LUND UNIVERSITY

Managing quality requirements in software product development

Berntsson Svensson, Richard

2009

Link to publication

Citation for published version (APA):
Berntsson Svensson, R. (2009). *Managing quality requirements in software product development*. [Licentiate Thesis, Department of Computer Science]. Lund University.

Total number of authors:
1

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Download date: 23. Nov. 2025

Managing Quality Requirements in Software Product Development

Richard Berntsson Svensson

Licentiate Thesis, 2009

Department of Computer Science
Lund University
Faculty of Engineering

ISSN 1652-4691
Licentiate Thesis 10, 2009
LU-CS-LIC:2009-2

Department of Computer Science
Faculty of Engineering
Lund University
Box 118
SE-221 00 Lund
Sweden

Email: richard.berntsson_svensson@cs.lth.se

Abstract

Software product development companies experience different challenges in managing quality requirements compared to functional requirements. In this context, quality requirements are defined as requirements that describe a restriction on the system, and specify how well the system performs its functions. In a market–driven development context with large markets, potential customers, and strong competitors push the software product development companies to release the software product to a certain market segment at the right time with higher level of quality than the competitors.

This thesis focuses on techniques and methods that support software product development companies that release their product to an open market. The goals are to find means to improve the ability to make early estimates of quality requirements with adequate accuracy, such as performance, in order to enhance high–level decision–making.

This thesis is based on empirical research, including both quantitative and qualitative research design. The research results include a systematic literature review of empirical studies on quality requirements, which presents the state of research. The results show that there is a gap in the research literature of how cost estimation of quality requirements is conducted. How quality requirements are handled in practice is discovered and described in a survey in requirements engineering for embedded systems. From the survey, issues emerge such as when the quality level is good enough, and how to get quality requirements into projects when functional requirements are prioritized. A case study within the embedded software domain investigates how quality requirements metrics are used in an industrial context, which concludes that for a method to be successful, it is important that it is flexible enough to handle the diverse nature of quality requirements. Finally, a model for cost–benefit analysis of quality requirements, called QUPER, was set into operation in a case study. The intent was to evaluate and improve the model for supporting requirements prioritization and quality requirements roadmapping at early stages of release planning.

Acknowledgements

The work presented in this thesis was funded by the Swedish Governmental Agency for Innovation Systems under the grant for MARS, Methods for early analysis and specification of non–functional system requirements on mobile terminals.

I would like to extend my sincere gratitude to my supervisor and collaborator Professor Björn Regnell, for giving me the opportunity to be part of the Software Engineering Research Group, and for his guidence and advice. I would also like to thank my assistant supervisor, Dr. Martin Höst, for his support and advice.

The research presented in this thesis was conducted in close cooperation between academia and industry. I would like to thank everyone involved at Sony Ericsson Mobile Communication AB for their commitment, in particular Thomas Olsson. I would also like to thank all participants and their companies who have helped in making the data collection possible for this thesis. The industry cooperation has been a valuable learning experience, and I would like to thank all involved for their help and patience.

I am grateful to the co–authors of my papers and others who have contributed. I would like to thank my colleagues in the Software Engineering Research Group, for an inspiring and supporting atmosphere. I would also like to mention the colleagues at the Department of Computer Science, thanks for providing an excellent environment to work in.

Last but not least I would like to thank my family for their understanding and support, and Najia for always believing in me and making me happy every single day.

Richard Berntsson Svensson
April 2009

Contents

Introduction

Software continually becomes more important and compose a large share of today's products. Many domains need to handle software development, e.g. developers of IT systems in banking, the automotive industry, telecommunication systems, and developers of commercial products. One example of a commercial product is the mobile phone. As software becomes more important, the complexity of the software products increases. The complexity is determined partly by functionality and partly by quality requirements, such as performance and usability (Chung et al. 2000). Major challenges are related to management and requirement aspects (Ebert 1998). Requirements engineering is important for ensuring that the right software product is developed within budget and the given time frame (Berntsson Svensson and Aurum 2006).

Even if a software product is developed on time and within budget, it may be seen as a failure due to poor quality, and end–users are often dissatisfied with software quality (Jung et al. 2004). In order to improve the overall quality of a software product, it is not enough to fulfill the functional requirements. For example, even if the product works, it may be difficult to use, or showing too many failures (Ebert 1998). Therefore, quality requirements play a critical role in software product development, and not dealing with quality requirements may lead to more expensive software products and longer–time–to–market (Cysneiros and Leite 2004). Despite their importance, quality requirements are often poorly understood, generally stated informally during requirements analysis, often contradicting, and difficult to validate when the software product has been developed (Chung et al. 2000).

The handling and balance of quality requirements are an important and difficult part of the requirements engineering process (Jacobs 1999). However, in market–driven development, the situation is even more complex (Aurum and Wohlin 2005) due to the continuous flow of requirements. The continuous flow of requirements is not limited to one project, and the requirements are generated from internal (e.g., engineers) and external (e.g., customers) sources (Gorschek and Wohlin 2006). Furthermore, to achieve high–quality in embedded software products, a combination of experience and knowledge from different disciplines is needed (Kusters et al. 1999).

This may lead to communication difficulties and difficulties in achieving the required quality level (Kusters et al. 1999).

The main goal of the research presented in this thesis is to increase the awareness and understanding of quality requirements and to enhance high-level decisions–making with regards to quality requirements in software product development. By developing and applying efficient methods for early analysis of quality requirements, release planning and roadmapping of quality requirements is expected to improve. The main contributions are: an investigation of the state of research in the area of quality requirements based on a systematic review (Kitchenham 2007), increased understanding of quality requirements in practice based on a qualitative survey, and a method for release planning and roadmapping decision–making evaluated in case studies.

The first part of this thesis is an introduction to the research area and the research focus. The introduction is organized as follows: in Section 1, the quality performance model, the model that is further evaluated in paper IV and V, is described. Section 2 describes the research focus, while related work related to the thesis is presented in Section 3. In Section 4, the research methodology used in this thesis is discussed. The main contributions of this thesis is presented in Section 5 together with threats of validity. Section 6 presents further research opportunities. The second part of this thesis contains the included papers of the thesis.

Included papers

The following five papers are included in the thesis:

I **Managing Quality Requirements: A Systematic Review**
Richard Berntsson Svensson, Martin Höst, and Björn Regnell
Submitted to Information and Software Technology, 2009

II **Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems**
Richard Berntsson Svensson, Tony Gorschek, and Björn Regnell
Accepted for publication at the 15th International Working conference on Requirements Engineering: Foundation for Software Quality (REFSQ09), 2009

III **Non–functional requirements metrics in practice – an empirical document analysis**
Thomas Olsson, Richard Berntsson Svensson, and Björn Regnell
Workshop on Measuring Requirements for Project and Product Success (MeReP07), 2007

IV **Supporting Roadmapping of Quality Requirements**
Björn Regnell, Richard Berntsson Svensson, and Thomas Olsson
IEEE Software Vol. 25, no. 2, pp 42–47, 2008

V **Introducing Support for Release Planning of Quality Requirements – An Industrial Evaluation of the QUPER Model**
Richard Berntsson Svensson, Thomas Olsson, and Björn Regnell
Second International Workshop on Software Product Management (IWSPM08), 2008

Contribution Statement

Mr. Berntsson Svensson is the main author for three of the included papers (paper I, II and V). This means responsibility for running the research process, dividing the work between co–authors, and conducting most of the writing. The research in paper I and V was performed mainly by Mr. Berntsson Svensson, who designed and conducted most of the work, as well as reported on the studies. For paper I and V, Mr. Berntsson Svensson wrote most of the paper with assistance from Dr. Martin Höst, and Professor Björn Regnell respectively. Paper II was produced in cooperation with another university. Most of the design was performed together with the co–authors, while most of the analysis, writing, and division of work was performed primarily by Mr. Berntsson Svensson.

For paper III, Mr. Berntsson Svensson's contribution is part of coding quality requirements, which was performed in parallel by all authors. All authors contributed in the discussions and writing; however, the first author contributed with more than half of the research effort, while the remaining authors' work was equally distributed. In paper IV, Mr. Berntsson Svensson's contribution is the development of the practical application of the QUPER model. In terms of writing, the authors contributed to an extent corresponding to the order of the author's names.

Related publications

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

VI **Can We Beat the Complexity of Very Large–Scale Requirements Engineering?**
Björn Regnell, Richard Berntsson Svensson, and Krzysztof Wnuk
14th International Working conference on Requirements Engineering: Foundation for Software Quality (REFSQ08), 2008
(This paper presents challenges faced in very–large–scale requirements engineering, which is the context of the included papers.)

VII **A Quality Performance Model for Cost–Benefit Analysis of Non–functional Requirements Applied to the Mobile Handset Domain**
Björn Regnell, Martin Höst, and Richard Berntsson Svensson
13th International Working conference on Requirements Engineering: Foundation for Software Quality (REFSQ07), 2007

(This paper is summarised in the Introduction, Section 1. The paper presents the QUPER model, which is further evaluated in paper IV and V.)

VIII **Successful Software Project and Products: An Empirical Investigation Comparing Australia and Sweden**
Richard Berntsson Svensson, Aybüke Aurum, Claes Wohlin, and Ganglan Hu
17th Australian Conference on Information Systems (ACIS06), 2006
(See paper IX)

IX **Successful Software Projects and Products**
Richard Berntsson Svensson, and Aybüke Aurum
IEEE/ACM 5th International Symposium on Empirical Software Engineering (ISESE06), 2006
(The results show the importance of requirements engineering, which aroused my interest in this field and encouraged me to start my PhD. studies.)

1 Background

This section presents the theoretical background of a conceptual model called QUality PERformance (QUPER) for cost–benefit analysis of quality requirements, which incorporates quality as a dimension in addition to the cost and value (benefit) dimensions used in prioritization approaches for functional requirements. The QUPER model is the foundation of paper IV and V in this thesis. Therefore, it is important to understand the main concepts of the model. The reminder of this section is based on Regnell et al. (2007).

In the context of market–driven requirements engineering (see Section 3.2), products are often developed using a product–line approach (Dikel et al. 1997) applying various types of upstream decision–making (Ebert 2005) that combine market considerations with implementation concerns in activities such as roadmapping (Regnell and Brinkkemper 2005), release planning (Carlshamre and Regnell 2000) and platform scoping (deBaud and Schmid 1999). There are approaches that address requirements prioritization in a market–driven context (see section 3.3); however, despite the importance of quality requirements in market–driven requirements engineering (Jacobs 1999), focus is often on functional aspects (Regnell et al. 2007). Therefore, the QUPER model was developed with the general objective to support management of quality requirements.

The model was developed based on findings from the requirements engineering interface between two case companies (Regnell et al. 2006), but in addition to these findings the need for a cost–benefit model including quality aspects to support roadmapping and scoping was identified. High–level goals were elicited in order to capture the conjectures on what would make such a model successful.

1.1 The QUPER Model

The QUPER model aims to support requirements prioritization and roadmapping of quality requirements at early stages of release planning when making high–level scoping decisions and creating roadmaps. The model is based on two hypotheses:

- *Quality is continuous*: Quality aspects are assumed to have the potential of being measured with a value on a continuous scale rather than being either included or excluded for a certain release.
- *Quality is non–linear*: For a quality aspect such as response time in a specific use case, different variants of the following questions regarding changes in quality level are relevant: Would a little faster be almost as valuable from a market perspective? Would a little slower be very much cheaper to implement? It is assumed that a change in quality level result in non–linear changes to both cost and benefit.

Based on the results reported in (Regnell et al. 2006), the need for a cost–benefit model including quality aspects to support roadmapping and scoping, and discussions with domain experts, the following goals for the QUPER model were selected as a guide to the model development step:

- *Robust to uncertainties*. In practical cases, the relations among quality attributes and their market value and implementation cost may be very complex and difficult to estimate with high accuracy. Although it may be possible to define release planning as a mathematical optimization problem, it may not be worthwhile to apply complex mathematics, if the input data is highly uncertain.
- *Easy to use*. The model should include only a few concepts that are easy to learn, remember, understand and use by practitioners without requiring mathematical skills.
- *Domain–relevant*. The model should be possible to combine with existing practice and possible to tailor to a particular domain. In a practical setting, a model for quality attribute roadmapping should be feasible to include as an add–on to the working practice without costly interference with existing processes, techniques and methods.

The QUPER model has two main concepts, breakpoints and barriers. A breakpoint is an important aspect of the non–linear relation between quality and benefit, while a barrier represents an interesting aspect of the non–linear relation between quality and cost. The two concepts of breakpoints and barriers form the basis of QUPER's three views: (1) the *benefit view*, (2) the *cost view*, and (3) the *roadmap view*. The three views are illustrated in Figures 1–3 respectively and subsequently described.

The QUPER benefit view (Figure 1) includes three breakpoints indicating principal changes in the benefit level with respect to user quality perception and market value. The three breakpoints are:

- *Utility breakpoint*. Represents the border between a quality level that is so low that a product is not accepted on the market as users find the quality level *useless*, and the level where a product starts to become *useful* and thus have a potential market value.
- *Differentiation breakpoint*. Marks the shift from the useful quality range to a quality level which only a few products (currently) reach, which makes them having a *competitive* market proposition.
- *Saturation breakpoint*. Implies a change in quality level from competitive to *excessive*, where higher quality levels have no practical impact on the benefit in the particular usage context considered.

The QUPER cost view (Figure 2) includes the notion of *cost barriers* to represent the non–linear nature of the relation between quality and cost. A

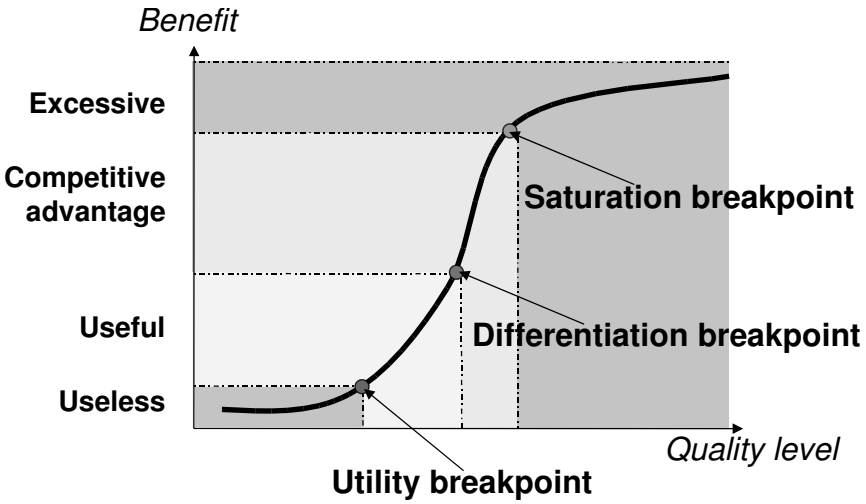

Figure 1: The QUPER benefit view

cost barrier occurs when the cost characteristic shifts from a plateau–like behavior where an increase in quality has a low cost penalty, to a sharp rise behavior where an increase in quality has a high cost penalty. Costs can e.g. be investments in development effort or cost per unit of hardware. A typical cost barrier may be the result of that a quality increase is not feasible without a large reconstruction of the product architecture, while a typical cost plateau is exemplified by the case where comparatively inexpensive software optimizations may result in high gains of performance.

The QUPER roadmap view (Figure 3) combines the benefit and cost views by position the breakpoints and barrier together ordered on the same scale. This view enables visualisation of benefit breakpoints and cost barriers in relation to the *current* quality level of a product and the qualities of *competing* products. This view also combine the notion of *targets* for coming releases with the aim of supporting roadmapping.

The quality levels on the horizontal axis of all three views are measured by quality indicators that may be specific with respect to different entities such as feature, use case, and market segment. The definition of quality indicators is the main issue in tailoring the QUPER model for a certain domain and for a certain (set of) products.

When applying the QUPER model in non–functional requirements prioritization and roadmapping, the following steps are envisioned:

1. Define quality indicators
2. For each quality indicator, and for each relevant qualifier (feature, use case, segment) make estimations of (a) benefit breakpoints and (b) cost barriers

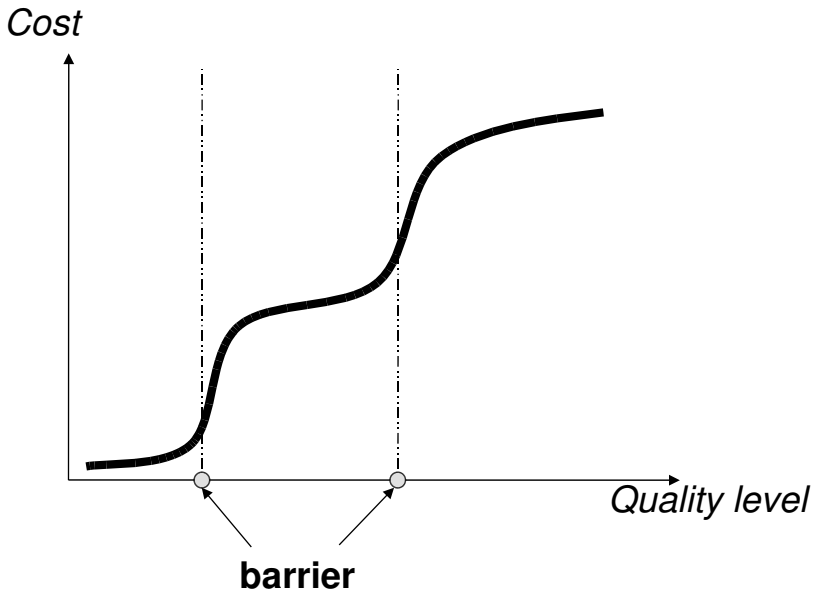

Figure 2: The QUPER cost view

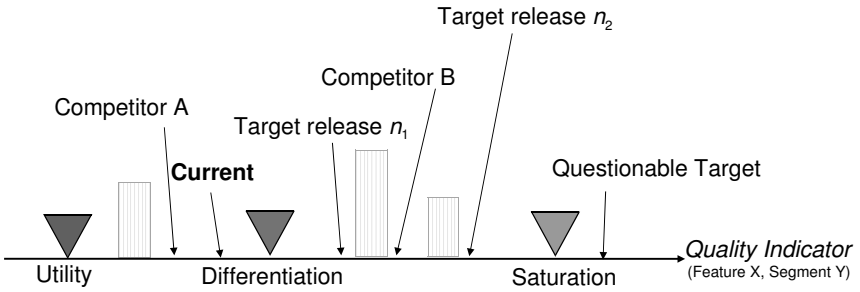

Figure 3: The QUPER roadmap view

3. Estimate the current quality of own product (for a given release) and the quality of competing products (at present or envisioned)
4. Visualize estimations, discuss and decide targets for coming releases
5. Communicate roadmaps as a basis for further requirements engineering
6. Revise roadmaps and iterate as estimates become more certain or circumstances change

1.2 Case Study

The feasibility and relevance of the QUPER model has been validated in the mobile handset domain through a series of interviews with experts (Regnell et al. 2007). The study is based on six cases in selected sub–domains representing examples of important parts of the different technology areas that are included in the mobile handset domain.

Local Connectivity. The local connectivity sub–domain includes the capabilities of a mobile phone to connect to local devices such as a personal computer while not requiring access to the mobile network. The following findings were made for this sub–domain:

- *Quality indicators*. The *data–transfer–rate* is an important quality indicator measured in bits per second. Interoperability, usability, security and reliability are also important quality aspects. One example of usability indicator is the *connection–setup–time*.
- *Benefit breakpoints*. Benefit breakpoints can be identified for several different use cases, such as transfer music and synchronizing calendar. Often there is a discrepancy between the theoretical maximum data–transfer–rate and what may be achievable in practice. Benefit breakpoints are dependent on market segments.
- *Cost barriers*. Different transfer technologies have different costs and achieving the next level often requires development efforts and/or application specific hardware with attractive cost–size–performance trade–off.

Positioning. The positioning sub–domain includes the capabilities of a mobile phone to know its geographical position and to provide services that are based on its position. The following findings were made for this sub–domain:

- *Quality indicators*. An important quality indicator is *time–to–first–fix*, defined as the time from initiation of a positioning request until location data is provided, and measured in seconds. Another important quality indicator is *position–accuracy*, defined as the error margin in the given positioning data measured in meters.
- *Benefit breakpoints*. The utility, differentiation and saturation breakpoints depend on which use case is considered. E.g., for finding places in a city, the time–to–first–fix utility breakpoint is more demanding than compared to navigation support on the sea. Utility and saturation is in some cases based on physical constraints such as distances between streets in a city.
- *Cost barriers*. Costs are dependent on both hardware and software issues. Development investments in network infrastructure to increase

performance also impact cost barriers. Anther cost factor in this domain is related to energy consumption that has impact on battery life.

Java Platform. The java platform enables a mobile device to run java applications that can be downloaded via local connectivity or over the network. The following findings were made for this sub–domain:

- *Quality indicators*. Real–time performance is a very important quality indicator that can be measured in many ways, for example *application-start–up–time*, *data–save–time*, etc. and can be measured in seconds. Also quality indicators such as 3D–*graphics–frame–rate* and *number–of–polygons–per–second* are important. Reliability is also important and is measured in *number–of–software–crashes–per–time–unit*.
- *Benefit breakpoints*. For graphics and streaming the benefit breakpoints can easily be identified. Also application–start–up–time has clear utility, differentiation and saturation breakpoints. Reliability and compatibility is more difficult to measure, however, by testing competing products it is possible to get a general picture.
- *Cost barriers*. Cost barriers in quality requirements are often related to development efforts directed towards performance optimization. It is often easy to detect existence of performance problems but not always easy to identify the best solution. A major challenge is to estimate the relation between invested performance optimization effort and the effect in terms of improved performance.

Mobile TV. Mobile TV is an area that is of strategic importance for future mobile products. Mobile TV is enhanced with interactivity that enables users to watch streamed TV programs live and interact with the show, with voting and chatting capabilities. The following findings were made for this sub–domain:

- *Quality indicators*. Quality indicators related to user experience of video streaming are central in this sub–domain. Typically, quality is indicated by *video–frame–rate* measured in number of image frames per second, but the subjective user experience is dependent on many factors, such as performance of coding and decoding including compression, error correction and radio reception sensitivity.
- *Benefit breakpoints*. Benefit breakpoints can be identified rather easily for mobile TV and depends on market segment and the nature of the streamed content. Some quality indicators related to performance tend to have a more either–or–nature in terms of the utility–differentiation–saturation scale.
- *Cost barriers*. Cost barriers are related both to dedicated hardware and optimization of software–implemented algorithms. Typically,

performance issues are central to development investments and passing utility breakpoints often requires breaking a cost barrier. Sometimes differentiation can be reached through software optimizations and sometimes dedicated technology platform support is needed.

Memory. Memory technology is central to many applications in mobile handsets. Memory is used not only for software that runs operating systems and applications but also for content such as personal information management, music, images, video and other files. The following findings were made for this sub–domain:

- *Quality indicators*. There are many different memory technologies and they differ with respect to quality indicators such as *memory–density* measured in bytes, *physical–size–of–package* measured in millimeters in three dimensions and *memory–data–transfer–rate* measured in bits per second.
- *Benefit breakpoints*. he benefit breakpoints are dependant on the actual use case. For example, multishot (consecutive photographing) require higher data transfer rates. Memory hardware needs to be planned far in advanced to be able to manage sourcing and supply as well as to enable integration into the technical platform. Memory is cutting cross many different use cases and other sub–domains are heavily dependent on memory qualities, which in turn affects the breakpoint levels in that they need to be qualified with use case and segment.
- *Cost barriers*. Cost is mainly related to hardware costs, although development costs for integrating new memory technologies into the technical platform is related to engineering effort and involves both hardware and software interfacing.

Radio Network Access. This sub–domain thus involves standardization issues and requirements on the technical platform that implements the access to the radio network. The following findings were made for this sub–domain:

- *Quality indicators*. Primary quality indicators are the *downlink–* and *uplink–data–transfer–rate*, as well as the *packet–latency* affecting quality of real–time data such as voice and video conferencing.
- *Benefit breakpoints*. Different use cases have very different characteristics in terms of benefit breakpoints. Also, different segments have different demands although shifting as new technology generations arrive.
- *Cost barriers*. The costs are connected to cost–per unit for hardware and protocol software, together with license fees. Another type of

cost is related to the risk of lost market opportunities, should technical platforms be delayed.

1.3 Discussion of Case Study Findings

In general, it was possible to define benefit breakpoints and cost barriers for all six sub–domains, supporting the relevance of the model. The interviewees acknowledged the usefulness of the model, although open issues where pointed out:

- How many and which quality indicators should be managed? This is a challenge on how to keep balance between the benefit of the information and the effort involved in acquiring and maintaining the information. It also deals with the challenge of tailoring the QUPER model to particular domains. The set of managed quality indicators of course depend on the domain, the products and its strategic use cases.

- How to combine different quality indicators and trade–off among them? This is a challenge of making prioritization among several quality indicators, possibly by using existing prioritization methods but for discrete values of the quality indicator, and possibly by using the breakpoints of different quality indicators and comparing them with other breakpoints of other quality indicators.

There were a number of factors encountered that where relevant to the qualification of quality metrics and affected the positions of breakpoints:

- *Use case*. Different use cases often have different quality demands.

- *Market segment*. Different market segments, e.g. comparing low–end to high–end, have different demands on quality.

- *Feature maturity*. As the products and markets mature and users get familiar with features, expectations on quality often rise.

A number of different types of costs were identified in the six cases:

- Development effort (software and hardware).

- Cost per unit (hardware and indirectly software).

- Footprint, physical size (hardware and indirectly software).

- Energy consumption (hardware and indirectly software).

- Missed market opportunities vs. competitors (potential earnings).

In general, cost seems to have a non–linear relationship to the level of quality, which supports the relevance of the QUPER cost model with its barriers. However, it seems as the nearest barrier often is easier to identify than the barriers beyond. It is not until a certain barrier is reached and passed that a more accurate location of the next barrier can be determined.

Many quality indicators are often related to standardized levels, which makes a continuous scale transformed into a set of ordered discrete levels. Taking standards into account in the definition of quality indicators seem inevitable in the telecommunications domain. However, the relation between a technical quality defined by a standard level and the perceived user experience in a real–life usage situation is not always straight forward.

When introducing prioritization techniques and roadmapping methodology it is stressed by informants that application of techniques and methodology needs to be simple and easy to learn and understand.

1.4 Summary

The goal of the QUPER model is to be useful by being simple and robust and yet relevant to high–level decision–making in activities such as roadmapping, release planning and scoping.

The contribution of the QUPER model is based on our observation that quality aspects and non functional metrics are often specified without explanation or rationale in existing practices. Lehtola and Kauppinen (2006) found that communication problems were a difficulty for understanding the importance of a requirement. Managers need to have an understanding of the whole picture of requirements priorities. QUPER addresses this challenge aiming at enriching the over all picture through a better understanding also of non–functional requirements.

The feasibility and relevance of the QUPER model is validated through interviews with experts in six cases representing sub–domains of the mobile handset domain. The validation indicates that QUPER is feasible and relevant to the selected domain.

The QUPER model formed the foundation of this thesis research focus, which is presented in the following section.

2 Research Focus

The research presented in this thesis is in the field of requirements engineering. Requirements engineering is a critical activity when developing software–intensive products (Castro et al. 2002). Software products consists of both hardware and software, such as embedded products (for example, mobile phones), or a software product can be pure software applications (Thayer 2002). According to Konrad and Gall (2008), the higher the complexity of the product under development, the more important re-

quirements engineering becomes. Several studies ((Wohlin et al. 2000b), (Boehm and Basili 2000), (Berntsson Svensson and Aurum 2006), (Berntsson Svensson et al. 2006)) have identified the importance of requirements engineering, the quality of the product, and customer satisfaction. The ability to develop a software product that meets customers' requirements, and offer high value to both their own business and to the customer increase the chance of market success (Barney et al. 2008). However, this provides that the software product is released to the market at the right time, and offers a higher level of quality than the competitors' products (Barney et al. 2008). The value of a software product is related to quality requirements (Barney et al. 2008), and is increased in direct proportion to the advantage over competitors' products (Alwis et al. 2003).

The research in this thesis is more specifically concerned with managing quality requirements when developing software products in relation to market–driven requirements engineering, software product management activities, and requirements prioritization, which is illustrated in Figure 4.

The different research areas in Figure 4 are further explained in the related work section (Section 3).

2.1 Research Goals

The goal of this research is to find means for improving the ability to make early estimates with adequate accuracy of quality requirements such as performance in order to enhance high–level decision–making. The main research questions that have been investigated are:

RQ1. What empirical evidence of managing quality requirements exist in the literature?

RQ2. Which are the current quality requirement challenges that need further investigation?

RQ3. How can QUPER be transferred to industry practice?

RQ4. To what extent does the use of QUPER as a part of release planning of quality requirements result in improvements with regards to high–level decision–making?

The relation between the research questions is illustrated in Figure 5.

RQ1 was posed in order to discover what empirical evidence exists of managing quality requirements in industry, and what areas need further investigation. Among the found evidence in the systematic review, issues regarding prioritization of quality requirements and how quality requirements are handled in software product management emerged.

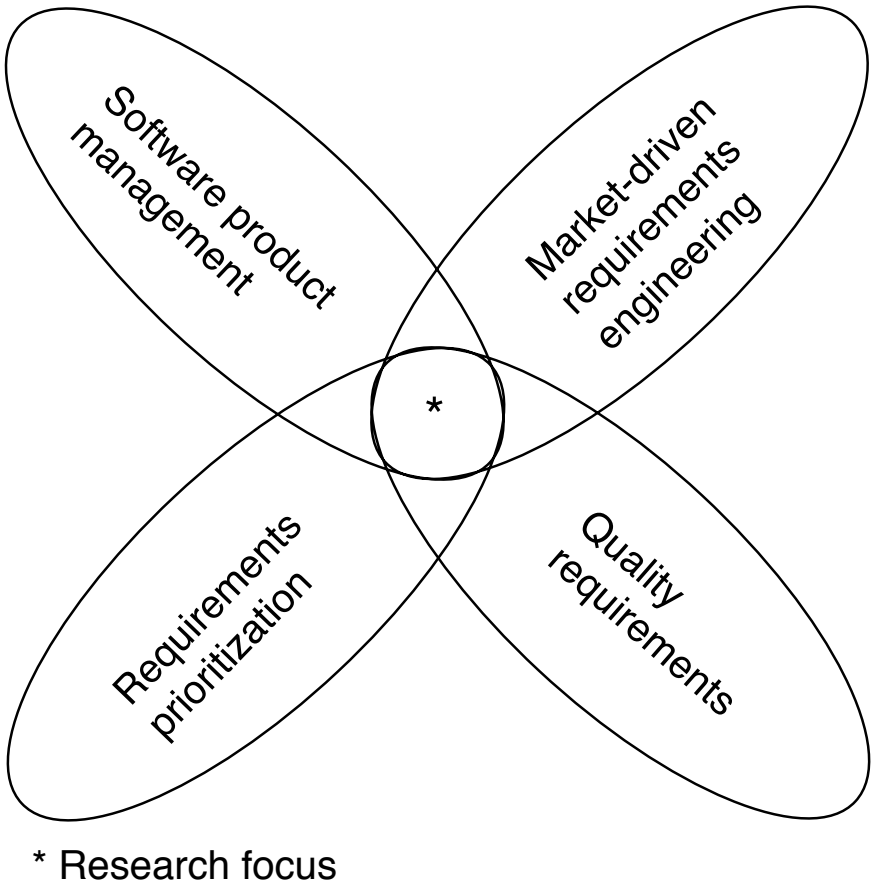

Figure 4: Research focus

RQ2 aims at discovering and understanding how quality requirements are managed by practitioners in industry. The results from RQ1 were used to create an interview instrument to identify how quality requirements are handled in practice, and discovering possible challenges. Two major challenges of handling quality requirements are identified, (1) how to get quality requirements into projects, and (2) when is the quality level good enough?

RQ3 examined how the QUPER model could be applied in practice. A set of guidelines including a step–by–step practical application is developed. The results were used to adapt the QUPER model to the case study in RQ4.

RQ4 aims at evaluating and improving the ability to make early estimates of performance requirements as input to release planning through the QU-

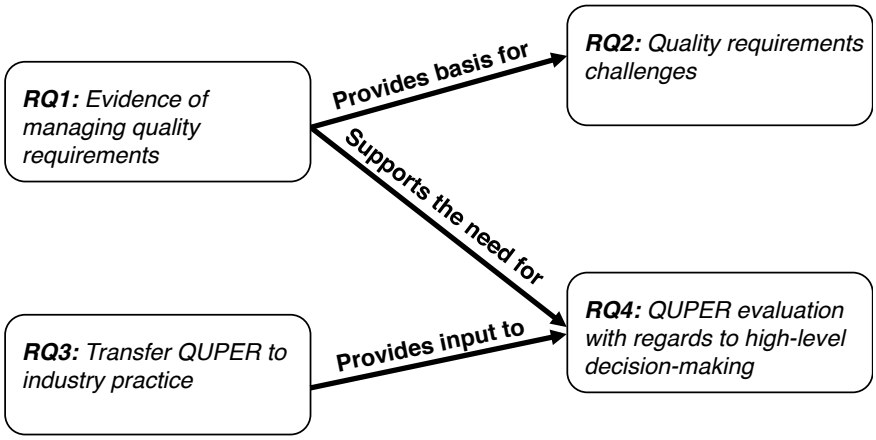

Figure 5: The main parts of this thesis

PER model. The QUPER model is applied in one case study to investigate its possibilities and limitations.

The used research methodology in each research question is presented in Section 4, and the results in relation to the research questions are discussed in Section 5. In the following subsection, related work relevant to the research focus is discussed.

3 Related Work

In this section, some theoretical background to the market–driven requirements engineering and quality requiremetns areas are provided, and the context of the research in this thesis is described.

3.1 Quality Requirements

In the literature, different types of requirements are discussed, and often classified as functional or non–functional requirements. These non–functional requirements are subsequently called quality requirements (QR). Functional requirements are defined as:

> *"Requirements that specify the functions of the system, how it records, computes, transforms, and transmits data"*
>
> (Lauesen 2002)

There are many definitions of quality requirements in the literature, the following presents a selection of definitions.

"Quality requirements specify how well the system performs its intended functions"

(Lauesen 2002)

"Quality requirements put restrictions on the system. That is, quality requirements or constraints describe a restriction on the system that limits our choices for constructing a solution to the problem"

(Pfleeger 2001)

"Quality requirements are not directly concerned with the specific functions delivered by the system. They may relate to emergent system properties such as reliability and response time. Alternatively, they define constraints on the system such as capabilities of I/O devices and the data representations used in system interfaces"

(Sommerville 2007)

"In software engineering, a software requirement that describes not what the software will do, but how the software will do it, for example, software performance requirements"

(Thayer and Dorfman 1990)

In Sommerville's definition, what data that should exists in the interfaces may be seen as data requirements, which is a subgroup of functional requirements (Lauesen 2002). Based on this, our definition of quality requirements does not include Sommervilles. In addition, Thayer and Dorfman define quality requirements as requirements that describe how the software will do it. We believe that how certain aspects should be done is more related to the design of software than requirements engineering. Therefore, Thayer and Dorfman's definition is not incorporated into ours. In this thesis, we define quality requirements as: *"Quality requirements describe a restriction on the system, and specify how well the system performs its functions"*, which is a combination of (Lauesen 2002) and (Pfleeger 2001) defintions.

In the literature, quality requirements have been categorized based on different characteristics. However, there is no formal definition, nor a complete list of quality requirements (Chung et al. 2000). Neither is there a universal classification of quality requirements characteristics, and different people use different terminologies (Chung et al. 2000). In 1976, Boehm et al. (1976) presented a tree of software quality characteristics. Fulfilling the parent quality characteristics in the quality tree, implies that the child quality characteristics are also fulfilled. Examples of parent quality requirements are reliability, modifiability, and human engineering. Later on,

Table 1: Characteristics and subcharacteristics in ISO/IEC 9126

Characteristics	subcharacteristics
Functionality	Suitability, accuracy, interoperability, security, functionality compliance
Reliability	Maturity, fault tolerance, recoverability, reliability compliance
Usability	Understandability, learnability, operability, attractiveness, usability compliance
Efficiency	Time behavior, resource utilization, efficiency compliance
Maintainability	Analyzability, changeability, stability, testability, maintainability compliance
Portability	Adaptability, installability, replaceability, coexistence, portability compliance

in 1985, Roman (1985) classified quality requirements into several classes such as performance constraints, life–cycle constrains, and economic constrains. Each class of quality requirements had several subclasses. Another classification divides quality requirements into three general groups: organizational, product, and external requirements (Sommerville 2007). An example of organizational requirements is delivery requirements, while legislative requirements belongs to the external group. For further elaboration of classifications of quality requirements, see Chung et al. (2000).

In addition to the classifications of quality requirements, several standards have been published, such as the ISO/IEC 9126 (9126-2001 E), McCall and Matsumoto (1980), and 830 (1998). In this thesis, the ISO/IEC 9126 standard has been used. The ISO/IEC 9126 standard defines a quality model that comprises of six characteristics and 27 sub–characteristics (see Table 1). Standards for quality requirements are described in (Lauesen 2002) and (Thayer and Dorfman 1990).

Why are quality requirements critical and difficult to manage?
Quality requirements address the issue of quality for software products. Not dealing, or ineffectively with quality requirements may result in a software product with poor quality, unsatisfied users, and more expensive software (Chung et al. 2000). Chung et al. (2000) identifies three aspects of quality requirements, first, QR can be *subjective*, which means that the quality requirement can be evaluated and interpreted differently. Some people may consider the QR to be accomplished, while others do not. Second, quality requirements can be *relative*, meaning that a some level of quality has been reached. For example, the system may have slow response time, medium response time, or high response time. Third, quality requirements

can be *interacting*, by accomplishing one quality requirement can have a positive or negative affect on other quality requirements. For example, improved security may affect the usability in a negative way. Difficulties to manage quality requirements are investigated by Borg et al. (2003) in two development organizations. Borg et al. (2003) found that quality requirement related problems occur throughout the entire development process. The results show that quality requirements are discovered too late, or not discovered at all; difficulties in prioritization of quality requirements; and difficulties to estimate cost and measures of quality requirements.

3.2 Market–Driven Requirements Engineering

Requirements engineering is a process that involves activities that are required to gather, create, and maintain a software product's requirements specification. According to Sommerville (2007), the requirements engineering process is defined by four high–level activities, as illustrated in Figure 6. For more details regarding the requirements engineering process, see Sommerville (2007).

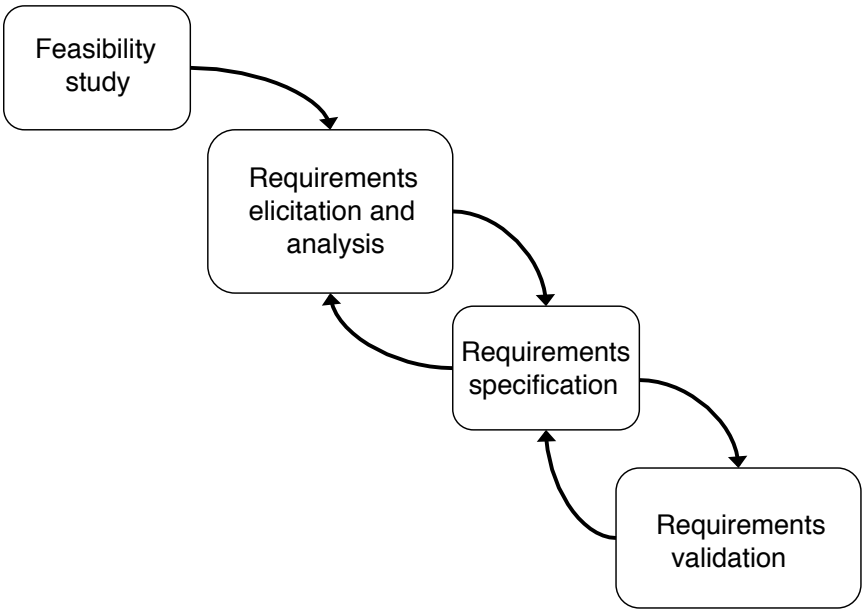

Figure 6: The requirements engineering process

A software product can be developed by two different approaches depending on the type of market, customer specific development (also called bespoke or contract–driven) or market–driven software product development (also called packaged software or commercial off–the–shelf). In cus-

tomer specific development, a supplier develops and delivers a software product to the customer. The requirements specification and a contract are negotiated and specify what the supplier shall deliver. The customer–specific requirements engineering process,thus covers the four activities of requirements engineering proposed by Sommerville (2007).

In market–driven development, the software product is developed for an open market instead of a single customer. The market–driven requirements engineering (MDRE) process consists of the same four activities in Figure 6. In addition, the MDRE process consists of specific activities such as release management and market analysis (Regnell and Brinkkemper 2005). Moreover, MDRE is often under the pressure of competitors' products and the evolvement of the market and product (Regnell and Brinkkemper 2005).

There is no clear distinction between market–driven and customer–specific development, for example, it is not unusual for a supplier to provide products to an open market and at the same time customizing their product for specific customers. The distinguishing features of MDRE in comparison to customer–specific RE is illustrated in Table 2, which is adapted from Regnell and Brinkkemper (2005) and Carlshamre (2002b).

Karlsson et al. (2007) published a study that focused on challenges in market–driven software development. Even though the study does not have a focus on quality requirements, challenges related to quality requirements are identified, including the handling of interdependencies of quality requirements. Moreover, problems with considering quality requirements in release planning are identified.

Thus, challenges of quality requirements in MDRE are identified; a study with main focus on quality requirements in practice at companies using market–driven development is conducted to increase the understanding of quality requirements specifically. How quality requirements are handled in practice is provided in Paper II.

3.3 Requirements Prioritisation

A product's quality is often determined by the ability to satisfy the needs of the customers/users (Bergman and Klefsjö 2003), (Schulmeyer and McManus 1999). All stakeholders and their requirements need to be identified and their conflicting preferences and expectations (Karlsson et al. 1997). When developing a software product for an open market (MDRE), it is not possible to involve all stakeholders to prioritize requirements. The requirements are generated from internal (e.g., engineers) and external (e.g., customers) sources (Gorschek and Wohlin 2006). Conflicting prioritize between stakeholders is an issue that is addressed by many software product managers (Berander and Andrews 2005). In these situations, it is important to handle different stakeholders in a structured way. Regnell et al. (2001a) suggest adjusting each stakeholder's influence by prioritizing different as-

Table 2: Overview of customer–specific RE and MDRE (Regnell and Brinkkemper 2005) and (Carlshamre 2002b)

	Customer–specific RE	MDRE
Objective	Fulfillment of a contract and compliance to the requirements specification	Deliver the right product at the right time
Success	Customer satisfaction and user acceptance	Determined by sales, market share, and product reviews
Life cycle	First development, then maintenance. Often one major release	Long series of releases and the product is undergoing continuous evolution
Elicitation	Collects information from one customer	Innovation of new requirements and market analysis
Specification	More formal	Less formal
Negotiation	Negotiation and conflict resolution	Focused on prioritization, cost estimation, and release planning
Validation	Continuously through the contract	Delayed until late stage in the development

pects. Which aspects depend on the strategy that is most suitable in the current market segment (Regnell et al. 2001a).

In most software product development, there are more candidate requirements than are possible to implement within the time and budget constrains (Berander 2004). Hence, the objective of requirements prioritization is to select and implement a sub–set of these requirements based on effort and value estimates, and still meet the stakeholders needs and to satisfy the customers (Karlsson and Ryan 1997). In addition, requirements interdependencies and the product's scope should also be taken into account. Moreover, requirements are often specified at different levels of abstraction (Gorschek and Wohlin 2006), and deciding on what level of abstraction should be used can be difficult. In small–scale or even in medium–scale requirements engineering (Regnell et al. 2008b), it may be possible to prioritize requirements on a low level of abstraction. However, in very large–scale requirements engineering (Regnell et al. 2008b) there are often too many requirements to prioritize. Regnell et al. (2001a) suggest grouping the requirements to make the prioritization easier.

Dependencies have an enormous impact on requirements prioritization, which makes the requirements prioritization process even more complex when including quality requirements. The increased complexity of prioritizing quality requirements is related to difficulties to trace quality requirements since they tend to have a global impact on the whole system, and an extensive network of interdependencies between them (Cleland-Huang et al. 2005). In addition, quality requirements can be in conflict with each other; therefore, trade–offs need to be made.

There are several prioritization techniques introduced in the literature. Karlsson et al. (1998) evaluated different methods for prioritizing software requirements involving pair–wise comparisons. The study concluded that the Analytical Hierarchical Process (AHP) (Saaty 1980) is superior but also time–consuming. In addition, AHP assumes that requirements are independent, even though that is seldom the case (Regnell et al. 2001b). Karlsson and Ryan (1997) suggested using a cost–value approach based on the AHP. This approach supports trade–off analysis, but is mainly used for functional requirements. However, quality requirements can be included as objects of prioritization in AHP. Quality Function Deployment (QFD) (Karlsson 1997) is a comprehensive, and customer and user oriented approach for requirements prioritization. To fully implement QFD, customers and users need to be visible; however, not all market–driven projects have access to their customers.

Thus, there are several requirements prioritization techniques that may support quality requirements, some more than others. A comparison of the QUPER model and other techniques is provided in paper IV and V, and in the related publication paper VII.

3.4 Release Planning and Roadmapping

Software product development is more and more commercialized as standard products (van de Weerd et al. 2006b), and less customized software is developed (van de Weerd et al. 2006a). At the same time, market–driven product development gains greater acceptance (AlBourae et al. 2006); therefore, a new role within software companies emerged, namely that of product manager (van de Weerd et al. 2006a). However, product management is not a new domain; it has been established in other sectors, such as manufacturing since the 19th century (Kilpi 1997). Only recently has software product management (SPM) received attention in the software industry (van de Weerd et al. 2006b). Software product management has specific challenges compared to product management in other sectors. van de Weerd et al. (2006b) identifies five specific challenges in software product management:

- Manufacturing and distributing of extra copies do not require extra cost
- Software can be changed easily, sold products can be updated by release updates
- Organization of requirements and tracing of changes in design is complex
- Software products have a high release frequency due to the ease of changing
- The software product manager has many responsibilities, but has no authority over the development team

The role of software product manager has emerged over recent years and appears to be of value; however, the role is complex to execute. The product manager has several important tasks, such as managing requirements, release planning, and launching products (van de Weerd et al. 2006a). The research in this thesis has been conducted in relation to two activities in software product management, namely, roadmapping and release planning. The relation between roadmapping, release planning, and requirements prioritization is illustrated in Figure 7. For further elaboration of SPM activities, see van de Weerd et al. (2006a) and van de Weerd et al. (2006b).

Regnell and Brinkkemper (2005) defines a roadmap as a document that provides a layout of the product release to come over a time frame of three to five years. There are many types of roadmaps described in the literature (Schalken and Brinkkemper 2004), and the one used in MDRE release planning is the Product–Technology Roadmaps, where the purpose is to map and align efforts towards a common goal. Roadmapping is a complex

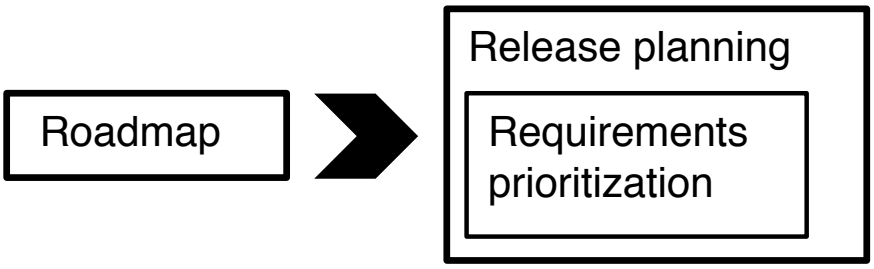

Figure 7: Software product management activities

activity due to the dependencies between the product and the related ones. A roadmap should communicate several aspects such as themes of a certain release (e.g. improving quality, performance), goals, and milestones (for releases).

Release planning is a process in software product management (as described in Figure 7). The software product manager is responsible for the release process (Regnell and Brinkkemper 2005). Release planning is a process applying various types of upstream decision–making that combine market considerations with implementation concerns (Regnell et al. 2007). Release planning involves activities such as selecting what features and requirements should be in a certain release (requirements prioritization), when it should be released, and at what cost (Ullah and Ruhe 2006). Thus, it is a major determinant of the success of a software product (Carlshamre 2002a). Figure 8 illustrates the release planning process, which is adapted from (van de Weerd et al. 2006b).

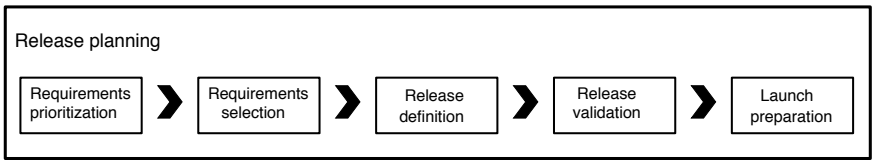

Figure 8: Release planning activities

Wohlin and Aurum (2005) identified relevant criteria for release planning, one criterion that is regarded relevant for all participants is the cost–benefit trade–off for implementing a requirements. This is similar to the cost–value approach by Karlsson and Ryan (1997) and to the QUPER model ((Regnell et al. 2007), (Regnell et al. 2008a), and (Berntsson Svensson et al. 2008)). Determining what requirements to include in a certain release is a complex process (Regnell and Brinkkemper 2005) due to that requirement needs to be collected from various sources. The selected product requirements are then input for the development process, which results in

a software product. According to (Ullah and Ruhe 2006), lacking of good release planning practices may results in unsatisfied customers and market loss, which makes release planning a major determinant of the success of a product.

Much research has been conducted about the actual process of determining requirements for a certain release. For example, by Carlshamre and Regnell (2000) in the use of the REPEAT process (Requirements Engineering Process At Telelogic). REPEAT is based on fixed release dates and intervals, which allows the requirements to be allocated to lists with a *"must"* part and a *"wish"* part. For further elaboration of the REPEAT, see Regnell et al. (1998). Other examples of release planning processes includes, the AHP (described in Section 3.3), stakeholders' opinions on requirements importance (Ruhe and Saliu 2005), and Carlshamre (2002a) used linear programming on which requirements interdependencies are added.

4 Research Methodology

This section gives an overview of the methodological approaches that are used in this thesis. Furthermore, the research strategies and methods used in the studies in this thesis are described. In addition, threats to validity of the results in this thesis are discussed.

4.1 Research Design

There are two main approaches to research: the *fixed* and the *flexible* research design (Robson 2002). The fixed research design, which is also called *quantitative*, is a highly pre–specified research design. In order to know in advance what to do, and how to do it, fixed design requires a conceptual framework or theory to be developed before getting into the main part of the research study. The researcher needs to collect all data before starting to analyze it. The fixed research design often quantifying a relationship or comparing two or more groups, where a solution or method is suggested as more appropriate than others.

The flexible research design, which is also called *qualitative*, is a less pre–specified research design than the fixed approach. Flexible research design evolves during the research process, and the data collection and analysis are intertwined. Qualitative data are typically non–numerical, instead, the data is mainly focused on words. However, qualitative data may include numbers. The flexible design studies objects in their natural setting, where issues of the real world are described.

In this thesis, both fixed and flexible research designs are used (see Table 3). Fixed and flexible research design can be further classified into research strategies. The following sub–section describes the used research strategies

in this thesis. Surveys and case studies can both be classified as fixed and flexible research design (Wohlin et al. 2000a).

4.2 Research Strategies

This section describes the used research strategies in this thesis, which are systematic review, surveys, case study, and action research.

Systematic Review: A systematic review is a method that enables assessment and interpretation of all available research that is relevant to a particular research question, topic area, or phenomena of interest (Kitchenham 2007). Reasons for carrying out a systematic review include, but are not limited to (Kitchenham 2007): to review existing literature in relation to a treatment or technology, to identify a gap in the existing literature, and to provide a context to appropriately place new research activities.

Systematic review is of fixed research design. There are two main reasons for classifying systematic review as fixed research design. First, in fixed design a conceptual framework or theory needs to be developed before getting into the main part of the research. Prior to undertake a systematic review, it is necessary to identify the needs for the review, define research questions, produce a review protocol including defined review procedures (planning), and predefined search strategy should be created. Second, fixed research design often quantifying a relationship. A systematic review summarizes the existing evidence concerning a treatment, which is about quantifying a relationship. A systematic review comprises of three main phases: planning the review, conducting the review, and reporting the review. For further elaboration, see Kitchenham (2007).

The advantages with systematic reviews are, a well–defined methodology, provides information about the effects of a phenomena across a variety of contexts and empirical methods, and the possibility to combine data using meta–analytic techniques. One disadvantage is that systematic reviews require considerably more effort that traditional literature reviews.

Surveys: Surveys can be both flexible and fixed. The classification depends on the design of the questionnaire (which data is collected) and if it is possible to apply statistical methods (Wohlin et al. 2000a). Surveys includes anything from open–ended interviews to questionnaires with closed questions. Questionnaires can reach a large set of population and provide easy to analyze data. One disadvantage with questionnaires is low response rate. Moreover, questionnaires have a risk of being misunderstood. Interviews have a higher response rate, and provides the interviewer with the possibility to explain and clarify misunderstandings. However, interviews have the disadvantage of being more time consuming and may introduce researcher bias.

The purpose of surveys is to understand, describe, and explain the pop-

ulation, from which a sample is selected (Wohlin et al. 2000a). Surveys are common in other research areas, such as social science, for example, for analyzing voting interests. The collected data from surveys are analyzed to be generalized to the population, from which the sample is drawn. However, the results from one organization may be difficult to generalize to other organizations.

Case study: A case study methodology is suited for many kinds of software engineering research (Runeson and Höst 2009). In addition, Wieringa and Heerkens (2007) lists case study as a well suited research methodology for requirements engineering research. However, the use of the term case study in software engineering research is of varying quality (Runeson and Höst 2009). The reported studies range from ambitious studies to small toy examples. There are several definitions of case study research in the literature, and in this thesis we use the following: "investigating contemporary phenomena in their context" (Runeson and Höst 2009).

A case study is of flexible research design; however, good planning is crucial for its success. A case study focus on the situation, individual, project, or organization that the researcher is interested in. The researcher collects detailed information and different data collection methods may be applied. Case studies differ from experiments in terms of identifying causal relationships; however, case studies provide a deeper understanding of the phenomena (Runeson and Höst 2009). The results from case studies are more difficult to interpret and generalize than the results from experiment (Wohlin et al. 2000a).

Action Research: Wieringa and Heerkens (2007) classifies research methods that can be used in requirements engineering research. One of the methods is action research. In action research, the researcher enters a project where tasks are performed by using the researchers technique/method. The purpose is to influence or change some aspects of the research focus. Moreover, action research aims to improve: practice, the understanding of practitioners, and the situation in which the practice takes place (Robson 2002).

The cycle of action research comprises of four steps (Robson 2002): (1) plan to improve current practice, (2) implementation of the plan (action), (3) observe effects, and (4) reflection. In fact, after the reflection step, the researcher evaluates the performance of the used technique or method and draws conclusions, which may lead to improvements of the technique or method. The emphasis on the situation and improving practice in a particular context, and to produce a change in that particular context, palace action research in the strategy of case studies (Robson 2002).

4.3 Research methods

Without proper data collection and analysis methods, the essence of the collected data may not be revealed nor possible to communicate. There are a variate of research methods to choose from, and the researcher's choice is dependent on the information sought after (Robson 2002). The following Section focus on the research methods used in this thesis.

Content analysis: The first method for data collection and analysis is review of written documents. The focus of content analysis is to gather information and generate findings. The gathered information (content) can be any written information. After the content has been gathered, it is analyzed and conclusions based on the content is reported. Content analysis differs from interviews (interviews are described below) in terms of being indirect. This means that the observer does not affect the documents.

In this thesis, papers I and III have collected data from existing documents. However, content analysis also includes analyzing the content from interviews, and content analysis has been used for this purpose in papers II, IV and V.

Interviews: In interviews, the researcher is the instrument for data collection. One advantage with interviews is the flexibility. The interviewer has the possibility to follow up answers, interpret the tone of the voice, expressions and intonations of the interviewee, which documents or written answers cannot reveal. One disadvantage with interviews is that they are rather time consuming.

Interviews can be classified into three interview types, fully structured, semi–structured, and unstructured (Robson 2002). Semi–structured interviews has predetermined questions; however, the interviewer can change the order and the wording of the questions. For elaboration of fully structured and unstructured interviews, see Robson (2002).

The interviews performed in this thesis have been of the semi–structured interview type (papers II, IV and V).

4.4 Research Classification

The results in this thesis have been reached through the use of the presented research strategies and methods. Table 3 provides a mapping between the presented papers, research questions, research strategies, research designs, and research methods.

In paper I, systematic review is chosen as research strategy. A reason to use systematic review is related to RQ1, "what empirical evidence of managing quality requirements exists in the literature". Since one of the main reasons for undertake a systematic review is to summarize the existing evidence concerning a treatment, in this research, quality requirements, this

Table 3: Research classification

Paper	Research Question	Research Design	Research Strategy	Research Method
I	RQ1	fixed	systematic review	content analysis
II	RQ2	flexible	survey	interviews
III	RQ2	flexible	case study	content analysis
IV	RQ3	flexible	action research	interviews
V	RQ3, RQ4	flexible	action research	interviews

research strategy's purpose is well aligned with RQ1. One may argue that a literature review, which most research starts with, could have been used instead. However, unless the literature review is thorough, it is of less scientific value. A systematic review synthesizes existing work in a manner that is thorough and fair, and a predefined search strategy is used. In addition, the methodology in systematic reviews is well–defined and minimizes the risk of literature bias. The used research method in paper I is content analysis. The main reason for using this method is that the collected data (publications in this study) is already produced, which makes the indirect method of content analysis (instead of direct observing or interviewing for the purpose of the study) well suited in paper I.

A Survey is used as research strategy in paper II. The reason for using a survey strategy is that the aim of paper II is to understand and describe how quality requirements are handled in practice, which is aligned with the purpose of surveys. In addition, surveys aim to explain the population, which is more difficult in an experiment since it is not possible to generalize the results outside the controlled conditions. Case studies focus on the situation or organization and provide a deep understanding; however, the results are difficult to generalize, thus the case study strategy is not well suited for the study in paper II. Interviews were used as research method approach because of the ability to interpret the tone of the voice, expressions, and intonations of the interviewee. Moreover, interviews provide the researcher with the ability to explain questions (if misunderstood) and to follow up answers.

The aim of paper III is to analyze a quality requirements specification in practice in an industrial context and to classify quality requirements that are used in industry. An in–depth analysis of a single case helps to understand the details of a specific context, therefore, case study is chosen as research strategy. The analyzed data is a real requirements specification with 2113 requirements from a case company, which was already produced. Considering the collected data and the aim of paper III, content analysis is the most suitable research method as it is an indirect method.

Action research was used as research strategy in both paper IV and V.

The aim of paper IV and V was to evaluate the introduction of a new method and to improve the situation of managing quality requirements. Action research aims to influence or change some aspects of the research focus, and the improvement of practice and the situation in which the practice takes place. In paper IV and V, we were involved in several steps to improve the practice of release planning of quality requirements by introducing the QUPER model. To apply QUPER in practice, the guidelines in paper IV were developed in cooperation between industry and academia. Moreover, the tailoring of the model (paper V) was carried out by the researchers together with practitioners at the case company. We participated in the process to set the QUPER model into operation, and introduced how to use the model in practice to the practitioners in several workshops. To evaluate the introduction of QUPER, interviews were used as research method approach because of the ability to interpret the tone of the voice, expressions, and intonations of the interviewee.

4.5 Validity

Even though the research in this thesis has been conducted with reliable, and well–known strategies, and methods, the result should be questioned and evaluated. The validity of the result should always be addressed. In this thesis, four perspectives of validity and threats as presented in Wohlin et al. (2000a) are considered: conclusion, internal, construct, and external validity. An overview of these validity threats is presented below.

Conclusion validity arise from the ability to draw accurate conclusions, i.e. the reliability of the results (Wohlin et al. 2000a). Conclusion validity is related to the repeatability of the study, such as the data collection procedures. That is, if the same study is repeated, and the results are the same, then the study has a high degree if reliability (Yin 2003).

Internal validity is related to issues that may affect the causal relationship between treatment and outcome, for example, a change in the subjects environment may affect the outcome without the researcher knowing about it (Wohlin et al. 2000a). If the researcher incorrectly concludes that the treatment affects the outcome without knowing that a third factor has caused the outcome, then the study has a low degree of internal validity (Yin 2003). Internal validity is a large threat to case studies due to that industrial environment changes over time (Yin 2003).

Construct validity is concerned with the relation between theories behind the research and the observations. The use of *multiple sources of evidence* and establish a *chain of evidence* may increase construct validity (Yin 2003) and ensure that the result is an effect of the treatment.

External validity is concerned with the ability to generalize the findings beyond the actual study. Results obtained in the context of a unique environment, or with a specific group of subjects may not be fully generalizable to other contexts and environments. However, qualitative studies rarely attempt to generalize beyond the actual setting since they are more concerned with explaining and understanding a phenomena.

Threats to validity of the result in this thesis are discussed in conjunction with research results in Section 5 and separately in each of the included papers.

5 Research Results

This section presents the main contributions of this thesis related to each research question. The discussed contributions are based on the conclusions from the results of the included papers. In addition, the main threats to validity to these conclusions related to each research question are summarized. Detailed contributions and threats to validity of each paper in this thesis can be found in the respective papers.

Main contributions of RQ1. What empirical evidence of managing quality requirements exist in the literature?

The first research contribution is the systematic review, which is described in paper I. The systematic review seeks to collect and compare existing empirical evidence on quality requirements in relation to the software product management domain. The systematic review generated research questions and laid the basis for studies regarding research question two (RQ2).

The systematic review presents the state of research in six areas of quality requirements. It also provides a basis for further research opportunities. One of the empirical evidence that is encountered is a gap in the research literate of how cost estimation of quality requirements is conducted. The results show that only one study is related to this area. However, the study does not address how cost estimation of quality requirements is conducted; instead, the study relates to the identification of foreseen cost barriers.

Another area that lacks empirical evidence is prioritization of quality requirements. None of the identified studies investigates how quality requirements are, and should be prioritized. Other encountered empirical evidence includes six different techniques for handling interdependencies among quality requirements, and six different techniques for eliciting quality requirements.

Main validity issues of RQ1
One major threat to construct validity is the exclusion of relevant empiri-

cal studies, which may influence the identified state of research. This threat was countered by a defined stringent search strategy, and that the second and third author, each reviewed a set of 20 percent of all excluded papers. Threats to conclusion validity might affect the conclusions drawn by introducing author bias. To minimize this threat, inclusion and exclusion criteria were explicitly defined.

Main contributions of RQ2. Which are the current quality requirement challenges that need further investigation?

The second contribution is described in two papers. Paper II describes an industrial survey conducted at five Swedish embedded software–developing companies. The survey gives insights into how quality requirements are handled in the studied companies, and the challenges they face. Moreover, what quality requirements aspects do the companies feel confident as being adequately handled are discovered. Paper II generated further research opportunities, which is described in Section 6, and in the paper. Paper II presents challenges managed by market–driven software developing companies, both from a product and project perspective. Some of the challenges encountered are: when is the quality level good enough, and how to get quality requirements into projects when functional requirements are prioritized?

Another challenge is that there seems to be a bespoke development mindset where the immediate project gets a higher priority than the long–term evolution of the product. This is confirmed by the implicit management of quality requirements, and the dismissal off–hand of quality requirements with little or no consequence analysis. However, the main problem is that quality requirements are not taken into consideration during product planning and thus not included as hard requirements in the projects.

Paper III regards how quality requirements metrics are used in an industrial context. Paper III describes a case study within the embedded software domain where an in–depth analysis gave an understanding of details of a quality requirements specification in a specific context. The results show that 40 percent of the 2113 requirements are quality requirements. Furthermore, references to various standards are commonly used for quality requirements. About half of the quality requirements are quantified in the requirements specification, which seems to confirm the findings in paper II. Paper III concludes that for a method to be successful, it is important that it is flexible enough to handle the diverse nature of quality requirements. This impacts all areas of requirements engineering, starting with elicitation and analysis to specification and validation.

Main validity issues of RQ2

In the two quality requirements studies in paper II and III, the major threat is concerned with external validity, i.e. whether the results are general-

izable to other contexts. The study in paper III only covers one specific case company, therefore, the generalizability may be questioned. However, qualitative studies rarely attempt to generalize beyond the actual setting since it is more concerned with explaining and understanding the phenomena. In addition, paper II seems to confirm part of the results discovered in paper III, by a qualitative survey from five companies. This increases the possibility to generalize the results beyond a specific context.

Main contributions of RQ3. How can QUPER be transferred to industry practice?

The third contribution is also investigated in two papers. Paper IV presents a general set of guidelines of applying QUPER in practice, which is developed in a case company. The guidelines involves six steps and a template for documenting the results of QUPER's steps. The QUPER model is proposed to be aligned with the case company's current scoping process, which uses pair–wise comparisons of features in cost–benefit analysis. The general set of guidelines is used as input to the adaption and evaluation of the QUPER model in paper V.

QUPER as presented in Section 1 and paper IV is generic in nature. Therefore, in paper V, an adaption of the six steps in applying QUPER in practice (paper IV) needed to be addressed prior to the model being evaluated in a case study (the contribution of the evaluation is presented below in RQ4). The adaption only includes the first four steps from paper IV. The reason is that the case company considered these steps as the most important ones to start with. A modification of the steps from paper IV is addressed, and the order is changed. The tailored QUPER steps envisions the following practical steps: (1) define quality aspects, (2) Estimate your product's current quality (for a given release) and the competing products' quality (at present or envisioned), (3) for each quality aspect and for each relevant qualifier, estimate the breakpoints, and (4) estimate candidate targets and discuss and decide on actual targets for coming releases.

Main validity issues of RQ3
A major threat of the practical application of QUPER is concerned with external validity. The development of the guidelines has been conducted in two case studies; however, the case company is the same. We believe that the general concepts of applying QUPER in practice are transferable to requirements engineering for other domains of market–driven software product development, but this needs to be investigated in further research by applying the same set of guidelines in other companies.

Main contributions of RQ4. To what extent does the use of QUPER as a part of release planning of quality requirements result in improvements with regards to high–level decision–making?

The fourth contribution is an evaluation and improvement of QUPER. Paper V reports on findings of the practical use of the QUPER model from interviews with four experts in one case study. The model is applied in four different areas within the case company and both possibilities and limitations are discovered.

The general view of QUPER is related to the saturation breakpoint, the subjects expressed that it is good to know when to stop improving the quality. Furthermore, the relation to the real world, that is, comparing their own product's quality against the competitors level of quality is one of the beneficial aspects of the QUPER model. This comparison provides a more extensive view of the quality requirements and therefore, provides a better basis and understanding of the rationale behind a certain level of quality. With regards to high–level decision–making, QUPER is found to provide more knowledge of the current market situation, which leads to more informed decisions. Moreover, due to a better overview of the market situation, QUPER is found to improve the decision–making, particular when introducing new products on a certain market.

However, the results encountered some limitations of the model. The limitations include difficulties to identify the differentiation and saturation breakpoints, and different people may interpret the breakpoints differently. In addition, the introduction of the QUPER model required more time to quantify quality requirements compared with the previous used process.

Main validity issues of RQ4

In paper V, one threat is concerned with internal validity, i.e. whether the causal relationship between treatment and outcome has been affected. The interview subjects' answers were recored by the researcher, which may have constrained the subjects in their answers. This threat was countered by guaranteeing complete anonymity.

Another threat is related to conclusion validity, that is, the ability to draw correct conclusions. To minimize this threat, the interviews were conducted at different departments and different geographical locations within the case company. Each interview was conducted in one work session, thus, answers were not influenced by internal discussions. Moreover, the relative small sample of interview subjects may affect the drawn conclusions and may not be representative for the whole case company.

A major threat is related to external validity, in this study, the applicability of QUPER in industry at companies other than the case company. Some of the discovered limitations and possibilities could to some extent be general for organizations that are developing products for markets.

Table 4: Research plan

Further Research	Description	Research Approach
FR1	Development and evaluation of QUPER's cost view in an educational and industrial environment	Data collection: literature study, interviews Evaluation: interviews, questionnaires, controlled experiment
FR2	Development and evaluation of QUPER's roadmap view	Data collection: literature study, interviews
FR3	How to incorporate dependencies into the QUPER model	Data collection: interviews with experts
FR4	Create a deeper understanding of quality requirements challenges	Data collection: interviews
FR5	A deeper understanding of the characteristics of quality requirements	Data collection: content analysis

6 Further Research

This section describes how the research can be continued in the future, and a research plan is presented. All included papers have possibilities of further research, which are presented and discussed in relation to each paper. Our research is intended to continue with the same focus as in this thesis, namely improvement of managing quality requirements, with focus on further evaluating and developing the QUPER model. The main goal is to provide methods and guidelines that can be incorporated into the QUPER model, where the specific research goals are: (1) to investigate cost estimations of quality requirements, and (2) to improve the support of quality requirements with regards to high–level decision making. In addition, another goal is to continue the studies in this thesis. These goals could be realized through empirical studies with both qualitative and quantitative approaches. A plan for further research opportunities of interest is outlined below, and the research plan is summarized in Table 4.

FR1: Cost estimations of quality requirements
The QUPER model described in the background section consists of three views. In paper IV and V guidelines and an evaluation of the benefit view were conducted. One research opportunity is to develop practical guide-

lines and conduct an evaluation of the QUPER cost view. Data could be collected by a literature study and interviews. The purpose would be to investigate what cost estimation models have been empirically validated and what challenges and needs are faced in industry. Based on the collected data, guidelines of how to use the cost view in a practical setting could be developed.

It is of interest to investigate to what extent the practical guidelines for the QUPER cost view can assist practitioners in high–level decision making. Therefore, the cost view could be evaluated regarding its usefulness as prediction of cost estimations of quality requirements. Data could be collected by questionnaires and interviews to obtain opinions from requirements engineers. In addition, two experiments could be conducted as a complement to the questionnaires and interviews. One experiment may use students as subject while the second may use practitioners from industry as subjects. The treatment could be arranged as follows: one group could have access to the cost view when estimating the cost of quality requirements, while the other group may not have any assistance during cost estimations.

FR2: Predicting future market trends/needs for quality requirements
In addition to QUPER's cost view (FR1), QUPER's roadmap view is to predict future quality levels based on a snapshot of today's market situation for products that are released in one, two, or even three years time. To further develop the QUPER model, the roadmap view is an important part. One research opportunity would be to develop a practical application guide and conduct an evaluation of the QUPER roadmap view. Data could be collected by a literature study with a focus on what marketing models and tools exist, and are applicable to the software engineering industry. In addition, an interview study with the purpose to identify how the industry predicts future market needs in terms of quality levels could be conducted.

FR3: Dependencies and quality requirements
As mentioned in Section 3.3, dependencies can have a major impact on requirements prioritization. In addition, requirements interdependencies has an important role in release planning (Carlshamre et al. 2000). In release planning, the selection of of requirements are usually based on the requirements prioritization process. However, the selection of one requirement may imply that other requirements may be selected as well (Dahlstedt and Persson 2005). The interdependencies may also affect the development cost for other requirements, for example, the performance of feature A should not be longer than 3 seconds. This requirement may increase the cost of implementing other requirements (Dahlstedt and Persson 2005). Interdependencies can have an impact on other requirements value, for example, an on–line manual may decrease the customer value of a printed manual.

In the QUPER model, requirements dependencies have not been taken into consideration during the implementation of the benefit view. Currently, the most important dependencies are managed on the basis of expert judgment in particular crucial cases (Regnell et al. 2008a). Trying to estimate all dependencies inevitably lead to a combinatorial explosion. It is important to understand more about quality requirements interdependencies before it is possible to find heuristics to efficiently support and incorporate them into the QUPER model.

FR4: Deeper analysis of challenges with quality requirements
Paper II takes a first step towards a deeper understanding of faced challenges in relation to quality requirements in industry. We intend to continue the investigation with a larger sample size that consists of 25–30 interviews (including the interviews in paper II). This would result in a more comprehensive understanding of faced challenges when managing quality requirements. Since paper II was written, data collection from additional interviews and companies have taken place.

FR5: Deeper analysis of quality requirements metrics
Paper III takes a first step towards understanding how quality requirements are specified, in particular which metrics are used in an industrial setting. We intend to continue the investigation by a deeper analysis of the collected data. In addition, one possibility is to map the collected quality requirements to the ISO 9126 standard (9126-2001 E). This would result in a more comprehensive understanding of the characteristics of quality requirements and how reliable the ISO 9126 classification is in a particular case.

References

ANSI/IEEE Std. 830. Guide to software requirements specification, 1998.

ISO/IEC 9126-2001(E). Software engineering – product quality – part 1: Quality model, 2001.

T. AlBourae, G. Ruhe, and M. Moussavi. Lightweight replanning of software product releases. In *Proceedings of the 1st International Workshop on Software Product Management*, pages 27–34, 2006.

D. Alwis, V. Hlupic, and R. Fitzgerald. Intellectual capital factors that impact of value creation. In *Proceedings of the 25th International Conference on Information Technology Interfaces*, pages 411–416, 2003.

A. Aurum and C. Wohlin. *Engineering and Managing Software Requirements*. Springer, 2005.

S. Barney, A. Aurum, and C. Wohlin. A product management challenge: Creating software product value through requirements selection. *Journal of Systems Architecture*, 54(6):576–593, 2008.

P. Berander. Using students as subjects in requirements prioritization. In *Proceedings International Symposium on Empirical Software Engineering*, pages 167–176, 2004.

P. Berander and A. Andrews. *Engineering and Managing Software Requirements*, chapter Requirements Prioritization, pages 69–94. Springer, 2005.

B. Bergman and B. Klefsjö. *Quality from Customer Needs to Customer Satisfaction*. Studentlitteratur, 2003.

R. Berntsson Svensson and A. Aurum. Successful software projects and products. In *IEEE/ACM 5th International Symposium on Empirical Software Engineering*, 2006.

R. Berntsson Svensson, A. Aurum, C. Wohlin, and G. Hu. Successful software project and products: An empirical investigation comparing australia and sweden. In *17th Australian Conference on Information Systems*, 2006.

R. Berntsson Svensson, T. Olsson, and B. Regnell. Introducing support for release planning of quality requirements – an industrial evaluation of the quper model. In *2nd International Workshop on Software Product Management*, 2008.

B.W. Boehm and V. Basili. Gaining intellectual control of software development. *Computer*, 33(5):27–33, 2000.

B.W. Boehm, J.R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings 2nd International Conference on Software Engineering*, pages 592–605, 1976.

A. Borg, A. Yong, P. Carlshamre, and K. Sandahl. The bad conscience of requirements engineering: An investigation in real–world treatment of non–functional requirements. In *Proceedings of the third Conference on Software Engineering and Practice in Sweden*, pages 1–8, 2003.

P. Carlshamre. Release planning in market–driven software product development: Provoking an understanding. *Requirements Engineering Journal*, 7(3):139–151, 2002a.

P. Carlshamre. *A usability perspective on requirements engineering – From methodology to product development*. PhD thesis, Linköping University, Sweden, 2002b.

P. Carlshamre and B. Regnell. Requirements lifecycle management and release planning in market–driven requirements engineering processes. In *Proceedings 11th International Workshop on Database and Expert Systems Applications*, pages 961–965, 2000.

P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Proceedings 5th IEEE International Symposium on Requirements Engineering*, pages 84–91, 2000.

J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements–driven information systems engineering: the tropos project. *Information Systems*, 27(6):365–389, 2002.

L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *NFR in Software Engineering*. Kluwer Academic Publishers, 2000.

J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina. Goal–centric traceability for managing non–functional requirements. In *Proceedings 27th International Conference on Software Engineering*, pages 362–371, 2005.

L.M. Cysneiros and J.C.S.P. Leite. Nonfunctional requirements: From elicitation to conceptual models. *IEEE Transactions on Software Engineering*, 30(5):328–349, 2004.

À.G. Dahlstedt and A. Persson. *Engineering and Managing Software Requirements*, chapter Requirements Interdependencies: State of the Art and Future Challenges, pages 95–116. Springer, 2005.

J.M deBaud and K. Schmid. A systematic approach to derive the scope of software product lines. In *Proceedings of the 21st International Conference on Software Engineering*, pages 34–43, 1999.

D. Dikel, D. Kane, S. Ornburn, W. Loftus, and J. Wilson. Applying software product–line architecture. *Computer*, 30(8):49–55, 1997.

C. Ebert. Requirements before the requirements: understanding the upstream impacts. In *Proceedings 13th IEEE International Conference on Requirements Engineering*, pages 117–124, 2005.

C. Ebert. Putting requirement management into praxis: dealing with non-functional requirements. *Information and Software Technology*, 40(3):175–185, 1998.

T. Gorschek and C. Wohlin. Requirements abstraction model. *Requirements Engineering Journal*, 11:79–101, 2006.

S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings 4th IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

H-W. Jung, S-G. Kim, and C-S. Chung. Measuring software product quality: A survey of iso/iec 9126. *IEEE Software*, 21(5):88–92, 2004.

J. Karlsson. Managing software requirements using quality function deployment. *Software Quality Journal*, 6(4):311–325, 1997.

J. Karlsson and K. Ryan. A cost–value approach for prioritizing requirements. *IEEE Software*, 14(5):67–74, 1997.

J. Karlsson, S. Olsson, and K. Ryan. Improved practical support for large–scale requirements prioritising. *Requirements Engineering*, 2:51–60, 1997.

J. Karlsson, C. Wohlin, and B. Regnell. An evaluation of methods for prioritising software requirements. *Information and Software Technology*, 39 (14–15):939–947, 1998.

L. Karlsson, Å.G. Dahlstedt, B. Regnell, J. Natt och Dag, and A. Persson. Requirements engineering challenges in market–driven software development – an interview study with practitioners. *Information and Software Technology*, 49:588–604, 2007.

T. Kilpi. Product management challenge to software change process: Preliminary results from three smes experiments. *Software Process Improvement and Practice*, 3(3):165–175, 1997.

B.A. Kitchenham. Guidelines for performing systematic literature reviews in software engineering version 2.3. Technical report, Keele University and University of Durham, 2007.

S. Konrad and M. Gall. Requirements engineering in the development of large–scale systems. In *Proceedings of the 16th IEEE International Requirements Engineering Conference*, pages 217–222, 2008.

R.J. Kusters, R.V. Solingen, and J.J.M. Trienekens. Identifying embedded software quality: Two approaches. *Quality and Reliability Engineering International*, 15:485–492, 1999.

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

L. Lehtola and M. Kauppinen. Suitability of requirements prioritization methods for market–driven software product development. *Software Process Improvement and Practice*, 11(1):7–19, 2006.

J.A. McCall and M. Matsumoto. Software quality metrics enhancements, vol. i–ii. Technical report, Rome Air Development Center, 1980.

S.L. Pfleeger. *Software Engineering – Theory and practice*. Prentice–Hall, 2001.

B. Regnell and J. Brinkkemper. *Engineering and Managing Software Requirements*, chapter Market–Driven Requirements Engineering for Software Products, pages 287–308. Springer, 2005.

B. Regnell, P. Beremark, and O. Eklundh. A market–driven requirements engineering process – results from an industrial process improvement programme. *Requirements Engineering*, 3(2):121–129, 1998.

B. Regnell, M. Höst, J. Natt och Dag, P. Beremark, and T. Hjelm. An industrial case study on distributed prioritization in market–driven requirements engineering for packaged software. *Requirements Engineering*, 6: 51–62, 2001a.

B. Regnell, B. Paech, A. Aurum, C. Wohlin, A. Dutoit, and J. Natt och Dag. Requirements mean decisions! – research issues for understanding and supporting decision–making in requirements engineering. In *Proceedings of 1st Swedish Conference on Software Engineering Research and Practise*, pages 49–52, 2001b.

B. Regnell, H.O. Olsson, and S. Mossberg. Assessing requirements compliance scenarios in system platform subcontracting. In *Lecture Notes in Computer Science*, volume 4034, pages 362–376, 2006.

B. Regnell, M. Höst, and R. Berntsson Svensson. A quality performance model for cost–benefit analysis of non–functional requirement applied to the mobile handset domain. In *Lecture Notes in Computer Science*, volume 4542, pages 277–291, 2007.

B. Regnell, R. Berntsson Svensson, and T. Olsson. Supporting roadmapping of quality requirements. *IEEE Software*, 25(2):42–47, 2008a.

B. Regnell, R. Berntsson Svensson, and K. Wnuk. We beat the complexity of very large–scale requirements engineering? In *Lecture Notes in Computer Science*, volume 5025, pages 123–128, 2008b.

C. Robson. *Real World Research*. Blackwell, 2002.

G-C. Roman. A taxonomy of current issues in requirements engineering. *IEEE Computer*, 18(4):14–23, 1985.

G. Ruhe and M.O. Saliu. The art and science of software release planning. *IEEE Software*, 22(6):47–53, 2005.

P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

T. Saaty. *The Analytical Hierarchy Process*. McGraw–Hill, 1980.

J. Schalken and S. Brinkkemper. Assessing the effects of facilitated workshops in requirements engineering. In *Proceedings of 8th IEEE International Conference on Empirical Assessment in Software Engineering*, pages 135–144, 2004.

G.G. Schulmeyer and J.I. McManus. *Handbook of Software Quality Assurance*. Prentice Hall, 1999.

I. Sommerville. *Software Engineering*. Addison–Wesley, 2007.

H. Thayer. Software engineering: a tutorial. *IEEE Computer*, pages 68–73, 2002.

R. Thayer and M. Dorfman. *Systems and Software Requirements Engineering*. IEEE Computer Society Press, 1990.

M.I. Ullah and G. Ruhe. Towards comprehensive release planning for software product lines. In *Proceedings of the 1st International Workshop on Software Product Management*, pages 51–55, 2006.

I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma. On the creation of a reference framework for software product management: Validation and tool support. In *Proceedings of the 1st International Workshop on Software Product Management*, pages 3–11, 2006a.

I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma. Towards a reference framework for software product management. In *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pages 312–315, 2006b.

R. Wieringa and H. Heerkens. Designing requirements engineering research. In *Proceedings of 5th International Workshop on Comparative Evaluation in Requirements Engineering*, pages 36–48, 2007.

C. Wohlin and A. Aurum. What is important when deciding to include a software requirement in a project or release? In *Proceedings of 4th International Symposium on Empirical Software Engineering*, pages 237–246, 2005.

C. Wohlin, P. Runeson, M. Höst, M.C. Ohlson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An introduction*. Kluwer Academic, 2000a.

C. Wohlin, A. von Mayrhauser, M. Höst, and B. Regnell. Subjective evaluation as a tool for learning from software project success. *Information and Software Technology*, 42:983–992, 2000b.

R.K. Yin. *Case Study Research: Design and Methods*. Sage Publications, 2003.

Paper I

Managing Quality Requirements: A Systematic Review

Richard Berntsson Svensson, Martin Höst, Björn Regnell
Department of Computer Science, Lund University, Sweden
{richard.berntsson_svensson,martin.host,bjorn.regnell}@cs.lth.se

Submitted to Information and Software Technology, 2009

ABSTRACT

It is commonly acknowledged that the handling and balance of quality requirements are important and difficult parts of the requirements engineering process, and playing a critical role in software product development. This paper presents a systematic review of empirical studies of quality requirements. The search strategy identified 2647 studies by searching the literature, of which 22 were found to be empirical research studies of acceptable quality, and related to the research questions. The studies are grouped into six areas: elicitation, dependencies, quality requirements metrics, cost estimations, prioritization, and software product management. The review investigates what is currently known about the benefits and limitations of methods for quality requirements. Moreover, the strength of evidence of the reviewed studies is evaluated. The main implication for research is a need for more empirical studies on quality requirements. In addition, replication of current methods/techniques for quality requirements is needed. For practitioners, the review provides an overview of what is currently known about methods for quality requirements. In addition, the identified methods give the practitioners the opportunity to compare to their own industrial environment..

Keywords: Empirical software engineering, systematic review, evidence–based, quality requirements, non–functional requirements

1 Introduction

The complexity of software systems is determined by both functionality and by quality aspects such as performance, reliability, accuracy, security, and usability (Chung et al. 2000). These quality aspects, or non–functional requirements are subsequently called quality requirements (QR). It is commonly acknowledged that the handling and balance of QR are important and difficult parts of the requirements engineering process (Jacobs 1999), and that QR are playing a critical role in software developmen (Chung et al. 2000).

One characteristic of QR is the specification of certain quality levels, and therefore QR are in many cases possible to quantify (Olsson et al. 2007). This is important, not only for understanding QR (Jacobs 1999), but also for planning (Regnell et al. 2007). Not dealing, or ineffectively dealing with QR may lead to more expensive software and longer-time–to–market (Cysneiros and Leite 2004), or in worst case, failures in software development ((Breitman et al. 1999), (Finkelstein and Dowell 1996)). Studies ((Jr. 1987), (Cysneiros and Leite 1999)) have showed that QR are expensive and difficult to handle, and according to Chung et al. (2000), QR are often poorly understood in comparison to less critical aspects of software development.

It is generally agreed that decisions about what QR to state on a product have large effects on the development project and the choice of architecture. This means that the area of QR is important to understand in more detail and to understand which dependencies there are between different QR.

This paper seeks to collect and compare existing empirical evidence on quality requirements to date. Also, an overview of findings, strength of findings, and implications for research and practice are studied. Kitchenham et al. (2009) published a review of all conducted systematic reviews. Among the 20 identified systematic reviews, not a single topic area was about quality requirements.

Since the area of QR is an important area for research there is a need for a review of the state of research in the area. This systematic review is intended to provide this state of research and in this way it can serve as a basis for further empirical research on the subject. A systematic review can, in general, both provide an understanding of the performance of different methods in an area, and provide a summary of which methods are available and how they are related. This paper is mainly concerned with the latter objective.

The paper is structured as follows: In section 2, an overview of quality requirements and existing reviews are presented. Section 3 describes the review method used in this systematic review. In section 4, findings of the review are presented while section 5 discusses benefits and implications for practice and research. Section 6 gives a summary of the main conclusions and provides recommendations for further research on quality

requirements.

2 Background and Related Work

This section first describes quality requirements in relation to elicitation, dependencies, metrics, cost estimation, prioritization, and software product management. Then, a summary of previous reviews of requirements engineering literature is provided.

2.1 Quality Requirements

Functional requirements describe the interaction between the system and its environment (Pfleeger 2001), while quality requirements specify how well the system must perform the functions (Lauesen 2002). Instead of specifying what the system must do, quality requirements put restrictions on the system (Pfleeger 2001). It is commonly acknowledged that the handling and balance of quality requirements are an important and difficult part of the requirements engineering process (Jacobs 1999). Furthermore, quality requirements are often poorly understood (Chung et al. 2000), despite that they are of major importance (Ebert 2005). According to Kotonya and Sommerville (1998), quality requirements are of major importance because they are restrictions on the systems, and therefore, functional requirements may have to be excluded to meet these quality requirements. Despite the knowledge of the importance of quality requirements, quality requirements receive little attention in the literature (Cysneiros and Leite 2004), and users are often dissatisfied with software quality (Jung et al. 2004). Quality requirements are a key differentiator between a company and its competitors (Doerr et al. 2005).

There are a number of important research areas with respect quality requirements. However, this systematic review focus on areas related to the software product management domain. That is, requirements elicitation, requirements metrics (quantification of quality requirements), cost estimation, requirements prioritization, requirements dependencies, and software product management in general. Research areas such as architecture and design are not within the scope of this systematic review. For further elaboration on the software product management domain, see van de Weerd et al. (2006b). In the following, the considered research areas are described, and the difficulties when considering quality requirements are presented.

Requirements elicitation is about finding and revealing requirements from different stakeholders for a software product. Elicitation of quality requirements involves additional challenges compared to elicitation of functional requirements. Lack of deep understanding of quality requirements of the application domain, and quality requirements are often specified in-

formally (Balushi et al. 2007), for example, the software product should be fast or the user interface should be easy to use.

Requirements metrics relates to quantification of requirements, to make them possible to verify. Olsson et al. (2007) found that 50% of all quality requirements in a requirement specification were specified with a metric. This may indicate difficulties in specifying verifiable quality requirements.

Cost estimation is about estimating the needed development effort for a particular requirement, function, or feature. Estimating the development effort for a particular quality requirement may be even more complex considering the complex interdependencies among them.

Requirements prioritization involves cost and value estimations of the requirements (Karlsson et al. 1998) and is the foundation for release planning (Ullah and Ruhe 2006) in software product management. Models that address requirements prioritization often emphasize functional aspects, for example, the cost–value approach for requirements prioritization (Karlsson and Ryan 1997).

Requirements dependencies: Priority of requirements is a major determinant in planning, but the fact that requirements are related to each other makes it difficult or sometimes even impossible, to schedule requirements based on priority only (Carlshamre et al. 2000). However, the situation is even more complex for quality requirements because they tend to have a global impact on the entire system (Cysneiros and Leite 2004).

Software product management includes several important activities, such as requirements management, release planning, and launching products (van de Weerd et al. 2006a). Ullah and Ruhe (2006), lacking of good release planning practices may results in unsatisfied customers and market loss, which makes release planning a major determinant of the success of a product. When dealing with quality requirements such as performance, usability, and reliability, we often end up in a difficult trade–off analysis. Aspects such as release targets, end–user experience, and business opportunities must be taken into account (Berntsson Svensson et al. 2008).

2.2 Related Work

Reviews of requirements engineering that are related to this systematic review are presented by Davis et al. (2006) and van Lamsweerde (2000). In addition, several systematic reviews ((Grimstad et al. 2006), (Jørgensen 2004), (Jørgensen and Shepperd 2007), (Kitchenham et al. 2007), (Mair and Shepperd 2005)) of cost estimations are presented.

Davis et al. (2006) conducted a systematic review of the effectiveness of requirements elicitation techniques. The results show that the most effective elicitation technique, in several domains, is interviews. In addition, interviews were found to be the most commonly used elicitation technique in practice. Several other elicitation techniques, such as, ranking and card sorting were found to be less effective than interviews. Davis et al. (2006)

found that careful preparation of interviews has much higher impact on the elicitation result than the analyst's experience, which contradicts the old ideas of requirements engineering.

van Lamsweerde (2000) reviewed the history of the main concepts and techniques to support requirements engineering activities and found, in the first 25 years of requirements engineering, that modeling appears to be important in requirements engineering. Several modeling techniques were introduced in the literature. Later on, integration of goal–based reasoning was introduced to requirements models. Two frameworks arose, the formal framework and the qualitative framework. The qualitative framework was the foundation for the NFR methodology for capturing and evaluating goal decompositions. The next step was the introduction of scenario–based elicitation. Scenarios were introduced due to the difficulties of eliciting goals.

van Lamsweerde (2000) identifies several issues that need more attention in the next 25 years. First, efforts to bridge the gap between requirement engineering research and research in software architecture are needed. Internet becomes more accessible, which enables more end–users to access applications. Based on this, van Lamsweerde suggests that "Define–it–yourself" approaches should be explored in the area of supporting requirements engineering. Moreover, requirements models need to capture more knowledge about aspects, concerns and activities in the requirements engineering process. In addition, a lot of research has been conducted on new languages and notions; van Lamsweerde suggests it is time to use the new languages and notions to build complex artifacts. Finally, tool support for requirements engineering developments need to be addressed.

Several systematic reviews of cost estimations have been conducted in recent years (Kitchenham et al. 2009). The first systematic review of the existing literature on cost estimations was published by Jørgensen in 2004 (Jørgensen 2004). Jørgensen (2004) concluded that expert estimation is the dominating strategy, and that there is no evidence to support the superiority of model estimates over expert estimates. In 2005, Mair and Shepperd (2005) conducted a systematic review of software engineering cost estimation and focused on regression and analogy techniques. Mair and Shepperd (2005) did not find clear evidence of which technique should be preferred. Grimstad et al. (2006) studied typical software effort estimation terminology, and concluded that a more precise terminology is needed. Jørgensen and Shepperd (2007) conducted a systematic review of software development cost estimation studies in 2007. A systematic review by Kitchenham et al. (2007) looked into under what circumstances companies would benefit on cross–company–based estimations models. However, Kitchenham et al. (2007) did not find any conclusive explanations of why some companies would benefit from using the models, but others would not.

The focus of the above mentioned reviews have not been on quality re-

quirements, nor has any systematic review of quality requirements previously been published (Kitchenham et al. 2009).

3 Review Method

A systematic review aims to identify, evaluate and interpret all available research to a particular research question or to examine empirical evidence that supports or contradicts theoretical hypotheses (Kitchenham 2007). The research method used in this study is a systematic review (Kitchenham 2007), which includes several stages: planning the review, identification of research questions, search strategy and search, selection of studies, quality assessment, data extraction, and data synthesis. In the reminder of this section, each stage of the systematic review is described in details.

3.1 Planning the Review

This systematic review started by developing a review protocol following Kitchenham's guidelines (Kitchenham 2007). One of the aims of the review protocol is to reduce the possibility of research bias (Kitchenham 2007). The review protocol specified the background to the systematic review, the research questions, search strategy, search process, inclusion and exclusion criteria, quality assessment criteria, data extraction, and method of synthesis. The aim of this systematic review was to provide an overview of empirical studies of quality requirements in software engineering, answering the research questions listed in Section 3.2.

3.2 Research Questions

This systematic review aims at summarizing the current state–of–the–art in quality requirements by answering the following research questions:

RQ1. What are the results of empirical investigations on quality requirements in relation to elicitation, metrics, cost estimation, prioritization, dependencies, and software product management?

RQ2. What empirical research methods have been used within the area?

3.3 Search Strategy and Search

The search strategy for a systematic review is a plan to identify a set of relevant publications in relation to the research questions. The search string used in this systematic review was constructed in the following steps:

1. Check keywords in relevant papers, the authors' knowledge of the

Table 1.1: Identified search terms

Category	Search words
C1: QR	non–functional requirements OR nonfunctional requirements OR non functional requirements OR quality attributes OR quality requirements OR non–functional software requirements OR quality characteristics OR quality factors OR qualities
C2: software	software
C3: elicitation	elicitation OR requirements gathering OR requirements acquisition
C4: dependency	dependenc* OR trade–off OR tradeoff OR trade–offs OR tradeoffs OR trade off OR trade offs OR interdependenc* OR change impact OR traceability OR relationships OR inter–dependencies OR conflict
C5: Metrics	metrics OR measurement
C6: cost	software development effort OR cost estimation
C7: prioritization	prioritization OR prioritizing OR prioritize OR prioritisation OR prioritising OR prioritise
C8: Software product management	release planning OR roadmapping OR road mapping OR roadmap OR scope OR scoping OR software product management OR software product

areas, and from previous systematic reviews that are related to our research questions?
2. Identify alternative spelling and synonyms for the keywords
3. A test search in databases was performed to validate and identify new keywords and synonyms
4. Keywords, alternative spelling, and synonyms were grouped into *categories* according to their search area
5. Use the Boolean "OR" operator to connect keywords and alternative spelling and synonyms for each *category*
6. Use the Booleans "AND" and "OR" operator to create the search string by connecting the different *categories*

Table 1.1 presents the identified categories of search terms from the construction of the search string. Keywords and synonyms for empirical studies were not included in the search string. Instead, a screening question in the quality assessment checklist was added to decide if a study is empirical or not, see Section 3.5.

Chung et al. (2000) defines about 160 different terms for quality requirements and several standards that define quality requirements. To include

of all of these specific terms would have created a too long and complicated search-string, and still we would not be sure that all of them are covered. Therefore, specific terms of quality requirements, such as performance and usability requirements were excluded. Thus, category 1 consists of different general terms for quality requirements. Categories 3 and 6 are inspired by previous systematic reviews by (Davis et al. 2006) and (Jørgensen and Shepperd 2007) respectively. The final search string was constructed in the following way:

Search–string = C1 AND C2 AND (C3 OR C4 OR C5 OR C6 OR C7 OR C8)

The search strategy included the following electronic databases:

- ACM Digital Library[1]
- Compendex and Inspec[2]
- IEEE Xplore[3]
- Wiley Inter Science Journal Finder[4]

In addition, the search–string was applied to two more electronic databases, ScienceDirect – Elsevier[5] and SpringerLink[6] . However, the search string was too long and complex to be used in these databases. Therefore, a subset of the search string was used in both ScienceDirect – Elsevier and SpringerLink, and compared the result from the five databases. All articles from the subset found in ScienceDirect – Elsevier and SpringerLink were also found in the included databases. Based on this test search, we concluded that articles in ScienceDirect – Elsevier and SpringerLink would be found among the five included databases.

The International Requirements Engineering conference, the major event in requirements engineering is indexed in the Compendex and Inspec databases, therefore, not manually searched. However, all volumes of the following proceedings were manually searched:

- International Workshop on Software Product Management
- Measuring Requirements for Project and Product Success
- Requirements Engineering: Foundation for Software Quality

[1] portal.acm.org
[2] www.engineeringvillage2.org
[3] ieeexplore.ieee.org
[4] www3.interscience.wiley.com
[5] www.sciencedirect.com
[6] www.springerlink.com

Whenever an electronic database did not allow the use of our search string or the use of complex Boolean applied to the titles, abstract, and keywords, we designed different search strings for each of these databases. Excluded from the search were expert opinions, summaries of tutorials and workshops, discussions, and comments.

In Section 3.4, the systematic review process is illustrated, and the number of papers identified at each phase. In phase 1, the search string was used to search for articles based on the title, abstract, and keywords in the included electronic databases. The search was performed the 10th October 2008 and resulted in a total of 2647 articles that included 1560 unduplicated articles. These 1560 articles formed the basis for the following phases in our selection process.

3.4 Selection of Studies

Our selection of studies process comprised of four phases, which is shown in Figure 1. Relevant studies from phase 1 (2647 studies) were entered into a reference database tool, while duplicates were excluded. For each of the following phases, separate databases and spreadsheets were created. Only papers written in English are included.

At phase 2, one researcher (the first author) read through the 1560 unduplicated titles of all studies from phase 1, to determine their relevance to this systematic review. We included papers that were about quality requirements or related to any of the following categories: elicitation, dependencies, metrics, cost estimation, prioritization or software product management, independently of whether they were empirical or not. The reason for this broad inclusion is that titles are not always a good indicator of what an article is about (Jørgensen and Shepperd 2007). To minimize the threat of excluding relevant papers, the first author randomly selected two sample sets (with different papers) of 10% of the excluded papers. The second and third authors were provided with one sample set each to include or exclude papers. Any disagreement between the authors was resolved by discussion that included all three researchers. Two papers were up for discussion; however, not a single paper was added to the included ones. At this phase, 727 papers were included.

At phase 3, papers were excluded, on the basis of abstract, if focus or main focus was not quality requirements and related to the same categories as in phase 2 or if they did not present any empirical data. However, according to (Dybå and Dingsyr 2008), abstracts can be of different quality and misleading. Therefore, all papers that indicated some form of empirical data were included for review in phase 4. All abstracts were reviewed by the first author, while the second and third author reviewed a total of 20% of all papers that were excluded by the first author. Any disagreement between the authors was resolved by discussion that included all three researchers. Four papers were up for discussion, and one paper was added

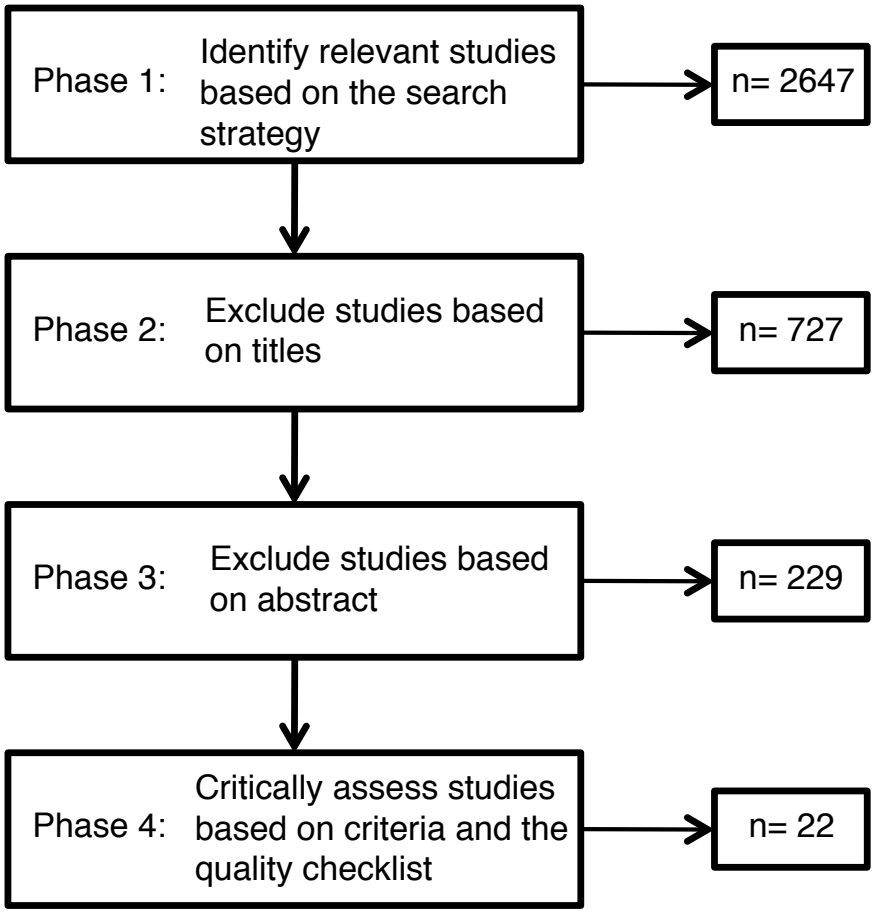

Figure 1.1: Phase of the selection of studies process

to the included papers after the discussion. After phase 3, 229 papers were included for the detailed quality assessment (see Section 3.5).

3.5 Quality Assessment

Each of the 229 studies from phase 3 was assessed by the first author according to 10 criteria, see Table 1.2. These 10 criteria are based on a researcher's checklist for case study by Runeson and Höst (2009). However, despite that the criteria are adopted from a case study checklist, all empirical studies that fulfilled the quality criteria were included in this systematic review. The 10 criteria covered three main aspects, (1) study design, (2) preparation for data collection, and (3) analysis of collected data. In addition, one criterion (Q1 in Table 1.2) was used to decide whether the study

Table 1.2: Quality assessment checklist

ID	Quality criteria
Q1	Is the study based on empirical research?
	Study Design
Q2	Are the research questions, objectives of study and aims well defined?
Q3	Is the studied context well defined?
Q4	Is it motivated that the research design is suitable to address the research questions?
	Preparation for data collection
Q5	Are data collection procedures sufficient for the purpose?
	Analysis of collected data
Q6	Are the analysis procedures sufficient for the purpose?
Q7	Are findings clearly stated, results credible, and conclusions justified?
Q8	Are different views taken on the case?
Q9	Are threats to validity analyses addressed in a systematic way?
Q10	Are conclusions, implications for practice and future research, reported suitably for its audience?

was based on empirical data or not.

One reason for including Q1 is that the term case study appears in software engineering research papers even though small toy examples have been used (Runeson and Höst 2009). In addition, studies that used the term case study in their title or abstract were identified; however, the used case was based on an example from literature. The definition of what constitutes a case study is not always obvious. Therefore, the following definition of case study was used in this systematic review: *An empirical method aimed at investigating contemporary phenomena in their context* (Runeson and Höst 2009).

The ten questions in Table 1.2 provided a measure of the quality of the studies and a measure of how confident we were about a study's findings. The first question (Q1) was graded on a "yes" and "no" scale, where "yes" was assigned the value of one (1) and "no" the value of zero (0). Questions 2 to 9 were graded on a "yes" (1), "partly" (0.5), and "no" (0) scale. The result of the quality assessment checklist is displayed in Table 1.3.

The first three questions (Q1, Q2, and Q3) were used as screening questions to exclude non–empirical research papers. If a study received a "no" score on Q1, or if both Q2 and Q3 received a "no" score, we did not continue with the quality assessment and the study was excluded.

Table 1.3: Quality Scores

Publication ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
P1	1	1	1	1	1	0.5	1	1	0	0.5	8
P2	1	1	1	1	1	1	1	1	0	1	9
P3	1	1	1	1	1	1	1	1	1	1	10
P4	1	1	0.5	1	0.5	0.5	1	0.5	0	0.5	6.5
P5	1	1	1	1	1	1	1	1	0	1	9
P6	1	1	1	1	1	1	1	1	0	1	9
P7	1	0.5	1	0.5	0.5	0	1	1	0	1	6.5
P8	1	0.5	0.5	1	1	0.5	0.5	1	0	0.5	6.5
P9	1	1	1	1	0.5	0.5	1	1	0	0.5	7.5
P10	1	1	1	0.5	0	0.5	1	1	1	1	8
P11	1	0.5	1	0	0	0	1	1	0	0.5	5
P12	1	1	0.5	0	0.5	0	1	0.5	0	0.5	5
P13	1	1	0.5	1	1	1	1	0.5	0	0.5	7.5
P14	1	1	0.5	1	1	0.5	1	1	0	0.5	7.5
P15	1	1	0.5	1	1	0.5	1	1	0	0.5	7.5
P16	1	1	1	1	1	1	1	1	0	1	9
P17	1	1	1	1	1	1	1	0	0.5	1	8.5
P18	1	1	1	1	0.5	0	1	1	0	1	7.5
P19	1	1	0.5	1	1	0.5	1	1	0	1	8
P20	1	0	0.5	0.5	0.5	0	1	1	0	0.5	5
P21	1	1	0.5	1	1	0.5	1	1	0.5	0.5	8
P22	1	1	0.5	0.5	0.5	0.5	1	1	0	0.5	6.5
Total	**22**	**19.5**	**17**	**18**	**16.5**	**12**	**21.5**	**19.5**	**3**	**16**	

Table 1.4: Data extraction form

General study description
Publication identification
Bibliographic information: author(s), year, title, publication information
Study aims and research questions
Methodology: design of study, sample description, context of study, data collection, data analysis
Tools/approaches/techniques used

Study findings
Result(s)
Conclusions
Limitations and threats to validity

3.6 Data Extraction and Synthesis

During the data extraction phase, data was extracted from 22 primary studies. A predefined data extraction form was created to collect the needed information to be able to address the research questions. The quality of each study was not part of the data extraction form since it was assessed during the quality assessment phase (Section 3.5). The extracted data was divided into two categories: (1) general study description, and (2) study findings, which is displayed in Table 1.4. In addition, each primary study was classified into one (or more) of the six categories described in Table 1.1 and Figure 1.2. In this systematic review, a descriptive (non–quantitative) synthesis (Kitchenham 2007) was conducted. The data synthesis was carried out individually by the first author. However, any uncertainty of the classification or the extracted data from the primary studies was solved through discussion within the research team.

3.7 Threats to Validity

In this section, threats to this systematic review are discussed. Similar to (Engström et al. 2008), we base this on the discussion of validity and threats presented in Wohlin et al. (2000). One type of threats mentioned in (Wohlin et al. 2000) is not relevant to this systematic review, which is external validity. External validity is concerned with the ability to generalize the findings beyond the actual study. The validity threats considered are: construct, conclusion, and internal validity threats respectively.

Construct validity: The construct validity is concerned with the relation between theories behind the research and the observations. One threat to

construct validity is exclusion of relevant studies. To address this issue, a stringent search strategy (Section 3) was defined, which included four phases. A quantitative inter–rater reliability study, e.g. based on the Kappa statistic, was not conducted at any of the four phases. However, a set of 20% of all excluded papers at each phase was reviewed by the second and third author. Only one paper from all phases was added to the included papers after the discussion. This indicates that the authors agreed to a high extent concerning which papers to include and exclude, which also indicates that there were no misunderstandings about constructs of the study.

Conclusion validity: Threats to conclusion validity arise from the ability to draw accurate conclusions. With respect to the conclusion validity, two concerns were raised; author bias may have been introduced in the quality assessment criteria and during data extraction. In order to minimize the threat to quality criteria, inclusion and exclusion criteria were explicitly defined. In addition, we had a focus of inclusion at phase two and three, meaning that papers were rather included than excluded. Any disagreement between the authors was resolved by discussions that included all three researchers until an agreement was reached. To address possible data extraction bias, we ensured that any dubious data extraction was discussed between all researchers until an agreement was reached.

Internal validity: This threat is related to issues that may affect the causal relationship between treatment and outcome. One concern related to internal validity is relevant for this study, namely unpublished studies may affect the result are not made available. These studies are difficult to obtain; however, inclusion of such studies would have increased the internal validity.

4 Results

This section describes the descriptive evaluation of the identified studies in relation to the research questions (see Section 3.2). Out of 2647 papers analyzed in the systematic review, 22 publications (P1–P22) were identified and categorized into six different topic areas in relation to quality requirements (QR), based on the empirical evidence that is presented in the study. The topic areas are elicitation, dependencies, cost estimations, metrics, prioritization, and software product management (SPM), which is displayed in Table 1.5.

Figure 1.2 provides an overview of the 22 identified publications in relation to the topic areas they have been applied to. Figure 1.2 shows that six publications are related to quality requirements and dependencies, while only three publications have looked into prioritization of quality requirements. Only one paper (P19) related to cost estimations of quality requirements was identified with our search criteria.

In the following subsections, first, a general analysis of the primary stud-

Table 1.5: Primary studies, P1–P22

Publication ID	Reference	Topic area where QR have been applied	Type of study
P1	Al-Kilidar et al. (2005)	Metrics	Experiment
P2	Andersson and Bosch (2005)	SPM	Case Study (Multiple)
P3	Berntsson Svensson et al. (2008)	Metrics, SPM	Case Study
P4	Boehm and In (1996)	Dependencies	Experiment
P5	Cleland-Huang et al. (2005)	Dependencies	Experiment
P6	Cysneiros and Leite (1999)	Elicitation	Case Study (Multiple
P7	Doerr et al. (2005)	Elicitation, Metrics, Dependencies	Case Study (Multiple
P8	Hassenzahl et al. (2001)	Elicitation	Case Study
P9	In and Boehm (2001)	Dependencies	Experiment
P10	In et al. (2001)	Dependencies	Case Study
P11	Jacobs (1999)	Metrics	Case Study
P12	JaeJoon et al. (2001)	Elicitation, Metrics	Case Study
P13	Johansson et al. (2001)	Prioritization	Survey
P14	Kusters et al. (1999a)	Elicitation	Case Study (Multiple
P15	Kusters et al. (1999b)	Elicitation	Case Study (Multiple
P16	Leung (2001)	Metrics, Prioritization	Survey, Experiment
P17	Olsson et al. (2007)	Metrics	Case Study
P18	Poort and de With (2004)	Dependencies	Case Study (Multiple
P19	Regnell et al. (2007)	Metrics, Cost, SPM	Case Study (Multiple
P20	Regnell et al. (2008)	Elicitation, Metrics, SPM	Case Study
P21	Sibisi and van Waveren (2007)	Prioritization	Survey
P22	Zulzalil et al. (2008)	Dependencies	Case Study (Multiple

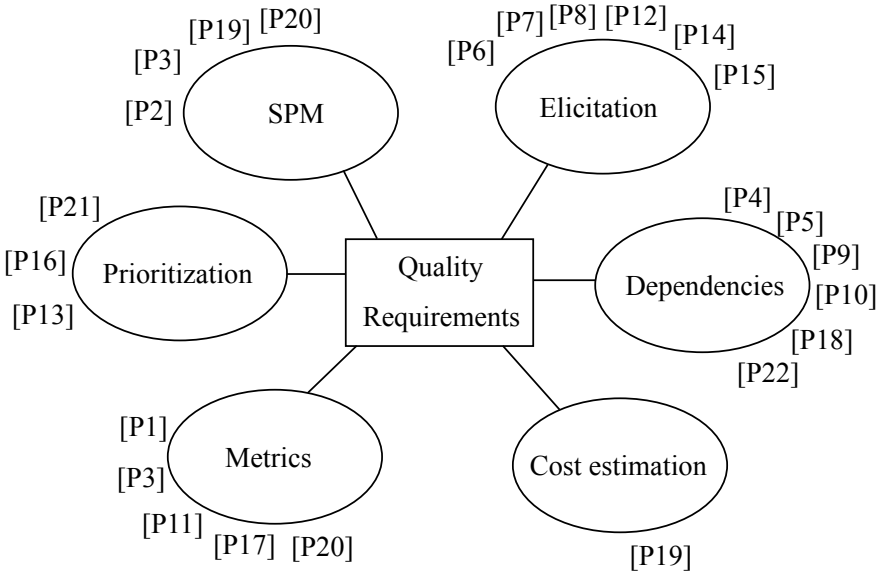

Figure 1.2: Overview of primary studies related to topic area

ies, then the main findings from the empirical studies in relation to each of the six topic areas are presented.

4.1 General Analysis of Primary Studies

In this subsection, the primary studies with regards to where they are published, which year they are published, the type of study (case study, experiment etc), sample description, and the quality of the primary studies are analyzed.

The two major requirements engineering conferences, the International Conference on Requirements Engineering and the International Working Conference on Requirements Engineering: Foundation for Software Quality, are represented among the publications. In addition, major software engineering conferences and journals are also represented. It is not surprising that the International Conference on Requirements Engineering has most publications (together with the IEEE Software journal). However, it is worth noticing that these two are the only ones with more than one publication, and that they only have two publications each. Among the included publications, 59% (13 papers) are published at conferences, 32% (7 papers) in journals, and the remaining 9% (2 papers) are published at workshops.

The identified empirical studies on quality requirements have been published between 1996 and 2008, which is visualized in Figure 1.3. In the year of 2001, most empirical studies of quality requirements were published (6

papers). The first published paper on empirical evaluation of quality requirements was P4. Figure 1.3 shows that publications that empirically evaluate quality requirements are increasing. However, one surprising finding is that not a single paper was published in the years of 2000, 2002, 2003, and 2006.

From Figure 1.3, it can be seen that about two–thirds (15 papers) of the included papers used case studies to empirically evaluate quality requirements, which makes it the most common research method. Seven of these papers used a single case study, while 8 papers used multiple case studies (between 2 and 4 cases).

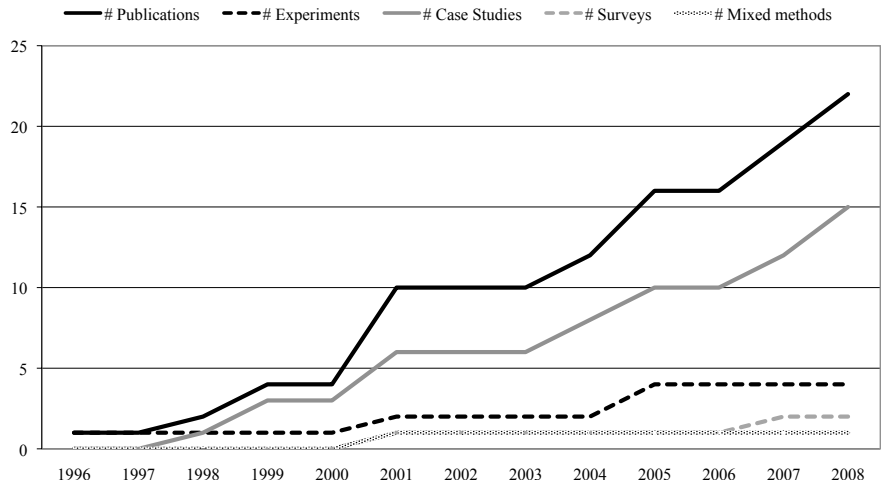

Figure 1.3: Cumulative number of publications, experiments, case studies, surveys, and mixed methods

Almost 60% (13 papers) used professionals when evaluating quality requirements, which makes professionals the most common type of sample (see Table 1.6). Five publications used systems or projects as their sample, while only 2 studies used documents.

As described in Section 3.5, an assessment of each of the primary studies according to 10 quality criteria was conducted. The results of the quality assessment checklist are shown in Table 1.3 (Section 3.5). Since only empirical studies were included in this systematic review, all publications received a ”yes” response on Q1. All studies had a more or less detailed description of the studied context in which the research was conducted. For two primary studies, the research design (Q4 in Table 1.3) was not described and therefore they received a ”no” response. Furthermore, in two studies, the data collection procedures (Q5 in Table 1.3) were not described, while five primary studies did not describe their analysis procedure (Q6 in Table 1.3). Surprisingly, only four of the 22 primary studies addressed va-

Table 1.6: Distribution of sample description

Sample description	Number of publications	Percentage
Professionals	13	59%
Systems/projects	5	23%
Documents	2	9%
Students	1	5%
Tools	1	5%

lidity issues (Q9 in Table 1.3), where two studies received a ”yes” response and two received ”partly”.

4.2 Elicitation

Table 1.7 provides an overview of six primary studies in relation to elicitation of quality requirements. The six papers evaluated six different elicitation techniques in one or more case studies. The type of systems where each elicitation technique has been evaluated is displayed in Table 1.7.

Cysneiros and Leite (P6) present a strategy to elicit quality requirements (ET1). The strategy is based on the use of a lexicon that will anchor functional and non–functional models, and drive quality requirements elicitation. The strategy comprises of four major activities. First, a lexicon based on the Language Extended Lexicon is built. Second, the functional model is built. Third, build the non–functional perspective, which adds the quality requirements to the created lexicon. Fourth, the functional and non–functional perspectives are integrated. The elicited quality requirements are then integrated into UML (Unified Modeling Language). The strategy was evaluated in three case studies, using three different systems, which is illustrated in Table 1.7. In all three case studies, ET1 elicited new quality requirements that generated between 20–25 percent of new classes for the existing systems. In addition, 46 percent of the existing classes (for all three systems) were in someway changed to satisfy the quality requirements. Cysneiros and Leite (P6) argue that quality requirements should not be dealt within the scope of functional requirements because quality requirements require detailed reasoning. Moreover, quality requirements have many interdependencies among them that may require trade offs among different design decisions.

On the other hand, Doerr et al. (P7) argue that elicitation of quality and functional requirements, and architecture must be intertwined. One reason is that the refinement of quality requirements is not possible without detail functional requirements and architecture. Therefore, Doerr et al., developed the NFR (non–functional requirements) method (ET2) and evaluated it in three case studies, using three different systems (see Table 1.7). The

Table 1.7: Primary studies of different elicitation techniques

Elicitation Technique ID	Used technique	Type of system(s)	Elicited quality requirement(s)	Publication ID
ET1	A strategy to elicit QR	(1) Light control system, (2) Students software project course, (3) Software for clinical analysis laboratories	(1, 2, 3) General QR elicitation	P6
ET2	An NFR approach	(1) Wireless control system, (2) Multi–functional printer system, (3) Geographical information system	(1) Efficiency, reliability, maintainability (2) Efficiency (3) Security	P7
ET3	SHIRA	Home automation system	General QR elicitation	P8
ET4	A strategy for elicitation	Factory–process–controlling–system	Performance, maintainability, fault tolerance	P12
ET5	Questionnaire–based strategy	(1) Outdoor payment terminal, (2) Omega fuel station management system	(1, 2) General QR elicitation	P14, 15
ET6	Multi–party chain strategy	(1) Outdoor payment terminal, (2) Omega fuel station management system	(1, 2) General QR elicitation	P14, 15

aim of the method is to achieve a set of measurable and traceable quality requirements. The main feature of the model include, a process for common treatment of high level quality requirements, quality models that capture experiences with quality requirements, detailed elicitation guidelines in terms of checklists and prioritization questionnaire, documentation guidelines, rationales to justify quality requirements, and requirement management support in terms of dependency analysis. In the wireless control system, the NFR method was successfully implemented, especially the prioritization questionnaire. The authors argue that new and important quality requirements were discovered with the NFR method. Moreover, only five of the 54 elicited quality requirements were not measurable. In addition, the NFR method found many new quality requirements that were missed before, which was found when comparing the new quality requirements with the original requirements specification. In the multi–functional printer system, the NFR method was found promising for elicitation of quality requirements. In this case study, the need for requirements management support was emphasized, which the NFR methods provide. In the geographical information system, quality requirements that had not been considered before, despite that they were of high importance for the project, were identified by the NFR method.

The Structured Hierarchical Interview for Requirement Analysis (ET3) was evaluated by using a home automation system by Hassenzahl (P8), which is illustrated in Table 1.7. Hassenzahl argues that an integrated approach, which gathers different aspects such as quality requirements, design approach, and the relationships among them, is important for ensuring a basic understanding of the design problem. The Structured Hierarchical Interview for Requirement Analysis (SHIRA) is an interviewing technique that consists of the following parts. First, the general idea of the system is introduced to the interviewee. Second, the interviewee chooses desired abstract quality requirements. Third, the interviewee is requested to list concrete qualities to the abstract quality requirement. Finally, the interviewee presents ideas of how to design the concrete qualities. The SHIRA method was evaluated by 18 interview session, where 172 concrete qualities and 474 design approaches were elicited. To further evaluate the usefulness of SHIRA, five independent experts assessed the results from the interviews. The experts found the results to be very useful; however, lack of information that defines usability requirement was criticized. The authors argue that SHIRA is a promising tool, but further assessments are needed.

JaeJoon et al. (P12) proposed a strategy to elicit quality requirements (ET4) for a factory–process–controlling system, which can be seen in Table 1.7. The proposed strategy has the following basic steps: (1) select quality requirements, (2) make consensus on the quality requirements, (3) develop scenario elicitation forms, (4) decide metrics for each quality requirements (e.g. performance is measured by the amount of data processed per time

unit), and (5) prioritize the quality requirements. The strategy (ET4) helps in identifying which components of the architecture model are related to which quality requirement. The authors compared to maintenance cost between the existing and the new architecture model (the existing used an old method to elicit quality requirements, while the new architecture model used the proposed strategy). The result shows that the maintenance cost was reduced by 10–50 percent in the new architecture model.

In Kusters et al. (P14, P15), two strategies for elicitation of quality requirements are presented and evaluated. The strategies are a questionnaire-based strategy (ET5) and the multi–party chain strategy (ET6). For the questionnaire–based (QB) strategy, a supporting tool and a questionnaire are developed to enable the elicitation of quality requirements. The procedure is to interview (one or more) persons that are connected to the development effort. The interview's aim is to characterize the environment where the system is supposed to run. Then, defined rules are applied to the defined environment, which will result in a quality profile. The multi–party chain (MPC) strategy requires three elements, a *party* (a person, group of persons, or an organization), a *role* (an area where the party is responsible), and *quality requirements* (identified by one or more parties). The process of applying MPC is as follow, first, all relevant parties, roles, and their relationships are identified. Second, a series of structured interviews are conducted. The goal of the MPC is to provide an effective and efficient communication of the systems quality requirements.

Both ET5 and ET6 were applied together in two different systems (Table 1.7). That is, first ET5 was used; the elicitation results were used as input to ET6. In the Outdoor payment terminal case, the result shows that an additional system test checklist was created in relation to reliability (one type of quality requirement). In the second case, the omega fuel station management system, the authors found that a substantial amount of improvements in relation to maturity and suitability (types of quality requirements) were undertaken, for example, additional effort in configuration management and testing. However, in both case studies, the authors argue that the quality of the results were insufficient, changes to the results from ET5 were needed. The authors argue that the combination of both ET5 and ET6 seems to work fine. Their disadvantages tend to cancel one another out.

4.3 Dependencies

In this subsection empirical evidence in quality requirements dependencies is covered. In Table 1.8, an overview of the identified dependency techniques/tools is provided. In addition to the techniques, one primary study (P22) investigates the existence of interdependencies among quality requirements.

Zulzalil et al. (P22) investigated interdependencies among four quality characteristics, namely functionality, usability, reliability, and efficiency for

Table 1.8: Comparison of dependency techniques/tools

Dependency Technique ID	Used technique/tool	Type of system(s)	Type of study	Publication ID
DT1	QARCC	Satellite ground stations	Experiment	P4
DT2	WinWin mode	Satellite ground stations	Experiment	P4
DT2	WinWin mode	15 digital library systems (students projects)	Case study	P9, P10
DT3	QARCC and S–COST	15 digital library systems (students projects)	Case study	P9, P10
DT4	Goal–Centric Traceability	Ice–Breaker system	Experiment	P5
DT5	NFD	Criminal investigation system, Dutch road–pricing system	Case study	P18

Table 1.9: Strong dependencies between quality requirements

Academic domain	**E–commerce domain**	**Museum domain**
Usability and Efficiency	Usability and Functionality	Usability and Reliability
Functionality and Efficiency	Usability and Reliability	Usability and Efficiency
	Functionality and Reliability	Functionality and Efficiency

web applications. Three different web applications domains were used, academic websites (six websites are used in the evaluation), e–commerce websites (five websites are used), and museum websites (four websites are involved in the evaluation). Zulzalil et al. discovered eight strong positive correlations among quality requirements, which are illustrated in Table 1.9. Moreover, the authors concluded that there are interactions among quality requirements and they are dependent on each other, either a positive or a negative dependency.

Techniques and tools for identifying interdependencies among quality requirements were found in five primary studies. Boehm and In (P4) compared the manual approach of the WinWin system (DT2) with the semi–automatic approach of the Quality Attribute Risk and Conflict Consultant (QARCC) tool for identifying quality requirements conflicts. The WinWin model is a general framework for identifying and resolving requirement conflicts. The WinWin model identifies conflicts by negotiating artifacts, such as win conditions, issues, and options. QARCC (DT1) is a knowledge–based tool for identifying possible conflicts among quality requirements. QARCC uses the same artifact as the WinWin model to identify conflicts. In an experiment, Boehm and In (P4) applied the QARCC tool in a system for satellite ground stations. The results show that the QARCC tool found both the significant identified conflicts by the WinWin model. Moreover, QARCC also identified eight more possible conflicts, where five of these were considered significant for the satellite ground station system. The authors argue that QARCC can help users, developers, and customers to identify conflicts among quality requirements.

In and Boehm (P9) describe the QARCC tool as a good tool for providing top–level suggestions about quality requirement conflicts; however, QARCC is lacking details. In an attempt to add more details to QARCC, the Software Cost Option Strategy Tool (S–COST) was developed to complement QARCC. S–COST uses the COCOMO cost drivers, cost estimates, and related experience to provide details of quality requirement conflicts involving cost. In et al. (P10) and In and Boehm (P10) evaluated the combi-

nation of QARCC and S–COST (DT3) in comparison to the WinWin model. A case study involving 15 digital library systems developed by graduate students was used for the comparison of identifying conflicting requirements. The result shows that DT3 identified 79% of all issues identified by the manual approach of the WinWin model, while DT3 could not identify 15% of issues found by WinWin. In addition, DT3 found 742% more issues than discovered by WinWin. Moreover, DT3 generated 97% of the same conflict resolutions options that was generated by the manual approach of the WinWin model (3% were not identified by DT3). Furthermore, DT3 surfaced 166% more solution strategies than identified by the WinWin model. The authors conclude that DT3 identifies more issues and options than the manual use of WinWin. However, some issues and options identified by DT3 were not significant. The authors argue when DT3 is trained to the systems nature, the non–significant issues and options will greatly reduce.

Cleland–Huang et al. (P5) propose the goal–centric traceability (GCT) technique to trace quality requirements. The GCT (DT4) uses a softgoal interdependency graph (SIG) to model quality requirements as goals, and helps developers to model quality requirements during software development. The GCT technique consists of four phases; the first one is goal modeling, which occurs during elicitation and specification of the software product. The second phase is impact detection, which is about understanding the impact of a change, and helping developers to evaluate the change. Third, in the goal analysis phase a re–analysis of the change, and an evaluation of its effect on the software products other goals are conducted. Finally, in the decision–making phase stakeholders review the impact to decide to proceed or not. In addition, stakeholders review the impact the change may have on other quality requirement goals. Cleland–Huang et al. (P5) empirically evaluated the GCT technique in an experiment of a system that consists of 180 functional requirements and nine quality requirements. The results show that GCT helps developers to manage the impact of functional change upon quality requirements. Moreover, the experiment revealed that GCT is feasible to dynamically retrieve traceability links for quality requirements.

Poort and de With (P18) developed the Non–Functional Decomposition (NFD) method for decomposing a system, based on the system's conflicts among requirements with an emphasis on quality requirements. The NFD (DT5) method comprises of the following steps. First are requirements gathered and prioritized, where any elicitation or prioritization technique can be used. It is important to show how the primary functional requirements are mapped to the supplementary requirements. Second, functions are grouped based on supplementary requirements. Third, conflicts among the supplementary requirements are identified. Then, function groups that deal with in–group conflicts are split. Finally, when in–group conflicts have been sorted out, the resulting grouping conflicts are the basis for the architectural decomposition. Poort and de With evaluated the NFD method in

two case studies. The results reveal that NFD turned out to be successful in terms of communicating to stakeholders how the design decisions are related to the supplementary requirements. Furthermore, resolving in–group conflicts showed the application of the method and principles of the NFD method.

4.4 Metrics

The metrics category has a focus on how metrics are used in relation to quality requirements, how many are quantified, and what scales are used. In addition, which methods are used to quantify, and to create verifiable quality requirements are investigated. In (P17), Olsson et al. empirically analyzed a requirement specification from the mobile phone domain. The aim was to analyze the quality requirements and how quality requirement metrics are used in practice. They found that 40% of the requirements are quality requirements. The authors found that 37% of the quality requirements use references to various standards. Moreover, a mixture of different scales was used to quantify the quality requirements, where absolute values (58% of all quantified quality requirements) were the most frequently used scale. That is, no interval is given, but an absolute number. Other used scales include creating a min–max interval, or creating a one–sided interval (either with an upper or a lower bound). Olsson et al. found, even within the same software product, that the treatment of quality requirements varies between different development areas. Therefore, the authors argues that different areas, both with respect to technical domain and type of quality requirements are unique in their character and require unique treatment in terms of tool support and method guidance. Furthermore, Olsson et al. states that the nature of specified intervals and scales for different areas are important for negotiation and prioritization. That is, the intervals and scales need to be aligned with the market and cost value.

The alignment between intervals and market values has been studied in P20 and P3. Both Regnell et al. (P20) and Berntsson Svensson et al. (P3) evaluated the application of the quality performance (QUPER) model in the mobile phone domain. Regnell et al. developed a guideline of how to apply the model in practice, with a focus on aligning intervals and market value. The guideline focus on identifying for which market segment the product should be released, what are the competitors' current level of quality, and what our product's current quality is. When the needed information has been gathered, a template for documenting the results, and the actual interval for a certain quality requirement was developed. The interval has three targets, low, mid, and high, where low means the minimum expected quality level needed, while high indicates that quality over this level is not needed. Berntsson Svensson et al. (P3) evaluated the practical guideline from Regnell et al. (P20) by interviewing four experts that used the QUPER model for three months. The experts viewed the rela-

tion between the intervals and market value as an important feature of the model. By adjusting the quality requirement interval based on the market segment and the competitors' quality provides a better basis for the quality requirement's actual metrics. Moreover, understanding the company's position on the market and more knowledge of the market situation were two comments about the practical application of QUPER.

Another method for quantifying quality requirements was studied by Jacobs (P11). Jacobs introduced a method called the Gild style in the telecommunication domain. The Gild style method is an adaption of the Planguage (Gilb 2005) method. To make quality requirements measurable, Jacobs used several concepts form the Planguage, such as scale (the unit in which the requirement should be measured) and meter (how the measurement will be performed). All used concepts were introduced, and visible in the requirements specification. The author argues that the whole concept of implementing the Gilb style was successful. Unclear requirement is no longer an issue for project using the new style. However, lack of good requirements is not just a technical problem, but also a behavioral problem. Jacobs reports that the culture changed within the company by implementing the Gilb style. In addition, writing requirements with the Gilb style put a focus on quality requirements, a common understanding of quality requirements is crucial according to Jacobs. By specifying the concept meter, Jacobs found that test cases were already defined during the requirements engineering phase.

The ISO/IEC 9126 (9126-2001 E) is an international standard for quality requirements. The aim of the standard is to ensure the quality of all software products. Al–Kilidar et al. (P1) evaluated the standard in terms of its ability to quantify and measure the quality attributes of a software design. Al–Kilidar et al. used 158 students in an experiment for the evaluation. The results show that the "common language" proposed by ISO/IEC9126 did not have a standard interpretation. The subjects had difficulties to interpret the quality characteristics in the standard. The authors argue that ISO/IEC 9126, in its present form, does not achieve any of its objectives.

4.5 Cost Estimation

Only one paper (P19) was related to cost estimations of quality requirements. Regnell at al. (P19) evaluated if it is possible to foresee when a major investment (in terms of cost) is necessary in order to improve the level of quality. The study found that all of the six interviewed experts in the mobile handset domain could identify these cost barriers. The cost of achieving a certain quality level is related to software optimization, hardware investment, development effort, investments in new architecture, and license fees. In addition, cost estimations is more uncertain in early stages of a new technology compared to technology that has been available and reached a certain market maturity. Another aspect that needs to be con-

sidered is the rapidly changes over time, which makes the expected cost barriers change over time, making trade–off analysis even more difficult.

It is worth noting that only one paper is related to quality requirements and cost estimations, particular since there are many reviews of cost estimations in the literature (Kitchenham et al. 2009). However, Regnell et al. (P19) do not address how cost estimations of quality requirements are conducted in industry; instead, P19 indentifies foreseen cost barriers representing the nonlinear nature of the relation between quality and cost. For further elaboration on cost estimation of quality requirements, see Section 5.1.

4.6 Prioritization

Three studies (P13, P16, and P21) address prioritization of quality requirements. Two studies (P13 and P16) address how stakeholders prioritize the importance of different quality requirements, while P21 created a process framework that can be used to prioritize the importance of quality requirements. All three studies use the ISO/IEC 9126 standard (9126-2001 E) as their reference of quality requirements.

Johansson et al. (P13) present the result of which quality requirements are considered the most expensive to obtain, and the most wanted ones in software platform development. A questionnaire was sent to the participants and non–directive interviews were used to capture unexpected information. Johansson et al. concluded that different stakeholders prioritize quality requirements differently, despite that the company's goals are the same for all stakeholders. In one of the studied companies, marketing considered reusability as the cheapest quality requirement to achieve; however, system designers and architects do not share this belief. Both the studied companies identified reliability as one of the most important quality requirements in a software platform. Johansson et al. advise companies that develops architecture of software platforms to consider three goals: (1) reliability is identified as the most important quality requirement, (2) a software platform should not be developed as part of another project, and (3) techniques that allows for precise communication and eliciting quality requirements from stakeholders are needed.

Leung (P16) studied which are the key quality requirements for intranet applications. A survey was used to determine the users' view of the importance of quality requirements. The survey was sent to various users such as end–users, developers, and information system professionals and a total of 30 responses were used in the study. The participants ranked the six quality characteristics from the ISO/IEC 9126 standard (9126-2001 E) and the three most important ones for intranet applications are: (1) reliability, (2) functionality, and (3) efficiency. In addition, the participants were also asked to rank the importance of the 32 sub–characteristics from the ISO/IEC 9126 standard. The five most important sub–characteristics

are: (1) availability, (2) accuracy, (3) security, (4) suitability, and (5) time behavior, which all belong to the three key quality characteristics.

Sibisi and van Waveren (P21) developed process framework for customizing software quality models. The proposed framework comprises four steps. In step 1, a generic quality profile questionnaire is created, which must be constructed to achieve its goal quality (users need). Step 2 is about building a specific quality profile. The quality profile presents the importance of each quality requirement. In step 3, a target quality profile is built, which is a modified specific quality profile (created in step 2) to fit the organizations own business goals. Finally, customization of a quality model is the last step. In this step, the target quality profile (from step 3) is used to prioritize quality requirements and thereby customizing a software quality measurement model, and decisions if quality requirements shall be removed or ignored are taken. The proposed framework was evaluated by two independent projects. The evaluation shows that the framework is valid for the six quality characteristics from the ISO/IEC 9126 standard (9126-2001 E), while improvements are needed at the sub–characteristic level.

4.7 Software Product Management

Four studies (P2, P3, P19, and P20) investigated quality requirements in relation to software product management activities. The focus of this category is on scoping, release planning, and roadmapping. Two main topics were investigated: identification of issues and problems in relation to scoping and variability management (P2), and the introduction and evaluation of the quality performance (QUPER) model (P3, P19, and P20).

In a study by Andersson and Bosch (P2), issues in scoping and variability management in software product families are analyzed. The results are based on case studies, using interviews and document studies, conducted at four Swedish companies that are involved in software product families. Andersson and Bosch (P2) identified five major issues, first, *evolving quality requirements*, which was a key problem in two of the case companies. The problem arises from how to expand a quality platform and maintain support levels for current platform users. Second, *service level for quality requirements support*, is also caused by the evolution and identified by two case companies. The major problem is to make applications that work on different levels to coexist on the same platform without causing behavioral problems. Third, *dynamic quality footprint* was identified by all four case companies. Problems related to dynamic quality footprint are, how to manage quality requirements as variation points, how to express quality requirements and their interdependencies, and the lack of descriptions of how to design evolvable architecture. Fourth, *quality support mismatch* was also identified by all companies as an important problem. This problem is a result of insufficient variability management and scoping, and integra-

tion problems. *Flexibility and extendibility of quality support* is the fifth and final identified problem. The flexibility and extendibility of quality support problem was regarded as an extremely important development for future quality platforms. One part of the problem is difficulties to modeling and designing an "open scope" platform. In addition, another part of the problem is the crosscutting nature of quality requirements and complex interdependencies among them.

The Quality performance model has been created, implemented and evaluated at a case company (P19, P20, and P3). Regnell et al. (P19) developed the quality performance (QUPER) model that incorporates quality as a dimension in addition to the cost and value dimensions used in prioritization approaches for functional requirements. The QUPER model aims to support decisions with regards to roadmapping, release planning, and platform scoping. The QUPER model consists of three views, the benefit view, the cost view, and the roadmap view. The benefit view includes three breakpoints indicating principal changes in the benefit level with respect to user experience and market value. The cost view includes foreseen cost barriers representing the nonlinear nature of the relation between quality and cost. The roadmap view combines the two previous views by positioning the information on the same scale. How to use the model in practice was studied by Regnell et al. (P20). The development of practical application was conducted in close collaboration with a case company. Applying QUPER in practice involves six steps, (1) define quality indicators, (2) for each indicator, estimate the breakpoints and barriers, (3) estimate the competitors and your own products quality, (4) estimate targets for coming releases, (5) approve and communicate the roadmaps, and (6) revise the roadmaps. However, only the first four steps were introduced at the company. In addition, a template for documenting the results of the model's first four steps was developed. Berntsson Svensson et al. (P3) evaluated the practical application of the model; however, the evaluation only includes the benefit view. The evaluation shows that the main benefit of the breakpoints was an understanding of when to stop to improve the quality level. Moreover, with regards to decision making, and especially in release planning, all interviewees confirmed that the richer understanding of the market with the identified breakpoints, the competitors and their own products quality level the more accurate the decisions are.

5 Discussion

This systematic review's overall goal is to collect and compare existing empirical evidence on quality requirements to date, and in addition, to provide state of research to serve as a basis for further empirical research on quality requirements. This review is the first systematic review on quality requirements. The following subsections address the benefits and limita-

tions of method for quality requirements and strength of evidence of the primary studies.

5.1 Benefits and limitations

The 22 identified primary studies are classified into six categories in relation to software product management activities. The review showed that more empirical studies on quality requirements in relation to the elicitation and dependency categories are conducted.

The studies that address elicitation of quality requirements do not provide a unified view of current practice, instead, a broad picture of experience and tested techniques is offered. Six different elicitation techniques for quality requirements are identified in this systematic review. Each technique is evaluated on one, two, or three different systems. All techniques were found to be promising for gathering quality requirements. Davis et al. (2006) found that interviews are the most effective elicitation technique for gathering requirements. In this systematic review, three of six techniques use interviews to elicit quality requirements. Moreover, Davis et al. (2006) found that interviews are the most commonly used elicitation technique, which is only partly confirmed in this study. With respect to limitations, Doerr et al. (P7) argue that elicitation of quality and functional requirements need to be intertwined with architecture. On the other hand, Cysneiros and Leite (P6) argue that quality and functional requirements should not be dealt within the same scope. The reason is that quality requirements require detailed reasoning. However, van Lamsweerde (2000) identified the need for bridging the gap between requirements and architecture, which supports the idea from Doerr et al. (P7).

With respect to handle dependencies of quality requirements, five different techniques are identified. No unified view of current practice is provided. In three studies, Boehm and In (P4); and In and Boehm (P9 and P10) compared the WinWin model with the, QARCC tool; and the QARCC tool in combination with the S-COST tool. All three studies showed that the WinWin model was less effective in identifying conflicts between quality requirements. However, no comparison of the QARCC tool and the combination of the QARCC and S-COST tools are conducted. The need for tool support in requirements engineering is support by van van Lamsweerde (2000). No other comparison of dependency techniques is conducted. With respect to limitations, the WinWin model is the only technique that has been evaluated in more than one study. Therefore, it is not possible to identify which technique/tool is most suitable to use for identification of interdependencies among quality requirements.

With respect to prioritization of quality requirements, only three studies are identified. However, it is worth noting that none of the identified studies address which technique or method that can be beneficial to use for requirements prioritization. This indicates the importance of understand-

ing how quality requirements are prioritized, which may be a gap in the literature.

Only one paper related to quality requirements and cost estimations is identified by this systematic review. One possible explanation is that our search string did not identify the existing studies. There is a risk that our search string did not identify relevant papers due to that a number of synonyms of estimations are used in the literature, which has been discussed by Jørgensen and Shepperd (2007). Jørgensen and Shepperd (2007) concluded that a wider search, including more synonyms, results in a too large set of studies to be meaningful. Another explanation for the lack of identified studies on cost estimations of quality requirements can be that cost estimations of quality requirements and functional requirements are conducted in a similar way. This review found that no study has been conducted on how to estimate the cost of quality requirements. To the best of our knowledge, based on our systematic review and the previous eight systematic reviews on cost estimations (Kitchenham et al. 2009), we believe that there is a lack of evidence of how cost estimations of quality requirements are conducted. One may argue that cost estimation, regardless if it is for functional or quality requirements, the same process can be used. However, studies ((Jr. 1987), (Cysneiros and Leite 1999)) have showed that quality requirements are expensive and difficult to handle, and according to Chung et al. (2000), quality requirements are often poorly understood. This indicates the importance of understanding how the procedure of cost estimation of quality requirements are conducted, thus there may be a gap in the literature.

5.2 Strength of evidence

Similar to (Dybå and Dingsyr 2008), we base the discussion of the strength of evidence on the GRADE (Grading Recommendations Assessment, Development and Evaluation) (Atkins et al. 2004) definitions of the overall strength of evidence. The GRADE system's overall strength of evidence is defined as high, moderate, low, or very low. According to GRADE, the overall strength of evidence is based on four categories, study design, study quality, consistency, and directness. In study design, experiments are considered as a high grade, while observational studies are considered as low. In this systematic review, only four experiments were identified, while the remaining studies are observational. Therefore, the total evidence of the combined studies, in relation to the category of study design in the GRADE system, is considered low.

With respect to the quality of the included studies methods were sometimes not well described; issues of bias, validity, and appropriate data analysis procedures were not always addressed (see Section 3.5). As many as 15 out of the 22 primary studies did not fully have/describe a sufficient data analysis procedure for the purpose of the study. In addition, only 2

out of the 22 primary studies did address validity issues in a systematic way, and 2 of the 22 primary studies mentioned validity threats. Based on these finding, it is concluded that there are limitations to the quality of the studies.

The consistency category is concerned with the similarity of estimates and effect across studies. Since most of the studies did not address threats to validity, in general we did not find direct evidence from the studies with no major validity threats. With respect to directness, i.e., to the extent which the interventions and outcome measures are similar, it was found that very few studies provided comparisons of interventions. Therefore, we believe that there are major uncertainties about the directness of the included primary studies. When combining the four categories from the GRADE system, our conclusion is that the strength of evidence is low.

6 Conclusion

This systematic review identified 2647 studies by searching the literature, of which 22 were found to be empirical research studies of acceptable quality, and related to the research questions. The studies are categorized into six areas in relation to quality requirements: elicitation, dependency, metrics, cost estimation, prioritization, and software product management.

A number of benefits and limitations of methods for quality requirements within each of these areas are identified. All identified methods for quality requirements are found to be promising. With respect to limitations, no unified view of current practices can be provided. The strength of evidence is not very high, which makes it difficult to offer specific advice to practitioners. Few studies are replicated, and thus the possibility to draw conclusions based on variations is limited. In order for practitioners to make use of the results, the context of where a method/technique has been applied must be considered and compared to the actual environment into which the method/technique is supposed to be applied.

A clear finding of this systematic review is the need to increase the number, and the quality of studies on quality requirements. In particular, prioritization of quality requirements warrants further attention. Not a single study looked into techniques of prioritization of quality requirements; instead, the studies looked into which quality requirements are considered most important. In addition, only one paper is related to cost estimations of quality requirements. However, the study does not address how cost estimations of quality requirements are conducted, nor does it identify what methods/techniques are appropriate for estimating the cost of quality requirements.

Future work for the research community is to: (1) empirically evaluate quality requirements in various requirements engineering and software product management activities to fill the identified gaps in literature; (2)

encourage systematic replications of studies in different context, and scaling up to more complex environments; (3) define how empirical evaluations of quality requirements should be reported, and what defines a case study.

Acknowledgements

This work was partly funded by VINNOVA (the Swedish Agency for Innovation Systems) within the MARS project. We especially thank Per Runeson for his valuable advice on the systematic review methodology.

References

ISO/IEC 9126-2001(E). Software engineering – product quality – part 1: Quality model, 2001.

H. Al-Kilidar, K. Cox, and B. Kitchenham. The use and usefulness of the iso/iec 9126 quality standard. In *Proceedings International Symposium on Empirical Software Engineering*, pages 122–128, 2005.

J. Andersson and J. Bosch. Development and use of dynamic product–line architectures. *IEE Software*, 152(1):15–28, 2005.

D. Atkins, D. Best, P.A. Briss, M. Eccles, Y. Falck-Ytter, S. Flottorp, G.H. Guyatt, R.T. Harbour, M.C. Haugh, D. Henry, S. Hill, R. Jaeschke, G. Leng, A. Liberati, N. Magrini, J. Mason, P. Middleton, J. Mrukowicz, D. O'connell, A. D Oxman, B. Phillips, H.J. Schunemann, T.T.T. Edejer, H. Varonen, G.E. Vist, J.W. Williams Jr., and Z. Stephanie. Grading quality of evidence and strength of recommendations. *BMJ*, 328(1490), 2004.

T.H. Al Balushi, P.R.F. Sampaio, D. Dabhi, and P. Loucopoulos. Elicito: A quality ontology–guided nfr elicitation tool. In *Lecture Notes in Computer Science*, volume 404542, pages 306–319, 2007.

R. Berntsson Svensson, T. Olsson, and B. Regnell. Introducing support for release planning of quality requirements – an industrial evaluation of the quper model. In *2nd International Workshop on Software Product Management*, 2008.

B. Boehm and H. In. Identifying quality–requirement conflicts. *IEEE Software*, 13(2):25–35, 1996.

K.K. Breitman, J.C.S.P. Leite, and A. Finkelstein. The world's stage: A survey on requirements engineering using a real–life case study. *Journal of the Brazilian Computer Scociety*, 6:13–38, 1999.

P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Proceedings 5th IEEE International Symposium on Requirements Engineering*, pages 84–91, 2000.

L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *NFR in Software Engineering*. Kluwer Academic Publishers, 2000.

J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina. Goal–centric traceability for managing non–functional requirements. In *Proceedings 27th International Conference on Software Engineering*, pages 362–371, 2005.

L.M. Cysneiros and J.C.S.P. Leite. Integrating non–functional requirements into data model. In *Proceedings 4th IEEE International Symposium on Requirements Engineering*, pages 162–171, 1999.

L.M. Cysneiros and J.C.S.P. Leite. Nonfunctional requirements: From elicitation to conceptual models. *IEEE Transactions on Software Engineering*, 30(5):328–349, 2004.

A. Davis, O. Dieste, A. Hickey, N. Juristo, and A.M. Moreno. Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review. In *Proceedings 14th IEEE International Requirements Engineering Conference*, pages 176–185, 2006.

J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki. Non–functional requirements in industry – three case studies adopting an experience–based nfr method. In *Proceedings 13th IEEE International Conference on Requirements Engineering*, pages 373–382, 2005.

T. Dybà and T. Dingsyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9–10):833–859, 2008.

C. Ebert. Requirements before the requirements: Understanding the upstream impact. In *Proceedings 13th IEEE International Conference on Requirements Engineering*, pages 117–124, 2005.

E. Engström, M. Skoglund, and P. Runeson. Empirical evaluations of regression test selection techniques: a systematic review. In *Proceedings of the Second ACM–IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 22–31, 2008.

A. Finkelstein and J. Dowell. A comedy of errors: The london ambulance service case study. In *Proceedings 8th International Workshop on Software Specification and Design*, pages 2–4, 1996.

T. Gilb. *Competitive Engineering*. Elsevier Butterworth–Heinemann, 2005.

S. Grimstad, M. Jørgensen, and K. Molokken-Ostvold. Software effort estimation terminology: the tower of babel. *Information and Software Technology*, 48(4):302–310, 2006.

M. Hassenzahl, R. Wessler, and K.C. Hamborg. Exploring and understanding product qualities that users desire. In *Proceedings of the 15th Annual Conference of the Human–Computer Interaction Group of the British Computer Society*, pages 95–96, 2001.

H. In and B.W. Boehm. Using winwin quality requirements management tools: a case study. *Annals of Software Engineering*, 11:141–174, 2001.

H. In, B.W. Boehm, and M. Deutsch. Applying winwin to quality requirements: a case study. In *Proceedings of the 23rd International Conference on Software Engineering*, pages 555–564, 2001.

S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings 4th IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

L. JaeJoon, H. Sucheol, K.C. Kang, C. Youngyeol, H. Yoonpyo, and H. Hwawon. Quality requirement elicitation for the architecture evaluation of process computer systems. In *Proceedings of the Eighth Asia–Pacific Software Engineering Conference*, pages 335–340, 2001.

E. Johansson, A. Wesslen, L. Bratthall, and M. Höst. The importance of quality requirements in software platform development–a survey. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, 2001.

M. Jørgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70(1–2):37–60, 2004.

M. Jørgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, 31(1):33–53, 2007.

F.P. Brooks Jr. No silver bullet: Essences and accidents of software engineering. *Computer*, 4:10–19, 1987.

H-W. Jung, S-G. Kim, and C-S. Chung. Measuring software product quality: A survey of iso/iec 9126. *IEEE Software*, 21(5):88–92, 2004.

J. Karlsson and K. Ryan. A cost–value approach for prioritizing requirements. *IEEE Software*, 14(5):67–74, 1997.

J. Karlsson, C. Wohlin, and B. Regnell. An evaluation of methods for prioritising software requirements. *Information and Software Technology*, 39 (14–15):939–947, 1998.

B. Kitchenham, E. Mendes, and G.H. Travassos. A systematic review of cross– vs. within–company cost estimation studies. *IEEE Transactions on Software Engineering*, 33(5):316–329, 2007.

B.A. Kitchenham. Guidelines for performing systematic literature reviews in software engineering version 2.3. Technical report, Keele University and University of Durham, 2007.

B.A. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering – a systematic literature review. *Information and Software Technology*, 51(1): 7–15, 2009.

G. Kotonya and I. Sommerville. *Requirements Engineering*. John Wiley & Sons, 1998.

R.J. Kusters, R.V. Solingen, and J.J.M. Trienekens. Identifying embedded software quality: Two approaches. *Quality and Reliability Engineering International*, 15:485–492, 1999a.

R.J. Kusters, R.V. Solingen, and J.J.M. Trienekens. Strategies for the identification and specification of embedded software quality. In *Proceedings of the Ninth International Workshop Software Technology and Engineering Practice*, pages 33–39, 1999b.

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

H.K.N. Leung. Quality metrics for intranet applications. *Information and Management*, 38(3):137–152, 2001.

C. Mair and M. Shepperd. The consistency of empirical comparisons of regression and analogy–based software project cost prediction. In *Proceedings of the International Symposium on Empirical Software Engineering*, pages 509–518, 2005.

T. Olsson, R. Berntsson Svensson, and B. Regnell. Non–functional requirements metrics in practice – an empirical document analysis. In *Workshop on Measuring Requirements for Project and Product Success*, 2007.

S.L. Pfleeger. *Software Engineering – Theory and practice*. Prentice–Hall, 2001.

E.R. Poort and P.H.N. de With. Resolving requirement conflicts through non–functional decomposition. In *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture*, pages 145–154, 2004.

B. Regnell, M. Höst, and R. Berntsson Svensson. A quality performance model for cost–benefit analysis of non–functional requirement applied to the mobile handset domain. In *Lecture Notes in Computer Science*, volume 4542, pages 277–291, 2007.

B. Regnell, R. Berntsson Svensson, and T. Olsson. Supporting roadmapping of quality requirements. *IEEE Software*, 25(2):42–47, 2008.

P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

M. Sibisi and C.C. van Waveren. A process framework for customising software quality models. In *Proceedings of IEEE AFRICON Conference*, pages 547–554, 2007.

M.I. Ullah and G. Ruhe. Towards comprehensive release planning for software product lines. In *Proceedings of the 1st International Workshop on Software Product Management*, pages 51–55, 2006.

I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma. On the creation of a reference framework for software product management: Validation and tool support. In *Proceedings of the 1st International Workshop on Software Product Management*, pages 3–11, 2006a.

I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma. Towards a reference framework for software product management. In *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pages 312–315, 2006b.

A. van Lamsweerde. Requirements engineering in the year 00: A research perspective. In *Proceedings of the 2000 International Conference on Software Engineering*, pages 5–19, 2000.

C. Wohlin, P. Runeson, M. Höst, M.C. Ohlson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An introduction*. Kluwer Academic, 2000.

H. Zulzalil, Z.M. Zain, A. Ghani, M.H. Selamat, and R. Mahmod. Relationships analysis between quality factors for web applications. In *Proceedings of the International Symposium on Information Technology*, 2008.

Paper II

Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems

Richard Berntsson Svensson[1], Tony Gorschek[2], Björn Regnell[1]
[1]Lund University, Department of Computer Science, Box 118, 221 00 Lund, Sweden
{Richard.Berntsson_Svensson,Bjorn.Regnell}@cs.lth.se
[2]Blekinge Institute of Technology, School of Engineering, PO Box 520, 372 25 Ronneby, Sweden
tony.gorschek@bth.se

Accepted for publication at the 15th International Working conference on Requirements Engineering: Foundation for Software Quality (REFSQ09), June 2009, Amsterdam, The Netherlands

ABSTRACT

[Context and motivation] In market–driven software development it is crucial, but challenging, to find the right balance among competing quality requirements (QR). **[Problem]** In order to identify the unique challenges associated with the selection, trade–off, and management of quality requirements an interview study is performed. **[Results]** This paper describes how QR are handled in practice. Data is collected through interviews with five product managers and five project leaders from five software companies. **[Contribution]** The contribution of this study is threefold: Firstly, it includes an examination of the interdependencies among quality requirements perceived as most important by the practitioners. Secondly, it compares the perceptions and priorities of quality requirements by product management and project management respectively. Thirdly, it characterizes the selection and management of quality requirements in down–stream development activities.

Keywords: Quality requirements; Non–functional requirements; Requirements engineering; Market–driven requirements engineering; Empirical study

1 Introduction

The complexity of software systems is determined by both functionality and by quality aspects such as performance, reliability, accuracy, security, and usability (Chung et al. 2000). These quality aspects, or non–functional requirements are subsequently called quality requirements (QR). It is commonly acknowledged that the handling and balance of QR are an important and difficult part of the requirements engineering (RE) process (Jacobs 1999), playing a critical role in software development (Chung et al. 2000). However, the situation is even more complex in a market–driven development situation (Aurum and Wohlin 2005). In market–driven development, the flow of requirements is not limited to one project, and the requirements are generated from internal (e.g., engineers) and external (e.g., customers) sources (Gorschek and Wohlin 2006). Also, to achieve high–quality in complex embedded systems, a combination of experience and knowledge from different disciplines is needed (Kusters et al. 1999). This may lead to communication difficulties and difficulties in achieving the required quality level (Kusters et al. 1999). QR often specify certain quality levels and QR are in many cases possible to quantify (Olsson et al. 2007). Quantification is important, not only for understanding QR (Jacobs 1999), but also for planning (Regnell et al. 2007). Not dealing, or ineffectively dealing with QR may lead to more expensive software and longer–time–to–market (Cysneiros and Leite 2004), or in worst case, failures in software development (Breitman et al. 1999), (Finkelstein and Dowell 1996). Studies (Jr. 1987), (Cysneiros and Leite 1999) have showed cases where QR are the most expensive and difficult aspects to handle, and according to Chung et al., QR are often poorly understood in comparison to less critical aspects of software development (Chung et al. 2000). To be able to improve how QR are handled it is important to understand their characteristics (Olsson et al. 2007), how they are used and prioritized in industry, as well as the challenges of dealing with QR. This paper presents an empirical study performed in industry to investigate these aspects as well as complement other RE surveys as few of them have focused on the specific challenges related to QR.

Two main perspectives on QR are studied in this paper (Gorschek and Davis 2008). First, the product perspective. Product managers are responsible for the overall product perspective and the selection of the overall planning of the product evolution and offering are elicited (for further elaboration see van de Weerd et al. (2006)). Second, the project perspective is studied through the project leader, responsible for managing and prioritizing within the realization phases. The two perspectives are also compared, studying the alignment between project and product managers. The purpose of this study is to discover and describe how QR are handled in practice, both from the product manager and the project leader's perspective, which is important since communication problems are a chal-

lenge in market–driven software development (Karlsson et al. 2007). In addition, the effects of not dealing, or ineffectively dealing with QR are also investigated. The paper presents the results of an empirical study that includes data collected from ten practitioners (five product managers and five project leaders) at five companies in Sweden.

The reminder of this paper is organized as follows. In section 2, the background and related work are presented. The research methodology is described in Section 3, while Section 4 presents the results and relates the findings to previous studies. Section 5 gives a summary of the main conclusions.

2 Background and Related Work

There are several surveys that concern or include RE related challenges. Curtis et al. (1988) reported the first significant field survey of practices. Even though the study does not have a focus on RE, challenges related to RE were identified, including communication breakdowns and conflicting requirements. Next, a study by Chatzoglou (1997) identifies problems with the RE process, the challenges presented are e.g., lack of resources and poor quality of tools and techniques in the RE process.

Lubars et al. (1993) published a field study on requirements modeling. The presented challenges include vaguely stated requirements and difficulties with prioritization of requirements. In addition, Lubars et al. (1993) identified challenges in relation to specification of performance requirements (a type of QR) such as the rationale is not always obvious and difficulties to associate performance requirements with parts of dataflow or control flow specifications. In addition, a field study by Kamsties et al. (1998) includes small and medium sized enterprises. The identified challenges include implementation of new requirements may cause unpredictable interaction with existing requirements, requirements are not traceable, and that requirements are too vague to test. Kamsties et al. (1998) also indentified a challenge related to specification of graphical user interfaces (usability requirements, a type of QR). Furthermore, Karlsson et al. (2007) published a study with solely focus on challenges in market–driven software development. The presented challenges include communication problems between marketing and development, and requirements prioritization. Karlsson et al. (2007) also indentified challenges in relation to QR. One challenge is related to QR interdependencies, which was identified as a major problem. Quality requirements can influence a large part of the functionality or other QR. This is not only related to finding the existing interdependencies, but also assessing to what extent that requirements affect each other, and determining how to deal with this. In addition, problems with considering quality requirements in release planning were identified.

Several studies (Carlshamre et al. 2000), (Chung et al. 2000), (Cleland-

Huang et al. 2005), have looked at requirements interdependencies; for example, Carlshamre et al. (2000) identified six different interdependency types in industry. Research related to classification and measurement of QR are also introduced in literature (Jacobs 1999), (Olsson et al. 2007). Olsson et al. (2007) conclude that for a method to be successful, it is important that it is flexible enough to handle the diverse nature of QR.

The focus of the above mentioned studies have not been primarily on QR, but QRrelated findings emerged as parts of the results. This paper presents a study with the primary focus on QR and how they are managed in the RE process.

3 Research Method

The study was carried out using a qualitative research approach (Robson 2002). Qualitative research aims to investigate and understand phenomena within its real life context. A qualitative research approach is useful when the purpose is to explore an area of interest, and when the aim is to improve the understanding of phenomena. The purpose of this study is to gain in–depth understanding of QR within market–driven embedded systems companies. The following research questions (see Table 2.1) provided a focus for our empirical investigation.

It is important to understand an organizations alignment in terms of QR, otherwise there may be a mismatch between product management (van de Weerd et al. 2006) and project leadership. Project leaders may down prioritize quality aspects that are considered important by product managers and vice verca. In addition, interdependencies are important to understand since QR may influence a large part of the system (Karlsson et al. 2007). Kamsties et al. (1998) found that requirements are often too vague to test, therefore, it is important to investigate if QR are quantified in industry. Also, dismissal of QR from projects may have an impact on the predicted return of investment, as well as the cost for the customers. Finally, QR are a difficult part of the RE process (Jacobs 1999), however; not all challenges in relation to QR may be of major concern for industry. Therefore, it is important to understand what challenges are critical and which ones are adequately handled today.

3.1 Research Design and Data Collection

The study uses semi–structured interviews enabling exploratory discussion between the researcher and the interviewee. The study was conducted in two stages: first the data from each company was collected and analyzed. Secondly, the combined data from all participating companies was collected and analyzed. The criteria for selecting companies were based on our corporate contacts within industry. Five market–driven software com-

Table 2.1: Research questions

Research Questions (QR = Quality Requirements)
RQ1: Is there any difference in the views of what quality requirements are the most important between product managers and project leaders?
RQ2: What interdependencies between QR are present in the companies?
RQ2.1: What types of interdependencies are deemed most important by practitioners, and is there any difference between the view of product managers and project leaders in this regard?
RQ2.2: To what extent are interdependencies elicited, analyzed and documented in industrial practice?
RQ3: Are QR specified in a measurable manner?
RQ4: To what extent are QR dismissed from projects after project initiation?
RQ4.1: If QR are dismissed, is any consequence analysis performed?
RQ5: What QR challenges are articulated as critical by the practitioners themselves?
RQ6: What QR aspects do the companies feel confident as being adequately handled today?

panies participate. From each company, one product manager (PM) and one project leader (PL) from the same project were interviewed, resulting in ten data points. The study consists of three phases: planning, data collection, and analysis.

Planning: The first phase of the study involved a brainstorming and planning meetings to design the study and to identify different areas of interests. A combination of maximum variation sampling and convenience sampling was used to select companies within our industrial collaboration network (Patton 2002). The included companies vary in respect to size, type of product, and application domain, a rudimentary characterization can be see in Table 2 (more details are not revealed for confidentiality reasons). The interview instrument was designed with respect to the different areas of interest and inspiration from (Karlsson et al. 2007). To test the interview instrument[1], two pilot interviews were conducted prior to the industry study.

Data collection: The study used a semi–structured interview strategy (Robson 2002). All interviews were attended by one interviewee and one interviewer. First, the purpose of the study and a general explanation of QR were presented and then questions about the different areas of interests in relation to QR were discussed in detail. All interviews varied between 40 and 90 minutes.

Analysis: The content analysis (Robson 2002) involved creating categories where interesting parts from the interviews were added and discussed. The first two authors examined the categories from different perspectives and searched for explicitly stated or concealed pros and cons in relation to how QR are handled in industry. The results from the analysis are found in Section 4.

3.2 Validity

In this section, threats to validity are discussed. We consider the four perspectives of validity and threats presented in Wohlin et al. (2000).

Construct validity: The construct validity is concerned with the relation between theories behind the research and the observations. The variables in our research are measured through interviews, including open–ended aspects where the participants are asked to express their own opinions. Mono–operation bias (Wohlin et al. 2000) was avoided by collecting data from a wide range of sources on the topic of the study. To avoid evaluation apprehension (Wohlin et al. 2000), complete anonymity from other participants, the companies, and researchers was guaranteed. Another validity threat lies in the question that asked interviewees to rank and include additional factors if the list provided to them was inadequate. Interviewees may have thought that it was easier to rank the provided factors than pro-

[1] http://serg.cs.lth.se/research/packages

Table 2.2: Company characteristics

	Aplha	**Beta**	**Gamma**	**Delta**	**Epsilon**
# of employees	˜100	˜3000	>5000	325	65
Domain	Control systems	Telecom	Telecom	Telecom	Control systems
Typical project cycle	18 months	48 months	24-36 months	Differs	9 months
# of reqs	>1000	˜7000	>20000	˜100 features	Differs
# of QR	˜10% QR	˜10% QR	QR unknown	˜10% QR	

pose new factors. This means that important interdependency types may be missing.

Conclusion validity: Threats to conclusion validity arise from the ability to draw accurate conclusions. The interviews were conducted at different companies and each interview was done in one work session. Thus, answers were not influenced by internal discussions. To obtain highly reliable measures and to avoid poor question wording and poor layout, several pilot studies were conducted.

Internal validity: This threat is related to issues that may affect the causal relationship between treatment and outcome. Threats to internal validity include instrumentation, maturation and selection threats. In our study, the research instrument was developed with close reference to literature relating to non–functional requirements, and influenced by a previously administrated and validated research instrument (Karlsson et al. 2007), which mitigates the instrumentation threat. In addition, maturation threats are handled by reducing the duration of interview sessions by collecting background information before the interview, and by keeping the interview session to 90 minutes.

External validity: This threat is concerned with the ability to generalize the findings beyond the actual study. Qualitative studies rarely attempt to generalize beyond the actual setting since it is more concerned with explaining and understanding the phenomena. However, understanding the phenomena may help in understanding other cases. The fact that most of the identified challenges are acknowledged by more than one company increases the possibility to generalize the results beyond this study. To avoid the interaction of selection and treatment, interviewees were selected according to their roles within the company, and companies were selected from different geographical locations.

4 Results and Analysis

This section presents the results discovered during the analysis of the interviews. The five following sub–sections present and discuss one research question each, corresponding to the research questions in Table 2.1.

4.1 Important Quality Aspects (RQ1)

In analyzing Research Question 1, this section examines the most important quality aspects, as illustrated in Figure 2.1. Based on Lauesen's comparison of ISO9126 and McCall quality factors (Lauesen 2002), we identified 23 different types of QR. We asked the interviewees to rank the top five most important aspects for their products based on their expertise and their own definition of the quality factor. (Our approach was not to impose preconceived definitions but to try to understand existing industrial

practice and practitioners' own interpretations of QR.) Looking at Figure 2.1, the quality aspect ranked first received five points, the one ranked second got four points and so on, and the one ranked fifth got one point. In total we see that interviewees agreed that usability (which got a total of 26 points) and performance (23 points) requirements are the two most important types of QR followed by compliance (13 points), flexibility (13 points), and stability (11 points).

One reason for the prioritization of usability, as explained by several interviewees, is that *"if the product is not usable we will not sell any products"*. One interviewee expanded the view by stating that it does not matter if you have the latest and coolest functionality, if the system is not easy to use, the customer will look at the competitors for an easy to use system. The reason why compliance was ranked as the third most important quality aspect is interesting. Several interviewees explained that compliance is important because *"we must be compliant with the requirement document"*. This interpretation of compliance differs from the one formulated by ISO9126 which states to adhere to standards, regulations and laws. This leads to a possible mismatch between the established academic interpretation of compliance and the industrial interpretation of it. Apart from the agreement that usability and performance were the two most important types of QR, PM and PL had different priorities. PMs ranked performance (14 points) as the most important quality aspect, followed by usability (12 points) and security/integrity (7points).

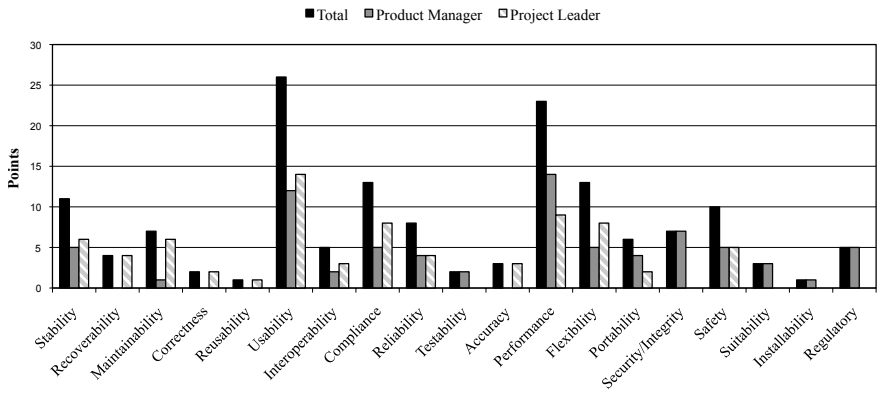

Figure 2.1: Importance of quality aspects

PLs ranked usability first (14 points), followed by performance (9 points), and compliance and flexibility (8 points). PMs uniquely identified security/integrity, testability, suitability, and installability as the most important quality aspects. On the other hand, PLs uniquely identified the following quality aspects: recoverability, reusability, correctness, and accuracy. The differences in priority between PM and PL may not be a surprise as some

mismatch can be expected. The two have different roles and perspectives, but it might nevertheless be an important insight. For example, not a single PL ranked security/integrity among the top five even if security/integrity was considered the third most important by the PMs. Worth observing is that the aspects of fault tolerance, conformance, replaceability, and analyzability was not in any of the PMs or PLs top five.

4.2 Interdependencies (RQ2)

Six different interdependency types are characterized (Carlshamre et al. 2000): (**1**) R1 *AND* R2: R1 requires R2 to function, and R2 requires R1 to function, (**2**) R1 *REQUIRES* R2: R1 requires R2 to function, but not vice versa, (**3**) R1 *TEMPORAL* R2: Either R1 has to be implemented before R2 or vice versa, (**4**) R1 *CVALUE* R2: R1 affects the value of R2 for a customer, (**5**) R1 *ICOST* R2: R1 affects the cost of implementing R2, and (**6**) R1 *OR* R2: Only one of R1, R2 needs to be implemented.

Although the interviewees had the option of adding new types of interdependencies, no new types were discovered during the interviews. All of the six presented interdependency types were used by the interviewees to characterize perceived interdependencies, both among different QR, and interdependencies among QR and functional requirements (FR), as illustrated in Table 2.3.

In general, the most common interdependency types identified among QR were: *OR*, *REQUIRES*, and *ICOST*, while the least frequent one identified was *TEMPORAL*. When the results from PM and PL were examined separately, the findings show a difference of opinion. PMs viewed *OR* and *ICOST* as the most common types, while PLs viewed *REQUIRES* as the most common one.

When examining the most frequent identified interdependency types among QR and FR, four of the six types were considered equally common, while the remaining two (*AND*, *OR*) types were considered least important. However, when examining the results from the PM and PL separately, the findings show an interesting difference. While PM considered *TEMPORAL* as the most common interdependency, PL viewed TEMPORAL as least frequent. On the other hand, PL identified OR as one of four (the other three are: *REQUIRES*, *ICOST*, and *CVALUE*) equally common interdependency types, but *OR* was viewed as least frequent by PM.

In the study by Carlshamre et al. (2000), three of five case companies viewed value related (*ICOST* or *CVALUE*) interdependency types as the most common. In the remaining two cases, functionality related (i.e., *AND* or *REQUIRES*) types were most common. Our results show a mix of value and functionality types as the most common ones (with the exception of Company Epsilon). The difference between the studies may be explained by the focus, i.e. we focused solely on interdependencies related to QR, while in Carlshamre et al. (2000) the focus was on requirements in general.

Table 2.3: Existing interdependency types divided by role

	Role	QR to QR	QR to FR
Alpha	PM	REQUIRES, **CVALUE**, ICOST	AND, REQUIRES, TEMPORAL, CVALUE, **ICOST**
	PL	NONE	NONE
Beta	PM	OR, AND, REQUIRES, TEMPORAL, CVALUE, **ICOST**	OR, AND, REQUIRES, TEMPORAL, CVALUE, **ICOST**
	PL	OR, **REQUIRES**, TEMPORAL, CVALUE, ICOST	OR, **REQUIRES**, TEMPORAL, CVALUE, ICOST
Gamma	PM	OR, AND, **REQUIRES**, CVALUE, ICOST	AND, **REQUIRES**, TEMPORAL, CVALUE, ICOST
	PL	OR, AND, REQUIRES, **CVALUE**, ICOST	AND, REQUIRES, **CVALUE**, ICOST
Delta	PM	OR, **ICOST**	OR, REQUIRES, TEMPORAL, CVALUE, **ICOST**
	PL	OR, **REQUIRES**, CVALUE, ICOST	OR, AND, **REQUIRES**, TEMPORAL, CVALUE, ICOST
Epsilon	PM	**OR**	**TEMPORAL**
	PL	**REQUIRES**	**OR**

In Chung et al. (2000) and Cleland-Huang et al. (2005), a softgoal interdependency graph (SIG) is used to show interdependencies among QR. The interdependency types used in the SIG are limited to *AND*, and *OR*, which is not inline with the findings in our study, as we found that six different interdependency types were present in the companies. Furthermore, the two types *AND*, and *OR* were only identified as present by 25% of the interviewees.

RQ2.1: What types of interdependencies are deemed most important by practitioners, and is there any difference between the view of product managers and project leaders in this regard? According to the interviewees in total, the most important interdependency type to identify among QR was REQUIRES, however, the PM and PL roles were not in agreement. PM considered *ICOST* as the most important, while *REQUIRES* was prioritized by the PL. Interestingly, in identifying the most important interdependency type among QR and FR, the total result was identical to interdependency types among QR. On closer examination the result between PM and PL vary in relation to interdependencies among QR and FR. PM pri-

oritized *ICOST*, but also uniquely identified *TEMPORAL* and *ICOST*. The PL prioritized *REQUIRES*, but also uniquely identified *OR* and *CVALUE*.

It is not surprising that PM and PL have different views on interdependency priority. According to Carlshamre et al. (2000), value related interdependencies are subjective; it may be difficult to state whether the cost exceeds the value for the customer, therefore, these types of decisions should be made by product committees. This is inline with the results in this study; PM considers *ICOST* as the most important type, while PL *REQUIRES*. One PL explained that *REQUIRES* is the most important interdependency type because *"functionality first, then the quality aspect of the functionality is relevant"*. Surprisingly, both among QR, and among QR and FR, *REQUIRES* is considered the most important to identify looking at the summation of all interviewees. This result is not inline with Carlshamre et al. (2000), which found that *ICOST* and *CVALUE* were the most important types of interdependencies in marketdriven developing companies, while *REQUIRES* was considered the most important in bespoke developing companies. One PM explained that *REQUIRES* is considered the most important interdependency to identify because *"this is the easiest type to miss, and therefore the most important to identify"*.

RQ2.2: To what extent are interdependencies elicited, analyzed and documented in industrial practice? The results show that in three of the five companies (Gamma, Delta, and Epsilon) both PM and PL confirmed that no elicitation, analysis, or documentation of interdependencies involving QR was conducted at all. In Company Alpha, the PM stated that all dependency activities were conducted, while the PL from the same company indicated that none of them were performed. In only one company (Beta) both PM and PL stated that activities to elicit, analyze and document interdependencies was performed. This result is inline with results from Karlsson et al. (2007), which found that interdependencies between requirements in market–driven software development are a major problem. The problem includes identification, how the requirements affect each other, and how to deal with them. The results are relevant since interdependencies among QR's are at the hart of managing explicit trade–offs among solution alternatives (Cysneiros and Leite 2004). In addition, Cleland-Huang et al. (2005) states that failing to trace QR expose a company to huge risks when a change is introduced. Furthermore,Kamsties et al. (1998) found that new requirements may cause unpredictable interaction with existing requirements, which indicates the importance of finding the interdependencies among requirements.

There can be several potential explanations of why interdependencies among QR are not actively looked for. Quality requirements tend to have a global impact on the entire system, therefore, QR are difficult to trace and because of the extensive network of interdependencies and trade–offs that exists among them responsibilities for their realization is often vague

Table 2.4: Quantification of quality requirements

Role	Alpha	Beta	Gamma	Delta	Epsilon
PM	Always	Never	Always	Sometimes	Always
PL	Sometimes	Sometimes	Always	Sometimes	Sometimes

((Chung et al. 2000), (Cysneiros and Leite 2004)). Other explanations were discovered during the interviews. Some interviewees stated that they have little focus on QR, while others stated that QR are assumed and therefore interdependencies are not actively looked for. In addition, one interviewee confirmed that their focus is on functional requirements and not QR. Others stated that dependencies are handled during other parts of the development process, for example, during the design, architecture, and implementation. However, they have more focus on functional requirements because functional requirements are easier to discover than QR.

One possible implication with this is that quality aspects such as usability and performance are not considered at the early stages of product and project planning. This can be an acceptable alternative, given that the companies consider quality aspects important only in the solution domain, and not from a product offering or business perspective. This is however contradicted by the results obtained during the prioritization of quality aspects (see Section 4.1), where the practitioners stated that several (or which usability was premiered) quality aspects were crucial for being able to sell the product at all.

4.3 Quantification of Quality Requirements (RQ3)

In analyzing research question 3, this section examines how often QR are specified in a measurable manner, as illustrated in Table 2.4.

Interestingly, four of the PLs claimed that QR were quantified *sometimes*, while in three of these cases the PMs view differed, stating *always* or *never*. In two out of five (Gamma and Beta) companies agreement between PM and PL could be observed. The disagreement may be an indication of communication problems between the PM and PL. Communication problems were also identified as a challenge in market–driven RE by several studies ((Fricker et al. 2008), (Fricker et al. 2007), (Karlsson et al. 2007)). In a study by Olsson et al. (2007), about half of the QR were found to be quantified which seems to confirm the findings. However, one interesting observation that can not be directly confirmed is the level of disagreement between PM and PL. It should be noted that each PM and PL pair worked for the same company, and moreover with the same project.

4.4 Dismissal of Quality Requirements (RQ4)

We asked the interviewees how often QR that were actually specified and selected for inclusion in a project were subsequently dismissed from project during development (see Table 2.5). The total average mean value of dismissed QR is 22.5%, meaning almost every fourth QR that has been included in a project is dismissed at some stage. When comparing PM and PL, the least (in the best situation) amount of dismissed QR is slightly higher for PM (5%) than for PL (3%). In worst case (Most in Table 5); the mean value of dismissed QR is 55% according to the PMs, while PLs believe that 45% are dismissed.

According to the interviewees, there are two trends of which types of QR that are more representative of the ones being dismissed. Firstly, QR that are not visible for the end customer, such as maintainability and testability are more often dismissed than other QR. Secondly, performance requirements are more often dismissed due to the difficulties of estimating them. One inherent contradiction can be seen in these two trends. For example, if the performance of a system is inadequate, the inadequacy of this quality aspect can be noticed by the customer through a slow system/product. No further elaborations were given on this contradiction.

The results reveal three main reasons for the dismissal of QR: (1) poor cost estimations, (2) lack of resources, and (3) that QR have lower priority than functional requirements (FR). Poor cost estimations is related to the difficulties to estimate the cost of QR that have a global impact on the system. The difficulties of estimating the cost of QR are related to lack of knowledge and understanding of how to manage QR in practice. Several interviewees frequently described that QR have lower priority, and that they do not spend much time on managing QR. Some of the interviewees explained that QR are seen as base requirements and therefore not considered. However, this focus has implications on the system, as explained by one PM, *"in most situations, QR are down prioritized by FR due to lack of knowledge of how important a system's quality is. By lowering the quality level, the value of the system decreases"*.

RQ4.1: If QR are dismissed, is any consequence analysis performed? According to the PMs, a consequence analysis is only conducted if the customers are affected. The consequence analysis may include new prioritization of all requirements and new cost estimations, as explained by one interviewee that *"if we have promised a certain quality, then we have to increase the cost for this project and accept a lower return of investment"*. Another consequence, as explained by a PM, is to *"first ask the customer if this is OK. If not, we talk to the developers to find out the reason why this cannot be done. Finally, we decide if we have to add or remove other requirements"*. Surprisingly, none of the PLs shared the view of the PMs. All PL claimed that nothing happens when QR were dismissed from the projects. One explanation, which was

Table 2.5: Dismiss rate of quality requirements

	Role	**Dismissal rate**			**Consequence Analysis**	**Reason for dismiss rate**
		Least	*Average*	*Most*		
Alpha	*PM*	10%	15%	20%	If customer is affected	Poor cost estimations
	PL	0%	50%	90%	No	Testing QR very late
Beta	*PM*	10%	20%	90%	If customer is affected	Lack of resources
	PL	1%	5%	20%	Yes	Lack of resources and poor cost estimations
Gamma	*PM*	NA	NA	NA	Check with stakeholders	Poor cost estimations and lack of resources
	PL	NA	NA	NA	No	Lack of resources and lower priority than FR
Delta	*PM*	0%	5%	10%	If customer is affected	Issues we cannot affect, e.g. network capacity
	PL	0%	10%	20%	No	Issues we cannot affect, e.g. network capacity
Epsilon	*PM*	0%	50%	100%	If customer is affected	Poor cost estimations and lower priority than FR
	PL	10%	25%	50%	No	Lower priority than FR

NA: Not available

qualified by one PL, is that *"we do not have time to re–analyze the consequence of QRs, other things are more important"*. Another explanation according to another PL is that *"we can deliver on time if QRs are dismissed"*.

A central issue here seems to be the difficulty to properly quantify as well as estimate the cost of implementing a QR, but more importantly the value of a QR. This might indicate a lack of estimation models/techniques for QR. The complexity is of course that a QR often implies a quality aspect of a system/product. Such a quality aspect is often not realized as a feature, but rather implies that all development be in line and adhering to the quality aspect. For example, performance is not dictated by one thing, but often by how the system is realized overall, including architectural considerations impacting the whole.

4.5 Quality Requirement Challenges (RQ5 and RQ6)

In analyzing research questions 5 and 6, this section examines what QR challenges and what QR aspects the practitioners identified. Figure 2.2 shows the two perspectives.

Three companies (Beta, Gamma, and Epsilon) stated that they are very good in terms of testing QR (QR that are well specified and quantified). This was confirmed by one interviewee: QR that are quantified are easy to test. Another interviewee explained that their company has a well established test organization and good methods for testing QR, both in lab and field environments. However, one of the identified challenges is difficulties in achieving testable QR, i.e. making QR well specified and quantified. This is not a surprising result and is confirmed in previous studies ((Kamsties et al. 1998), (Karlsson et al. 2007), (Lubars et al. 1993)). Apart from the agreement of testing QR, each company identified issues in relation to QR that are adequately handled today. One surprising finding was that one company (Delta) stated that they are good at rejection of QR. The product manager explained that *"we are very good in negotiation of QRs, which is to make sure that QRs are not part of the contract."*. The result reveals two major challenges that are faced by the companies, (1) how to get QR into the projects, and (2) when is the quality level good enough? All companies faced the same problem of getting QR into the project. The challenge is that QR have to contend with FR, where FR often emerge as victors. Problems with considering QR were also found by Karlsson et al. (2007).

A reason may be that having an extra function is considered more valuable than to improve the quality of the system. However, this focus may backfire as the customers may want a certain quality level of the systems that are bought. One interviewee confirmed that *"we have been very technology focused, we did not care about QRs, but now it has backfired and we have to put a lot of focus on the QRs."*. In addition, QR are considered as obvious, or even as base requirements and therefore not quantified or specified. The second main challenge is to decide when a certain quality level is good

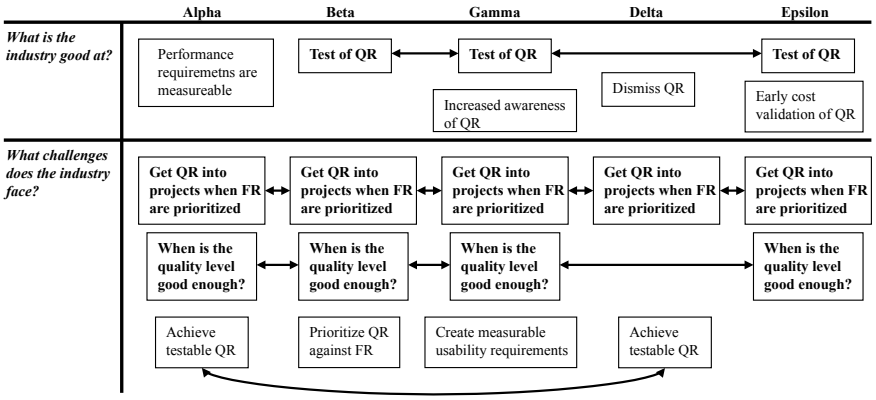

Figure 2.2: Challenges and non–challenges in the companies

enough, when are you finished with a QR? The interviewees expressed their concern of how to decide when the quality is good enough. Should the performance be two seconds, 1.5 seconds, or even one second, who can decide that? One interviewee said, who can decide if 1 or 5 Mbits are most appropriate?

Two companies (Alpha and Delta) identified achieving testable QR, and one company (Gamma) viewed creating measurable usability requirements as challenges. One reason for identifying these challenges may be related to the quantification of QR (Section 4.3), which shows that 60% of the interviewees stated that QR are *never*, or *sometimes* specified in a measurable manner. Another identified challenge (Company Beta) is prioritization of QR. Prioritization of QR involves other challenges than prioritization of FR, which were further explained by one interviewee by the statement that performance and usability requirements are different in nature and very difficult to compare. How do we prioritize performance requirements of two seconds against the subjective appraisal of usability requirements, was asked by one interviewee.

5 Conclusions

In conclusion, this article presents the results of an empirical study that examines Quality Requirements (QR) in practice in five software companies. Data is collected from five product managers and five project leaders at the companies. To the best of our knowledge, there are no other multi–case survey studies that examine QR in practice.

The findings reveal that usability and performance requirements are deemed the two most important types of QR by the interviewed practitioners. In addition, we found that the companies do not actively look for interde-

pendencies among QR, and we did not encounter QR–specific elicitation, documentation, or analysis (RQ2). The findings highlight three important challenges (RQ5): (1) how to get QR into the projects when functional requirements are prioritized, (2) how to know when the quality level is good enough, and (3) how to achieve testable QR. Our results indicate that QR are often not quantified (RQ3), thus difficult to test. However, the interviewees consider that the companies are good in terms of testing the (few) QR that are quantified (RQ6).

There seems to be a bespoke development mindset where the immediate project gets a higher priority than the long–term evolution of the product (which would be interesting for further studies). This is confirmed by the implicit management of QR, and the dismissal off–hand of QR with little or no consequence analysis (RQ4). This contradicts the interviewees' initial view (RQ1) where quality aspects were labeled as critical, but looking at practice, the project–oriented perspective and the urge of offering more functionality in the immediate release dominates.

The interviewees expressed that the limited focus on QR can have long–term consequences as well; increased maintenance costs and degradation in usability with feature growth are but a few examples. However, the main problem is that QR are not taken into consideration during product planning (pre–project) and thus not included as hard requirements in the projects. This implies that no explicit trade–off can be made, making the realization of QR a reactive rather than proactive effort. Product management may thus not be able to plan and rely on quality aspects to achieve competitive advantages, but mainly respond to emerging QR problems.

References

A. Aurum and C. Wohlin. *Engineering and Managing Software Requirements*. Springer, 2005.

K.K. Breitman, J.C.S.P. Leite, and A. Finkelstein. The world's stage: A survey on requirements engineering using a real-life case study. *Journal of the Brazilian Computer Scociety*, 6:13–38, 1999.

P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Proceedings 5th IEEE International Symposium on Requirements Engineering*, pages 84–91, 2000.

P.D. Chatzoglou. Factors affecting completion of the requirements capture stage of projects with different characteristics. *Information and Software Technology*, 39:627–640, 1997.

L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *NFR in Software Engineering*. Kluwer Academic Publishers, 2000.

J. Cleland-Huang, R. Settimi, and O. BenKhadra. Goal-centric traceability for managing non-functional requirements. In *Proceedings 27th International Conference on Software Engineering*, pages 362–371, 2005.

B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31:1268–1287, 1988.

L.M. Cysneiros and J.C.S.P. Leite. Integrating non-functional requirements into data model. In *Proceedings 4th IEEE International Symposium on Requirements Engineering*, pages 162–171, 1999.

L.M. Cysneiros and J.C.S.P. Leite. Nonfunctional requirements: From elicitation to conceptual models. *IEEE Transactions on Software Engineering*, 30:328–349, 2004.

A. Finkelstein and J. Dowell. Comedy of errors: The london ambulance service case study. In *Proceedings 8th International Workshop on Software Specification and Design*, pages 2–4, 1996.

S. Fricker, T. Gorschek, and P. Myllyperkö. Handshaking between software projects and stakeholders using implementation proposals. In *Lecture Notes in Computer Science*, volume 4542, pages 144–159, 2007.

S. Fricker, T. Gorschek, and M. Glintz. Goal–oriented requirements communication in new product development. In *2nd International Workshop on Software Product Management*, 2008.

T. Gorschek and A. Davis. Requirements engineering: In search of the dependent variables. *Information and Software Technology*, 50:67–75, 2008.

T. Gorschek and C. Wohlin. Requirements abstraction model. *Requirements Engineering Journal*, 11:79–101, 2006.

S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings 4th IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

F.P. Brooks Jr. No silver bullet: Essences and accidents of software engineering. *Computer*, 4:10–19, 1987.

E. Kamsties, K. Hörnmann, and M. Schlich. Requirements engineering in small and medium enterprises. In *International Proceedings Conference on European Industrial Requirements Engineering*, pages 84–90, 1998.

L. Karlsson, Å.G. Dahlstedt, B. Regnell, J. Natt och Dag, and A. Persson. Requirements engineering challenges in market-driven software development - an interview study with practitioners. *Information and Software Technology*, 49:588–604, 2007.

R.J. Kusters, R.V. Solingen, and J.J.M. Trienekens. Identifying embedded software quality: Two approaches. *Quality and Reliability Engineering International*, 15:485–492, 1999.

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

M. Lubars, C. Potts, and C. Richter. A review of the state of the practice in requirements modelling. In *Proceedings 1st IEEE International Symposium on Requirements Engineering*, pages 2–14, 1993.

T. Olsson, R. Berntsson Svensson, and B. Regnell. Non-functional requirements metrics in practice - an empirical document analysis. In *Workshop on Measuring Requirements for Project and Product Success*, 2007.

M.Q. Patton. *Qualitative Research and Evaluation Methods*. Sage Publications, 2002.

B. Regnell, M. Höst, and R. Berntsson Svensson. A quality performance model for cost-benefit analysis of non-functional requirement applied to the mobile handset domain. In *Lecture Notes in Computer Science*, volume 4542, pages 277–291, 2007.

C. Robson. *Real World Research*. Blackwell, 2002.

I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma. Towards a reference framework for software product management. In *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pages 312–315, 2006.

C. Wohlin, P. Runeson, M. Höst, M.C. Ohlson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An introduction*. Kluwer Academic, 2000.

Paper III

Non–functional requirements metrics in practice – an empirical document analysis

Thomas Olsson[1], Richard Berntsson Svensson[2], Björn Regnell[1,2]
[1]Sony Ericsson Mobile Communication, Lund, Sweden
{thomas.olsson,bjorn.regnell}@sonyericsson.com
[2]Dept. of Computer Science, Lund University, Sweden
{bjorn.regnell,rbsv}@cs.lth.se

Published at the Workshop on Measuring Requirements for Project and Product Success (MeReP07), November 2007, Palma de Mallorca, Spain

ABSTRACT

Non–functional requirements (NFR) complement functional requirements with quality aspects and are a central part of software engineering. This paper presents an empirical case study that characterizes the specification of NFR. The study is performed on a large set comprising 2113 requirements on subcontracted technology platforms in the mobile phone domain. The analysis is qualitative using an emerging coding scheme for investigating of frequency and patterns in NFR specifications. It can be observed that as many as 40% of the requirements are NFR. Standards are commonly referenced in NFR and directly quantified NFR as well as NFR without a concrete metric are common. Results on distribution of NFR in different technical areas are discussed. Two hypotheses are identified for further investigations: (1) different areas of NFR are special in their character and require unique treatment, and (2) if interval and scale patterns are aligned with market value breakpoints and cost barriers then prioritization and scoping can be made more effective.

1 Introduction

It is commonly acknowledged that non–functional requirements (NFR) are an important and difficult part of requirements engineering (Doerr et al. 2005) (Jacobs 1999). Non–functional requirements (also known as quality requirements) complement functional requirements with quality aspects (Chung et al. 2000). A characteristic of NFR is that they specify certain quality levels and can hence often be quantified. This is important not only for understanding the requirements (Jacobs 1999), but also for prioritization and planning (Regnell et al. 2007).

The aim of the study presented in this paper is to empirically analyze NFR specification in practice and to investigate how NFR metrics are used in an industrial context. The study is performed at Sony Ericsson, one of the leading developers of mobile phones. An in–depth analysis of a single case helps us to understand the details of a specific context, and enables comparison with other similar case studies. To our best knowledge, similar studies on NFR metrics do not yet exist, but future empirical studies on the important area of NFR may hopefully be conducted. Another outcome of this research is the classification approach as such. We demonstrate how a classification can be performed in practice, enabling companies to detail their knowledge of their own requirements.

The case study presented here is a document content analysis study (Robson 2002), focused on understanding how non–functional requirements are specified, in particular which metrics that are being used. A document analysis is an unobtrusive study of an artefact (Robson 2002). A specification consisting of 2113 requirements is carefully analyzed in a bottom–up manner. The classification scheme is built up as the research progresses. The study is qualitative in the sense that we do not want to confirm any theory, but rather seek to identify patterns and generate hypotheses for further research.

The research questions that guide the presented work are:

1. What are the characteristics of different types of requirements?
2. How are NFR specified?
3. How are NFR quantified?

The paper is structured as follows: Section 2 provides an account of the case study analysis method with its goals, questions and metrics as well as the coding scheme that is applied in the qualitative data analysis. Section 2 also includes the data analysis with descriptive statistics of distribution of coding categories. Section 3 discusses the implications of the data analysis and summarises lessons learned and hypotheses generated. Section 4 gives an account of related work relevant to NFR specification. Finally, Section 5 concludes the paper.

2 Case study analysis

2.1 Research methodology

This case study is an open–ended and exploratory document analysis (Robson 2002). The focus is on understanding how non–functional requirements are specified, in particular which metrics that are being used. A document analysis is an unobtrusive study of an artefact. Analysing the content is a quantified codification of the artefact (Robson 2002).

The coding scheme and relevant aspects to code are derived as the study progresses. To avoid moulding the characterization to a particular standard or classification, the case study does not use a pre–defined characterization scheme. Instead, the questions and metrics are openly defined, see Table 3.1.

To collect the metrics, a requirements document is analyzed. A content analysis (Robson 2002) is performed to codify and quantify the specification. The requirements are analyzed and coded with respect to the different metrics that is collected. The document analysis is performed in the following steps:

1. **A preliminary coding was performed to categorize aspects of interest to be coded in more detail.** An overall coding of the entire set of requirements was performed. The goal is to have a first categorization of the requirements into classes of functional and non–functional and to explore which aspects to code further. In sub–sequent steps, the effort is focused on the non–functional requirements.

2. **The emerging codes are discussed and consolidated.** The overall coding was revised and consolidated. The revision consisted of attaining orthogonal categories and agreeing on the meaning of the categories. The consolidation also consisted of raising the level of confidence in the coding. The subjectively perceived coding confidence varied from ”very low” to ”very high” in five levels. It was agreed that the confidence should be at least judged ”high” to be considered acceptable. The coding was performed by all three researchers and discussed until an agreement was reached, with the intention to increase the coding accuracy.

3. **Detailed coding, initial iteration.** After having identified which categories to code, a more detailed coding of e.g. domain, scales and class followed. In the first iteration, the goal is o get an initial understanding of the requirements. The emerging codes and groups were analysed to derive a consistent and reliable codification. Not all quality requirements were coded, as the purpose was mainly to derive a suitable and consistent coding. The analysis was performed using card sorting (Nurmuliani et al. 2004), (Upchurch et al. 2001). The card sorting was performed as a group activity among the authors.

Table 3.1: Goals, questions and metrics (Al-Kilidar et al. 2005)

Goal	Question	Metric
1. What are the characteristics of different types of requirements?	1. Which types of requirements are present in the requirements document?	1. Type of requirements (functional, non–functional) 2. Distribution across different areas of the specification
	2. How many are there of the different types of requirements?	3. Type of requirements 4. Number of requirements of certain types
	3. Are there areas with more or less NFR than others?	5. Type of requirements 6. Distribution across different areas of the specification
2. How are NFR specified?	4. Which metrics are being used?	7. Direct or indirect quantification 8. Type of metric 9. Distribution of metric usage
	5. How is the usage of standards in the NFR?	10. Name of standard 11. Direct or indirect quantification
3. How are NFR quantified?	6. On which kind of scales are metrics defined?	12. Types of intervals 13. Type of scales

4. **Final detailed coding.** The purpose of the final detailed coding iteration is to code all NFR in the specification. This coding provides the final result presented here.

The coding was performed in parallel by all three authors. To ensure consistency and reliable results, there was an overlap in the coding among authors and the consolidating steps 2 and 3.

The process of coding and consolidating ran over several months. The effort spent on the case study is in the range of 3–4 person weeks. The first author spent more than half of that effort. The remaining was equally distributed between the two other authors.

2.2 Description of the case

The case study is performed at Sony Ericsson. Sony Ericsson operates in the mobile phone industry, developing several millions phone per years for a wide range of markets and customers. The individual products are developed on a common platform with a product line engineering approach. Hence, also the NFR are specified mainly for the platform and not the individual products. Some parts of the platform are developed by different sub–contractors, some by Sony Ericsson itself.

Mobile phones are embedded real–time system consumer products developed for a mass–market. The platform requirements that are investigated in this study are characterized by a number of aspects relevant to NFR, such as metrics and scale but also usage of standard references. By having a better understanding of the company practices regarding NFR, we support our long–term goal to develop effective support which works in a practical context.

This case study investigates a requirement specification given to a sub contractor of Sony Ericsson. This subcontractor provides mobile platform technology for integration into mobile products. The specification contains areas such as radio, multimedia and network. There are both hardware and software requirements in the specification. The areas range from being pure hardware related to being pure software related on a sliding scale. The specification is focused on enabling technologies, rather than end–user requirements.

In total, the requirements document contains 2113 requirements. There is a mixture of functional and non–functional requirements. The specification is written in natural language English. The document is structured into different areas (headings) and sub areas. The structure and depth of sub areas differ among the areas. A requirement typically consists of 1–5 sentences. The specification is reused over time. New requirements are added and obsolete requirements are deleted. The sub–contractor uses the specification as the basis for a statement of compliance in the negotiation process with Sony Ericsson. The specification has been used over a longer period of times for several generations of platforms. Hence, the requirements have been reviewed and used extensively and are of an appropriate quality.

2.3 Coding scheme

The scheme is used to codify the requirements. The scheme consists of three main requirements types

- Pure Functional requirements (PF) – used for the common understanding of functional requirements.
- Pure Quality requirements (PQ) – these requirements are NFR that

do not add functionality, but specifies a quality level on functional requirements or puts requirements on the sub–contractor as such.

- Both functional and quality aspects mixed (F&Q) – this category is used when NFR and functional aspects are combined and intermingled in the same requirement. This also includes cases where the requirement includes references to a (part of a) standard that contains both quality and functional aspects.

As the requirements of class F&Q also contain quality aspects, these requirements are also considered in the detailed analysis of the NFR classes. Therefore, the union of the requirements sets PQ and F&Q are given a general type "Q". The detailed coding is focused on the Q set, see the step description in Section .2.1.

In addition to the general types above, a number of other aspects are coded:

- Standard reference in Quality requirement (SQ) – used to indicate whether requirements of the class Q reference a standard or not. For example, "The MPEG coding standard shall be supported".

- Direct Quality level (DQ) – requirements that are quantified or use a metric. This might be a certain time requirement or sensitivity level, e.g. "640x480 (VGA) resolution shall be supported". It can also be the case that a specific level in a standard is referred, e.g. "GPRS class 1 according to the 3GPP standard shall be supported".

- Indirect Quality level (IQ) – even though it is a non–functional requirement, a metric might not be used. It can for example be a general standard reference or for example a maintainability requirement. The sum of all DQ and IQ requirements are all the Q requirements.

For the DQ requirements, the scale characteristics are also coded as follows:

- Lower/upper bound – a DQ requirement that specifies a one–sided interval, either an upper or a lower bound. For example, there are requirements on phone start–up time (upper bound) and data transfer rates (lower bound)

- Min–Max – a DQ requirement specifying a double–sided interval, i.e. both an upper and a lower bound. This is for example used for sensitivity and accuracy requirements.

- Absolute – a DQ requirement that specified an absolute value, i.e. no interval is used. Display resolution is one example of an absolute DQ.

- Discrete/Continuous – indicates whether the scale discrete or continuous. For certain requirements, e.g. memory, only some values are available. For others, e.g. response time, any value on the scale can be used.

2.4 Data analysis

The overall distribution of requirements can be found in Figure 3.1 and in Table 3.2. The data have for confidentiality reasons been made anonymous in terms of which areas the requirements cover. We use the main heading in the specification to separate the document into areas. These areas are used throughout the presentation. There are in total 28 areas, which correspond to heading level one in the requirements specification. As this heading structure has evolved over time, it is a mixture of different topics. Some of the topics are closely related to a specific domain, e.g. audio or telephony, some more general such as performance or architecture.

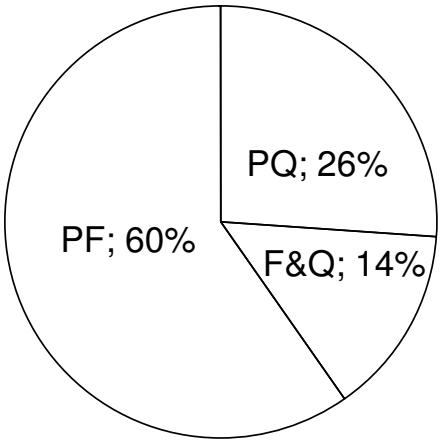

Figure 3.1: Total distribution of requirements types (functional requirements (PF), non–functional requirements (PQ) and both (F&Q))

As can be seen in Table 3.2 and Figure 3.1, functional requirements are the dominating type of requirements. PF requirements represent 60% of the entire requirements mass. 26% of the requirements are pure non–functional requirements (PQ) and the rest (14%) are both functional and non–functional (F&Q). Hence, Q requirements (PQ + F&Q) are 40% of the requirements.

Looking more closely into the different areas in the document, it can be seen that the distribution of the types differs across the document, see Figure 3.2. Interesting to note is that areas 1 and 2 in the figure have two requirements each and except for area 1, there is at least one Q requirement in the area (cf. Table 3.2). Note that the number of Q requirements for area

Table 3.2: Number of requirements per area

Area	PF	PQ	F&Q	Sum	Q	SQ	DQ
1	2	0	0	2	0	0	0
2	0	2	0	2	2	0	2
3	3	1	0	4	1	0	1
4	1	7	0	8	7	0	2
5	8	3	0	11	3	0	2
6	16	1	0	17	1	0	1
7	6	6	6	18	12	0	11
8	7	12	0	19	12	3	2
9	21	3	1	25	4	0	2
10	24	0	4	28	4	4	3
11	2	20	10	32	30	1	22
12	13	12	10	35	22	4	17
13	19	16	1	36	17	11	14
14	22	3	14	39	17	13	0
15	35	3	4	42	7	5	4
16	30	9	3	42	12	8	2
17	32	4	7	43	11	2	5
18	16	8	23	47	31	20	6
19	60	0	3	63	3	0	2
20	40	29	3	72	32	0	12
21	18	53	3	74	56	8	4
22	7	81	2	90	83	40	3
23	63	38	3	104	41	26	25
24	55	37	20	112	57	0	35
25	177	4	40	221	44	14	39
26	198	20	6	224	26	18	11
27	197	42	15	254	57	39	24
28	185	139	128	449	264	97	216
All	1257	553	303	2113	856	313	467

28 is off the chart in Figure 3.2, Figure 3.3 and Figure 3.5. It contains a total of 264 Q requirements. For presentation purposes, the scale is reduced.

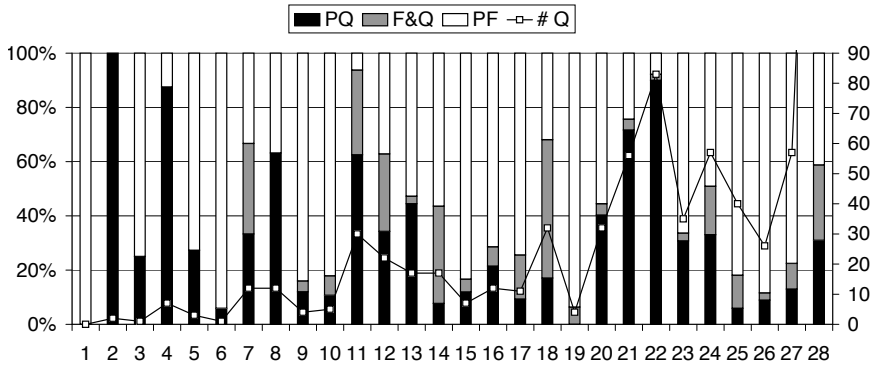

Figure 3.2: Distribution of types pf requirements per area (functional (PF), non–functional (PQ) and both (F&Q)) as well as number of NFR (# Q)

Overall, the mobile phone domain uses a lot of standard references in the requirements specification, see Figure 3.3. It might both be that there is a general fulfillment requirement, e.g. ”the platform must comply with the legal standard XYZ” or a direct reference to a level in the standard ”Level 2 of the standard ABC should be fulfilled”. The distribution shows that different areas utilized standards in a varying way. 11 areas have no reference to any standard in their NFR. However, 50% of the Q requirements have a standard reference in 9 areas.

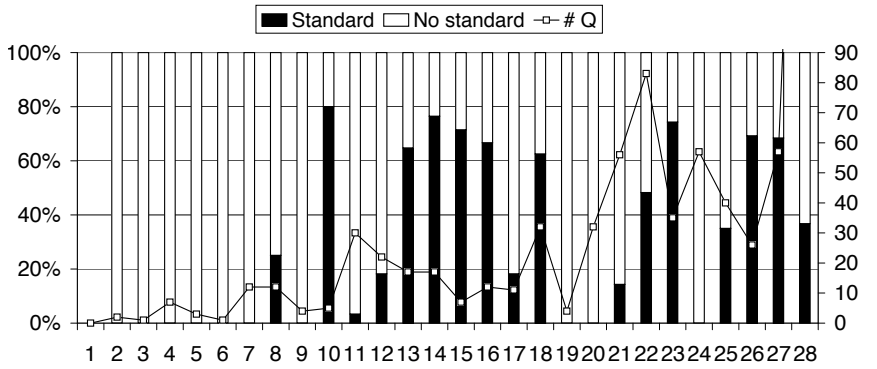

Figure 3.3: The proportion of Q requirements that reference a standard as well as the number of NFR (# Q)

A non–functional requirement might be specified using a direct quantification (DQ). For example, ”The platform should be able to decode MP3 of 128kb/s”. Alternatively, the quality level is indirect (IQ), e.g. ”The plat-

form should comply with the 3GPP standard". The distribution of DQ and IQ requirements can be seen in Figure 3.4.

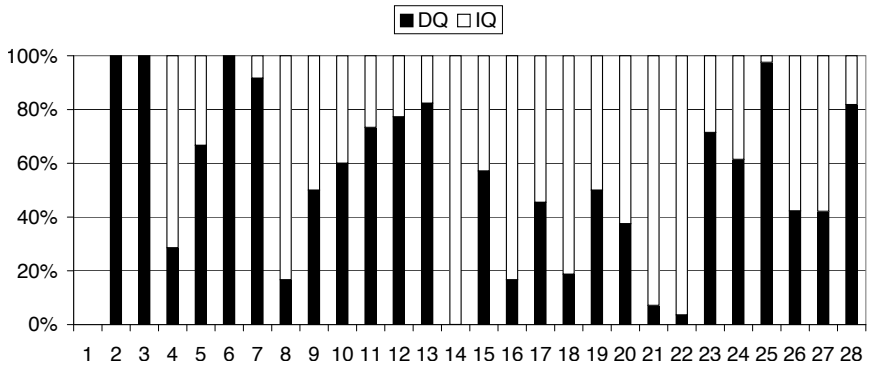

Figure 3.4: The relative number of Q requirements that have a direct quantification (DQ) or not (IQ)

As with the SQ requirements, it is clear that the different areas are not the same with respect to directly quantified requirements. Most areas have a mixture of IQ and DQ. When combining the standard and metric categories, 49% of the DQ requirements have a standard reference, while 31% of the IQ requirements have a standard reference. Figure 3.5 depicts how many percent of the requirements have a standard reference. The figure also shows how many Q requirements there are in each area.

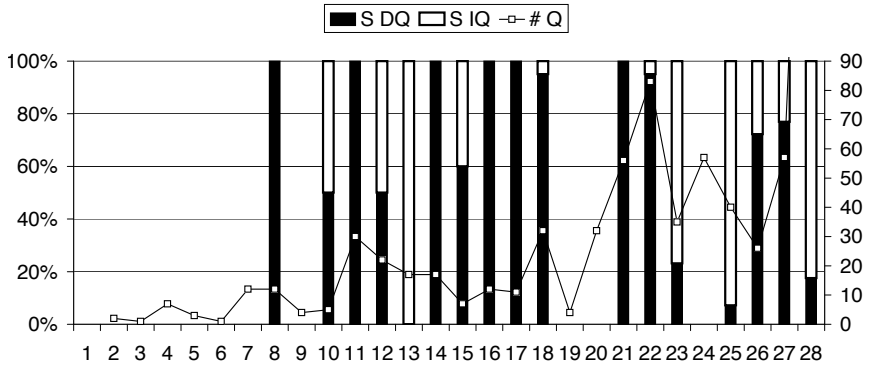

Figure 3.5: Distribution of standard references, divided on DQ and IQ

Domains of NFR were coded for all the requirements. This can be for example audio or telephony. The heading structure (areas) is related to domain, but the heading structure is mixed with different topics, sometimes cross–cutting. Hence, the domains need to be coded in a more consistent and coherent manner than the heading structure. As can be seen, the number of NFR varies across different domains, see Figure 3.6.

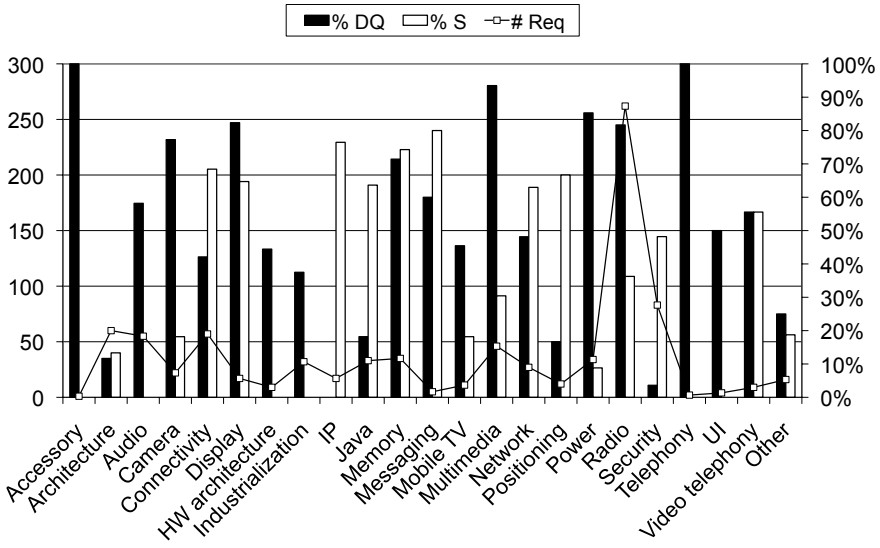

Figure 3.6: Domains of Q requirements

Finally, scales and types of NFR were coded for all DQ requirements. Looking more closely on the DQ requirements and the scales on which the quantification is specified, a mixture can be seen (see Figure 3.7):

- Absolute – no interval given, but an absolute number (58%)
- Min–max – a lower and an upper bound is specified, creating a min–max interval (7%)
- Upper bound – an upper bound is specified, creating a one–sided interval (24%)
- Lower bound – as with upper bound, a one–sided interval (12%)

In addition to the interval bounds, the scale might be either discrete or continuous. Memory is an example of a discrete scale. Memories typically come in sizes based on the power of 2, e.g. 64MB or 256MB, and combinations of such sizes. In comparison, e.g. transfer speeds are typically not limited to discrete steps but can assume any value within the available range, e.g. download capacity can vary from 100Kbps to 2Mbps. Area 28 is off the scale with a total of 214 DQ requirements on a continuous scale.

3 Discussion of findings

Figure 3.1 and Figure 3.2 show the distribution of pure functional, pure quality and mixed functional and non–functional requirements per area.

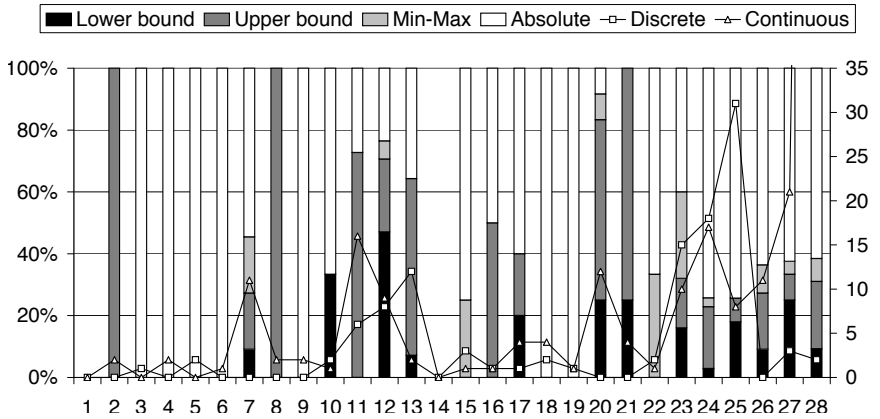

Figure 3.7: Interval (bars) and scale (line) patterns

The spread is large ranging from 100% pure functional to 0% pure functional and from 100% pure quality to 0% pure quality. The mixed functional and quality category ranges from 0% up to 83%. This further strengthens the view that the areas are very different and suggest that each area needs unique treatment of its NFR.

Figure 3.3, Figure 3.4 and Figure 3.5 also show vast differences among areas in terms of the fraction of direct metrics and the fraction of NFR that have references to standards external to the requirements specification. This implies even more that the different areas are different in nature and require tailored treatment. Similar observations have been made in (Doerr et al. 2005), (Kerkow et al. 2005).

The differences among areas may be explained by the following conjectures:

- The areas are technically very different ranging from having their major emphasis on hardware or mixed hardware and software or mainly software.
- The number of requirements in each group varies greatly, ranging from 2 to 449 with a median of 40 and a mean of 75 requirements per area.
- Some areas are easier to measure than others. Typically, physical measures in areas related to hardware are more straight–forward than subjective experience of software performance.
- Some areas have more maturity than others, in terms of how long requirements belonging to the area have been present in the platform.
- Some areas may have more dedicated resources than others in terms

of number of engineers working with the requirements of a particular area.

- Similarly, some areas may have been more thoroughly elicited, specified or validated. This is natural since time is a limited resource and the work needs to be prioritized.
- Some areas may be more critical from a quality viewpoint than others.
- Some areas may be more critical to important stakeholders than others.

This leads us to the first hypothesis: **[H1]** *Different areas of non–functional requirements, both with respect to technical domain and type of non–functional requirement, are unique in their character and require unique treatment in terms of tools support and method guidance.*

More work is needed to identify and classify the different areas and to find patterns within similar areas, across domains. If such patterns exist, these can be used to tailor the support for working with NFR within a certain area. This confirms observations from other case studies (Doerr et al. 2005). However, the underlying assumptions of the different areas and their unique behaviour are not well known and warrant further studies.

When regarding the distribution of different type, there are many NFR that are related to the architecture or associated documentation in contrast to requirements on the system as such. Performance requirements are the most common type of NFR in this case study, whereas reliability requirements are largely lacking. This leads us to the following observations:

- In the mobile phone domain size and power consumption are central along with limited computer capacity, which can explain why performance requirements are very common. Furthermore, mobile phones are not what are usually considered to be safety critical systems, such as trains or aeroplanes. This explains the limited emphasis on reliability requirements. If a different domain was analyzed, the results would probably have looked different, suggesting that different domains require different solutions.
- There are a number of NFR that are identified as having a concurrency aspect. This leads us to two conjectures: it is not always clear to which type a NFR belongs to and that NFR might belong to more than one type at the same time. Hence, when modelling quality, we need to consider that requirements might be cross–cutting and not possible to sub–divide in a strictly hierarchical manner.
- Many requirements are put on the architecture. In many cases Sony Ericsson has an architecture which needs to be taken into consideration for the sub–contracted parts. If those parts do not fit into the

overall system architecture, obvious problems will occur. This can explain why there are so many NFR in architecture related classes.

- Another reason that many requirements are related to the architecture might be that some requirements cannot be revealed to the sub–contractor for confidentiality reasons. Hence, rather than stating the end–user requirements, architecture or design requirements are specified.

Figure 3.7 is based on coding of direct NFR (DQ) and reveals that there are several different patterns for specifying intervals. It can be seen from the distribution that it is rather common that direct quality metrics are qualified with more information than just a simple absolute value. Furthermore, Figure 3.7 shows the coding of direct NFR (DQ) and reveals that there are two principally different scales used; a continuous pattern and a discrete pattern.

Based on the data analysis we make the following conjectures regarding intervals and scale patterns:

- In many cases, there is a threshold. Values up to (or down to) a certain value are all acceptable. One example is transfer rate (lower bound). Another example is start–up time (upper bound).
- Sometimes complex interrelationships exist among different NFR. For example, to maximize the conversation quality, the radio output should be as high as possible. However, the radio output level needs to be kept down to reduce current consumption and electromagnetic field affecting the person using the phone. This leads to a min–max interval.
- It is often difficult to specify an exact value for a certain quality level. Therefore, quality is often specified using an interval to indicate an approximate value within a certain range.
- There are cases where a specific level is sought, e.g. with video resolution. Due to for example compatibility issues, size restrictions and user quality perception, perhaps VGA 30fps is the only acceptable compromise, neither more nor less.
- When referring to a standard, a specific level is sometimes defined. This is then an absolute level in the standard and not an interval. In fact, the scale as such in this case would most likely be nominal and an interval scale would not make sense.
- In certain areas, prominently hardware, the scales are discrete rather than continuous. Hardware components typically have pre–defined steps and combinations which can be used. Hence, we are not free to select for example 34MB of memory. Software, on the other hand,

is prominently continuous, e.g. data transfer rate. But there are also examples where scales for software NFR are discrete, e.g. audio and video encoding which is typically standardized for interoperability purposes.

The nature of the intervals and scales for different areas are important when defining the quality levels and when negotiating and prioritizing then. This leads us to a second hypothesis: **[H2]** *Interval and scale patterns need to be aligned with the market value breakpoints and cost barriers (Regnell et al. 2007) to allow effective prioritization and negotiation.*

Each quality indicator needs to be investigated to figure out the optimal way of specifying requirements using intervals. The quality indicators patterns will be different for different areas. To better support prioritization and negotiation (e.g. as in (Regnell et al. 2007)) these patterns need to be understood for the domain. Furthermore, the complex interaction among the different quality indicators and NFR need more study to effectively support negotiation and prioritization.

The coding of NFR is a non–trivial task. It has been reported that when using the ISO 9126 it is inherently difficult to get a reliable classification (Al-Kilidar et al. 2005). Similar problems were identified in this case study. We address this by tailoring the coding scheme to the particular case. This made it easier to attain a reliable coding. The down–side of using a tailored code is that the comparability with other studies is hampered. However, each domain is unique in one way or another and we believe that the coding also needs to be tailored to the domain (Doerr et al. 2005), (Kerkow et al. 2005).

There are a number of threats to the validity of the results. First, the coding reliability may in some cases be low due to the deep domain knowledge required but sometimes lacking. This is addressed by independent coding among three researcher and consolidation based on perceived confidence.

Second, the transferability of the result can be discussed as this study only covers one specific case. Preferably, a standardized coding should be used to enable comparison with other cases. To the authors' knowledge, there are as of yet no similar studies performed. Hence, there is no standardized coding and no cases to compare with. Hopefully, the experience and outcome of this study will inspire and aid other researchers in performing similar studies, as there is a lack of empirically grounded understanding of NFR.

4 Related work

Research in non–functional requirements has concentrated on modelling and representation of NFR. However, research related to classification and

measurements of NFR are also introduced in literature. In this section, a selection of classification methods and measurements are presented.

There are case studies reporting using different NFR approaches (Balushi et al. 2007), (Doerr et al. 2005), (Jacobs 1999), (Sedigh-Ali et al. 2001). However, even though industry case studies were sometimes conducted, little or no information is shared on the characteristics of the NFR.

Balushi et al. (2007) developed a tool called ElicitO. The purpose of the tool is to empower requirements analysts during elicitation interviews. ElicitO is based on domain ontology to support elicitation activities. The domain ontology uses characteristics and metrics from the standard quality model ISO/IEC 9126 (9126-2001 E).

In the IESE NFR method (Kerkow et al. 2005) based on the ISO 9126 (9126-2001 E), quality models are used to document the understanding of quality attributes. The method has shown its usefulness in several industrial case studies (Doerr et al. 2005). The basis of the method is the concept of quality models. These models capture the behaviour of a specific quality attribute and break it down hierarchically. However, the quality models used in the method is not based on empirical data. Furthermore, detailed support on scales, intervals and interrelationships are largely missing.

According to Sedigh-Ali et al. (2001), commercial–off–the–self (COTS) based systems require metrics for quality indicators such as complexity, performance and so forth. Obtaining system–level indicators of quality is complicated for COTS products. Sedigh–Ali et al presents 13 system–level metrics for COTS based systems. How to measure these metrics are described, but no concrete or precise measures are defined. Furthermore, the 13 system–level metrics in (Sedigh-Ali et al. 2001) are not empirically validated.

Coding of NFR can be a difficult and unreliable process. Al-Kilidar et al. (2005) empirically evaluated ISO 9126 (9126-2001 E) as a mean to classify NFR. They found both shortcomings in the content of the standard and unreliable codes in an experiment. There is a lack of studies on the matter of coding NFR which needs to be addressed.

5 Conclusion

This paper presents an empirical study on non–functional requirements (NFR) in a requirements specification within the embedded software domain based on a document content analysis of 2113 requirements. As many as 40% of the requirements in the specification are non–functional. The distribution across the areas on the specification varies, but only a few areas completely lack NFR. In many cases (14%), the requirements specify both a functional and a non–functional aspect.

Standards are commonly used and for NFR, as many as 37% of the NFR references a standard. About half of the NFR are specified using a metric.

Large parts of the quantified NFR are specified on an interval, both single– and double–sided.

A general conclusion is that for a method to be successful, it is important that it is flexible enough to handle the diverse nature of NFR. This impacts all areas of requirements engineering, starting with elicitation and analysis to specification and validation. Further research is needed into different sub–domains to be able to identify general patterns and trends which can be used to facilitate industry in their work with NFR. This case study results are specific to the domain investigated and to the product platform under study and the results cannot be directly transferred to another context. However, a number of hypotheses of general interest can be stated based on this case study, as discussed in Section 3.

We used a domain–specific method for coding NFR which can be applied instead of basing a classification on a general model such as ISO 9126. Even though it requires an initial tailoring to be useful, once the coding scheme is defined, it is our experience that domain–specific coding can be reasonably reliable and efficient.

To improve how NFR are handled, we need to understand the characteristics of them, and this case study is aimed as a step towards a richer picture of NFR rooted in empirical findings in a specific domain. Further case studies are needed in other domains and on other requirements specifications to enable generalizations outside the domain of this study.

Acknowledgments

This work was partly funded by VINNOVA (Swedish Agency for Innovation Systems) within the MARS project.

References

ISO/IEC 9126-2001(E). Software engineering – product quality – part 1: Quality model, 2001.

H. Al-Kilidar, K. Cox, and B. Kitchenham. The use and usefulness of the iso/iec 9126 quality standard. In *Proceedings International Symposium on Empirical Software Engineering*, pages 122–128, 2005.

T.H. Al Balushi, P.R.F. Samoaio, D. Dabhi, and P. Loucopoulos. Elicito: A quality ontology–guided nfr elicitation tool. In *Lecture Notes in Computer Science*, volume 4542, pages 306–319, 2007.

L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *NFR in Software Engineering*. Kluwer Academic Publishers, 2000.

J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki. Non–functional requirements in industry – three case studies adopting an experience–based nfr method. In *Proceedings 13th IEEE International Conference on Requirements Engineering*, pages 373–382, 2005.

S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings 4th IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

B. Kerkow, J. Doerr, B. Paech, T. Olsson, and T. Koenig. *Requirements Engineering for Sociotechnical Systems*, chapter Elicitation and Documentation of Non–Functional Requirements for Sociotechnical Systems, pages 284–302. Hersey: Information Science Publishing, 2005.

N. Nurmuliani, D. Zowghi, and S.P. Williams. Using card sorting technique to classify requirements change. In *Proceedings of the 12th IEEE International Requirements Engineering Conference*, 2004.

B. Regnell, M. Höst, and R. Berntsson Svensson. A quality performance model for cost–benefit analysis of non–functional requirement applied to the mobile handset domain. In *Lecture Notes in Computer Science*, volume 4542, pages 277–291, 2007.

C. Robson. *Real World Research*. Blackwell, 2002.

S. Sedigh-Ali, A. Ghafoor, and R.A. Paul. Software engineering for cots–based systems. *IEEE Computer*, 34(5):44–50, 2001.

L. Upchurch, G. Rugg, and B. Kitchenham. Using card sorts to elicit web page quality attributes. *IEEE Software*, 18(4):84–89, 2001.

Paper IV

Supporting Roadmapping of Quality Requirements

Björn Regnell[1], Richard Berntsson Svensson[1], Thomas Olsson[2]
[1]Lund University [2]Sony Ericsson

Published in IEEE Software Vol. 25, no. 2, pp 42–47, 2008

ABSTRACT

When dealing with quality requirements, you often end up in difficult trade–off analysis. You must take into account aspects such as release targets, end–user experience, and business opportunities. At the same time, you must consider what is feasible with the evolving system architecture and the available development resources. Our experience from the mobile–phone domain shows that much can be gained if development team members share a common framework of quality indicators and have a simple, easy–to–use model for reasoning about quality targets. To support quality–requirements analysis, the Quper (quality performance) model combines cost and benefit views into a roadmap of each important quality indicator for the particular domain. The practical application of Quper involves six steps.

1 Introduction

Would slightly better performance be significantly more valuable from a market perspective? Would significantly better performance be just slightly more expensive to implement? When dealing with performance, usability, reliability, and so on, you often end up in difficult trade–off analysis. You must take into account aspects such as release targets, end–user experience, and business opportunities. At the same time, you must consider what is feasible with the evolving system architecture and the available development resources.

Our experience from the mobile–phone domain shows that it is helpful when software development team members share a common framework of quality indicators when discussing roadmaps for future system releases. They should also have a simple, easy–to–use model for reasoning about options as input to cost–benefit and trade–off analysis. To support roadmapping of quality requirements when developers plan systems for software–intensive consumer products, we developed the Quper (quality performance) model (Regnell et al. 2007).

Quality requirements are of major importance in the development of systems for software–intensive products (Jacobs 1999). To be successful, a company must find the right balance among competing quality attributes. How should you balance, for example, investments for improved usability of a mobile phone's phone book and better mobile positioning? In the context of quality requirements, decision making typically combines market considerations and design issues in activities such as roadmapping (Regnell and Brinkkemper 2005), release planning (Carlshamre and Regnell 2000), and platform scoping (deBaud and Schmid 1999). Models that address requirements prioritization in a market–driven context often emphasize functional aspects. (For a comparison of other relevant techniques with Quper, see the sidebar.) Quper provides concepts for reasoning about quality in relation to cost and value and can be used in combination with existing prioritization approaches.

2 Related Techniques

Several models related to requirements prioritization and cost–benefit analysis can help requirements engineers select product requirements. Karlsson and Ryan (1997) suggest a cost–value approach based on the Analytic Hierarchy Process (Saaty 1980). Their approach uses pair–wise comparisons to rate requirements on the basis of value and cost; a 2D graph displays the value against the cost (Saaty 1980). This approach deals mainly with functional requirements, but the prioritization can also include quality attributes. However, it does not explicitly address the continuous nature of quality levels. Our Quper (quality performance) model introduces

a third dimension capturing the quality level.

Noriaki Kano and his colleagues developed a model for evaluating patterns of quality (Kano et al. 1984), (Matzler and Hinterhuber 1998) . The evaluation is based on the customer's satisfaction with specific quality attributes and is displayed in a 2D graph. Like Quper, this model views quality as nonlinear. However, it does not include the cost dimension. In addition, Quper is related to roadmapping and includes benefit breakpoints and cost barriers to indicate important aspects of quality relations (for more on breakpoints and barriers, see the main article).

Quality Function Deployment is a customer– and user–oriented model for product development (Karlsson 1997). For a full implementation of QFD, customers and users must be visible. However, not all market–driven projects have access to their customers and users. In addition, QFD is a complex and comprehensive model that might require a company to completely change its practices. Quper is a simple reference model with a few concepts, aimed to be easy to use in combination with a company's current practices.

Tom Gilb's Planguage has roadmap–related concepts such as "past," "record," and "trend" in templates for quality requirements (Gilb 2005). You can use Quper in combination with Planguage to express breakpoints and barriers as well as targets related to, for example, competing products in different market segments.

Rick Kazman and his colleagues developed the Architecture Trade–off Analysis Method to find trade–offs among quality attributes that affect each other at the architecture level (Kazman et al. 1998). While ATAM considers decision making at the architecture level, Quper combines market considerations and design issues in activities such as roadmapping, release planning, and platform scoping.

For a general discussion on service value and quality in engineering, see The Changing Nature of Engineering (Aslaksen 1986).

3 QUPER

We have observed that quality is often continuous and nonlinear. That is, instead of viewing the quality level as a binary property of "good" or "bad," you can view it as having different shades on a sliding scale. For example, a small decrease in response time might require significant investments in architectural evolution; a small change in the user interface might significantly improve usability. On the basis of these observations and discussions with domain practitioners, we determined that Quper should be

- *robust to uncertainties* (we concentrate on principal properties rather than precise predictions),
- *easy to use* (the model should include a limited number of simple–to–

understand concepts), and

- *domain relevant* (it should be adaptable to a particular domain, and you should be able to incorporate it in your working practices without costly interference with existing processes, techniques, and methods).

3.1 Basic concepts

A *breakpoint* is an important aspect of the relation between quality and benefit–for example, when mobile–phone start–up time shifts from normal expectations to outperforming most competitors. A *barrier* is an important aspect of the relationship between quality and cost–for example, when better mobile–gaming performance requires an expensive rebuild of the architecture. These two concepts are the basis for Quper's three views.

The *benefit view* (see Figure 4.1) includes three breakpoints indicating principal changes in the benefit level with respect to user experience and market value. The *utility breakpoint* marks the border between useless and useful quality. "Useless" means that a product isn't accepted on the market and users do not recognize its value. After passing the utility breakpoint, a product starts to become useful and thus have a potential market value. The *differentiation breakpoint* marks the shift from useful to competitive quality. Only a few products have such quality, which makes them have a competitive market proposition. The *saturation breakpoint* indicates a change from competitive to excessive quality. At the excessive–quality level, higher quality has no practical effect on the benefit in the particular usage context.

The *cost view* (see Figure 4.2) includes foreseen barriers representing the nonlinear nature of the relation between quality and cost. For a specific quality aspect in a specific context, we approximate the quality–cost relation to have two different steepness ranges. A barrier occurs when the cost shifts from a plateau–like situation where an increase in quality has a low cost penalty, to a sharp rise where an increase in quality has a high cost penalty. A typical cost plateau is when comparatively inexpensive software optimizations might produce high gains of performance. A typical barrier might be that an increase in quality is not feasible without a large reconstruction of the product architecture. A quality aspect might have many barriers, depending on the context and the type of cost considered. Costs can, for example, be investments in development effort or the cost per hardware unit.

Quper generally aims to avoid making complete predictions of the inherently difficult relations between a system's benefit, cost, and quality. Instead, we simplify the problem by finding reasonably good predictions of a limited set of breakpoints and barriers (with ranges to indicate error margins, if appropriate). Every quality indicator must be considered care-

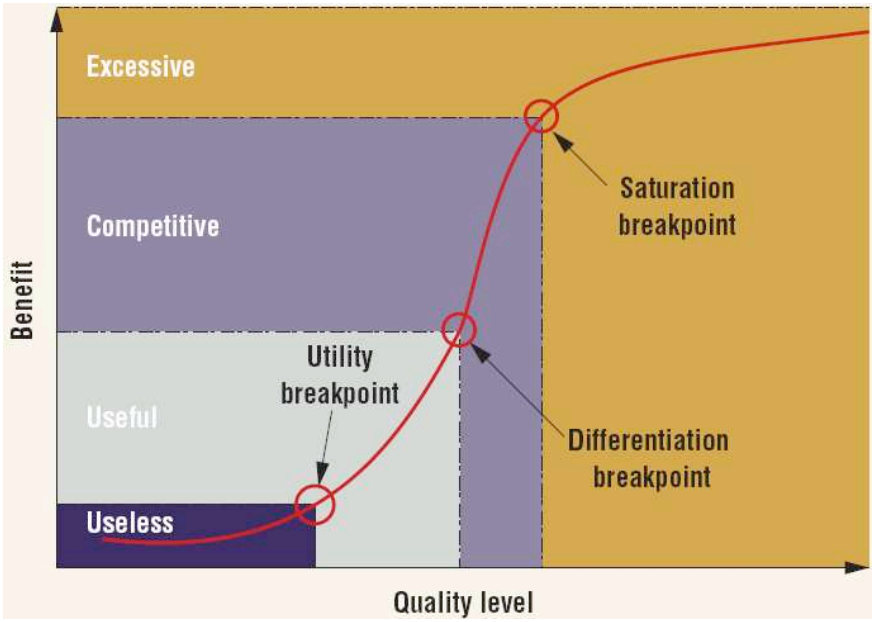

Figure 4.1: Benefit view

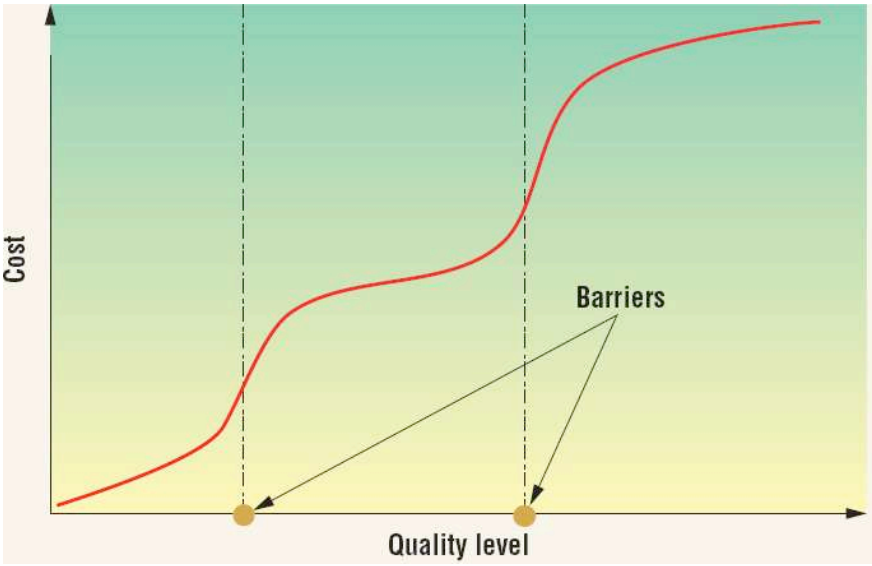

Figure 4.2: Cost view

fully to find special cases. The curves might not always look like those in Figures 4.1 and 4.2, and breakpoints and barriers might sometimes need special treatment and might change over time.

The *roadmap view* (see Figure 4.3) combines the two previous views by positioning the breakpoints and barriers on the same scale. So, you can visualize the breakpoints and barriers in relation to your product's current quality and the competing products' quality. To support roadmapping, this view also incorporates targets for coming releases.

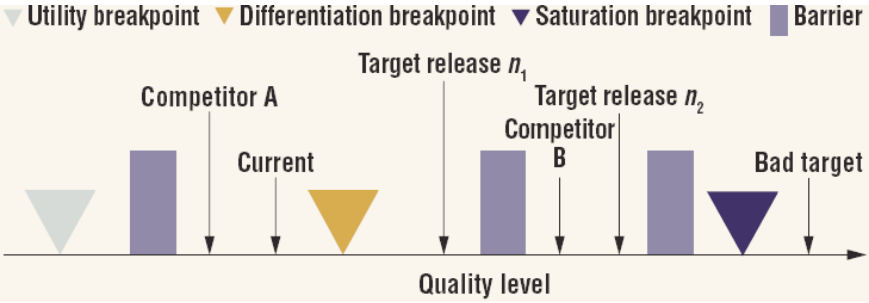

Figure 4.3: Roadmap view

In all three views, the horizontal axis indicates the quality level. The quality indicators might be specific to different entities, such as features, use cases, and market segments. Defining these indicators is the main issue in tailoring Quper for a certain domain or product.

3.2 Quper steps

Applying Quper in practice involves six steps:

1. Define the quality indicators.
2. For each indicator and for each relevant qualifier (for example, a specific feature, use case, market segment, competitor, or platform capability), estimate the breakpoints and barriers.
3. Estimate your product's current quality (for a given release) and the competing products' quality (current or future).
4. Estimate targets for coming releases, propose candidate targets, and decide on actual targets.
5. Approve and communicate roadmaps as a common vision with realistic targets for downstream systems and software engineering.
6. Revise the roadmaps and iterate any necessary steps as estimates become more certain or circumstances changes. Align the iterations with the release frequency.

We are introducing Quper at Sony Ericsson, concentrating initially on steps 1–4. We plan to eventually deploy steps 5 and 6 as well as estimate the cost barriers from step 2. We propose to align Quper with the current scoping process, which uses pairwise comparison of features (high–level functional requirements) in cost–benefit analysis (Karlsson and Ryan 1997). Figure 4.4 shows how to document the results of steps 1–4 (to preserve confidentiality, the requirements in the figure are fictitious).

Description	Quality indicator:	Time to play music (sec.)	
	Quality type:	Performance	
	Definition:	Measured from when the player presses the play button until the music begins playing, using a 20-Gbyte memory stick of type X with 100 tracks with an average duration of 3 min.	
Current market expectations	Utility:	5 sec.	
	Differentiation:	1.5 sec.	
	Saturation:	0.2 sec.	
Reference products	Competitor product X:	4 sec.	
	Competitor product Y:	2 sec.	
	Our product Z:	3 sec.	
			Target rationale
Candidate targets for release *n*	Low target:	4 sec.	We might not need to be better than product X.
	Mid target:	2 sec.	This target is possible without new architecture but needs some software optimization. We are already one of the best, so a higher-quality product's market value might be limited.
	High target:	1 sec.	If we create a new architecture, this target (which is better than differentiation) will be easy to reach. Users might require this feature within 2 years.
Contact person	Sven Svensson		

Figure 4.4: A template for documenting the results of QUPER's first four steps

In step 1, when defining quality indicators, it is important to identify relevant qualifiers and consider their consequences for the particular indicator. For example, different mobile phones offered to different market segments have different requirements for image quality. Or, a competitor might recently have released a mobile phone with better gaming performance, thereby changing users' perceptions of gaming quality. Furthermore, today's hardware isn't the same as tomorrow's. This has implications for performance requirements, as software features might run much faster. Also, users' evolving expectations might influence quality targets, as users expect better performance and quality in the latest mobile phones.

In step 2, for each quality indicator, define the current market expectations in terms of the values of the breakpoints in Figure 4.1. First, determine the utility breakpoint–the lowest acceptable value on the current market. Then, determine the saturation breakpoint, which represents quality levels clearly considered excessive in the current market. Finally, determine the differentiation breakpoint; values above it will give you a market advantage. For example, for a mobile phone, taking 10 minutes to start is useless, while less than one minute is useful. A competitive advantage could be to start the phone within 10 seconds, while 10 milliseconds could be considered excessive.

In step 3, after identifying all the quality levels, identify reference levels based on actual products– your own or your competitors'. These levels furfurther calibrate your estimates and give you objective measures to relate your targets to.

In step 4, targets are requirements with potential quality commitments. Different quality indicators might have a different number of relevant targets. Figure 4.4 shows three targets: low, mid, and high, which have different implications regarding cost and benefit.

4 Lessons learned

Before deploying Quper, we investigated it with experts in six mobile–phone subdomains: local connectivity, positioning, Java platforms, mobile TV, memory, and radio network access. We were able to define breakpoints and barriers for several quality indicators in all the subdomains. Also, the interviewees recognized the model's usefulness. They stressed that new prioritization techniques and roadmapping methodology must be simple and easy to learn and understand.

4.1 Quality indicators

Important quality indicators include these:

- for local connectivity, various data–transfer rates measured in bits per second;
- for positioning, the position accuracy (the error margin in the given positioning data) measured in meters; and
- for mobile TV, the video frame rate measured in number of image frames per second.

However, the interviewees raised two main issues. First, how many and which quality indicators should you manage? This issue is about the balance between the information's benefit and the effort for acquiring it. The number of quality indicators that you must identify depends on issues such as the domain, the product, and the most–strategic use cases.

Second, how do you manage dependencies between quality indicators? This issue is about prioritizing the indicators. You can use existing prioritization methods, but only for discrete values of the indicator. You can also compare one indicator's breakpoints with those of another. Currently, the most important dependencies are managed on the basis of expert judgment in particularly crucial cases. Trying to estimate all the dependencies inevitably leads to a combinatorial explosion. We need to understand more about dependencies before we can find heuristics to efficiently support them.

There are two types of metrics: one with lower values for higher quality (for example, the time to first fix, for mobile positioning) and one with higher values for higher quality (for example, frame rate, for mobile TV). To visualize quality indicators in the benefit view (see Figure 4.1), we recommend the second type of metric. You can achieve this, for example, by inverting the first type of metric or reversing the scale.

Taking standards into account when defining quality indicators seems inevitable in the mobilephone domain. However, the relation between a technical quality defined by standards and the user's perceived experience isn't always straightforward.

4.2 Breakpoints

We've encountered three factors relevant to breakpoint positions:

- Different use cases and market segments have different quality demands. You can express this in Quper by defining use–case–specific targets and market–segment–specific targets.

- As we mentioned before, when the products and market mature, users get familiar with features, and they expect higher quality. You can express this in Quper by having breakpoints that change over time.

- Sometimes, as the market matures, the utility and saturation breakpoints remain stable but the differentiation breakpoint rapidly becomes more demanding.

4.3 Barriers

Our experience indicates that cost has a nonlinear relationship to quality, and we found it relevant to discuss specific barriers in that relationship. However, our experiences also revealed that the next barrier you have to deal with seems much easier to identify than ones further into the future. You might have to overcome that first barrier before you can more accurately estimate future barriers.

We identified several types of costs related to different cost categories in the six subdomains. Development effort is one of the most critical cost types for software. For hardware issues (more or less indirectly linked to software), we found examples of these cost types: cost per unit, footprint, physical size, and energy consumption. We also identified the general cost of missed market opportunities.

4.4 Benefits

On the basis of our case study experiences, we conclude that Quper can provide a richer picture of quality targets. It also enables developers to put quality requirements into context and calibrate them against market and investment estimations in a coherent way. Instead of treating quality requirements separately, Quper lets developers prioritize them in combination with functional requirements and relate their priorities already during platform scoping.

5 Summary

Quper is based on our observation that existing practices often specify quality aspects without explanation or rationale. Communication is important in requirements prioritization. Quper addresses this issue, aiming to enrich the overall picture of quality requirements through a better shared understanding. Its validation in the mobile–phone domain indicates that it is feasible and relevant for this specific domain. We believe it can also be relevant in other domains where software–intensive products are developed for open markets. The main benefit in using Quper, we believe, is better release–planning decisions. We envision that Quper will support the inherently difficult prediction of the impact of different quality targets.

Acknowledgments

This work was partly funded by VINNOVA (the Swedish Agency for Innovation Systems) within the MARS project. We especially thank Per Runeson for valuable advice on initial drafts of this article.

References

E.W. Aslaksen. *The Changing Nature of Engineering*. McGraw–Hill, 1986.

P. Carlshamre and B. Regnell. Requirements lifecycle management and release planning in market–driven requirements engineering processes. In *Proceedings 11th International Workshop on Database and Expert Systems Applications*, pages 961–965, 2000.

J.M deBaud and K. Schmid. A systematic approach to derive the scope of software product lines. In *Proceedings of the 21st International Conference on Software Engineering*, pages 34–43, 1999.

T. Gilb. *Competitive Engineering*. Elsevier Butterworth–Heinemann, 2005.

S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings 4th IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

N. Kano, S. Nobuhiro, S. Takahashi, and S. Tsuji. Attractive quality and must–be quality. *Hinshitsu*, 14:39–48, 1984.

J. Karlsson. Managing software requirements using quality function deployment. *Software Quality Journal*, 6(4):311–325, 1997.

J. Karlsson and K. Ryan. A cost–value approach for prioritizing requirements. *IEEE Software*, 14(5):67–74, 1997.

R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere. The architecture tradeoff analysis method. In *Proceedings 4th IEEE International Conference on Engineering of Complex Computer Systems*, pages 66–78, 1998.

K. Matzler and H.H. Hinterhuber. How to make product development projects more successful by integrating kano's model of customer satisfaction into quality function deployment. *Technovation*, 18(1):25–38, 1998.

B. Regnell and J. Brinkkemper. *Engineering and Managing Software Requirements*, chapter Market–Driven Requirements Engineering for Software Products, pages 287–308. Springer, 2005.

B. Regnell, M. Höst, and R. Berntsson Svensson. A quality performance model for cost–benefit analysis of non–functional requirement applied to the mobile handset domain. In *Lecture Notes in Computer Science*, volume 4542, pages 277–291, 2007.

T. Saaty. *The Analytical Hierarchy Process*. McGraw–Hill, 1980.

Paper V

Introducing Support for Release Planning of Quality Requirements – An Industrial Evaluation of the QUPER Model

Richard Berntsson Svensson[1], Thomas Olsson[2], Björn Regnell[3]
[1,3]Lund University, [2],3Sony Ericsson
Richard.Berntsson_Svensson@cs.lth.se,Thomas.Olsson@
@sonyericsson.com,Bjorn.Regnell@cs.lth.se

Published at the Second International Workshop on Software Product Management (IWSPM08), September 2008, Barcelona, Spain

ABSTRACT

In market–driven product development and release planning, it is important to market success to find the right balance among competing quality requirements. To address this issue, a conceptual model that incorporates quality as a dimension in addition to the cost and value dimensions used in prioritisation approaches for functional requirements has been developed. In this paper, we present an industrial evaluation of the model. The results indicate that the quality performance model provides helpful information about quality requirements in release planning. All subjects stated that the most difficult estimations may be more accurate by using the quality performance model.

1 Introduction

Market–driven product development and release planning is becoming increasingly common in the software industry (Ullah and Ruhe 2006), (Carlshamre 2002). As market–driven product development gains greater acceptance (AlBourae et al. 2006), a new role within software companies emerged, namely that of product manager (van de Weerd et al. 2006). Product management is rather complex where the product manager has several important tasks, such as requirements management, release planning, and launching products (van de Weerd et al. 2006). Release planning is a process applying various types of upstream decision–making that combine market considerations with implementation concerns (Regnell et al. 2007). Release planning involves aspects such as selecting what features and requirements should be in a certain release, when it should be released, and at what cost (Ullah and Ruhe 2006). According to Ullah and Ruhe (2006), lacking of good release planning practices may results in unsatisfied customers and market loss, which makes release planning a major determinant of the success of a product.

Models that address requirements prioritization in a market–driven context often emphasize functional aspects, for example, the cost–value approach for requirements prioritization (Karlsson and Ryan 1997). Other methods are based on release planning and software product management (Greer and Ruhe 2004), (van de Weerd et al. 2006). However, to the best of our knowledge very little research has looked into prioritization of quality requirements in release planning, despite that quality requirements are of major importance in market–driven requirements engineering, as reported e.g. in a case study in the telecommunication domain (Jacobs 1999).

Would slightly better performance be significantly more valuable from a market perspective? Would significantly better performance be just slightly more expensive to implement? When dealing with performance, usability, reliability and so forth, we often end up in a difficult trade–off analysis. Aspects such as release targets, end–user experience, and business opportunities must be taken into account. To support release planning and roadmapping of quality requirements, we developed the quality performance (QUPER) model (Regnell et al. 2007), while applying QUPER in practice is reported in (Regnell et al. 2008a).

This paper presents one case of QUPER tailoring, implementation, and most important evaluation, conducted at Sony Ericsson, one of the leading mobile handset developers. The main purpose is to investigate the implementation of QUPER in industry. The very large–scale industry (Regnell et al. 2008b) trials allow us to validate the QUPER model's usefulness in a non–simulated environment in real projects using real requirements. The main objective and contribution of the paper is to show how QUPER can be used in one company and in particular the focus is on an evaluation of the industrial introduction of the model.

The paper is structured as follows: Section 2 gives a short introduction to the QUPER model. In section 3, the tailoring of the QUPER model is presented. In section 4, the company and its product development situation where QUPER is used is presented. Section 5 presents the research methodology while the results from the evaluation are presented in section 6. Related work is presented in section 7 and section 8 gives a summary of the main conclusions.

2 QUPER

The development of QUPER was carried out at two case companies in the mobile handset domain with a supplier–integrator relationship. Industry needs and possibilities for improvement were identified. The QUPER model was developed in three main steps (Regnell et al. 2007):

Step 1: *Problem definition*. The goal was to understand different requirement decision scenarios by focusing on the interface between the two case companies. The result of this work is reported in (Regnell et al. 2006). In addition, the need for a cost–benefit model including quality aspects to support roadmapping and scoping was identified.

Step 2: *Model definition*. The model definition was based on the input from step 1. The QUPER model was defined comprising three views: a benefit view, a cost view, a roadmap view, and the concepts of benefit breakpoints and cost barriers.

Step 3: *Model validation*. An evaluation of the model was carried out in six cases of selected subdomains through interviews with experts.

The quality performance model is a feature prioritization model that includes a third dimension related to quality, as a complement to the two dimension cost and value that are used in prioritization of functional requirements (Karlsson and Ryan 1997). The model aims to support prioritization and roadmapping of quality requirements at early stages of release planning when making high–level scoping decisions and creating roadmaps.

The QUPER model is based on the observations that quality is *continuous* and *non–linear*. The quality level is typically not viewed as either good or bad, but rather as something with different shades of goodness on a sliding scale. In addition, we assume that a change in quality level may result in non–linear changes to both cost and benefit, and that this non–linearity is of interest to release planning and roadmapping. Based on these observations, the following goals for QUPER were selected as a guide to the development step:

- *Robust to uncertainties*, concentrating on principal properties rather than precise predictions.
- *Easy to use*, the model should include only a few concepts that are easy to learn, remember, and understand by practitioners.

- *Domain relevant*, the model must be possible to combine with existing practice and possible to tailor to a particular domain.

The QUPER *benefit view* (Figure 5.1) includes three breakpoints indicating principal changes in the benefit level with respect to user experience and market value. A breakpoint is an important aspect of non–linear relation between quality and benefit. The *utility breakpoint* represents the border between a quality level useless and useful quality. Useless means that the quality is so low that the product is not accepted on the market. The *differentiation breakpoint* represents the shift from useful to *competitive* quality, which makes them have a competitive market proposition. The *saturation breakpoint* imply a change in quality level from competitive to *excessive* quality, where higher quality levels have no practical impact on the benefit in the particular usage context considered.

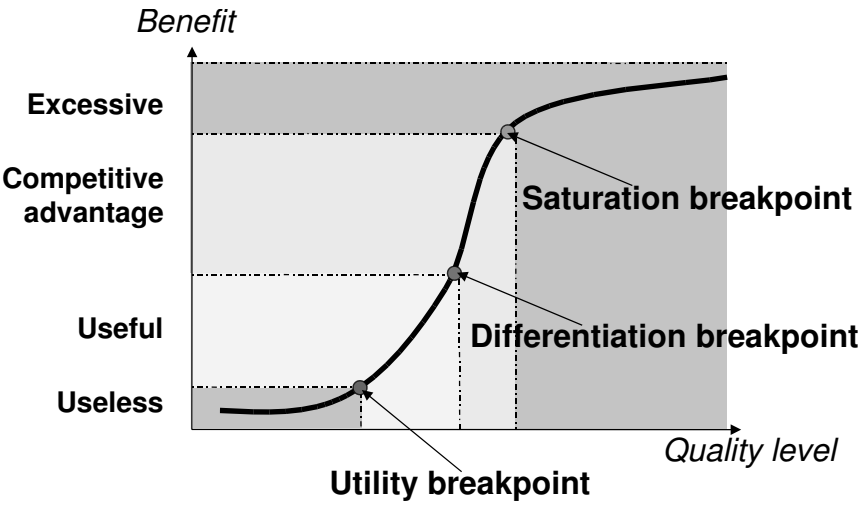

Figure 5.1: The QUPER benefit view

The QUPER *cost view* (Figure 5.2) includes the notation of cost barriers that represents the non–linear relation between quality and costs. For a specific quality aspect in a specific context, we approximate the quality–cost relation to have two different steepness ranges. A typical cost barrier may be the result of that a quality increases is not feasible without a large reconstruction of the product architecture, while a typical cost plateau is exemplified by the case where comparatively inexpensive software optimizations may result in high gains of performance.

The QUPER *roadmap view* (see Figure 5.3) combines the benefit and cost views by position the breakpoints and barrier together ordered on the same scale. This view enables visualization of benefit breakpoints and cost barriers in relation to the current quality level of a product and the qualities

of competing products. This view also combines the notation of targets for coming releases with the aim of supporting roadmapping.

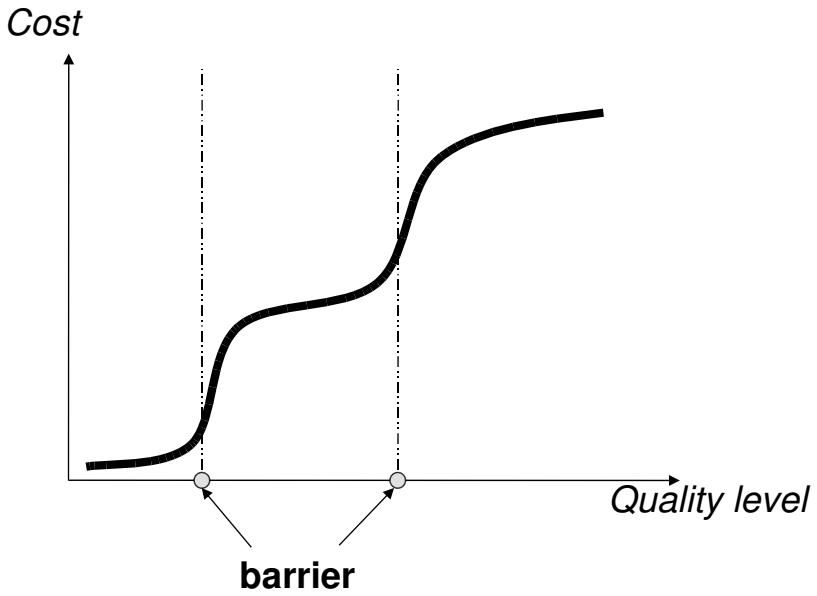

Figure 5.2: The QUPER cost view

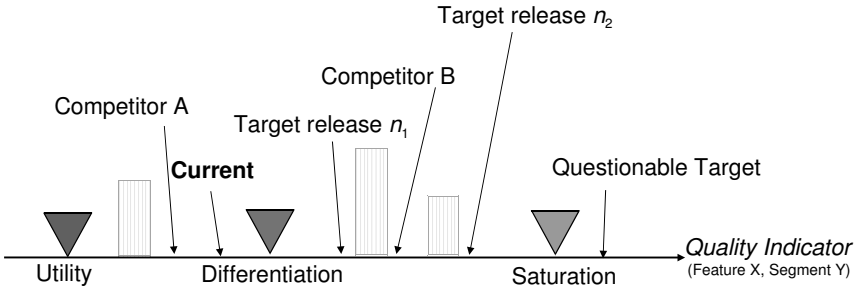

Figure 5.3: The QUPER roadmap view

3 QUPER tailoring

QUPER as presented in section 2 is generic in nature, therefore an adaption of the six steps in applying QUPER in practice (Regnell et al. 2008a) needs to be addressed prior to the model being set into operation at Sony Ericsson. This evaluation of QUPER only includes the QUPER benefit view (Figure 5.1) because it is considered the most important part of the QUPER

model for Sony Ericsson to start with. We envision the following four steps of how to use the QUPER benefit view at Sony Ericsson:

1. Define quality aspects.
2. Estimate your product's current quality (for a given release) and the competing products' quality (at present or envisioned).
3. For each quality aspect and for each relevant qualifier, estimate the breakpoints.
4. Estimate candidate targets and discuss and decide on actual targets for coming releases.

In step 1, when defining quality aspects, it is important to identify relevant qualifiers and consider their consequences. For example, different mobile phones offered to different market segments have different requirements for image quality. Furthermore, today's hardware is not the same as tomorrows. This has implications for performance requirements, as software features might run much faster.

In step 2, after identifying quality aspects, identify reference levels based on actual products, your own and competitors'. These reference levels further calibrate the estimations to provide objective measures to relate to the breakpoints.

In step 3, define the current *market expectations* in terms of breakpoints (Figure 5.1). First, determine the utility breakpoint – the lowest acceptable value on the current market. Then, determine the saturation breakpoint, which represents quality levels considered excessive in the current market. Finally, determine the differentiation breakpoint; values above this level give market advantages.

In step 4, targets are requirements with potential quality commitments. The actual requirement is an interval that is specified by two targets, *min* (the lowest acceptable quality) and *max* (the highest needed quality). Different quality aspects may have different number of relevant targets.

4 Case study description

Sony Ericsson develops mobile handsets for a global market. Sony Ericsson employs more than 5,000 people. In total, Sony Ericsson has more than 20,000 requirements. A modern mobile phone contains a complex set of features, ranging from traditional voice calls and SMS to multimedia usage and personal organizer. Compared to general purpose computers, user interfaces are much more limited as well as computing power and memory. Therefore, user interfaces need to be tailored to the device and performance needs to be optimized for the hardware constraints. Hence, in addition to scoping functional features, qualities of features are important and a large part of the effort invested.

Sony Ericsson employs a platform development process. Based on the platform, a number of products are developed. The first part of the platform process is the roadmap extraction, and each technology area has roadmaps. Based on the market and planned launch date of the first product on the platform, a selection of features on the roadmap is selected. The different technology areas are individually prioritized in terms of market value. For each area, based on the available resources, an initial scope is defined. Then the different technology areas are prioritized by different stakeholders to get their priorities. Finally, a project priority for the platform project in question is compiled by merging the different stakeholder views into a project feature list. Based on the project priority, the scope is adjusted to ensure that the features with highest return of investment are part of the project scope. Both the market value estimation and cost estimation is performed on high–level features.

Once the project scope is established, the high–level features are refined into requirements. The refinement of features includes both functional as well as quality requirements. Once the features are refined, cost estimations are redone. Also, if there have been any changes to the market, impacting either market value estimations or selection of features, market value estimations are also redone. Using the updated and more accurate estimations, the return of investment is recalculated and the project priority reanalyzed, to ensure that the most important features are part of the project scope.

5 Evaluation methodology

The research was carried out in cooperation between Lund University and Sony Ericsson. The study was carried out using an action research (Robson 2002) approach. Action research aims to influence or change some aspects of the research focus. Furthermore, action research involves the improvement of: practice, the understanding of practitioners, and the situation in which the practice takes place (Robson 2002). In this research, we are involved in improving the practice of release planning of quality requirements by introducing the QUPER model at Sony Ericcson. In addition, we improve the understanding of how practitioners use the QUPER model and its environment where the practice takes place. The general objectives of the research are to evaluate:

- The QUPER model in an industrial setting
- How easy the model adapts to existing processes
- What value the QUPER model may bring to release planning.

Four interview subjects were chosen to represent four areas (one from each area) to give a rich picture. The areas were selected to include dif-

ferences with respect to level of dependencies to hardware. The interview subjects are leaders for the selected areas. The study consists of the following three steps.

5.1 Step 1 – Interview (part 1)

Planning: Step 1 involved a brainstorming and planning meeting to plan the study and to identify different areas of interests for the evaluation. The interview instrument was designed with respect to the different areas of interests. To test the interview instrument, three pilot interviews with experts from Sony Ericsson were carried out to adapt and improve the instrument. A summary of the used interview instrument[1] is presented in Table 5.1.

Data collection: The study uses a semi–structured interview strategy (Robson 2002). All interviews were carried out individually by the first author. First, the purpose of the study was presented and then questions about their previous process were discussed in detail. All interviews were recorded and varied between 20 and 40 minutes in length. Transcripts of all interviews were made in order to facilitate and improve the analysis process.

Analysis step 1: The content analysis (Robson 2002) involved creating categories where interesting parts from the transcripts were added and discussed. The first author examined the categories from different perspectives and search for explicitly stated or concealed pros and cons with their current process of handling performance requirements.

5.2 Step 2 – Workshop

Presentation: QUPER and how to use QUPER in practice was presented in a workshop. During the workshop, a selection of requirement engineers and managers (including the subjects that participated in interview – part 1) were present. These representatives are selected based on their roles and expertise by the local managers. As they were invited, they were asked to prepare for the workshop by reading requirements from their real projects. In total, six workshops were conducted at different geographical locations and varied between 60 and 90 minutes in length. During the workshop, the first author provided help and feedback to the subjects about applying QUPER on their requirements.

Apply QUPER in real projects: As the workshop is concluded, the main goal is to achieve an understanding of how to use QUPER on real requirements in coming projects. The evaluations of the QUPER model were conducted about 3 months after the QUPER model and its practical application were introduced at Sony Ericsson. The reason for the time delay was that we wanted the subjects to use QUPER in their projects prior to the

[1] http://serg.cs.lth.se/research/packages

Table 5.1: The interview instrument

Link to Table 5.2	Question(s)
About the previous process	
Did PR exist?	Did performance requirements exist?
How are PR handled?	In what way are PR handled?
Challenges with PR	What challenges do you face when working with PR? What has been difficult?
Deciding relevant metrics	How did you decide relevant metrics
About the QUPER model	
General view	What is your general view of QUPER?
Challenges and difficulties	What challenges did you face when working with QUPER? What was difficult?
Using QUPER	Would you like to use QUPER?
Decision–making	Does QUPER lead to better decision making? (why, why not)
Time spent	How much time did it take to use QUPER?
Estimation accuracy	Do you think the estimates will be more accurate with QUPER? (why, why not)
Other issues	Can you think of any challenges that we have not covered?

PR: Performance Requirements

evaluation.

5.3 Step 3 – Interview (part 2)

Data collection: The semi–structured interview approach was continued. All interviews were carried out individually by the first author. Questions about the QUPER model were discussed in detail. The interview subjects were the same subjects as participated in step 1. All interviews were recorded and varied between 25 and 35 minutes in length. Transcripts of all interviews were made in order to facilitate and improve the analysis process.

Final analysis of all data: Since we sought a comprehensive view of the complete data set, the data from step 1 was analyzed together with the data from step 3. The interview transcripts were coded by the first author. The transcripts were analyzed and interesting quotations were marked. For the analysis, all transcript files with quotations were complied and printed. The results from the analysis are found in section 6.

5.4 Validity evaluation

In this section, we discuss the threats to validity in research projects presented in Wohlin et al. (2000), and the measures taken in the presented study to increase validity.

Conclusion validity: The conclusion validity is concerned with the ability to draw correct conclusions. The interviews were conducted at different departments and different geographical locations within the company and each interview part was done in one work session. Thus, answers were not influenced by internal discussions. The subjects selected may not be representative of the role they represent at the company. To minimize this misrepresentation, subjects were selected in cooperation with senior managers.

Internal validity: This threat may have a negative effect on the casual relationship between treatment and outcome. As the evaluations of QUPER were performed with different interview subjects, they expressed their opinions and views regarding the current process of working with performance requirements and about QUPER. As their answers were recorded by the researcher this may have constrained people in their answers. Recorded answers were only to be used by the researcher, i.e. not to be showed or used by any other party. To avoid evaluation apprehension, complete anonymity from other participants was guaranteed.

External validity: The external validity is concerned with the ability to generalize the results, in this case the applicability of QUPER in industry at companies other than Sony Ericsson. Some of the problems introduced as a motivation behind QUPER could, to some extent be general for organizations faced with developing products for markets. However, it is not possible to generalize the results from this evaluation based on the case study of Sony Ericsson; although the concepts and the practical application of the QUPER model as described in this paper and in Regnell et al. (Regnell et al. 2008a), (Regnell et al. 2007), makes it possible for any organization to adapt the concepts behind QUPER to fit their organization.

6 Evaluation results

Table 5.2 illustrates the result from this study. All areas, except email, had specified performance requirements. A general tendency observed is that performance requirements were indirectly controlled by standards or hardware components and/or suppliers. There are three main reasons why the email area did not have any performance requirements: (1) performance was continuously tested by the testing department, (2) the operating system supplier performed performance testing, and (3) no structured process of how to handle performance requirements existed. However, internal performance requirements are now introduced in the email

area. One reason is the introduction of the QUPER model, which provides a structured process of handling performance requirements, and more control over the requirements in terms of understanding why a particular quality level is needed and the relation to the competitors.

In general, the areas handled performance requirements in two ways: (1) looked at different standards stated performance and (2) the performance was provided by either hardware suppliers or the market department. Those quality levels were accepted without an understanding of why they were important. The acceptance of provided quality levels is one of three major challenges that were identified by the subjects. One subject stated:

> *"We wrote use cases [for a particular feature] based on what the user expected and needed from the new feature. We did not release this feature because the hardware could not deliver what we thought was good enough quality. We did not know if this quality level was acceptable in the market or how good our competitors' quality level was."*

By introducing QUPER, an overview of the markets and the competitors' current quality level is visible in the roadmap view, which has helped to understand what good enough quality is. This is confirmed by one subject:

> *"With the QUPER model we would have had an understanding of what is good enough quality in the market and how good our competitors are. Maybe the quality level we had would have been good enough at this particular time and we could have released it."*

The second identified challenge was related to specification of performance requirements. The subjects identified a need to be able to have an interval for the specified quality level. However, even if this was possible, one subject asked what is good enough. The concept behind QUPER is to identify the current market situation (breakpoints and analyzing the competitors) and then specify the performance requirements. By specifying performance requirements according to the QUPER model, a richer picture and understanding of what is good enough are provided. This was confirmed by one subject by stating that the QUPER model provides a more extensive view to work after.

The third challenge was how to specify performance requirements that are quantifiable, representative of the "real world", and under what conditions they should be fulfilled.

In the first step of applying QUPER in practice (section 3), relevant market segment and hardware platforms needs to be considered as well as the consequences for the performance requirement, and thereby consider under what conditions the requirements should be fulfilled. This was inline with one subject that stated:

Area	*Network access*	*Email*	*Video systems*	*Positioning*
Number of requirements	>1000 system requirements	~1000 system requirements	~500 system requirements	~40 use cases
Number of PR	~10%	None	~10%	Unknown
About the previous process				
Did PR exist?	(1) Yes, related to network access and references to standards that include PR	(1) No, it is something new (2) PR were continuously tested by the testing department, therefore not specified	Yes, mainly low level PR	(1) A few loosely defined (2) indirectly controlled by hardware suppliers
How are PR handled?	(1) Meetings with 4-5 people, for major problems, meetings could last for several weeks or months	Not applicable	(1) Looked at codec's to see what we could handle (2) input from the market department	Accepted what the hardware suppliers promised to deliver
Challenges with PR	(1) Specify the right conditions for PR (2) representative PR of the real world	No structured process for handling PR, therefore did we not have any PR	Not possible to specify an interval for the PR, what is good enough?	No understanding of what is acceptable on the market, have to trust the hardware suppliers
Deciding relevant metrics	(1) refer to different standards (2) operators provided metrics	Not applicable	Decided by hardware and standards	(1) Considered the user (2) rely on technology restrictions and hardware suppliers
About the QUPER model				
General view	(1) Saturation breakpoint important input: valuable to know when to stop improving the quality (2) like the concept of breakpoints	(1) Extensive work with competitor analysis (2) good first impression (3) takes time to identify relevant metrics	(1) Performance beyond saturation makes no difference, good to know when to stop (2) not only for PR, QUPER is applicable for all quality requirements (3) good to have a structured process	(1) An important idea (2) good to know when to stop improving the quality (3) good model for hardware PR
Challenges and difficulties	**(1) Difficult to identify differentiation and saturation breakpoints** (2) what value should the breakpoints have? **(3) easy to understand, not a complicated model**	**(1) People may interpret the breakpoints differently** (2) what value should the breakpoints have? (3) no major problems	(1) Identifying the different breakpoints **(2) is the time spent really worth it?** (3) easy to understand and apply	(1) QUPER comes natural **(2) easy to learn** (3) very pedagogical
Using QUPER?	(1) QUPER recognizes that real mobile networks do not necessarily have clean conditions that standards assume (2) relates to the real world (3) already using QUPER in real projects	(1) A more extensive view **(2) benefit of comparing our products against our competitors** (3) may feel it is too much work to do (4) we are using QUPER	**Using QUPER in real projects not only for more accurate PR, but also to understand the advantages of competitors**	(1) Provides better basis for PR (2) already using QUPER in real projects
Decision-making	(1) In sense of understanding our position on the market **(2) QUPER may be used as input for decisions about introduction of new product to the market**	**The roadmap view provides a good overview of the current market, which helps in decision making**	**(1) More informed decisions in sense of breakpoints and competitors** (2) better backing when stated as market leaders	**(1) More knowledge of the market situation** (2) not totally rely on hardware suppliers
Time spent	It took more time to use QUPER compared to the previous process. However, all subjects stated it will be less time consuming when QUPER has been used for a while. All new processes and models takes longer time to use in the beginning			
Estimation accuracy	**All subjects stated that the most difficult and insecure PR estimations may be more accurate by using QUPER**			
Other issues	No	No	(1) Cost view of competitors (2) evolution over time, a static snapshot of the current market is not enough, how does the market looks like in 2 years?	A mathematical equation that describes the benefit curve.

PR: *Performance Requirements*

Table 5.2: Evaluation results

> *"QUPER recognizes the fact that in a real mobile network you do not necessarily have the clean conditions that the standards specify."*

The subjects liked the concept of the QUPER model, especially the breakpoints. The main benefit of the breakpoints was the saturation breakpoint, where the quality level changes from competitive to excessive quality, meaning that higher quality levels have no practical impact on the benefit in the particular usage context considered. However, one problem was identified related to the saturation breakpoint. One subject stated:

> *"Do not only look at the saturation breakpoint and stop improving the performance just because QUPER says stop. If we can go beyond this breakpoint without increasing the cost and time spent, why should we not improve the performance?"*

This indicates that the evolution of the saturation breakpoint over time should be considered when revising breakpoints regularly.

Another interesting point made by one subject was that QUPER is not only applicable to performance requirements, but also can be applied to all quality requirements.

One goal with the QUPER model was that it should be easy to learn. All subjects confirmed that the QUPER model is easy to understand and learn, and is a straight forward model that is not complicated. According to one subject, QUPER is very pedagogical and makes it easy to explain and discuss with others. In addition, a common terminology among the staff improves the communication. The QUPER model is introduced at Sony Ericsson and will be used as the process for handling performance requirements.

In general, estimations of performance requirements may be more accurate when using the QUPER model according to all subjects. The most difficult and insecure performance estimations will have the highest increase of accuracy. However, none of the subjects believed that their best estimations (the easiest performance requirements to estimate) will be more accurate when using QUPER. However, using the QUPER model takes more time and requires more effort than the previous process of handling performance requirements. The difference is related to competitor analysis and identification of the breakpoints. On the other hand, the subjects believe they have more control of the performance requirements and understand why a particular metrics is used in one market segment. One reason is the introduction of breakpoints and competitor analysis. By identifying all breakpoints and the competitors' quality level, and visualize all information in the roadmap view, the subjects experienced more control of both the performance requirements as well as the current market segment.

The evaluation of the QUPER model indicates improvements in decision making, especially in release planning. All subjects agreed that the richer the understanding of the market with identified breakpoints, the quality

level of their own and their competitors' products, the more accurate the decisions are. The subjects believe the QUPER model will be of major help in release planning, which was stated by one subject:

> *''The QUPER model can be used as input for release planning and decision making; and when we should introduce a product to a particular market segment"*

Another subject stated when asked if the QUPER model may help in decision making:

> *"Yes because you know more about the market and you are not 100% controlled by the hardware suppliers."*

According to another subject, the QUPER model is especially an important input when making decisions about what time a product with a certain quality level should be released. In addition, the QUPER model helps to understand when the market has matured over time, which is when the breakpoints have changed, and the test results show lower performance than expected. It is still possible to know that we are better than our competitors and therefore release the product, stated one subject. Another important feature of the QUPER model in decision making was the roadmap view, which provides the decision makers with a good overview of the market.

During the interviews, one main challenge of applying QUPER was identified, difficulties to identify the values of the differentiation and saturation breakpoints. When to stop calibrating those breakpoints? One subject relied on a measurement report that was conducted by an industrial organization together with the expertise within the area. However, by using QUPER over a longer period of time, all subjects believed this will not be a challenge. The first time a new model is used is always difficult before knowing what to do and how to do it. In addition, another challenge was raised by one subject; different people may have different understanding and opinion of the breakpoints value. This will be a smaller problem in the future when the staff has used QUPER for a longer period of time, which was confirmed by all subjects.

7 Related work

Several models related to requirements prioritization and cost–benefit analysis may help product managers select requirements for a certain release. The contributions in this area include: Kano (Kano et al. 1984), planguage (Gilb 2005), quality function deployment (QFD) (Karlsson 1997), and a cost–benefit approach (Karlsson and Ryan 1997) based on the analytical hierarchical process (AHP) (Saaty 1980). Kano et al. (1984) developed a

model for evaluating patterns of quality. Similar to the QUPER model, Kano's approach views quality relationships as non–linear. The Kano model, however, does not include a cost dimension as in the QUPER model. In addition, Kano's model is not related to roadmapping, benefit breakpoints, or cost barriers to indicate important aspects of quality relations.

Gilb's planguage (Gilb 2005) has roadmap related concepts such as past, record, and trend in templates for quality requirements. QUPER could be used together with planguage to express breakpoints, barriers, and targets related to, for example, competing products in different market segments.

QFD (Karlsson 1997) is a comprehensive, customer and user oriented approach to product development. To fully implement QFD, customers and users need to be visible; however, not all market–driven projects have access to customers and users (Karlsson 1997). Furthermore, QFD measures quality attributes using a scale where no clear distinctions between the values are provided. While QFD is a complex and comprehensive methodology that may require a complete change of current practice, QUPER is a simple reference model to be used in combination with current practice to support communication of quality attributes using a few, easy concepts.

Karlsson and Ryan (1997) suggested using a cost–value approach for requirements prioritization based on the AHP (Saaty 1980). This approach is mainly used for functional requirements; however, quality requirements can of course be included as objects of prioritization in AHP. The QUPER model thus goes further by introducing a third dimension related to the continuous nature of quality attributes.

8 Conclusions

In this article we have tailored, implemented, and evaluated the QUPER model at Sony Ericsson by applying it in real projects, using real requirements, by industry professionals. The overall result indicates that the QUPER model is relevant in high–level decisionmaking for quality requirements in an activity such as release planning. The concepts of breakpoints, competitor analysis, and identification of own products quality level provides a greater understanding of the current market segment and why a certain quality level is needed in a particular release. The goal of the model is to be useful by being simple and it must be possible to combine QUPER with current practices. The conducted evaluation shows that QUPER is easy to understand and learn, straight forward, and not complicated to apply in Sony Ericsson's current practice. In fact, all subjects stated that they are and will use QUPER. In addition, the concepts behind QUPER improve the communication among staff regarding requirements prioritization.

The main identified challenge was difficulties to identify and specify the values for the differentiation and saturation breakpoints. Furthermore, different understanding of the breakpoints value among the staff was raised

as a challenge.

The evaluation indicates that QUPER is feasible and relevant to the selected domain. We also believe that the general concepts of QUPER are transferable to release planning for other domains of market–oriented product development, but this needs to be investigated in further research. Further research also includes, an additional evaluation of the QUPER model involving more areas and subjects with different roles. Furthermore, a practical application and evaluation of the QUPER cost view will be investigated. In addition, evolution of the market needs to be investigated, how to use a snapshot of today's market when predicting future quality levels.

References

T. AlBourae, G. Ruhe, and M. Moussavi. Lightweight replanning of software product releases. In *Proceedings of the 1st International Workshop on Software Product Management*, pages 27–34, 2006.

P. Carlshamre. Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering Journal*, 7(3):139–151, 2002.

T. Gilb. *Competitive Engineering*. Elsevier Butterworth-Heinemann, 2005.

D. Greer and G. Ruhe. Software release planning: An evolutionary and iterative approach. *Information and Software Technology*, 46(4):243–253, 2004.

S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings 4th IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

N. Kano, S. Nobuhiro, S. Takahashi, and S. Tsuji. Attractive quality and must-be quality. *Hinshitsu*, 14:39–48, 1984.

J. Karlsson. Managing software requirements using quality function deployment. *Software Quality Journal*, 6(4):311–325, 1997.

J. Karlsson and K. Ryan. A cost-value approach for prioritizing requirements. *IEEE Software*, 14(5):67–74, 1997.

B. Regnell, H.O. Olsson, and S. Mossberg. Assessing requirements compliance scenarios in system platform subcontracting. In *Lecture Notes in Computer Science*, volume 4034, pages 362–376, 2006.

B. Regnell, M. Höst, and R. Berntsson Svensson. A quality performance model for cost-benefit analysis of non-functional requirement applied to the mobile handset domain. In *Lecture Notes in Computer Science*, volume 4542, pages 277–291, 2007.

B. Regnell, R. Berntsson Svensson, and T. Olsson. Supporting roadmapping of quality requirements. *IEEE Software*, 25(2):42–47, 2008a.

B. Regnell, R. Berntsson Svensson, and K. Wnuk. We beat the complexity of very large-scale requirements engineering? In *Lecture Notes in Computer Science*, volume 5025, pages 123–128, 2008b.

C. Robson. *Real World Research*. Blackwell, 2002.

T. Saaty. *The Analytical Hierarchy Process*. McGraw-Hill, 1980.

M.I. Ullah and G. Ruhe. Towards comprehensive release planning for software product lines. In *Proceedings of the 1st International Workshop on Software Product Management*, pages 51–55, 2006.

I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma. On the creation of a reference framework for software product management: Validation and tool support. In *Proceedings of the 1st International Workshop on Software Product Management*, pages 3–11, 2006.

C. Wohlin, P. Runeson, M. Höst, M.C. Ohlson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering: An introduction*. Kluwer Academic, 2000.