proposals appeared more difficult. More investigation into support for finding improvement proposals is needed.
- *Generalisation of root cause categories*. If many case studies applying PARSEQ are carried out in various contexts, it may be possible to derive a complete and generally applicable set of root cause categories that are common reasons for inappropriate decisions. This knowledge may be very valuable in the research of requirements engineering methods in the product software domain.

Acknowledgements

The authors would like to thank the participating anonymous organisation for the industrial requirements engineering expertise and confidential data, without which this study would not have been possible. We would also like to thank Magnus Höglund at Focal Point for contributing to this work with his valuable time and knowledge.

References

[1] Birk, A., Dingsoyr, T., Stålhane, T., "Postmortem: Never Leave a Project without It", *IEEE Software*, pp.43-45, May/June 2002.

[2] Carmone, F.J., Kara, A., Zanakis, S.H., "A Monte Carlo Investigation of Incomplete Pairwise Comparison Matrices in AHP", *European Journal of Operational Research*, Vol 102, pp. 538-553, 1997.

[3] Carlshamre, P., Regnell, B., "Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes", *IEEE International. Workshop on the Requirements Engineering Process* (REP'2000), Greenwich, UK, September 2000.

[4] Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Natt och Dag, J., "An Industrial Survey of Requirements Interdependencies in Software Release Planning", *IEEE International Conference on Requirements Engineering* (RE'01), pp. 84-91, 2001.

[5] Cleland, D.I., *Project Management,* McGraw-Hill, 1995.

[6] Fenton, N.E., *Software Metrics - A Rigorous Approach*, Chapman & Hall, 1994.

[7] Higgins, S. A., de Laat, M., Gieles, P. M. C., Guerts, E. M., "Managing Product Requirements for Medical IT Products", *IEEE International Conference on Requirements Engineering* (RE'02), pp. 341-349, 2002.

[8] Karlsson, J., Olsson, S., Ryan, K., "Improved Practical Support for Large-scale Requirements Prioritising", *Requirements Engineering,* Vol 2, pp. 51-60, 1997.

[9] Karlsson, J., Ryan, K., "A Cost-Value Approach for Prioritizing Requirements", *IEEE Software*, pp. 67-74, Sept/Oct 1997.

[10] Karlsson, J., Wohlin, C., Regnell, B. "An Evaluation of Methods for Prioritizing Software Requirements", *Information and Software Technology*, Vol 39(14-15): 939-947, 1998.

[11] Karlsson, L., Dahlstedt, Å.G., Natt och Dag, J., Regnell, B., Persson, A., "Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study", *International Workshop on Requirements Engineering: Foundations of Software Quality* (REFSQ'02), Essen, Germany, September 2002.

[12] Lauesen, S., Vinter, O., "Preventing Requirements Defects: An Experiment in Process Improvement", *Requirements Engineering* Vol 6:37-50, 2001.

[13] Paulk, M. C., Weber, C. V., Curtis, B., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley, 1995.

[14] Potts, C., "Invented Requirements and Imagined Customers: Requirements Engineering for Off-the-Shelf Software", *Proceedings of the Second IEEE International Symposium on Requirements Engineering* (RE'95), pp. 128-30, 1995.

[15] Regnell, B., Beremark, P., Eklundh, O., "A Market-Driven Requirements Engineering Process - Results from an Industrial Process Improvement Programme", *Requirements Engineering*, 3:121-129, 1998.

[16] Rus, I., Lindvall, M., "Knowledge Management in Software Engineering", *IEEE Software*, pp.26-38, May/June 2002.

[17] Sawyer, P., "Packaged Software: Challenges for RE", *Proc. 6th Int. Workshop on Requirements Engineering: Foundations of Software Quality* (REFSQ'00), Stockholm, Sweden, pp 137-142, June 2000.

[18] Ulrich K.T., Eppinger, S.D., *Product Design and Development*, McGraw-Hill, 2000.

[19] Yeh, A., "Requirements Engineering Support Technique (REQUEST): A Market Driven Requirements Management Process", *IEEE Second Symposium of Quality Software Development Tools*, pp. 211-223, New Orleans USA, May 1992.

PAPER V

Simple Is Better? -An Experiment on Requirements Prioritisation

Lena Karlsson, Patrik Berander, Björn Regnell, Claes Wohlin

Proceedings of the 3rd Conference on Software Engineering Research and Practise in Sweden (SERPS'03), Lund, Sweden, October 2003.

V

Abstract

The process of selecting the right set of requirements for a product release is highly dependent on how well we succeed in prioritising the requirements candidates. There are different techniques available for requirements prioritisation, some more elaborate than others. In order to compare different techniques, a controlled experiment was conducted with the objective of understanding differences regarding time consumption, ease of use, and accuracy. The requirements prioritisation techniques compared in the experiment are the Analytical Hierarchy Process (AHP) and a variation of the Planning Game (PG), isolated from Extreme Programming. The subjects were 15 Ph.D. students and one professor, who prioritised mobile phone features using both methods. It was found that the straightforward and intuitive PG was less time consuming, and considered by the subjects as easier to use, and more accurate than AHP.

1. Introduction

Software requirements need to be prioritised when the elicitation process has yielded more requirements than can be implemented at once. There exist a number of different techniques and tools to use for requirements prioritisation. However, some software organisations may not have enough resources to buy or develop a tool and therefore it is interesting to investigate techniques that do not need computer support.

This paper describes an experiment aimed at comparing two requirements prioritisation techniques. The intention with the experiment is to compare a rudimentary prioritisation technique (Planning Game) with a more elaborate one (Analytical Hierarchy Process). The main variables that were investigated were the difference in time-consumption, accuracy, and ease of use. The experiment was performed during a one-day session with 15 Ph.D. students and one professor as subjects. Instead of real requirements, the subjects prioritised features of mobile phones, which is a well-known product with a range of features to choose from.

In order to investigate the trade-off between low price and high value, the prioritisation was performed with respect to both Price and Value. The experiment also aimed at investigating if the preferred choice of prioritisation technique depended on the number of features involved.

As expected, the results indicate that the more rudimentary technique was less time-consuming and a majority of the subjects found it easier to use. Most subjects also found the results from the rudimentary technique more accurate, which is a bit surprising.

The paper is structured as follows. Section 2 explains and discusses the matter of requirements prioritisation in general and the two compared techniques in particular. Section 3 describes the design of the experiment and brings up some validity issues. Further, Section 4 presents the results discovered in the experiment while Section 5 discusses what the results may imply. Finally, the paper is concluded in Section 6.

2. Requirements Prioritisation

The ultimate goal of any software organisation is to create systems that meet the stakeholder demands. Since there are usually more requirements than can be implemented, decision makers must face the dilemma of

selecting the right set of requirements for their next product release. In order to select the correct set of requirements, the decision makers must understand the relative priorities of the requested requirements [18]. By selecting a subset of the requirements that are valuable for the customers, and can be implemented within budget, organisations can become more successful on the market. There are several different techniques to choose from when prioritising requirements. Some techniques are based on more or less structured sorting algorithms, while others use pair-wise comparisons or numeral assignment [6].

The two techniques compared in this paper are (1) the Analytical Hierarchy Process (AHP) that is based on pair-wise comparisons [14], and (2) the Planning Game (PG) [1] that uses a sorting algorithm. The two techniques are further described below.

2.1 Analytical Hierarchy Process (AHP)

AHP is a decision-making method that involves comparing all possible pairs of requirements, in order to determine which of the two is of higher priority, and to what extent. If there are *n* requirements to prioritise, the total number of comparisons to perform is *n(n-1)/2*. This relation results in a dramatically increasing number of comparisons as the number of requirements increases. However, due to redundancy of the pair-wise comparisons, AHP is rather insensitive to judgement errors. Furthermore, AHP includes a *consistency check* where judgement errors can be identified and a *consistency ratio* can be calculated.

In AHP, any system structure can be abstracted into a hierarchy that explains the system's components and their functions. Hence, AHP takes the whole system into account during decision-making since it prioritises the components on each level in the hierarchy [14].

Karlsson *et al.* [10] performed an evaluation of six different prioritisation techniques based on pair-wise comparisons, including AHP. The authors concluded that AHP was the most promising approach because it is based on a ratio scale, is fault tolerant, and includes a consistency check. AHP was the only technique in the evaluation that satisfied all these criteria. Furthermore, it includes a priority distance, i.e. a *ratio scale*, while the other approaches only provided the preferred order. However, because of the rigour of the technique, it was also the most time-consuming one in the investigation.

Since the major disadvantage of AHP is the time consumption for large problems, different investigations have been performed in order to decrease the number of comparisons, and thus the time needed [4, 15]. The results of these have been that it is possible to reduce the number of comparisons with as much as 75 % [7]. However, when reducing the number of comparisons, the number of comparisons that are redundant is also reduced, and hence the possibility to identify inconsistent judgements [10].

2.2 Planning Game (PG)

In the last years, there have been an increased use and interest in agile methodologies, such as Extreme Programming (XP). Agile methodologies are based on streamlined processes, attempting to reduce overhead such as unnecessary documentation. The interest and use of agile methodologies have been both from industry and academia. Tom De Marco has aligned to this interest and have expressed that *"XP is the most important movement in our field today"* [2].

XP is composed of 12 fundamental practices and the Planning Game (PG) is one of them. For the purpose of this experiment we have isolated PG despite that the practices affect each other according to [1].

PG is used in planning and deciding what to develop in a XP project. In PG, requirements (written on so called Story Cards) are elicited from the customer. When the requirements are elicited, they are prioritised by the customer into three different piles: (1) those without which the system will not function, (2) those that are less essential but provide significant business value, and (3) those that would be nice to have [1].

At the same time, the developers sort the requirements by risk into three piles: (1) those that they can estimate precisely, (2) those that they can estimate reasonably well, and (3) those they cannot estimate at all. Further, the developers estimate the time required to implement each requirement.

Based on the time-estimates, or by choosing the cards and then calculating the release date, the customers prioritise the requirements within the piles and then decide which requirements that should be planned for the next release [12].

The result of this easy and straightforward technique is a sorted vector of requirements. This means that the requirements are represented as a

ranking on an *ordinal scale* without the possibility to see how much more important one requirement is than another.

2.3 Cost-Value Trade-Off

When prioritising requirements, it is often not enough to just prioritise how much value the requirement has to the customers. Often other factors such as risk, time, cost and requirements interdependencies should be considered before deciding if a requirement should be implemented directly, later, or not at all. For example, if a high-priority requirement would cost a fortune, it might not be as important for the customer as the customer first thought [11]. This means that it is important to find those requirements that provide much value for the customers at the same time as they cost as little as possible. Or as Wiegers puts it: *"Prioritisation means balancing the business benefit of each requirement against its cost and any implications it has for the architectural foundation and future evolution of the product"* [18].

Karlsson and Ryan [8] use AHP as an approach for prioritising both Value and Cost in order to implement those requirements that give most value for the money. The data can be further used to provide graphs to visualise the Value-Cost ratio between the requirements.

In PG, a similar approach is taken when requirements are prioritised based on both customer value and implementation effort. The information that could be extracted from PG should hence be possible to use in the same way as it was used in [8] with the difference that the result from PG is based on an ordinal scale instead of a ratio scale.

3. Experiment Design

This section describes the experiment approach and execution as well as the analysis performed by the researchers. Finally, it is concluded with a number of validity issues.

3.1 Experiment Approach

The experiment was carried out with a *repeated measures design, using counter-balancing* [13, 19]. The 16 subjects in the convenient sample included 15 Ph.D. Students in their first or second year, and one

professor. The experiment was carried out during a one-day session, which included an introduction to the task, the experiment itself, a post-test, and finally a concluding discussion of the experiment implementation. In addition, before the experiment a pre-test was performed, and a few weeks after the experiment a second post-test was conducted.

The two requirements prioritisation techniques described above (Section 2.) were used as input to the experiment, but were modified in order to be further comparable. The system aspect of AHP was not considered, and thus there is only one level of the hierarchy in this investigation. [14]

In PG, the piles were labelled according to Value and Price: (1) Necessary, (2) Adds to the value and (3) Unnecessary, and (1) Very high price, (2) Reasonable price and (3) Low price, respectively. In practice, PG is performed by a customer representative and a developer, but in this experiment each subject had to play both roles.

3.1.1 Research Hypotheses. The goal of the experiment is to compare two prioritisation techniques and to test the following hypotheses:

1. The average time to conclude the prioritisations is larger when using AHP.
2. The ease of use is considered higher for PG.
3. AHP reflects the subjects' views more accurately.

The objective dependent variable *average time to conclude the prioritisations* was captured by measuring each subject's time to conclude the tasks. The subjective dependent variables *ease of use* and *reflecting the subjects' views* were captured by questionnaires after the experiment.

3.1.2 Pilot Experiment. A pilot experiment was performed before the main study to evaluate the design. Six colleagues participated and they prioritised ten features each, with both techniques. After this pilot experiment, it was concluded that the experiment should be extended to 8 and 16 features in order to capture the difference depending on the number of factors to prioritise. Another change was to let the subjects use the techniques and criteria in different orders to eliminate order effects. Further, changes to the AHP sheets included to remove the scale and instead use the "more than" and "less than" signs so that the participants

would not focus on the numbers, and to arrange the pairs randomly on each sheet.

3.1.3 Pre-Test. Before the session, the subjects were exposed to a *pre-test* in order to get a foundation for sampling. A questionnaire was sent out by e-mail in order to capture the knowledge about mobile phones and the subjects' knowledge and opinion of the two prioritisation techniques. The pre-test was used to divide the subjects by random into two groups with as similar characteristics as possible.

Another objective with the pre-test was to investigate how well the subjects could apprehend the price of mobile phone features. Nine of the 16 subjects stated that they consider buying a new mobile phone at least every second year, and therefore we believe that their knowledge of mobile phone prices is fairly good.

3.1.4 Experiment Execution. The domain in this experiment was mobile phones and according to the pre-test, all subjects were familiar with this context. The factors to prioritise were mobile phone features, for example SMS, Games, WAP, Calendar, etc. In this experiment, the prioritisation criteria were *Value for me*, which corresponds to how important and interesting the subject find the feature, and *Added price on the phone*, which is an estimation of how much the feature might add to the actual mobile phone price. Note that this is not the same as development cost, which would be difficult for laymen to estimate.

The Value criterion has probably been regarded by most of the subjects when buying or considering buying a mobile phone. The Price criterion may also be accounted for since considering buying and comparing mobile phones gives a clue of how much the features add to the price. Thus, there is a trade-off between Value and Price when buying a mobile phone.

One intention of the experiment was to investigate if a different number of requirements would affect the choice of preferred technique. Therefore, half of the subjects were asked to prioritise 8 features, while the other half prioritised 16 features. Another intention was to investigate if the order in which the techniques were used would affect the choice of preferred technique. Therefore, half of the subjects started with AHP and half started with PG. The order of the Value and Price was also distributed within the groups in order to eliminate order effects.

The experiment was conducted in a classroom with the subjects spread out. Each subject was given an experiment kit consisting of the AHP sheets and the PG cards.

For AHP, one sheet per criterion and person had been prepared, with all possible pair-wise combinations of the features to compare. For the purpose of eliminating order effects, the order of the pairs was randomly distributed so every subject got different order of the comparisons. With 16 features to compare, there was 16(16-1)/2 = 120 pair-wise comparisons for Value and Price, respectively. With 8 features, there was 8(8-1)/2 = 28 pair-wise comparisons for both Value and Price. In between each pair in the sheets there was a scale where the difference of the requirements' Value or Price was circled, see Figure 1. In order to be able to try different scales, no scale numbers were written on the sheets. Instead, a scale with 9 different "more than", "equal" and "less than" symbols was used. The further to the left a symbol was circled, the more valuable (or expensive) was the left feature than the right one. If the features were equally valuable (or expensive) the "equal" symbol was circled.

For PG, the subjects were given two sets of cards (one for Value and one for Price) with one mobile phone feature written on each. The cards were sorted into three piles, separately for the Value criterion and the Price criterion, see Figure 2. The piles represent (1) Necessary, (2) Adds to the

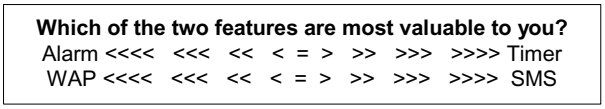

Figure 1. *Example of AHP sheet*

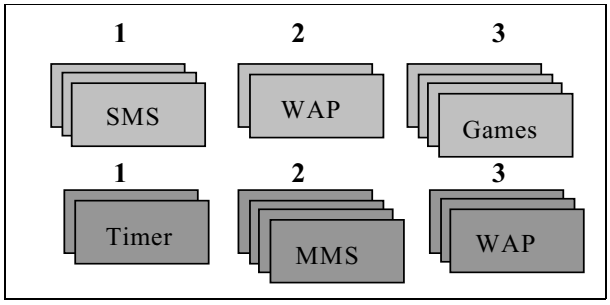

Figure 2. *Example of PG cards*

value and (3) Unnecessary, for the Value criterion, and (1) Very high price, (2) Reasonable price and (3) Low price, for the Price criterion.

Within the piles, the cards were then arranged so that the most valuable (or expensive) one is at the top of the pile and the less valuable (or expensive) are put underneath. Then the three piles were put together and numbered from 1 to 8 and 1 to 16 so that a single list of prioritised features was constructed for each criterion.

The subjects were given approximately 2 hours to conclude the tasks, which was enough time to avoid time-pressure. During the experiment, the subjects were instructed to note the time-consumption for each prioritisation. Further, the subjects had the possibility to ask questions of clarification.

3.1.5 Post-Test 1. The subjects handed in their experiment kit after finishing the tasks and were then asked to fill out a post-test. This was made in order to capture the subjects' opinions right after the experiment. The test included the questions below, as well as some optional questions capturing the opinions about the techniques and the experiment as a whole. The questions were answered by circling one of the symbols "more than", "equal" or "less than".

1. Which technique did you find easiest to use?
2. Which technique do you think gives the most accurate result?
3. Which technique do you think is most sensitive to judgemental errors?

3.1.6 Post-Test 2. After completing the analysis, the subjects were, in a second post-test, asked to state which technique they thought gave the most accurate result. They were sent two sheets (one for Value and one for Price) with two different lists of features, corresponding to the results from PG and AHP prioritisation. The post-test was designed as a *blind-test*, thus the subjects did not know which list corresponded to which technique, but were asked to select the list that they felt agreed the best with their views. The ratio scale from AHP was not taken into consideration, and neither was the pile distribution from PG. This was necessary in order to get comparable lists.

3.2 Analysis

The analysis of the experiment was divided between two independent researchers, in order to save time and to perform spot checks in order to further improve validity. The analysis was performed with Microsoft Excel™ and the computing tool MATLAB™.

Two different scales were tried for the AHP analysis: 1-5 and 1-9. According to Zhang [20] the scale 1-5 is better than 1-9 at expressing human views and therefore the scale 1-5 was used when compiling the prioritisation ranking lists. However, in section 4.6 we present consistency ratios for both scales.

Furthermore, Saaty [14] has calculated *random indices* (RI) that are used in the calculation of the consistency ratios. Unfortunately, this calculation only includes 15 factors while this experiment included as many as 16 factors. However, the RI scale was extrapolated and the RI for 16 factors was set to 1.61.

3.3 Validity

The experimental design involves some threats to validity, which we have tried to prevent. Using the *counter-balancing* design, the order effects have been balanced out since the subjects were randomly given different orders to perform the techniques and using the criteria. Therefore, we believe that the order of the techniques and criterion will not affect the results.

It is also possible that the subjects could become fatigued during the experiment. Especially the subjects who perform the tasks with 16 features may get tired or bored, which in turn may affect the concentration. This has been tested during the analysis, by calculating the consistency for AHP and the results indicate that there is no significant difference in consistency depending on the number of features (see Table 7).

Another possibility is that the subjects get practice during the experiment and unconsciously get an opinion on the context using the first technique, which will affect the result for the second technique. Especially when using PG first, it may affect the AHP performance. This is in fact the case, which is illustrated by the consistency ratio being better for those who started with PG (see Table 9).

Group pressure and the measure of each subject's time to complete the task might impose time-pressure, which can affect the results. However, it

may not be a large problem since there is no major correlation between the time and the consistency in the results (see Section 4.5). Therefore we can argue that time-pressure will not affect the performance of the prioritisation.

The number of subjects was only 16, which reduces the generalisability, i.e. there is a threat that the findings are specific to this particular group or context. On the other hand, Ph.D. students may have similar views as the requirements engineers and customers who are intended to use the techniques in practice [5]. It is also likely that the subjects are not taking the prioritisation as seriously as a requirements engineer or customers would in a real project (see Section 4.7).

Unfortunately, the scales with "more than" and "less than" in the AHP sheets were accidentally switched so that it could be interpreted in the opposite way than was intended (see Figure 1). This caused some confusion during the experiment. However, the interpretation was explained and clarified and therefore this should not be considered a threat to validity.

It would have been valuable to start the session with an introduction explaining each feature in the prioritisation since some subjects were unfamiliar to some of them. However, the subjects had their own interpretation of the features, which was the same throughout the experiment and therefore this should not affect the result.

4. Results

This section presents some of the results found during analysis. First, the three hypotheses are discussed, then some other interesting findings are described.

4.1 Hypothesis 1: The average time to conclude the prioritisations is larger when using AHP.

As expected, the time to conclude the prioritisation is larger with AHP than with PG, for both criteria. As Table 1 shows, the difference in time between the two techniques is 6.1 minutes for 8 features and 14.7 minutes for 16 features. The time increase in percent from 8 to 16 features for AHP is 88 %, while the same for PG is only 48 %. Thus, a

Table 1. Average time consumption for the prioritisation

Nbr of features	Criteria	AHP	PG	Diff.
8	Value	7.8 min	3.6 min	4.2 min
	Price	6.4 min	4.5 min	1.9 min
Total		**14.2 min**	**8.1 min**	**6.1 min**
16	Value	12.6 min	6.5 min	6.1 min
	Price	14.1 min	5.5 min	8.6 min
Total		**26.7 min**	**12.0 min**	**14.7 min**
% increase		88 %	48 %	

Table 2. Time consumption per feature

Nbr of features	AHP	PG
8	53.5 s/feature	30.5 s/feature
16	50.0 s/feature	22.5 s/feature

larger number of objects to prioritise affect the time-consumption for AHP more than for PG, at least with 8 and 16 features.

As Table 2 suggests, the subjects have in average put less time per feature when they had more features to prioritise. It is particularly interesting to see that it takes less time per feature to perform PG with 16 features than with 8. One could expect that it should be more complex to perform PG with more features but this result show that it is even faster with more features. However, there might be a breakpoint when the number of features is too great and it becomes hard to obtain an overview.

Four hypothesis tests (see Table 3) were performed, for 8 and 16 features respectively, and one for each criterion. Due to the small, and not normally distributed, sample we chose a nonparametric test, the *Wilcoxon*

Table 3. Wilcoxon tests for the time difference

Nbr of features	Criteria	p=0.05
8	Value	0.0117
	Price	**0.0781**
16	Value	0.0098
	Price	0.0039

test. The hypothesis tests show that on the 5 %-level there is a significant time difference for three of the four cases. In the fourth case, the Price criterion on 8 features, the test shows that the difference is only significant on a higher level. This is illustrated in Table 3, where the p-value is higher than the critical values on the 5 %-level in three of the four cases.

4.2 Hypothesis 2: The ease of use is considered higher for PG

Immediately after the experiment the subjects filled out the first post-test that, among other things, captured the opinions of the techniques' ease of use. Among the 16 subjects, 12 found PG more or much more easy to use than AHP. Only 3 found them equally easy and 1 stated that AHP was more easy to use, see Table 4. Hence, 75 % of the subjects found PG easier to use.

It seems as if the subjects prioritising 16 features are a bit more sceptical to PG than those prioritising 8 features. This could indicate that the more features the more difficult to keep them all in mind.

Table 4. Results from the first post-test: Ease of use

Nbr of features	AHP Much more	More	Equal	More	PG Much more
8	0	0	1	3	4
16	0	1	2	1	4
Total	**0**	**1**	**3**	**4**	**8**

4.3 Hypothesis 3: AHP reflects the subjects' views more accurately

Right after the experiment, the subjects performed the first post-test that captured which technique the subjects expected to be the most accurate. As Table 5 illustrates, a majority of the subjects expected PG to be better, while less than a fifth expected AHP to be better.

For most subjects, the actual ranking that was captured in the analysis differed somewhat between the two prioritisation techniques. In order to evaluate which technique that gave the most accurate results, a second

Table 5. Results from the first post-test: Expected accuracy

Nbr of features	Favour AHP	Equal	Favour PG
8	1	3	4
16	2	1	5
Total	**3**	**4**	**9**
Total %	19 %	25 %	56 %

post-test was sent out to the subjects. This was done a few weeks after the experiment was performed, when the analysis was finished.

As Table 6 shows, the most common opinion was that PG reflects the subjects views more accurately. Half of the ones that have stated that both techniques are equally accurate actually had the same order in the lists. An interesting observation is that this implies that PG was actually not as good as the subjects expected even if it was clearly better than AHP.

Table 6. Results from the second post-test: Perceived accuracy

Nbr of features	Criteria	Favour AHP	Equal	Favour PG
8	Value	0	2	6
	Price	4	3	1
16	Value	3	1	4
	Price	2	2	4
Total		**9**	**8**	**15**
Total %		28 %	25 %	47 %

4.4 Judgement Errors

Another question at the first post-test was which technique the subjects expected to be most sensitive to judgemental errors. The objective was to find out the subjects' views, although it has been shown that AHP is insensitive to judgemental errors due to the redundancy in the pair-wise comparisons [10, 14]. However, among the subjects 75 % expected AHP to be most sensitive. Perhaps this is because the AHP-technique "feels like

pouring requirements into a black-box" as one of the subjects stated. It may be difficult to trust something that you are not in control of.

4.5 Consistency Ratio

The consistency ratio (CR) describes the amount of judgement errors that is imposed during the pair-wise comparisons. The CR is described with a value between 0 and 1 and the lower CR value, the higher consistency. Saaty [14] has recommended that CR should be lower than 0.10 in order for the prioritisation to be considered trustworthy. However, CR exceeding the limit 0.10 is used frequently in practice [8].

The CR limit above is only valid for the scale 1~9, and in this experiment the scale 1~5 was used instead. Therefore, the limit for acceptable CR will be lower. The average consistency ratios for scale 1~5 are presented in Table 7. Calculations show that the 0.10 limit for scale 1~9 correspond to approximately 0.04 in scale 1~5. However, it was decided not to exclude any of the prioritisations, even though CR was high, in order to keep all available data.

In order to investigate if the time spent on each comparison affects the consistency, the correlation between the parameters was calculated. There is a minor correlation between the time and the consistency, positive for the Value criterion and negative for the Price criterion, see Table 8. However, since the correlation coefficients are so small, no conclusions

Table 7. Mean consistency ratio

Nbr of features	Criteria	Scale 1~5
8	Value	0.11
	Price	0.10
16	Value	0.08
	Price	0.12

Table 8. Correlation between time and consistency

	Value	Price
8 Features	0.06	-0.25
16 Features	0.26	-0.21

can be drawn except that the consistency is not particularly influenced by the time consumption.

4.6 Order Effects

There is a chance that the order in which the two techniques are used can influence the result. Table 9 shows that the mean consistency ratio is lower for the subjects who used PG before AHP. This shows that using PG may provide an image of ones preferences that are not possible to get from using AHP. Therefore it may be easier to be consistent when PG precedes AHP.

Table 9. Order effect on consistency

	Mean consistency
AHP-PG	0.23
PG-AHP	0.18

However, the hypothesis tests show that the difference is not significant on the 5 %-level. Due to the small, and not normally distributed, sample we chose a nonparametric test, the *Mann-Whitney test,* see Table 10. The critical values on the 5%-level are all larger than the p-value, and therefore we can draw the conclusion that there is no significant difference depending on the order. This finding validates that the experiment analysis have not suffered from any order effects since there is no difference between the two groups.

Table 10. Mann-Whitney test for the order effects

Nbr of features	**Criteria**	**p=0.05**
8	Value	0.2429
	Price	0.2429
16	Value	0.6571
	Price	0.7571

4.7 Distribution in Piles

In PG the subjects were supposed to distribute the features in three different piles, dependent on Value and Price. In average, the respondents distributed 41 % of the features in the middle pile (independent of criterion). This is a result that might not correspond well to how the features would have been distributed in a real case. One could assume that customers would put most of the features in the highest priority pile, which is often the case when customers need to prioritise between their wishes [11, 17, 18]. Therefore, this result might be somewhat misleading and further studies should clarify if this assumption is correct.

4.8 Prioritising the Price Criterion

One of the problems that was identified before the experiment, was that the respondents may find it difficult to prioritise the Price criterion, since it is hard to know the price of different features. However, the results show that the mean standard deviation in PG was lower when prioritising the Price criterion than the Value criterion, see Table 11. This result shows that the respondents have been more united when prioritising Price than Value, which is a rather expected result since the Price is a somewhat more objective criterion. Therefore, it is concluded that the suspicion of the Price criterion as a threat, might have been overstated.

Table 11. Mean standard deviation

	8 Features	**16 Features**
Value	1.73	3.02
Price	1.25	2.79

4.9 Qualitative Answers

In the post-test performed right after the experiment, the subjects had the opportunity to answer some optional questions about their general opinion. Opinions about AHP include “effort demanding but nice”, “it feels like a black-box wherein you pour requirements”, “good but boring”, “it feels like you loose control over the prioritisation process”, and “straightforward”. Opinions about PG are for example “fast and easy”, “lets the respondent be creative”, “intuitive”, “prone to errors”, “good

overview", and "logical and simple". These opinions correspond well to the results of the captured subjective dependent variables: ease of use and expected accuracy, discussed in prior sections.

4.10 Price-Value Graphs

In order to illustrate the possibility of using the Cost-Value approach for requirements selection, two examples of Cost-Value graphs are available in Figure 3 and 4 (AHP and PG with 8 features). However, in this experiment, we use the term Price instead of Cost. The graphs are made in order to visualise the results from the experiment and to see how much the two techniques differ in Price-Value graphs.

The three areas in the graphs represent different grades of contribution [9] and the lines visualise which Value-Price ratio each requirement has, as explained in [8]. The upper line in each graph divides those features that had more than 2 in Value-Price ratio from those that had between 2 and 0.5. The lower line in each graph divides those features that had between 2 and 0.5 from those with a ratio below 0.5 [8]. The Price and Value markings for AHP are based on the mean of the subjects' relative weight of the features. In PG, the markings are based on the median of the subjects' ranking number.

In the case with 8 features, the two methods provide the same result when it comes to which feature that are located in which area of the graph. The features Alarm and Vibrating call alert have in average a high Value-Price ratio (above 2) and therefore they would give high contribution to the fictive product. The features Colorscreen and WAP have a low Value-Price ratio (below 0.5), and would bring low contribution to the product. Finally, Calendar, Games, Notebook and Timer bring medium contribution (between 0.5 and 2 in Value-Price ratio).

The results indicate that it is possible to provide Value-Price (and Cost-Value) graphs with both PG and AHP. However, further studies are needed in order to validate if this result applies to other prioritisations.

In practice, the Cost-Value diagram would be used to guide the decision-maker in the difficult requirements selection. Other factors such as market segmentation, product focus and time constraints, will also influence the requirements selection.

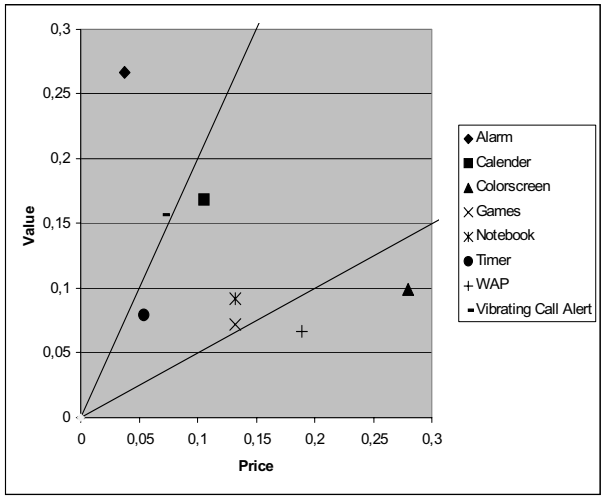

Figure 3. *Price-Value graph for AHP with 8 features*

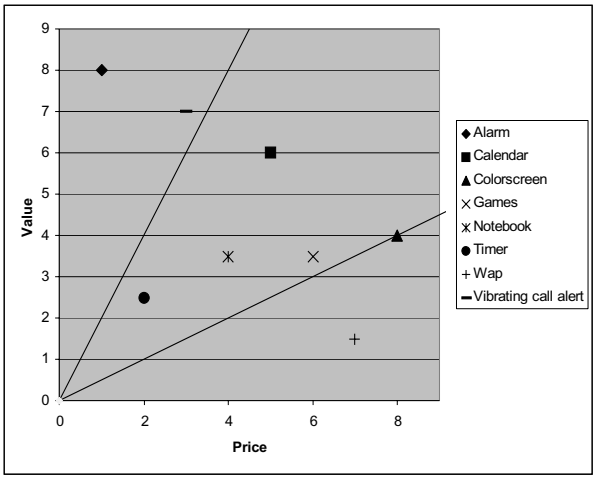

Figure 4. *Price-Value graph for PG with 8 features*

5. Discussion

Prioritisation is a very important activity in requirements engineering because it lays the foundation for release planning. However, it is also a difficult task since it requires domain knowledge and estimation skills in

order to be successful. The inability to estimate implementation effort and predict customer value may be one of the reasons why organisations use ad hoc methods when prioritising requirements. For a prioritisation technique to be used it has to be fast and easy to manage since projects often have strict time and budget pressure. Therefore, a strong argument for PG is that the time consumption is reasonable and the usage easy and intuitive.

In this experiment two groups prioritised 8 and 16 features, respectively, in order to investigate if there is a breakpoint between 8 and 16 where one of the methods is more efficient than the other. It was suspected that a greater number of requirements would eliminate the valuable overview in PG, since it would be difficult to keep all features in mind. However, this experiment only shows a slight tendency of less overview when prioritising 16 features. Therefore, it is suspected that the breakpoint is at an even higher number of features.

Another interesting observation in this experiment was that the time-consumption did not affect the consistency in AHP. One could assume that if someone stresses through the comparisons, the consistency would be worse. However, this is only initial results and with more difficult features to prioritise, the results might be different.

In practice, it is common that a larger number of requirements need to be prioritised. When the number of requirements grow, it is hard to get an overview. Therefore, visualisation is very important in order to share information. This experiment showed that it should be possible to visualise the result of both AHP and PG. However, it should be further evaluated how the ordinal scale in PG affects the visualisation.

In a real project, it may also be more valuable to use the ratio scale in order to, in more detail, differentiate requirements from each other. Thus, it may not be sufficient to determine which requirement that is of higher priority, without knowing to what extent. However, without tool support, AHP will be very time-consuming with a greater number of requirements, both to perform and to analyse.

Due to the small sample and the specific domain the results cannot be generalised to an industrial situation. Although the subjects may have opinions similar to decision-makers in industry, the context of mobile phone features is a bit too simplistic. The main weakness is that mobile phone features are on a high level and rather independent, while requirements in a real case often have interdependencies.

In the experiment performed by Karlsson *et al.* [10], AHP was ranked as the superior technique in relation to the others. The main reasons were that AHP had reliable results, was easy to use, was fault tolerant and was based on a ratio scale. This experiment shows that PG is superior to AHP on all of these criteria except for that it is not based on a ratio scale. Therefore, it is interesting to imagine a combination of the two techniques.

In order to decrease the number of comparisons, AHP could be used on the three piles, separately. Another possibility is to use AHP only on those requirements that end up in the middle pile in PG. This would imply that PG is used first, to divide the requirements into three groups according to the PG approach described earlier. The high priority group of requirements will most certainly be implemented, the low priority group will be postponed and looked into in a following release, while the ones in the middle need special treatment to determine the outcome.

This approach agrees with what Davis [3] has written about the *requirements triage* where he recommends requirements engineers to focus on the difficult requirements and skip the ones that will either be implemented or rejected anyway. In this manner, AHP can be used on the requirements that are difficult to estimate and need a more precise scale for determining its cost and value. The technique's ratio scale and fault tolerance would then come to its right.

The discussion above is based on the assumption that most requirements are not put into the same pile, which might be common in an industrial situation. Therefore, some constraints might be needed in order to force the piles to be rather evenly distributed. With three piles, this could for example mean that no pile is allowed to have less than 25 % of the requirements.

Based on the results from this experiment, it could not be concluded if a combination of the two techniques is efficient or not, or how such a combination should look like. However, we strongly believe that such a combination could be valuable and that it is worth evaluating. Therefore, it is recommended that a combination is tried in a separate experiment or case study, with more data points.

6. Conclusions

This paper describes an experiment aimed at comparing two requirements prioritisation techniques regarding time consumption, ease of use and accuracy in the result. The investigated techniques are the elaborate Analytical Hierarchy Process (AHP), which is based on pair-wise comparisons and has a ratio scale, and the elementary Planning Game (PG), which is based on pile sorting and has an ordinal scale.

The results reveal that the intuitive and quick PG technique is superior with regard to time consumption, ease of use, and accuracy. The mean time consumption was higher when using AHP and the result was statistically significant in three of four cases. PG was considered easier to use by 75 % of the subjects, although the results indicate that AHP is more preferred by those who prioritised a greater number of requirements. A blind-test performed after the experiment showed that 47 % found the priority order from PG more accurate, while 28 % favoured the order from AHP. 25 % found both priority orders equally accurate. However, it was concluded that a combination of the two techniques would further improve prioritisation. By first using PG to get an overall picture of the problem and then use AHP for the most difficult decisions, you would, with reasonable effort, get an accurate priority list.

The generalisability of the study is limited due to the small sample and the specific context. A real project has requirements interdependencies, and time and budget pressure to consider, which cause the decision-making to be far more difficult. However, we believe that PG is valid as prioritisation technique, although it does not have the same elaborate and valuable attributes as AHP.

The main disadvantage of the experiment being the difficulty to generalise to industrial projects, it would be valuable to try the experiment out in a case study. The participating organisation would then get knowledge about prioritisation and perhaps find a technique that suits their needs.

The presented experiment design could also be used on more subjects to get a larger data set and thereby a stronger basis for conclusions. There are, as discussed, several other prioritisation techniques that would be interesting to look into and compare to the presented techniques as well.

Acknowledgements

The authors would like to thank the subjects, without which this study would not have been possible. The authors would also like to thank Daniel Karlström for giving valuable comments on an earlier version of the paper.

References

[1] Beck, K., *Extreme Programming Explained*, Addison-Wesley, 1999.

[2] Beck, K., Fowler, M., *Planning Extreme Programming*, Addison-Wesley, 2001.

[3] Davis, A., M., "The Art of Requirements Triage", *IEEE Computer*, Vol. 36, pp. 42-49, 2003.

[4] Harker, P. T., "Incomplete Pairwise Comparisons in the Analytical Hierarchy Process", *Mathl. Modelling*, Vol 9, pp. 837-848, 1987.

[5] Höst, M., Regnell, B., Wohlin, C., "Using Students as Subjects - A Comparative Study of Students and Professionals in Lead-Time Impact Assessment", *EASE'00 - Empirical Software Engineering*, Vol. 5, Issue 3, pp. 201-214, 2000.

[6] Karlsson, J., "Software Requirements Prioritizing", *Proceedings of ICRE*, pp. 110-116, 1996.

[7] Karlsson, J., Olsson, S., Ryan, K., "Improved Practical Support for Large-scale Requirements Prioritising", *Requirements Engineering,* Vol 2, pp. 51-60, 1997.

[8] Karlsson, J., Ryan, K. "A Cost-Value Approach for Prioritising Requirements", *IEEE Software,* pp. 67-74, Sept/Oct, 1997.

[9] Karlsson, J., Ryan, K. "Supporting the Selection of Software Requirements", *Proceedings of IWSSD,* pp. 146-149, 1996.

[10] Karlsson, J., Wohlin, C., Regnell, B., "An Evaluation of Methods for Prioritising Software Requirements", *Information and Software Technology,* Vol 39, pp. 939-947, 1998.

[11] Lauesen, S., *Software Requirements - Styles and Techniques*, Addison-Wesley, 2002.

[12] Newkirk, J. W., Martin, R. C., *Extreme Programming in Practice,* Addison-Wesley, 2001.

[13] Robson, C., *Real World Research*, Blackwell, 1997.

[14] Saaty, T. L., *The Analytical Hierarchy Process*, McGraw-Hill, 1980.

[15] Shen, Y., Hoerl, A. E., McConnell, W., "An Incomplete Design in the Analytical Hierarchy Process", *Mathl. Comput. Modelling*, Vol 16, pp.121-129, 1992.

[16] Siegel, S., Castellan, J. N., *Nonparametric Statistics for the Behavioral Sciences,* 2nd edition, McGraw-Hill International Editions, 1988.

[17] Sommerville, I., Sawyer, P., *Requirements Engineering - A Good Practice Guide*, John Wiley & Sons Ltd, 1997.

[18] Wiegers, K., *Software Requirements*, Microsoft Press, 1999.

[19] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A., *Experimentation in Software Engineering - An Introduction*, Kluwer Academic Publishers, 2000.

[20] Zhang, Q., Nishimura, T., "A Method of Evaluation for Scaling in the Analytical Hierarchy Process", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics,* Vol. 3, pp. 1888-1893, 1996.

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 Søbirk, *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

147 **Supporting Software Inspections through Fault Content Estimation and Effectiveness Analysis**
Håkan Petersson, *Ph.D. thesis,* September 2002.
ISRN LUTEDX/TETS- -1055- -SE+237P

148 **Empirical Evaluations of Usage-Based Reading and Fault Content Estimation for Software Inspections**
Thomas Thelin, *Ph.D. thesis,* September 2002.
ISRN LUTEDX/TETS- -1056- -SE+210P

149 **Software Information Management in Requirements and Test Documentation**
Thomas Olsson, *Lic. thesis,* October 2002.
ISRN LUTEDX/TETS- -1057- -SE+122P

150 **Increasing Involvement and Acceptance in Software Process Improvement**
Daniel Karlström, *Lic. thesis,* November 2002.
ISRN LUTEDX/TETS- -1058- -SE+125P

151 **Changes to Processes and Architectures; Suggested, Implemented and Analyzed from a Project viewpoint**
Josef Nedstam, *Lic. thesis,* November 2002.
ISRN LUTEDX/TETS- -1059- -SE+124P

152 **Resource Management in Cellular Networks -Handover Prioritization and Load Balancing Procedures**
Roland Zander, *Lic. thesis,* March 2003.
ISRN LUTEDX/TETS- -1060- -SE+120P

153 **On Optimisation of Fair and Robust Backbone Networks**
Pål Nilsson, *Lic. thesis,* October 2003.
ISRN LUTEDX/TETS- -1061- -SE+116P

154 **Exploring the Software Verification and Validation Process with Focus on Efficient Fault Detection**
Carina Andersson, *Lic. thesis,* November 2003.
ISRN LUTEDX/TETS- -1062- -SE+134P

155 **Improving Requirements Selection Quality in Market-Driven Software Development**
Lena Karlsson, *Lic. thesis,* November 2003.
ISRN LUTEDX/TETS- -1063- -SE+132P