Supporting Release Planning of Quality Requirements: The Quality Performance Model

Richard Berntsson Svensson

Doctoral Dissertation, 2011

Department of Computer Science
Lund University

LU-CS-DISS:2011-3
Dissertation 36, 2011
ISSN 1404-1219
ISBN 978-91-976939-4-3

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

Email: Richard.Berntsson_Svensson@cs.lth.se
WWW: `http://www.rbsv.eu`
WWW: `http://www.quper.org`

Typeset using LaTeX
Printed in Sweden by Tryckeriet i E-huset, Lund, 2011

ABSTRACT

In a competitive environment, as experienced by market-driven organizations, it is important to plan software product releases with time-to-market in mind. To increase the chances of market success, software products need to be released to the market, not only at the right time, but also with higher level of quality than the competitors' products. Thus quality requirements can be seen as a key competitive advantage. An especially challenging problem for an organization that develops software products offered to a market is to set the right quality target in relation to future market demands and competitor products. When is the quality level a competitive advantage? The goal of this thesis is to increase the awareness and understanding of quality requirements, and to find means for improving the ability to make early estimates of quality requirements, e.g., performance requirements, in order to enhance high-level decision-making related to release planning and roadmapping.

This thesis is based on empirical research using a flexible research design. The research contains two qualitative surveys of quality requirements challenges in market-driven software product development organizations based on interviews with practitioners. From these surveys, 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 are classified and specified (including quantification) in an industrial context, which concludes that for a method to be successful, it is important that the method itself is flexible enough to handle the diverse nature of quality requirements. A model called QUPER (QUality PERformance) is evolved and evaluated in two organizations. The model aims at supporting release planning of quality requirements, and was found relevant by both organizations. Finally, a prototype tool for the QUPER model was developed and evaluated in a software organization. The QUPER prototype tool was found to provide a clear overview of the current market situation by the generated roadmaps, and to reach an alignment between practitioners, e.g., product managers and developers, of what level of quality is actually needed.

ACKNOWLEDGEMENTS

The work presented in this thesis was partly 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 and by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se).

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 guidance and advice. I would also like to thank my assistant supervisor, Professor 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 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, in particular Associate Professor Tony Gorschek. 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 and friends for their understanding and support.

LIST OF PUBLICATIONS

This thesis is comprised of 6 papers that are either published or in print (4 papers), or submitted and under review (2 papers) for publication. In addition, 15 published papers were not included in the thesis, but are related to the thesis.

Publications Included in the Thesis

I **Quality Requirements in Industrial Practice - an extended interview study at eleven companies**
Richard Berntsson Svensson, Tony Gorschek, Björn Regnell, Richard Torkar, Ali Shahrokni, and Robert Feldt
IEEE Transactions on Software Engineering, 2011, in print.

II **Prioritization of Quality Requirements: State of Practice in Eleven Companies**
Richard Berntsson Svensson, Tony Gorschek, Björn Regnell, Richard Torkar, Ali Shahrokni, Robert Feldt, and Aybüke Aurum
In Proceedings of the 19th IEEE International Requirements Engineering Conference (RE'11), Trento, Italy, pp. 69–78, 2011.

III **How are quality requirements specified? A document analysis case study**
Richard Berntsson Svensson, Thomas Olsson, and Björn Regnell
Submitted to Information and Software Technology, 2011.

IV **The Quality Performance Model: Supporting Release Planning of Quality Requirements**
Richard Berntsson Svensson, Björn Regnell, and Jane Cleland-Huang
Submitted to IEEE Transactions on Software Engineering, 2011.

V **Setting quality targets for coming releases with QUPER - an industrial case study**

Richard Berntsson Svensson, Yuri Sprockel, Björn Regnell, and Sjaak Brinkkemper
Requirements Engineering, 2011, in print.

VI **A Prototype Tool for QUPER to Support Release Planning of Quality Requirements**
Richard Berntsson Svensson, Pontus Lindberg Parker, and Björn Regnell
In the Fifth International Workshop on Software Product Management (IWSPM'11), Trento, Italy, 2011.

Related Publications

VII **An Empirical Study on the Importance of Quality Requirements in Industry**
Jose Luis de la Vara, Krzysztof Wnuk, Richard Berntsson Svensson, Juan Sanchez, and Björn Regnell
In Proceedings of the 23rd International Conference Software Engineering and Knowledge Engineering, (SEKE'11), Miami Beach, USA, pp. 438–443, 2011.

VIII **Cost and Benefit Analysis of Quality Requirements in Competitive Software Product Management: A case study on the QUPER model**
Richard Berntsson Svensson, Yuri Sprockel, Björn Regnell, and Sjaak Brinkkemper
In Proceedings of the fourth International Workshop on Software Product Management, (IWSPM'10), Sydney, Australia, pp. 40–48, 2010.

IV **Towards Modeling Guidelines for Capturing the Cost of Improving Software Product Quality in Release Planning**
Richard Berntsson Svensson, Björn Regnell, and Aybüke Aurum
In Proceedings of the 11th International Conference on Product Focused Software Development and Process Improvement, (PROFES'10), Limerick, Ireland, pp. 20–23, 2010.

X **Managing Quality Requirements: A Systematic Review**
Richard Berntsson Svensson, Martin Höst, and Björn Regnell
In Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications, (SEAA'10), Lille, France, pp. 261–268, 2010.

XI **Quality Requirements: Trade-off Analysis of Benefits and Cost**
Richard Berntsson Svensson
12th Australian Workshop on Requirements Engineering (AWRE'09), Sydney, Australia, 2009.

XII **Investigating Upstream versus Downstream Decision-Making in Software Product Management**
Krzysztof Wnuk, Richard Berntsson Svensson, and Björn Regnell
In Proceedings of the third International Workshop on Software Product Management (IWSPM'09), Atlanta, USA, pp. 23–26, 2009.

XIII **Stakeholders Perception of Success: an Empirical Investigation**
Evgenia Egorova, Marco Torchiano, Maurizio Morisio, Claes Wohlin, Aybüke Aurum, and Richard Berntsson Svensson
In Proceedings of the 35th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA'09), Patras, Greece, pp. 210–216, 2009.

XIV **Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems**
Richard Berntsson Svensson, Tony Gorschek, and Björn Regnell
In Proceedings of the 15th International Working conference on Requirements Engineering: Foundation for Software Quality (REFSQ'09), Amsterdam, Netherlands, pp. 218–232, 2009.

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

XVI **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 (IWSPM-'08), Barcelona, Spain, 2008.

XVII **Can We Beat the Complexity of Very Large-Scale Requirements Engineering?**
Björn Regnell, Richard Berntsson Svensson, and Krzysztof Wnuk
In Proceedings of the 14th International Working conference on Requirements Engineering: Foundation for Software Quality (REFSQ'08), Montpellier, France, pp. 123–128, 2008.

XVIII **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
In Proceedings of the 13th International Working conference on Requirements Engineering: Foundation for Software Quality (REFSQ'07), Trondheim, Norway, pp. 277–291, 2007.

XIX **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 (MeReP'07), Palma de Mallorca, Spain, 2007.

XX **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), Adelaide, Australia, 2006.

XXI **Successful software project and products: An empirical investigation**
Richard Berntsson Svensson, and Aybüke Aurum
In Proceedings of the IEEE/ACM fifth International Symposium on Empirical Software Engineering (ISESE'06), Rio de Janeiro, Brazil, pp. 144–153, 2006.

Contribution Statement

The author of this thesis is the main author of all of the six included papers. This means responsibility for running the research process, dividing the work between co-authors, and conducting most of the writing. Paper I and II were produced in cooperation with other universities. In both cases, 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 the main author. For Paper III, the main author'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 main author wrote most of Paper III. The research in Paper IV was performed mainly by the main author, who designed and conducted most of the work, as well as reported on the study. Paper V was produced in cooperation with another university. Most of the design was performed together with the co-authors, while most of the analysis and writing was performed primarily by the main author. The execution of the study was performed by the second author of Paper V. Paper VI describes a tool which was designed by the main author, but developed by the second author of paper VI. The paper is written primarily by the main author.

CONTENTS

INTRODUCTION

Software continually becomes more important and composes a large share of today's products, thus gaining more importance in our lives. As a result, our dependence on software intensive systems in everyday life increases. Several different domains need to handle software development, e.g., developers of IT systems in banking, the automotive industry, train control systems, telecommunication systems, and developers of consumer products (e.g., mobile phones). As software become more important, the complexity, which is determined partly by functionality and partly by quality requirements (QR) [41], of the software products increases. Moreover, the intangible and flexible nature of software causes software products to be over-represented in project failure statistics, where typical problems include budget overruns, missed deadlines, and poor quality.

The handling and balancing of QR play a crucial role in software product development. In particular in market-driven organizations, which release their products to an open market with numerous potential customers and users. The ability to develop software products that meet customers' needs and expectations, and offers high value increases the likelihood of market success. In addition, to increase the chances of market success, the software product needs to be released to the market at the right time with higher level of quality than the competitors' products, thus QR can be seen as a key competitive advantage. However, despite their importance, QR are often poorly understood, generally stated informally in a non-quantifiable manner, often contradicting, and difficult to validate.

In market-driven software product development, customer expectations are gathered, analyzed, specified, and validated in the market-driven requirements engineering (MDRE) activity. Market-driven requirements engineering is recognized as a key area for software product development companies since it lays the foundation for successful planning, prioritization and development of the software product before it is released to the market. In a competitive environment, as experienced by market-driven organizations, it is important to plan the product releases with time-to-market in mind. Release planning is the processes of deciding which features should be included in the next releases, when it should be released (time) and at what cost this should be achieved. An especially challenging problem for an organization that develops software products offered to a market is to set the

right quality target in relation to future market demands and competitor products, when is the quality level good enough, when is the quality level a competitive advantage?

The main goal of this research is to increase the awareness and understanding of quality requirements, and to find means for improving the ability to make early estimates of quality requirements, e.g., performance requirements, in order to enhance high-level decision-making related to release planning and roadmapping. The main contributions are: increased understanding of QR challenges in the software industry based on a qualitative survey, understanding of QR specification and prioritization based on a case study and a qualitative survey, and a model for supporting release planning and roadmapping of QR, more specifically with the goal to provide concepts for qualitative reasoning of orders of magnitude rather than precise mathematical formulas, evaluated in two companies from different domains in two countries using action research.

The remainder of this section is structured as follows. Section 1 provides background information and related work on the investigated areas. First, software product management (SPM) is presented, followed by one of the main activities in SPM, namely MDRE. Then, the release planning process, which is part of both SPM and MDRE, is presented. Finally, background information and related work on quality requirements are provided. Section 2 identifies research gaps in the related work and discuss how the research questions address these research gaps. Moreover, the research questions arising form the research gaps are stated and motivated, and the relation between the research questions and included papers is presented. The research methodologies used to answer the research questions are presented in Section 3. Furthermore, the motivations of chosen the research methodologies are discussed. The results are summarized for each individual paper, and linked to research questions and research gaps in Section 4, while Section 5 draws together the obtained results in order to provide answers to the research questions. Section 6 describes how the research could be continued in the future. Finally, Section 7 concludes the thesis.

1 Background and Related Work

This section gives some background to software product management, requirements engineering focusing on market-driven requirements engineering, release planning, and quality requirements. Each section is summarized in a number of challenges/issues.

1.1 Software Product Management

Software product development is more and more commercialized as standard products [178], and less customized software is developed [177]. At the same

time, market-driven product development gains greater acceptance [8]; therefore, a new role within software companies has emerged, namely that of product manager [177]. However, product management is not a new domain; it has been established in other sectors, such as manufacturing since the 19th century [109]. Only recently has software product management (SPM) received attention in the software industry [178], for example, in companies like Microsoft [52] and Alcatel [65]. Product management is defined as:

> *"The discipline and role, which governs a product (or solution or service) from its inception to the market/customer delivery in order to generate biggest possible value to the business."*
>
> [64]

Product management covers all phases of the life-cycle, from strategy definition down to delivery, until retirement of the product. Figure 1, which is adapted from [64], illustrates product management and the lifecycle.

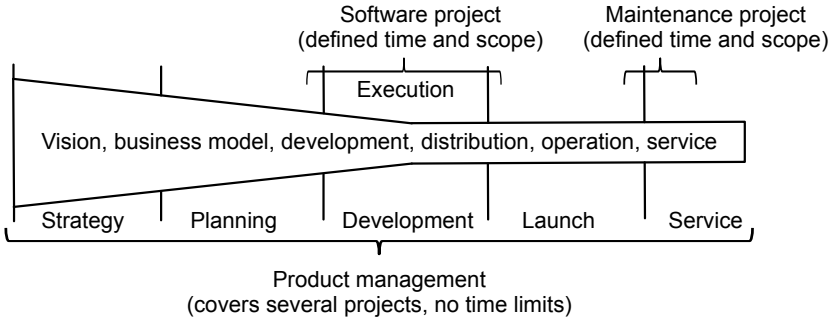


Figure 1: Product management and the lifecycle [64]

The role of the product manager has been studied, analyzed, and reported in the literature. Gorchels reports a general description of the role of product manager, the responsibilities, and the product manager's interfaces to other roles, such as marketing [74]. In another study, Cooper describes the importance of the product manager in order to drive the new product introduction and to drive the product lifecycle [51]. However, these general descriptions of the product manager role do not specifically deal with software products. Others have studied the relation between software engineering (SE) and business perspectives, for example, Lehtola et al. describe how roadmapping links to product requirements [119].

In software engineering, the role of software product manager has emerged over recent years and plays an important and critical role in the success of a software company's product [50]. The success of the product is dependent on the

skills and the competence of the product manager [64], and to create successful products, the software product manager has to manage product requirements, release planning, and launching products [50], [64], [177]. Moreover, the software product manager has to understand and define stakeholders [50], and market needs must be successfully identified and translated into the scope of the product [63]. However, deciding what requirements to include in a product's scope is not a trivial task [27], [182], and some requirements may be postponed to a later point in time [79], [183]. Reaching this, often uneasy compromise, sometimes means that the development of already made commitments may need to be sacrificed at the expense of wasted effort.

To reach a more mature SPM process, methods and standards are needed [19], but most software engineering standards only look into the engineering aspects within a single project [64]. Not even some of the best-known methods, such as the Capability Maturity Model (CMM) [132] and its follow-up CMMI [47] include product management related processes [39]. The ISO standards for systems life cycle processes [87] and software life cycle processes [86], and the IEEE standard for application and management of the systems engineering process [84] do not detail upstream activities before project start or product launch, nor do they mention how requirements are decided and how the system is defined.

Later research in the field of SPM has lead to the development of SPM frameworks, the reference framework for software product management [177], [178], the software product management framework [85], the situational assessment method (SAM) [18], and the software product management competence model [20].

The SPM framework [85], which is visualized in Figure 2, has seven main activities such as product strategy, product planning, development, and marketing. Each of the seven main activities has several sub activities. For example, product planning includes MDRE (described in Section 1.2), release planning (described in Section 1.3) and roadmapping, while development includes project management and quality management. The SPM maturity matrix [20] is a key component in SAM which is used to determine the SPM maturity level of an organization. In addition, the SPM maturity matrix identifies areas that the organization needs to improve to be able to reach a higher maturity level.

The SPM competence model [20], which is illustrated in Figure 3, consists of four main business functions, namely: *requirements management*, *release planning*, *product planning*, and *portfolio management*. The four main business functions are based on a structure where a portfolio (represented in portfolio management) consists of products, a product (represented in product planning) consists of releases (represented in release planning), and releases consists of requirements (represented in requirements management). Each business function includes a number of focus areas (see Figure 3). Each of the four main business functions and its focus areas are described below.

Portfolio management is related to decision-making about introduction of new products (based on market trends and product development strategy), the

Strategic Management	Product Strategy	Product Planning	Development	Marketing	Sales and Distribution	Evolution and Service
Portfolio Management	Positioning and Product Definition	Product Life-Cycle Management	Project Management	Marketing Mix Optimization	Customer Relationship Management	Technical Support
Corporate Strategy	Delivery Model	Market-Driven Requirements Engineering	Engineering Management	Marketing Planning	Sales Strategy and Planning	Marketing Support
Innovation Management	Sourcing	Release Planning	Project Requirements Engineering	Product Launch	Channel Preparation	Sales Support
Resource Management	Business Case and Costing	Roadmapping	Quality Management	Customer Analysis	Sames Management	Services Preparation
Market Analysis	Pricing			Opportunity Management	Operational Distribution	Services Provisioning
Product Analysis	Performance and Risk Management			Operational Marketing		
	Ecosystem Management					
	Legal and IPR Management					

Figure 2: Software product management framework [85]

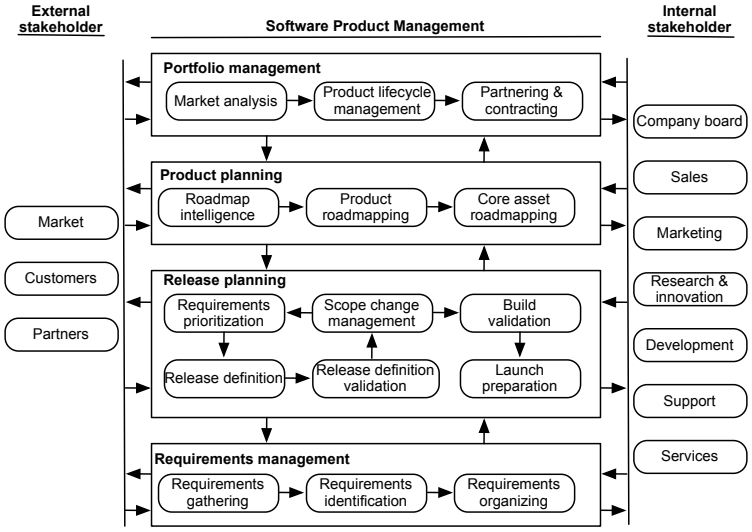


Figure 3: Software product management competence model [20]

product lifecycle, and establishing partnerships. As illustrated in Figure 3, portfolio management has three focus areas. The first focus area is *market analysis* where information about the market is gathered for decision-making about an organization's product portfolio. The second focus area is *product lifecycle management*, which includes key decision-making about product life and product changes across the entire product portfolio. Third, *partnering & contracting* focuses on the establishment of partnerships, pricing, and distribution.

Product planning includes the gathering of information and creation of a roadmap for a product, or product line, and its core assets. Product planning consists of three focus areas, *roadmap intelligence*, which is about gathering supporting information for creating a roadmap, *product roadmapping*, which is the actual creation of the roadmap, and *core asset roadmapping*, which includes the planning of the development of components that are shared by multiple products.

Release planning is a process applying various types of upstream decision-making that combines market considerations with implementation concerns [143] in order to successfully create and launch a release. The business function release planning starts with *requirements prioritization* where requirements are prioritized (see Section 1.3 for a more detailed description of requirements prioritization), and is followed by *release definition*, which includes selecting requirements for the next release. *Release definition validation* is focused on the validation of the selected requirements by internal parties. *Scope change management* is the focus area for handling scope changes that may occur during the development. Once the development department has approved the release definition, the built release is validated (*build validation*) before it is released, and a *launch preparation* package is constructed. For a more detailed description of release planning, see Section 1.3.

Requirements management includes the activities of continuos management of requirements that are not part of a release. Requirements management consists of three focus areas, eliciting (*requirements gathering*), identifying (*requirements identification*), and organizing (*requirements organizing*) the incoming requirements from all of the internal and external stakeholders. For a more detailed description of requirements managements, see Section 1.2.

Software Product Management - Challenges

Although several of the existing product management challenges may exist in SPM, specific challenges have been identified in SPM ([20], [27], [64], [178], [182]). A selection of the identified challenges are:

- **Changing software and release frequency**
  Software products can be changed easily, sold products can be updated by release updates, and distribution of extra copies of the software product do not add extra cost for the organization [178]. However, since software products can be more easily changed and updated compare to other products, the

release frequency is higher. Hence, organizing and tracking the changes in the design are complex [178].

- **Lack of authority**
  The software product manager has many responsibilities; however, he/she often has no authority over the development team, which requires consent from several players in the decision-making process [178].

- **Lack of knowledge**
  Despite the importance of software product management, and the product manager's function, little education [64], [178] and knowledge [20] exist in the area of SPM. According to [20], [64], not even a reference body of knowledge, such as PMBOK [139] and SWEBOK [1] exists for SPM. Therefore, for a product manager to learn the necessary skills and functions of the role, they need to learn 'on-the-job' [20].

- **Overscoping**
  The difference between release planning and scoping can in general be described as, release planning prioritizes features to releases while scoping identifies and categorizes features [176]. That is, scoping identifies individual products and assign features to the products [164], which means creating the product portfolio [163] without taking resources, capacities, or technological constraints into consideration [176]. While release planning prioritizes the features and assign them to releases. The scoping process is a challenge in SPM due to the continuous flow of requirements. The challenge lies in the changes to the scope that happens throughout the entire requirements engineering process, including both scope inclusions and exclusions [182]. One of the main causes for overscoping is an unclear vision of the overall goal [27].

1.2 Market-Driven Requirements Engineering

This section offers an introduction to requirements engineering (RE) and then moves on to introduce market-driven requirements engineering (MDRE) where some of the differences between customer-specific RE and MDRE are presented. In addition, challenges faced in MDRE are presented.

Requirements Engineering

Traditionally, RE takes place in the beginning of every project, and results in a specification that defines the product to be developed. This view is based on the *waterfall model* [150], where requirements engineering is followed by design, implementation, testing, and maintenance activities. However, this cascade process may not be the most appropriate in practice, since the flexible nature of software requires the development process to be more iterative and evolutionary. New and

changed requirements appearing during development calls for continuous RE efforts.

There are many definitions of RE in literature, for example, Sommerville [170] describes RE as an process of understanding and defining required services for the system, and identifying possible constraints of the system's operation and development. On the other hand, Kotonya and Sommerville [112] compares RE to system analysis, which is mainly concerned with analyzing and specifying business systems. However, while system analysis is mainly focused on the business aspects, RE is often concerned with both business and system concerns for the system to be developed. The importance of RE is stressed by Aurum and Wohlin [12] as one of the most crucial stages in software design and development when the critical problem of designing the right software product for the customer is tackled. Aurum and Wohlin [12] describe that RE is concerned with the identification of goals for a proposed system, and the operation and conversion of these goals into services and constraints.

Requirements engineering can be described as a process that involves activities that are required to gather (eliciting), create (specifying), and maintain a software product's requirements specification. In literature, several different RE processes and activities are proposed ([23], [112], [120], [122], [127], [129], [170], [171]). Moreover, the proposed RE process models have different structures: linear, linear with iterations between activities, and iterative.

RE processes are often depicted with a linear, incremental model where common RE activities, such as elicitation, analysis, and specification are combined under different headings, nevertheless they still follow a similar linear transition. Kotonya and Sommerville [112] suggest a linear RE process model that indicates iterations between the activities, which is illustrated in Figure 4. According to Kotonya and Sommerville [112], the activities in the model overlap and are often performed iteratively.

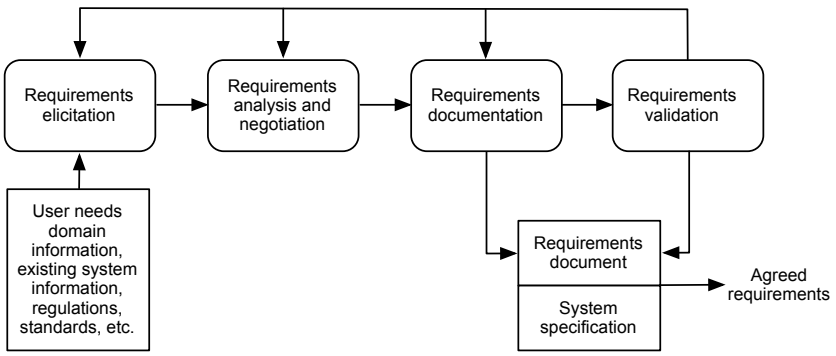


Figure 4: Kotonya and Sommerville [112] RE process model

Macaulay [122] presents a pure linear RE process model (see Figure 5). The model does not indicate overlapping or iterations of activities as suggested by Kotonya and Sommerville [112].

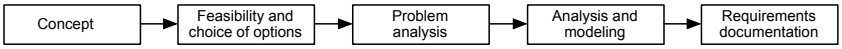


Figure 5: Macaulay [122] RE process model

The third RE process model structure, iterative, is suggested by Loucopoulos and Karakostas [120]. Their model depicts the RE process as iterative and cyclical, which is illustrated in Figure 6. The model shows interactions between elicitation, specification, validation, user, and the problem domain. Although Loucopoulos and Karakostas [120] model includes similar activities as Kotonya and Sommerville [112], and Macaulay [122], the order in which they occur is non-linear.

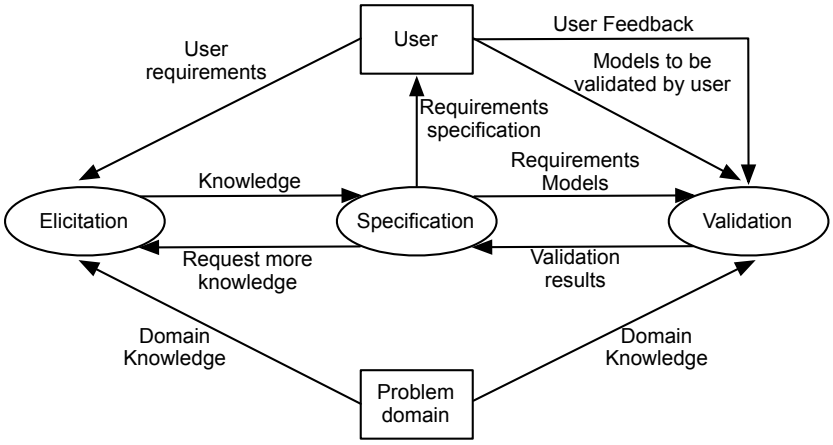


Figure 6: Loucopoulos and Karakostas [120] RE process model

Sommerville [170] defines the RE process in four main activities, which is illustrated in Figure 7. The feasibility study is performed before the requirements elicitation starts, and the activities are performed iteratively in order to handle changing requirements. In addition, requirements management is performed continually throughout the product lifecycle to understand and control requirements changes. Each of the activities is described in more details below.

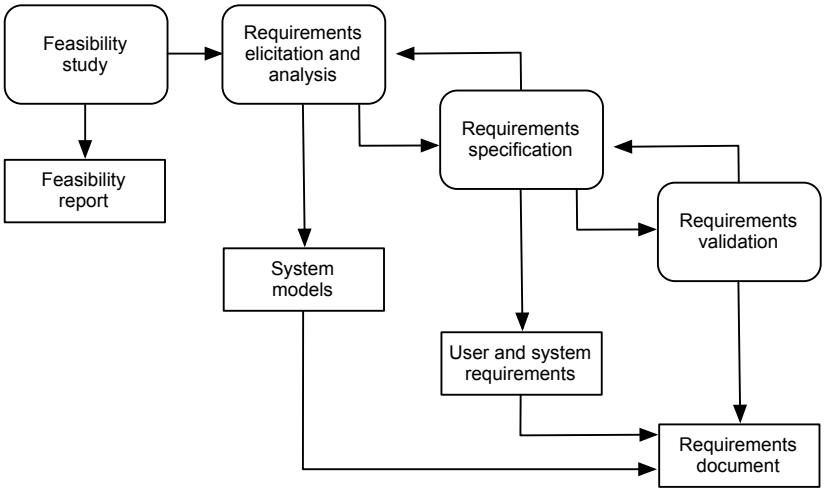


Figure 7: Sommerville [170] RE process model

Feasibility study is performed to decide weather or not it is worth carrying on with the development. The system should contribute to the overall objectives of the organization and be possible to implement with the current technology and within the given cost and schedule constraints.

Requirements elicitation and analysis starts with gaining application domain understanding and then moves on to gathering (eliciting) requirements from different sources. The main sources of information are usually the stakeholders of a product. The term *stakeholder* is comprised of any person or group who will be affected by the system, directly or indirectly, e.g., customers, end-users, engineers producing and managing the product, business managers, and third party bodies such as regulators. Within the elicitation area there are many elicitation techniques, for example (see Lauesen [115] for more techniques and detailed descriptions):

- *Interviewing* is the most effective and commonly used elicitation technique for gathering requirements [55]. Interviewing is a good elicitation technique for gathering knowledge about the present work and problems in the domain [115]. Although interviewing may not be a good option for identifying goals and critical issues, other information, such as opinions about what is realistic and where potential conflicts may lie, could be elicited.
- *Observation* can be a good way of gathering how things are done instead of asking stakeholders what is done and how. Moreover, the context and environment in which the stakeholder works may be of importance. Ob-

servation vastly improves the knowledge of current work and associated problems [115]. However, critical issues and tasks may not be captured by observations.

- *Brainstorming* can also be valuable for elicitation of requirements. Brainstorming sessions gather a group of people to let people come up with ideas where a facilitator takes notes of all ideas. An important rule of brainstorming is *not* to criticize any idea [115]. The focus of a brainstorming session should be on goals and requirements for the new product to be developed. Other similar group elicitation techniques include group interviews and joint application design.

Next, the requirements are classified, and conflicting views among stakeholders are resolved as some requirements are more important than others. In addition, different stakeholders have different power over the decisions being made [48], [112]. Another activity here is usually prioritization of requirements to ultimately decide which requirements should be included in the requirements specification. For an overview of prioritization techniques, see [12], [22].

Requirements specification involves documenting the elicited functional and non-functional requirements in detail (see Section 1.4 for requirements classification). Requirements can be documented in several forms, e.g., use-cases, requirements modeling [123], and formal specifications [137], but specification in natural language is most common. Moreover, the specification may include purpose and scope of the product, user characteristics, and development constraints.

Requirements validation is in the final stage of RE and involves validating the requirements, i.e., check the requirements to certify that they represents the description of the system which the customer wants. Moreover, requirements validation involves finding problems with the requirements. Requirements validation can be performed with different techniques, for example:

- Reviews and inspections can be used to identify defects in the requirements, assure traceability, and improve quality.
- Checklists can be used to assure that everything is there, check the structure of the requirements specification, and to identify missing parts and/or inconsistencies [115].

Requirements Management is an ongoing process lasting the entire development lifecycle, and is the process of managing changes to a system's requirements. During the development of software systems, requirements change. One reason is that stakeholders' understanding of the system and what it should do constantly

changes. Furthermore, new requirements inevitably emerge and need to be analyzed, and decisions whether or not a change should be accepted must be made.

Requirements management is the process of understanding, reviewing, analyzing, controlling, and communicating to all stakeholders changes to a system's requirements. A part of the management process is keeping track of individual requirements and maintain links between dependent requirements.

Market-Driven Requirements Engineering

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

In customer-specific development, the product is ordered by a specific customer and the supplier develops and maintains the product for that customer. A requirements specification and a contract are negotiated to specify what the supplier shall deliver. The customer-specific requirements engineering process, thus covers the four activities of requirements engineering proposed by [170].

In market-driven development, the software product is developed for an open market instead of a single customer. The market-driven requirements engineering process consists of the same four activities as in Figure 7. In addition, the MDRE process consists of specific activities such as release management and market analysis [142]. Moreover, MDRE is often under the pressure of competitors' products and the evolvement of the market and product [142]. The characteristics of market-driven development are described in [121], [138], [160].

Market-driven requirements engineering is in many ways similar to customer-specific RE. Although many practices and necessary skills are the same, there are several crucial differences between customer-specific RE and MDRE which need to be recognized to fully understand the challenges faced in a market-driven environment. According to [160], there are differences of such kind that theses need to be made explicit. However, 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 wanting to pay for tailoring of the products. The major differences include the characteristics of stakeholders [159], release planning and managing the constant flow of new requirements [37], [138].

The main distinguishing feature that separates MDRE from customer-specific RE is that there is no customer, but rather a market(s) consisting of any number of customers, which influences other aspects of MDRE such as elicitation and analysis. Eliciting requirements from the market (potential users, an imagined group of people who may fit the profile of the products users) is another major difference between MDRE and customer-specific RE [138], [161]. The elicitation of requirements is mainly managed through marketing, users groups, and trade pub-

Table 1: Examples of stakeholders that generate requirements [141]

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

lication reviews [37]. However, certain entities within the market can be identified, for example, key-customers may have the possibility to put forward requirements directly to the developing organization due to their importance. Table 1 provides examples of stakeholders that generates requirements, where some stakeholders may be counted in millions (e.g., consumers), while others may be counted in hundreds. Figure 8 gives a simplified overview of different types of requirements and their relations.

Many requirements are *invented* by the developers [138], e.g., based on product vision, to add new and unique features to a product. The developing organization is the main stakeholder, and hence decides which requirements should be included in the product. However, in order to keep, or increase market shares, the requirements that most customers want need to be selected, which puts an emphasis of marketing in the market-driven development environment [57].

The characteristics of MDRE include, e.g., that the mass market product often has a lifecycle with several consecutive releases and it lasts a long as there is a market for it. Therefore, release planning is a major determinant of the success of a product [34]. Within market-driven development, *time-to-market* is an important attribute [161]. If the product is not released to the market at the right time, there is a risk of losing market shares and customers to competitors. Release planning can be described as selecting an optimal subset of requirements for the next releases that maximizes customer value taking into account available resources [34]. Due to the constant flow of new requirements from both internal and external sources in market-driven development, decisions of which requirements that should be implemented, and which are not vital for the coming release needs to be made [140]. Having accurate estimates of implementation cost and re-

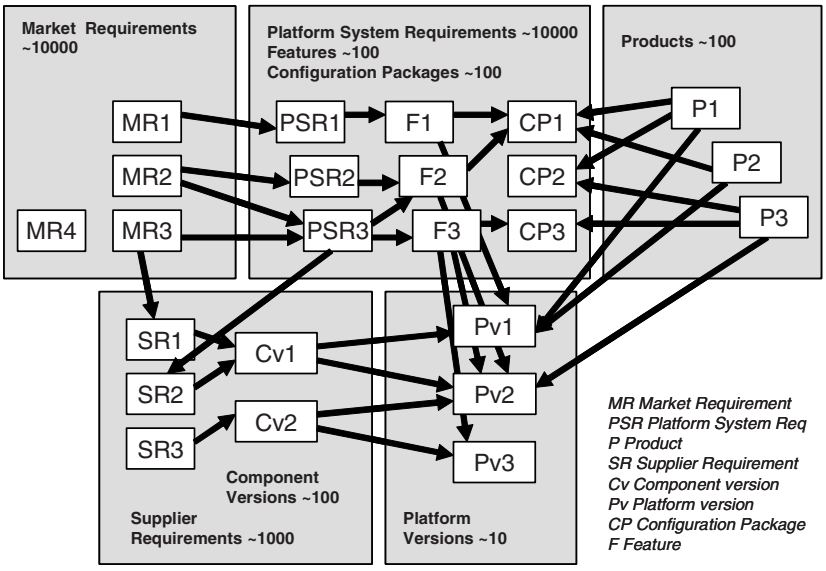


Figure 8: Orders of magnitude in different artifacts of a specific VLSRE case [141]

sources is directly related to the ability to perform requirements prioritization and release planning [144]. Regarding estimation techniques, several methods such as algorithmic models [28], estimation by analogy [167], and COCOMO II [28] have been introduced in literature. However, the use of expert judgment is the dominant estimation method employed in industry [93]. The objective of requirements prioritization is to get input for requirements selection for subsequent release planning. Some examples of prioritization techniques are listed below. For a detailed description of release planning and release planning methods, see Section 1.3.

- *Planning Game* is about ranking and grouping requirements on an ordinal scale [17]. Usually based on the criteria value, cost, and risk.

- *Pair-wise comparisons*, as suggested by [99], is based on the Analytical Hierarchy Process [157]. In this technique, pairs of requirements are compared according to their importance, and an understanding of each requirements share of the total value is provided.

- *Numeral Assignment* is the most traditional and common prioritization technique [22], [117]. Numerical assignment is based on grouping requirements into different categories, where three groups are common in practice [171].

Table 2: Overview of customer-specific RE vs. MDRE [142]

	Customer-specific RE	MDRE
Initiation	RE process is initiated and terminated based on a development project	RE process is continuous, projects are initiated as needed
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
Domain knowledge	Supplier and customer can cooperate to ensure that the domain is understood	Supplier has to be domain experts, or having internal experts
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

However, using categories like *high*, *medium*, and *low* may confuse the stakeholders, e.g., high and medium may mean different things for different stakeholders [179].

Distinguishing Features of Market-Driven Requirements Engineering

To get an overview, the main distinguishing features of MDRE in comparison to customer-specific RE is illustrated in Table 2, which is based on [142].

Market-Driven Requirements Engineering - Challenges

Several challenges in relation to MDRE have been identified in literature ([35], [36], [78], [82], [105], [117], [138], [144]). A selection of the identified challenges are discussed below.

- **Communication gap between marketing and development**
  Communication between different parts of the organization, especially between marketing and development is seen as a challenge in MDRE [105]. Both marketing and development are substantial requirement sources in themselves; however, their views of specifying requirements differ. While marketing's view of specifying requirements is to write down future ideas of functionality, development expects clear and detailed requirements. Moreover, it is important to provide the sales department with sufficient information before the sales staff promise certain functionality to customers.

- **Requirements interdependencies**
  Requirements interdependencies (a.k.a. dependencies) are usually used for bundling related requirements that need to be implemented together [105], [35], which influences requirements selection and release planning. Another challenge related to requirements interdependencies is duplicated requirements, i.e., the same requirements comes up several times, but in different shapes [105]. According to [36], the most important interdependencies types to identify in MDRE are value related, e.g., one requirement affects the value of a second requirements for a customer, and when one requirement affects the cost of implementing another requirement.

- **Constant flow of new requirements**
  In market-driven product development, a constant flow of new requirements must be dealt with [105]. Moreover, the flow of requirements is not limited to, e.g., one project, but rather continuos in nature [78]. It is important to gather all new and potentially valuable requirements, which are generally generated by multiple sources, both *internal* (e.g., engineers) and *external* (e.g., customers) [78], [138]. The large amount of requirements may be a potential threat (requirements overload [82]), hence, efficient means to manage the constant flow of requirements is important.

- **Inventing requirements**
  Requirements can either be invented by the developing organization or based on actual customer (the market) needs/requests. The trade-off between an organization's own innovated requirements and requirements suggested by the customer is a challenge [105], and may have consequences. For example, in [105], the results show that a risk for a newly started technology-focused company is that the focus is on having some new technology and that there is no need to think of the customers. The consequences may be that the customers are unsatisfied, hence the company may not sell any products. On the other hand, only considering the customers' needs/wants may lead to missing innovations that could potentially generate great value in the future.

- **Requirements Prioritization**
  Another challenge in MDRE is related to requirements prioritization [35], [117], [144], especially when information form different market segments needs to be collected, combined, synthesized, and trade-offs between their priorities should be made [35], [144]. Moreover, only a limited number of requirements can be implemented in one product, but the product should; however, meet customers' expectations and reach the markets in time [117].

- **Release planning**
  Estimation and prioritization activities are pre-requisites for release planning taking time-to-market into account. Release planning requires consideration of may different aspects, such as customer and user value, development cost, and marketing positioning. Furthermore, even more aspects need to be considered for embedded systems, e.g., component size and availability [105]. For a more detailed description of challenges in release planning, see Section 1.3.

1.3 Release Planning

In software engineering, humans make decisions based on both explicitly and implicitly known objects and constraints. Any computational technique, in isolation, is unlikely to provide meaningful results since only a small part of the reality can be captured in these techniques [128]. Humans, such as domain and solution experts, are more likely to address hidden factors that are part of the human decision-making [128].

Release planning, which is both a cognitively and computationally difficult problem [128], is classified as a wicked problem [34] since different kinds of uncertainties make it difficult to formulate and solve the problem. Moreover, the objective of release planning is to "maximize the benefit"; however, the difficulty lies in how to give a measurable definition of "benefit" [158].

Before the release planning process starts, a product roadmap can be used to document and communicate the plans for future releases [111]. Regnell and Brinkkemper [142] define 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 [162], and the one used in release planning is the Product-Technology Roadmaps, where the purpose is to map and align efforts towards a common goal. A roadmap should communicate several aspects such as themes of a certain release (e.g. improving quality, performance), goals, and milestones (for releases). Roadmaps can be seen as a concentration of the strategies for certain software product that depicting the long-term plans. Not only should the product strategies reflect current market and customer priorities, but also the goals that are set for a certain software product. The release planning process of selecting requirements for a certain release should be performed within the defined strategies of the product [12], [78].

In incremental software development, one key question is to decide which features can be offered in which release. These decisions depend on customer needs, technological constraints, and the resources and time frame available to implementing the features [5], [151]. In addition, these decisions are dynamic in nature as the features are under continuous change [172]. The process of selecting an optimum subset of features to be delivered in a release where given constraints is considered, is called strategic release planning [4], [153].

Release planning is the process of planning for the next release of an evolving product [151], applying various types of upstream decision-making (decisions that are made before the requirements are settled and the implementation phase starts [181]) that combine market considerations with implementation concerns [143]. Release planning involves activities such as selecting what features and requirements should be in a certain release, when it should be released, and at what cost [176].

Software release planning is conducted at two levels, *strategic* and *operational* [6]. It is important to have a good release plan on both the strategic, as well as the operational level [133]. Not having a good release planning process may cause late delivery, unsatisfied customers, budget overruns, and decreasing competitiveness [5]. Hence, release planning is a major determinant of the success of a product.

In strategic release planning, managers decide what features to implement in which release to meet technical and resource constraints to achieve maximum customer satisfaction. Operational release planning takes place after the strategic release planning process with the focus of deciding the allocation of developers to tasks to develop the selected features. In addition to these two levels, Al-Emran et al. adds a third level, performing dynamic re-planning on the operational level, which involves decision-making about re-allocation of developers due to, e.g., resource changes, changed requirements, and wrong assumptions of features value [5].

The release planning process should be continued as long as the software product evolves. Figure 9 (which is adapted from [151]) illustrates the release planning process (at strategic level) as part of an ongoing product evolution. In Figure 9, the planning is conducted for the next three releases for a certain product.

Features (an abstraction from requirements) are input to the release planning process and needs to be prioritized. Requirements prioritization is the process of assigning a priority to the features, which may be done by multiple stakeholders that need to consider multiple criteria [151]. Requirements prioritization is described in the following section.

Requirements Prioritization

A product's quality is often determined by the ability to satisfy the needs of the customers/users [24], [165]. All stakeholders and their requirements need to be identified, as well as their conflicting preferences and expectations [101]. When

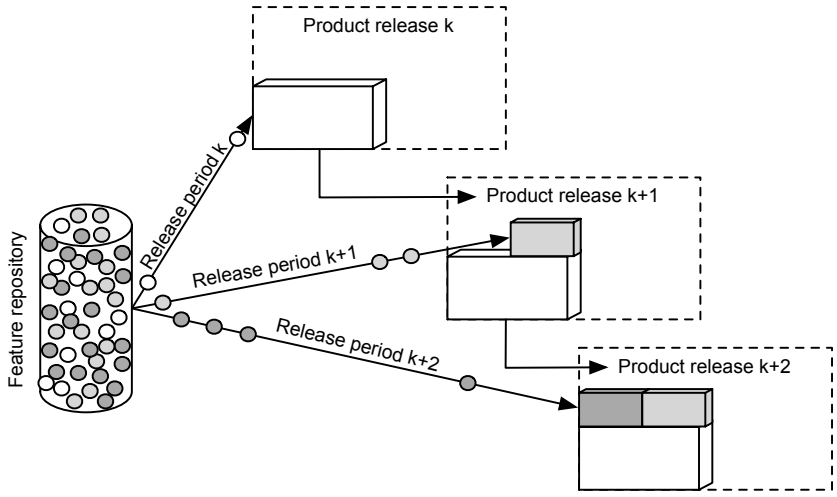


Figure 9: Release planning of selecting features to releases [151]

developing a software product for an open market, it is not possible to involve all stakeholders to prioritize requirements. Conflicting priorities between stakeholders are an issue that is addressed by many software product managers [22]. In these situations, it is important to handle different stakeholders in a structured way. Regnell et al. suggest adjusting each stakeholder's influence by prioritizing different aspects. Which aspects depend on the strategy that is most suitable in the current market segment [144].

In most software product development, there are more candidate requirements than are possible to implement within the time and budget constrains [21]. 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 [102]. Moreover, other issues to consider during the requirements prioritization are, importance to users and customers and implementation order [118]. 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 [78], and deciding on what level of abstraction should be used can be difficult. In small-scale or even in medium-scale requirements engineering [141], it may be possible to prioritize requirements on a low level of abstraction. However, in very large-scale requirements engineering [141], there are often too many requirements to prioritize. Regnell et al. suggest grouping the requirements to make the prioritization easier [144].

Dependencies have a large impact on requirements prioritization, which makes the requirements prioritization process even more complex when including qual-

ity 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 [45]. In addition, quality requirements can be in conflict with each other, therefore, trade-offs need to be made.

Although there are several prioritization techniques introduced in the literature, requirements prioritization in industry is often performed informally [117]. A selection of requirements prioritization techniques are summarized in Table 3. For a more detailed description of requirements prioritization techniques, see [22]. Karlsson et al. evaluated different methods for prioritizing software requirements involving pair-wise comparisons [103]. The study concluded that the Analytical Hierarchical Process (AHP) [157] is superior but also time-consuming. In addition, AHP assumes that requirements are independent, even though that is seldom the case [145]. Karlsson [99] 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.

Release Planning Methods

There are several release planning methods in the literature, varying from informal approaches such as planning games in agile development [49] to more rigorous and formal methods as described in [13], [79], [104] [154], [158]. A selection of release planning methods is presented below. For more details regarding release planning methods, see e.g., [91], [151], [173].

Svahnberg et al. identified 24 methods for strategic release planning, where 10 methods are extensions of others, thus 14 original methods were identified [173]. Of the 24 identified methods, 16 are related to the EVOLVE-family (EVOLVE [79], EVOLVE+ [152], EVOLVE* [154], Evolutionary EVOLVE+ [128], S-EVOLVE* [158], EVOLVEext [153], and F-EVOLVE* [124]). The EVOLVE family is illustrated in Figure 10.

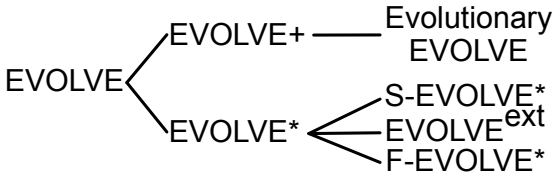


Figure 10: The EVOLVE-family [173]

Table 3: A selection of requirements prioritization techniques

Technique	**Description**
Planning game [17]	Grouping requirements on an ordinal scale using the criteria value, cost and risk
Cumulative voting [116]	Based on assigning fictional money to requirements. When the money has been distributed, the requirements are ranked on a ratio scale
Numerical Assignment [99]	Based on grouping requirements into different categories, where three groups are common in practice. The result is presented on an ordinal scale
pair-wise comparisons [103]	A technique where all pairs of requirements are compared using the criteria cost and value. Based on the Analytical Hierarchical Process (AHP) [157] and the result is on a ratio scale
Cost-value approach [102]	A prioritization technique based on AHP that uses a two-dimension graph that displays the requirements value against its cost
Quality Function Deployment [100]	Is a comprehensive, customer and user oriented approach. The QFD process starts by organizing the project, including the formation of a cross- functional team, followed by the establishment of relationships among requirements and then prioritization

EVOLVE [79] is an evolutionary and iterative method that looks ahead for more than one release. The method balances stakeholders' conflicting opinions to achieve the highest degree of satisfaction with the available resources. EVOVLE generates more than one alternative for a release where each alternative represents a trade-off between stakeholder expectations and the total benefit.

EVOLVE* [154] is a hybrid intelligent framework with the objective to create synergy between computational intelligence and the knowledge and experience of human experts. The method generates an optimal feature assignment to different releases that maximizes stakeholder satisfaction by balancing the trade-offs between release time, effort and value.

Software product release planning through optimization and what-if analysis [3] is a method that applies mathematical programming to provide a solution for the next release problem. The method's main planning criterion is projected revenue of the features, and different types of dependencies, such as implication and exclusion, are taken into account. The method emphasize on different scenarios that are related to number of teams needed to implement the feature.

Combining optimized value and cost with requirements interdependencies [34] selects requirements based on the trade-off between a requirement's cost and value, while considering the requirement's interdependencies.

Optimizing value and cost [94] is a cost-value requirements analysis method that uses mathematical programming, which is an improvement of the cost-value approach by [102] that uses AHP. Optimizing value and cost applies a rigorous algorithm to balance the cost and value of the requirements to decide which requirements should be included in the next release.

The **REPEAT** (Requirements Engineering Process at Telelogic) process [35] 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 [140].

The incremental funding method [58] aims at delivering functionality in chunks of customer-value. The method uses a single value function, namely Net Present Value to determine how the next release of selected features affects the potential income for the software development organization. This method is focused on the maximization of the overall financial value.

An approach using integer linear programming [2] is based on the assumption that the best release is the one that provides the maximum projected revenue compared to available resources in a given period of time. The approach takes

candidate requirements, estimated revenue for each requirement, and available resources as input. Moreover, requirements dependencies are taken into account.

In Saliu and Ruhe [158], seven different release planning methods using algorithms are compared and evaluated. The evaluated methods are: estimation-based management framework for enhancive maintenance, incremental funding method, cost-value approach for prioritizing requirements, optimizing value and cost in requirements analysis, the next release problem, planning software evolution with risk management, and EVOLVE*. The main difference between the methods is in how many properties that are considered. In addition, three main deficits in the evaluated methods were discovered: (1) no major focus on system constraints, (2) not enough decision support tools that are fully developed (except for Release-Planner, which is a tool based on the overall architecture of EVOLVE*), and (3) largely focused on ”fixed release intervals”.

Release Planning - Challenges

Several challenges in relation to release planning have been identified in literature (e.g., [15], [91], [151], [155], [158]). A selection of the identified challenges are presented below.

- **Size and uncertain information**
  Release planning is a very complex and challenging problem due to its size in terms of, number of involved stakeholders, the variety of variables that need to be considered, competing objectives and different types of constraints, and the uncertainty of the information it is relying upon [151], [158]. In addition, a proper understanding of the planning objectives and the important stakeholders' feature preferences is needed [155].

- **Prioritization**
  A review [91] of existing literature identified prioritization as one of the main challenges in release planning. To determine priorities of requirements has proved to be difficult, and coping with multiple aspects, such as importance, cost, time, and risk, has turned out to be difficult in practice [91]. One issue faced by decision-makers is, which aspect is important when prioritizing requirements. Moreover, the number of aspects affecting a requirement's priority seem to expand in software product companies compared to companies in project business [117]. Furthermore, to be able to collect ”real” information on the important aspects has been difficult [118].

- **Requirements interdependencies**
  Another release planning challenge found in [91] is requirements interdependencies, which tend to grow in time. The complexity of requirements interdependencies is even more difficult and dramatic when products are

introduced to new markets or customer segments [92]. The reason is that customers from different geographical locations, or from different market segments, often have different preferences of a product's functionality [91]. In addition, customers at different locations often have different technological maturity levels.

- **Problem definitions and viewpoints**
  Changing problem definitions and viewpoints are challenges in the release planning process [91]. For example, the criteria for what a successful solution is keep changing all the time. Reasons for these changes include, (1) constant evolving technologies, which enable new possibilities, (2) changes in regulations, and (3) competitors, or other market changes, force the software product company to change its focus [91]. Under these circumstances, it is important to react quickly and efficiently in order to exploit the opportunities of new technologies, which is a driving factor for a software product's success [151].

1.4 Quality Requirements

Similar to RE, there are several definitions of the term "requirement" in the literature. A general definition of the term requirement is given by Singer [169]: *a requirement is a portrait of a user's needs*. The definition by Singer [169] excludes all stakeholders except the users; however, the definition includes that requirements can both be *explicit* as well as *implicit* (aka tacit requirements [115]). Explicit requirements are requirements that stakeholders can express, while implicit requirements are requirements that stakeholders may not know they need, or requirements that are obvious to stakeholders so they take them for granted.

Requirements engineering should focus on *what* is needed, and leave the *how* for the designers. This is reflected in the definition of the term requirement by Davis [56]: *a user need or a necessary feature, function or attribute of a system that can be sensed from a position external to that system.* Kotonya and Sommerville [112] have a similar definition of what requirements are, *defining what the system is required to do, and the circumstances under which it is required to operate*.

Distinctions between different requirements are found in the literature. The classification of requirements is discussed below.

Classification of Requirements

In the literature (e.g., [112], [113]), different types of requirements are discussed, and often classified as functional or non-functional requirements. However, in a study by Olsson et al. [130], 14% of all requirements in a requirements specification from a case study were classified as both functional and non-functional requirements. That is, when non-functional requirements and functional aspects

are combined and intermingled in the same requirement. These non-functional requirements, including requirements that are a combination of functional and non-functional requirements, are subsequently called quality requirements (QR).

The definition of the term *functional requirement* (FR) has broad consensus in the literature. Functional requirements are characterized by the focus of what is needed, e.g., *"what the product must do"* [146], and *"what the system must do"* [170]. However, there is no consensus about how to define the term *quality requirement*. There are several definitions of QR in the literature, for example, Glinz [73] compiled a list of 13 selected definitions of QR. Table 4 provides an overview of a selection of QR definitions which includes a subset of the definitions in [73]. For a more detailed discussion about definitions and classifications of QR, see to [40].

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 [41]. Neither is there a universal classification of quality requirements characteristics, and different people use different terminologies [41]. In 1976, Boehm et al. [29] 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, in 1985, Roman [149] classified quality requirements into several classes such as performance constraints, lifecycle 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 [170]. 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 [41]. In addition to the classifications of quality requirements, several standards have been published, such as the ISO/IEC 9126 [88], McCall and Matsumoto [125], and IEEE 830 [10]. 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 5). Standards for quality requirements are described in [115] and [175].

According to [73], the problems with QR lies in their *definition*, *classification*, and *representation*. Glinz [73] propose a faceted classification (see Figure 11), which separates the concepts of *kind*, *representation*, *satisfaction*, and *role*, to overcome the classification and definition problems of QR. In the faceted classification [73], requirements are divided into four concerns, *functional requirements*, *performance requirements*, *specific quality requirements*, and *constraints*. The reason that performance requirements are a concern of its own is that performance requirements are typically treated separately in practice [73].

Table 4: Definitions of quality requirements

Source	Definition
Wiegers [179]	A description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behavior
Lauesen [115]	Quality requirements specify how well the system performs its intended functions
Mylopoulus et al. [126]	"... global requirements on its development or operational cost, performance, reliability, maintainability, portability, robustness, and the like. (...) There is not a formal definition or a complete list of nonfunctional requirements."
Pfleeger [136]	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
Jacobson et al. [90]	A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability. A requirement that specifies physical constraints on a functional requirement.
Sommerville [170]	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
Davis [56]	The required overall attributes of the system, including portability, reliability, efficiency, human engineering, testability, understandability, and modifiability
Glinz [73]	A non-functional requirement is an attribute of or a constraint of a system
Thayer and Dorfman [175]	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
Kotonya and Sommerville [112]	Requirements which are not specifically concerned with the functionality of a system. They place restrictions on the product being developed and the development process, and they specify external constraints that the product must meet

Table 5: 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

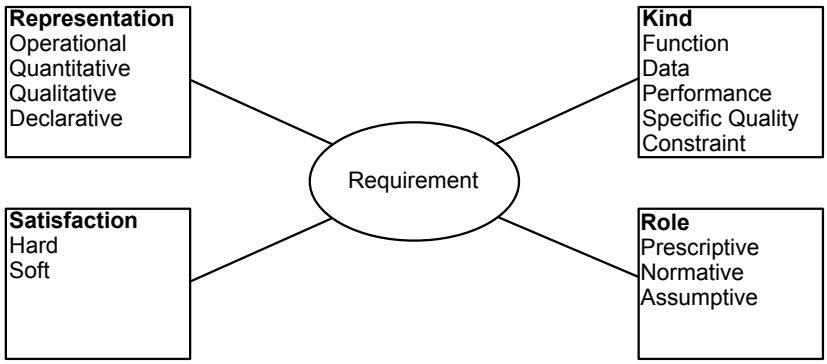


Figure 11: A faceted classificaton [73]

Importance of Quality Requirements

The body of research that currently exists in the RE and SE arena is focused on the notions and techniques for FR, even though the functionality is not useful or usable without the necessary quality characteristics [40]. For software product to be successful, it is not enough to fulfill the FR, for example, even if the software product works, and all FR have been implemented, the product may be difficult to use, or too costly to maintain [62]. Moreover, when it comes to customer satisfaction, end-customers are often dissatisfied with software quality [95]. Not dealing, or ineffectively dealing with QR may lead to more expensive software and longer-

time-to-market [54], or in worst case, failures in software development [32], [68]. For example, In 1992, the London Ambulance Service made newspaper headlines after their implementation of a Computer-Aided Dispatch System broke down, which lead to information system failure [69]. Two key issues were identified for the failure, lack of robustness and poor performance [68]. This is an example to illustrate the potential size, cost and extent of the impact of unsuccessful management of QR. It only takes a small system to crash, or not perform to the required standard of quality to trigger a whole network of interrelated systems to experience failure. In addition, studies (e.g., [33], [53]) 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 [41].

Another aspect of the importance of QR is illustrated in Figure 12. The general trend in the embedded system industry is a tremendous shift from isolated software product developers that compete, to a landscape with more complexity in the relation between different players on the market. They both collaborate, and compete at the same time and the most critical requirements in this situation are often QR such as performance and reliability. The ability to develop a software product that meets customers' requirements, and offers high value to both the development company and the customer, increases the likelihood of market success substantially, thus QR can be seen as a key competitive advantage [9], [15].

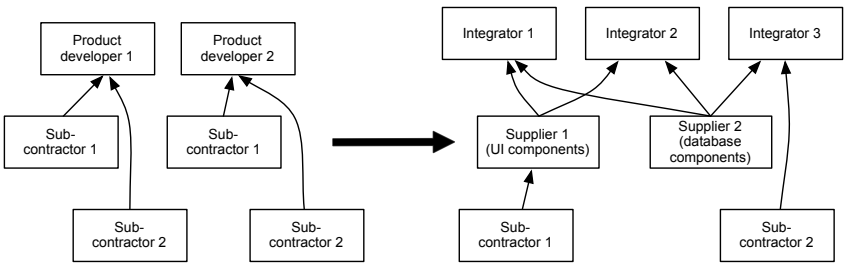


Figure 12: General trend in the embedded system industry

A third aspect of why QR are important is dynamic interdependencies, i.e., quality requirements, which may influence most (if not all) parts of the functionality as well as other QR of the product [105]. In addition, QR interdependencies may impact the order of development and the design of the product. Due to the global nature of QR, they are hard to build into a design and are often treated post facto in terms of metrics that are applied to the final product [66]. 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. The ability of a software-intensive system to meet a set of QR is to a large degree depending on the software architecture [38], and the SA thus constraints the achievement of various QR [16]. Consequently, QR are a driving force for architectural design [16]. Despite their importance, QR are often discovered late in the development process in an ad-hoc fashion, which may lead to problems such as architectural solutions failing to take into account critical QR, and systems may fall short of meeting users' real needs [46]. Therefore, one of the most critical tasks for a software architect is to create a design that can meet the QR that are vital to the success of the software product [70].

Difficult to Manage Quality Requirements

To be able to handle QR at all, quality requirements need to be gathered in the first place, but the general opinion is that QR are difficult to capture. Borg et al. [30] investigated the difficulties in managing QR at two development organizations, the results showed that QR are discovered too late, if discovered at all. The next problem with managing QR is the specification, QR are written with vague words, for example, *"the system shall be easy to use"*, which makes the requirement impossible to test.

Another difficulty is the nature of QR. Chung et al. [41] 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 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.

Most of the attention in both RE and SE has been centered on notions and techniques for defining and providing the functions of a product [40], [112], hence QR are only treated as technical issues related to the design and testing of a software product [40]. The reason most of the existing RE methods and approaches do not adequately cover QR is because it is difficult to manage QR [112], for example, certain QR are unknown at the requirements phase, that QR tend to conflict each other, and difficulties in determining when QR are optimally met. Despite the difficulties in managing QR, and lack of focus in the RE and SE arena, there are several methods and approaches that address the treatment of QR. These methods and approaches are presented below.

Managing Quality Requirements

There are several methods and techniques in the literature that address the handling and managing of QR in various ways and areas. A selection of these methods are

presented below.

Elicitation methods
Elicitation methods support stakeholders to reason about, identify, and negotiate QR. These methods rely upon brainstorming or the use of checklists. A selection of elicitation methods for QR is presented below.

- A method for eliciting, analyzing, and tracing QR using a language extended lexicon is proposed in [54]. The method is based on the use of a lexicon that will anchor FR and QR models, and drive QR elicitation. The method 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 quality perspective, which adds the QR to the created lexicon. Fourth, the functional and quality perspectives are integrated. The elicited QR are then integrated into UML (Unified Modeling Language).

- An elicitation method where functional use cases are created, and then identify associated QR by the use of a checklist is proposed in [61]. The aim of the method is to achieve a set of measurable and traceable QR. The main features of the method include, a process for common treatment of high level QR, quality models that capture experiences with QR, detailed elicitation guidelines in terms of checklists and prioritization questionnaire, documentation guidelines, rationales to justify QR, and requirement management support in terms of dependency analysis.

- A similar use case approach to [61] is proposed by Kaiya et al. [97]. However, Kaiya et al. use the goal-question-metric (GQM) model to explore quality requirements and their interdependencies.

- Misuse-oriented quality requirements engineering (MOQRE) [131] is a method for eliciting and prioritizing QR. One outcome of MOQRE is a misuse tree which shows the relationships between quality goals, misuses, countermeasures, and business goals. The misuse tree helps in making dependencies explicit, which can be exploited during the prioritization process.

Modeling and analysis methods
In addition to elicitation methods, several QR modeling and analysis techniques are proposed in the literature.

Goal-Oriented methods focus on the actual software development process where the software product's goals are the focus. Thus, goal-oriented methods can be seen as process-oriented approaches. Examples of goal-oriented methods includes: The NFR framework [41], Kaos [114], and i* [186]. For a more detailed overview of goal-oriented methods, see, e.g., [113].

The NFR Framework is one of the most comprehensive method for QR. The method defines quality goals, potential implementation solutions, and interdependencies between QR. The important quality goals are decomposed by the use of the softgoal interdependency graph (SIG) [41], [45] using *AND* and *OR* refinement. In addition to the NFR framework, techniques such as Kaos and i* provide an environment for the analysts to model QR and evaluate their constraints and tradeoffs. Both Kaos and i* use a structured approach for brainstorming and documenting QR; however, there is no guarantee that the important stakeholders' concerns will be included in the goal models. While the NFR Framework is a framework towards satisfying goals, the Kaos method concentrates on goal satisfaction and building complete requirements (including QR) models. In the Kaos method, goals are identified and refined, and objects and actions are identified based on the refined goals. The i* framework is applicable in requirements engineering as well as in the business process modeling.

Evaluating QR of the software architecture. The ability of a software product to meet a set of QR is depending on the software architecture (SA), thus the SA constraints the achievement of various QR. In the literature, there are various SA design methods, for example:

- Architecture Trade-off Analysis Method (ATAM) [108], the most well-known and used method for architectural analysis [60], is developed to find trade-offs among QR that affect each other at the architectural level. For example, high performance may be in conflict with high flexibility since typically incorporating flexibility has the price of lower performance. ATAM identifies high priority QR, which are business critical and develop a SA concept that can support the QR.
- Cost-Benefit Analysis Method (CBAM) [11], [107] is built upon ATAM. CBAM elicits scenarios expressing important QR and prioritize the scenarios using a 100-point method, where the top 50% scenarios are analyzed in more detail. Each scenario is estimated in terms of their worst-case to best-case quality response level, and architectural strategies are developed. Finally, cost and schedule estimations are performed for each architectural strategy where the return-of-investment (ROI) is computed. The architectural strategy with the highest ROI that can fit within the budget is selected for implementation.
- Quality Attribute Workshop (QAW) [14] is similar to ATAM in terms of eliciting business critical QR. QAW explicitly captures the QR that have the largest impact on the SA.
- Scenario-Based Architecture Analysis Method (SAAM) [106] uses scenarios for evaluating quality attributes, and is applied to a final version of the SA, but prior to the detailed design.

- Quality Attribute-Oriented Software Architecture Design Method (QASAR) [31] is a method for software architecture design that employs explicit assessment of, and design for QR of a software product.

Measurement and metrics
To measure and verify the performance of a software product before it is released to the market/customer is important, thus the ability to specify QR that are measurable is essential [89]. In the literature, only two methods for specifying measurable QR are empirically evaluated [26], the QUPER model (for details about the QUPER model, see Paper IV, V and VI) and the Gilb style method. The Gild style method, which is an adaption of the Planguage [71] method was introduced at a telecommunication company [89]. To make QR measurable, 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) were used. The Gilb style method has concepts for identifying measurements for the supplier's current products and what the market expects. In addition, the method specifies QR by using an interval estimate.

Besides these methods, different standards to ensure the quality of all software products exist. The IEEE Standard for a Software Quality Metrics Methodology [83] is an approach for applying metrics on different levels. The method establishes QR and identifies, implements, analyses, and validates the measurements, and it spans over the entire software lifecycle. In addition, the ISO/IEC 9126 [88] is an international standard for quality requirements where the aim is to ensure the quality of all software products. However, the "common language" proposed by ISO/IEC9126 do not have a standard interpretation as it is difficult to interpret the quality characteristics in the standard [7].

Traceability
One of the main tasks of requirements management is to assure requirements traceability from start throughout the artifact's lifetime. Traceability of QR is difficult, which is due to the fact that extensive interdependencies and trade-offs exist between them. A selection of traceability methods related to QR are presented below.

- Goal-centric traceability (GCT) is a technique to trace QR [45]. In GCT a softgoal interdependency graph (SIG) is used to model QR as goals and operationalizations. The SIG is a framework which helps developers to model QR during software development. In a SIG, goals are the QR to be satisfied, whereas operationalizations are development, design or implementation techniques that help in satisfying QR [41].

- The event-based traceability approach (EBT) was proposed by Cleland-Huang et al. in [43], [44] to handle FR and QR, and it provides a solution to the traceability update problem. The main reason for developing EBT was to provide maintenance of traceability relationships. Cleland-Huang et al. define traceability relationships as publisher-subscriber relationship. In this

relationship, dependent objects, i.e. artifacts, have to subscribe to their respective requirements on which they are dependent. Whenever a requirement change occurs, an event message is published, which is then notified to all dependent objects.

- In the scenario-based approach, traceability is established by mapping scenarios with design elements [42], [148]. Scenarios are created to trace only the interesting cases therefore they might not provide complete coverage. However, scenarios are frequently used by several architectural assessment methods like ATAM.

- A technique using design patterns for tracing QR was proposed by [80]. The technique was utilized by [42] in a model to depict traceability links between a SIG and underlying object-oriented design. This model is based on the application of pattern detection algorithms within a subset of high-level explicitly traced classes. The technique supports traceability of any QR that can be implemented as a design pattern.

Quality Requirements - Challenges

Several challenges in relation to QR have been identified in the literature ([26], [30], [54], [61], [81], [98], [105], [121]). A selection of the identified challenges are discussed below.

- **Elicitation of QR**
  To elicit and discover QR for a software product is not a trivial task. The difficulties of gathering QR is reported in [30]. Another challenge lies in how QR should be elicited. Cysnerios and Leite argue that QR should be dealt with within the scope of FR since QR require a more detailed reasoning [54]. On the other hand, Doerr et al. argue that elicitation of QR, FR and SA must be intertwined because the refinement of QR is not possible without detailed FR and SA [61]. Moreover, Hassenzhal et al. suggest to gather different aspects such as QR, design approach, and the relationships between them [81]. According to [26], there is no unified view of current QR elicitation practice.

- **Specification of QR**
  Writing understandable requirements in general, and specifically QR using natural language is an issue, particular when different stakeholders use different vocabulary [98], [105], [121]. In terms of QR specific challenges in relation to specification, challenges of writing performance requirements [121] and usability requirements [98] have been reported in the literature. The difficulties with performance requirements are, for example, the rationale is not always obvious and there are difficulties to associate performance

requirements with parts of dataflow or control flow specifications. In addition, difficulties in achieving testable (quantifying) QR, i.e. making QR well specified and quantified, is a challenge faced by practitioners [25].

- **QR dependencies**
  One challenge is related to QR interdependencies, which was identified as a major problem in Karlsson et al. [105]. 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.

- **QR in the release planning process**
  In release planning, consideration of several aspects such as customer value, market position, and architecture are required. Therefore, it is important to consider QR before applications, as the architectural aspects may change the structure of the system [105]. One issue, which also affects the release planning process is QR interdependencies (which is described above). Problems with considering quality requirements in release planning were identified as a challenge in market-driven software development [105].

2 Research Focus

Software product development is more and more commercialized as standard products. Software products may consist of both hardware and software, such as embedded products (e.g., mobile phones), or a software product can be pure software applications [174]. The software product manager plays an important role in the success of a software company's products. The software product manager manages product requirements (through MDRE, see Section 1.2), release planning (as described in Section 1.3), and launching products. In addition, the software product manager needs to understand stakeholder and market needs (part of portfolio management, see Section 1.1) and translate them into the scope of the product (see SPM in Section 1.1). 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 [15]. 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 [15]. If the product is not released to the market at the right time, there is a risk of losing market shares and customers to competitors; hence release planning is a major determinant of the success of a product. In addition, the value of a software product is related to quality requirements (described in Section 1.4) [15], and is increased in direct proportion to the advantage over competitors' products [9]. Hence, a challenging problem for an organization that develops software products offered to an open market is to set the

right quality targets in relation to future market demands and competitor products, when is the quality level a competitive advantage?

The main goal of this research is to increase the awareness and understanding of quality requirements, and to find means for improving the ability to make early estimates of quality requirements, e.g., performance requirements, in order to enhance high-level decision-making related to *release planning* and *roadmapping*. Research gaps (RG) have been identified based on literature reviews on quality requirements and release planning in market-driven software product development. We observed that:

- **RG1:** Quality requirements can be seen as a key competitive advantage in software product development for an open market. Despite their importance, it is generally acknowledged that QR are difficult to capture, often poorly understood, and generally stated in a non-quantifiable manner. To be able to improve how QR are handled, it is important to understand their characteristics and the challenges of dealing with QR in industry. Challenges associated with QR have been addressed in part in the literature, for example, elicitation and specification of QR, and managing interdependencies between QR. However, none of the studies in literature, with the exception of [25] and [30] have primarily focused on QR. Even though [25] and [30] solely focused on QR, only five respectively two companies were included in the studies.

- **RG2:** Requirements prioritization is an important part of release planning. For a software product to be successful, it is important to find the right balance among competing QR. Although several methods and tools for requirements prioritization have been developed, evaluated, and compared in the literature, requirements prioritization is seen as a challenging part of MDRE, and as a part of the release planning process. However, research on prioritization of QR is limited.

- **RG3:** A characteristic of QR is that they specify certain quality levels, and can hence often be quantified. At the same time, there is a general opinion that QR are difficult to capture and define. Moreover, another QR related problem that has been discussed in the literature is where to document QR, separated from FR, attached to FR, or even the option of treating performance requirements separately? However, no study has looked into how QR are quantified, how they are specified, or if QR generally are stated in a non-quantifiable manner in an industry requirements specification.

- **RG4:** In market-driven software product development it is important to deliver an ”optimal” subset of features and quality in a certain release, at the right time, which makes these decisions (release planning) a major determinant of the success of the product. Release planning is viewed as a challenge in SPM and MDRE, and the constant flow of requirements increases

the complexity of release planning due to its size. In addition, changing problem definitions, e.g., the criteria for what a successful solution is keep changing all the time. An especially challenging problem is to set the right quality targets (i.e., release planning of QR), when is the quality level good enough? Although several release planning methods and tools have been developed, no method, or tool, addresses quality levels and cost constraints of QR.

The identified research gaps (RG1-RG4) makes the general need for studying quality requirements in a market-driven software product development environment explicit, and in particular the specification of future quality requirements' targets for future releases.

The first research gap (RG1) is addressed in this thesis by investigating experienced challenges in relation to quality requirements by practitioners in eleven market-driven software development companies (research question 1 (RQ1) in Table 6). The second research gap (RG2) is covered in this thesis by investigating the prioritization process of quality requirements in detail at eleven market-driven software development companies (RQ2 in Table 6). Since difficulties in elicitation, specification, and quantification of quality requirements were identified as reasons for difficulties in prioritization of quality requirements, research gap RG3 is covered in this thesis as we examined a requirements specification at one case company in detail (RQ3 in Table 6). The fourth research gap (RG4) is addressed by; first, developing and evaluating a model for supporting release planning of quality requirements in close collaboration with industry (RQ4 in Table 6), second, evaluating the model in relation to improvements of high-level decision making, e.g., release planning (RQ5 in Table 6). Third, the possibility to tailor the model to different industrial environments was investigated (RQ6 in Table 6).

The resulting main research questions (RQ) investigated in this thesis are as follows:

The relationships between the research questions and the included papers in this thesis are depicted in Figure 13.

RQ1 was posed in order to discover and understand QR challenges faced by practitioners, and to select focus for the research. Among the found challenges, issues regarding the prioritization and how to get QR into projects when FR are prioritized emerged.

RQ2 aims at discovering and understanding how QR are prioritized in industry, and to what extent state-of-the-art in research has penetrated industrial practice. The difficulties of prioritization of QR were identified, where lack of well specified and quantified QR, and when are the QR good enough emerged.

Table 6: Main research questions

Research questions (RQ)
RQ1: Which challenges related to quality requirements are experienced by practitioners in the market-driven software development industry?
RQ2: How are quality requirements prioritized in the market-driven software development industry?
RQ3: How are quality requirements specified in the market-driven software development industry?
RQ4: How can the specification of quality requirements be improved when setting future quality targets in a market-driven software development environment?
RQ5: 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?
RQ6: How can QUPER be tailored to suit industrial environments?

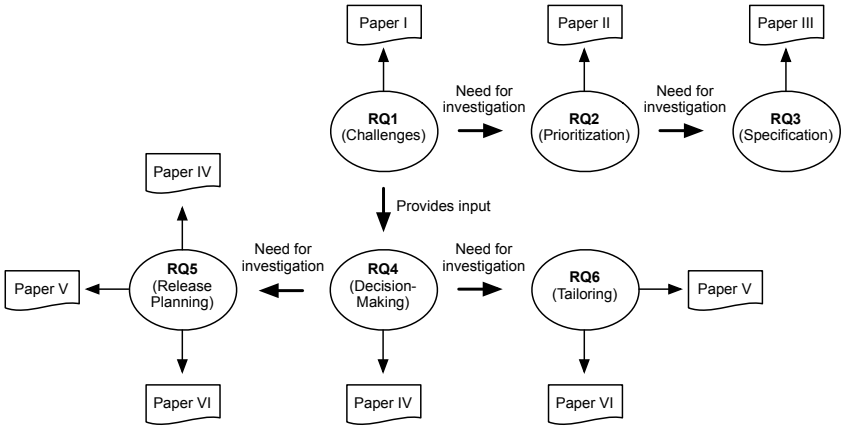


Figure 13: The relationships between research questions and papers

RQ3 examined how QR are specified in industry, how they are documented, and if QR often are stated in a non-quantifiable manner.

RQ4 was posed in order to support primarily decision-makers in taking care of and working with future QR targets in relation to market needs and competitors.

RQ5 aims at evaluating and improving the release planning process of QR by a

method (QUPER) for supporting release planning of QR. The method was evaluated in three studies, at two organizations with different characteristics in order to investigate its possibilities and limitations.

RQ6 examined how the QUPER model could be tailored to suit organizations in industry. QUPER was tailored to fit, not only two organizations in industry, but two different domains, namely, the electronic payment processing and mobile handset domains.

3 Research Methodology

In the pursuit to answer the research questions posed in Section 2, it is essential to utilize certain research methodologies as the research methodology provides the link between research questions and the data used to answer them. Thus, a methodology must be chosen that provides the necessary data to answer the stated research questions. This section gives an overview of the methodological approaches that are used in this thesis.

The research presented in this thesis employes an empirical research approach as described by [184], where findings are verified through observation and experience. The observation of human behavior and their interactions with technology in the real world (industry) cannot sufficiently be investigated by an analytical research paradigm [156]. As the research questions in this thesis have applied and practical objectives, this thesis focuses on empirical research methodologies, which are described below.

3.1 Research Design

There are two main approaches to empirical research: *fixed* and *flexible* research design [147]. 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, quantitative research requires a conceptual framework or theory to be developed before getting into the main part of the research study. Quantitative research is generally used to answer questions about the relationships between variables, e.g., quantifying a relationship or comparing two or more groups, for the purpose of explaining, predicting, or controlling the phenomena. Statistical methods are commonly used to establish or confirm hypotheses, and produce generalized findings, which is the greatest power of quantitative research. In quantitative research, the researcher needs to collect all data before starting to analyze it.

The flexible research design, which is also called *qualitative*, allows to change parameters of the design based on new information during the course of the study, e.g., change of research questions or data sources. Hence, qualitative research design may evolve 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. Qualitative research seeks to better understand and explain complex situations in their natural setting, where issues of the real world are described. Where quantitative research use statistical methods to confirm proposed theories, qualitative research use observations and inductive reasoning to build theory from the ground up. One reason for this difference is an assumption that reality cannot be divided into discrete measurable variables.

In this thesis, only qualitative research design is used (see Table 7). The contributions of this thesis are the investigation of quality requirements challenges faced by practitioners in industry, and the implementation of the quality performance model in a real-world setting. As the aim of this thesis was to improve and gain an in-depth understanding, hence qualitative research is more suitable than research in the breadth (quantitative research design).

3.2 Research Methods

There are a number of research methods for conducting empirical research in the software engineering discipline. According to Kitchenham et al. [110], the most common research methods are case studies [156], surveys [67], and experiments [184]. However, other research methods can be used, for example, simulations [168], and action research [180]. Wieringa and Heerkens [180] classifies research methods that can be used in requirements engineering research, where two of the methods are case studies and action research. The research methods of choice for this thesis were qualitative surveys, case studies and action research.

Qualitative Surveys

Surveys can be both flexible and fixed, the classification depends on the design of the questionnaire (which data is collected) [184]. Survey instruments can take four forms, self-administrated questionnaires, interviews, structured recorded reviews, and structured observations [67]. 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 population, from which a sample is selected [184]. However, in small samples, qualitative surveys are useful. Qualitative surveys concentrates on collecting in-depth information about the subjects' opinions and experiences expressed in their own words [67]. In general, qualitative surveys do not aim for a representative, or to generalize the results for the entire population.

Case Study

A case study is an in-depth investigation of a phenomena in its real-life context [185] focusing on a specific case. The cases are objects of the real world which are studied in a natural setting, i.e., real software organizations, software projects, a product, a group of people, or an individual. Case studies are typically flexible design studies [156]; however, good planning is crucial for its success. A flexible design allows to change the design based on new information gathered while executing the case study, e.g., leading to a change of research questions or data sources. Although the small sample size makes the results from case studies difficult to interpret and generalize, case studies are useful in exploratory research where little is known about an area. Data collection methods for case studies can be divided into three categories [156]: (1) direct methods (e.g., observations and interviews), (2) indirect methods (e.g., automatically monitoring software tools), and (3) independent analysis if work artifacts (e.g., requirements specifications). The analysis of the collected data of case studies ranges from purely qualitative analysis where raw data from interviewees is categorized and coded to the exclusive use of statistical inference.

The use of the term case study in software engineering research is of varying quality [156]. The reported studies range from ambitious studies to small toy examples. There are several definitions of case study research in the literature, but in this thesis, the following definition is used: *"investigating contemporary phenomena in their context"* [156].

Action Research

Action research, with its purpose to influence or change some aspects of the research focus, is closely related to case study [156]. Moreover, action research aims to improve practice, the understanding of practitioners, and the situation in which the practice takes place [147]. In action research, a researcher enters the project where tasks are performed by using the researchers method. Action research comprises of four steps [147]:

1. Plan how current practice can be improved
2. Implement the plan
3. Observe the effects
4. Reflection

After step four (reflection), the researcher evaluates the performance of the used method and draw conclusions, which may lead to improvements of the technique or method.

In literature, the degree of the researcher's active involvement in the team work in action research differs. While [180] states that it is not a practitioner's use

of the technology that is studied, instead it is the researcher's, [147] states that the collaboration between researchers and the practitioners who are the focus of the research, and their participation in the process are seen as central to action research. However, according to [156], case study is purely observational while action research is focused on the change process, which is supported by [147]. Hence, in software process improvement [59] and technology transfer studies [75], the research method should be categorized as action research [156].

In this thesis, Paper IV, V, and VI are classified as action research; however, if the degree of involvement from [180] is considered, the research method in Paper IV, V, and VI may be classified as *pilot projects* or *case studies*. On the other hand, looking at the degree of involvement from [147], and that action research is distinguishable in terms of its purpose, which is to influence or change an aspect of the research focus, the method can be characterized as action research. Moreover, according to the definition in [156], the used methods are action research since Paper IV, V, and VI focus on improving the process of supporting release planning of quality requirements, and transferring the technology of the QUPER model to industry.

3.3 Data Collection Methods

The collected data in empirical studies could be quantitative (e.g., numbers and classes) or qualitative (e.g., words, descriptions, and pictures). Although only qualitative research design is used in this thesis, which tend to be based on qualitative data, quantitative data has been collected. A combination of qualitative and quantitative data may provide a better understanding of the studied phenomenon [166].

Without proper data collection and analysis methods, the essence of the collected data may not be revealed nor possible to communicate. There is a variate of data collection methods to choose from, and the researcher's choice is dependent on the information sought after [147]. The following data collection methods have been used in this thesis.

Interviews

Data collection through interviews is important in case studies [156]. In interview-based data collection, the researcher, guided by an interview protocol, asks questions to a set of subjects about the phenomena of interests in the case study. Interviews can be divided into three types, *fully structured* (interviewee has to stick with the questions), *semi-structured* (interviewee has a guide, but can change the order and the wording of the questions), and *unstructured* (rough definitions of topics to be covered) [147]. 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 an-

swers cannot reveal. One disadvantage with interviews is that they are rather time consuming.

In this thesis, semi-structured interviews were used to allow for flexibility in the interview sessions. For elaboration of fully structured and unstructured interviews, see [147], [156].

Archival Data

Archival data refers to, for example, documents, financial records, previously collected measurements in an organization, and organizational charts [156]. Archival data collection differs from, e.g., interviews, in that it is an indirect data collection method. That is, instead of directly observing or interviewing a practitioner, data is collected from something that is produced for some other purposes [147]. While [185] distinguish between documents and archival records, [156] threat documents and archival records together and distinguish between qualitative data (e.g., documents) and quantitative data (e.g., records).

In this thesis, data was collected from written documents, namely a requirements specification.

Self-Administrated Questionnaires

Self-administrated questionnaires, which contains questions that the subjects answer by themselves, can either be mailed or answered on site in, e.g., an office [67]. Questions in a self-administrated questionnaire can have two primary forms , *closed questions* and *open questions*, where closed questions are usually preferable to open questions [147]. Closed questions provide the subjects with predefined answers, where the aim is either to test the subjects' performance (e.g., IQ tests) or to gain insight into what the subject feel or believe about something [147]. When the aim is to get an insight about what subjects feel or believe, the most common measurement scale, according to [147], is *attitude measurement*. In attitude measurement, several different types of scales, e.g., *arbitrary scale*, can be used; however, the summated rating (or *Likert*) scale is more widely adapted. The most common response categorization system is a five-level Likert-scale alternatives: 'strongly disagree', 'disagree', 'neutral', 'agree', 'strongly agree'.

Open questions are useful when issues are still unknown since they let respondents describe the phenomena as it is seen by them, instead of how the researcher believes it may look like. Moreover, some respondents may prefer to state their own views in their own words, and this may provide the researcher with quotable answers [67]. One disadvantage with open questions is that responses may be difficult to compare and interpret.

3.4 Data Analysis Methods

Data analysis for quantitative and qualitative data is conducted in different ways. Since case study and action research are of qualitative research deign, qualitative data analysis methods are commonly used [166]. However, as described in Section 3.3, quantitative data has been collected, hence statistical analysis methods have been used in this thesis. The following data analysis methods have been used in this thesis.

Content Analysis

Content analysis is a method for analyzing and interpreting data from qualitative surveys [67], but can also be used in the analysis of qualitative interviews and questionnaire data (e.g., coding open-ended questions) [147]. The focus of content analysis is to gather information and generate findings. The gathered information (content) can be any written information and different categories containing content are constructed for analysis. After the content has been gathered and categories been constructed, it is analyzed and conclusions based on the content is reported. A content analysis can be based on *inductive* or *deductive* analysis [67]. In inductive analysis, dominant themes of the collected data are looked for and identified. The researcher, by using inductive reasoning and experience, reviews the data for unifying ideas. In deductive analysis, the researcher preselects the themes and sub-themes that he/she thinks are the most likely to occur. The researcher walkthrough the collected data and records every support for the preselected themes and sub-themes. In addition, the researcher has the possibility to add new themes and/or sub-themes as such themes may emerge.

Statistical Analysis

In this thesis, two statistical analysis methods were used, namely, *descriptive statistics* and *inferential statistics*.

Descriptive statistics, such as mean values, standard deviation, and box-plots, are used to describe and present the data that has been collected in order to aid analysis [147], [184]. The goal of descriptive statistics is to get a feeling of how the collected data is distributed, and may be used before hypothesis testing is conducted. For more information about descriptive statistics, see for example [147], [184].

The objective of inferential statistics (hypothesis testing) is to reach a conclusion that extend beyond the immediate data alone (to see if it is possible to reject a certain null hypothesis), for example, trying to infer from the sample data what the population may think. Thus, inferential statistics are used to make inferences from the data to more general conditions. Test can be classified as *parametric* and *non-parametric* tests [184]. Parametric tests are based on a specific distribution of the data, that the data is normally distributed and that the data can be measured at least on an interval scale [184]. Non-parametric tests do not make the same assumption

as parametric tests, and are more general. In literature, several tests are available for parametric and non-parametric tests, for example, t-test and Wilcoxon. For more information about different tests, see [184].

In this thesis (Paper I), a non-parametric Wilcoxon rank sum test [184] with significance tested at the 0.05 level was used to judge between which sets of data there is any discernable difference and thus reduce the number of such possible differences that would otherwise be implied.

3.5 Research Classification

The results in this thesis have been reached through the use of the presented research design and methods. Each paper is linked to research questions, research design, research methods, data collection methods, and data analysis methods, which is illustrated in Table 7.

In Paper I and II, qualitative surveys are used as research methods. The reason for using a qualitative survey is that the aims of Paper I and II are to understand and describe experienced challenges by practitioners in relation to quality requirements, which is aligned with the purpose of qualitative surveys. Interviews were used as data collection method because it would best meet the objectives of the studies. Interviews help to ensure common information on pre-determined areas is collected, but allow the interviewer to probe deeper where required. Moreover, interviews provide the researcher with the ability to explain questions (if misunderstood) and to follow up answers. Considering the collected data and the aims of Paper I and II, content analysis is the most suitable data analysis method. In addition to content analysis, statistical analysis was used in Paper I since some of the collected opinions were quantified with the purpose of judging which sets of data there is any discernable difference and thus reduce the number of such possible differences that would otherwise be implied.

The aim of Paper III is to understand how quality requirements are specified, in particular how QR are quantified, and which types of requirements exist in a requirements specification from 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 2,178 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 data analysis method as it is an indirect method.

Paper IV presents the first complete version of the QUPER model, including detailed practical guidelines of how to apply QUPER in practice. The aim of Paper IV is, not only to present a complete version of the model, but also to evaluate its usefulness and applicability in an industrial environment. Paper IV is based upon previous work (Paper IX, XI, XV, XVI, and XVIII) where both case study and action research methods have been used, hence, the version presented in Paper IV uses a mixed case study and action research method. In both Paper IV

Table 7: Research classification

Paper	Research question	Research design	Research method	Data collection method	Data analysis method
I	RQ1	qualitative	qualitative survey	interviews	content analysis and statistical analysis
II	RQ2	qualitative	qualitative survey	interviews	content analysis
III	RQ3	qualitative	case study	archival data	content analysis
IV	RQ4, RQ5	qualitative	case study and action research	interviews and self-administrated questionnaire	content analysis and statistical analysis
V	RQ5, RQ6	qualitative	action research	interviews	content analysis
VI	RQ5, RQ6	qualitative	action research	interviews and self-administrated questionnaire	content analysis and statistical analysis

and the related papers, the researcher has been involved in several steps towards the development and implementation of the QUPER model. For data collection, both interviews and self-administrated questionnaires were used. Considering the collected data and the aim of Paper IV, content analysis is the most suitable data analysis method. In addition, since quantitative data was collected, statistical analysis (descriptive statistics) was used to describe the subjects' opinions about the QUPER model in relation to supporting release planning of quality requirements.

Action research was used as research method in both Paper V and VI. The aims of Paper V and VI were to evaluate the introduction of a new method and to improve the process of supporting release planning of 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 V and VI, we were involved in several steps to improve the practice of release planning of quality requirements by introducing the QUPER model. 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. Moreover, the prototype tool (Paper VI) was developed in cooperation between academia and industry, and we participated in a live demonstration of the tool at the case company. To evaluate the introduction of QUPER, interviews were used as data collection method approach because of the ability to interpret the tone of the voice, expressions, and intonations of the interviewee, and content analysis was used to analyze the collected data. In addition, Paper VI use statistical analysis (descriptive statistics) to describe the subjects' opinions of the collected data from the self-administrated questionnaire.

3.6 Technology Transfer Model

Technology transfer and industry-relevant research involves more than producing research results, it demands close cooperation and collaboration between industry and academia throughout the entire research process [75]. Moreover, technology transfer from research to industry can be, and is a real problem. According to [72], the problem can be described as, problems are formulated in academia, and solutions are put forward and validated in academia, which results in industrial inapplicability. Hence, there may be no real benefit for industry if the research is not based on problems that are identified in industry, and if solutions are not evaluated in an industrial environment prior to transferring the solution to industry [72]. Furthermore, technology transfer problems have also been identified in requirements engineering research [96].

The development of the QUPER model is based on the technology transfer model [75] in attempts to counter the technology transfer problems from academia to industry. The overall used research process in the development of QUPER is illustrated in Figure 14.

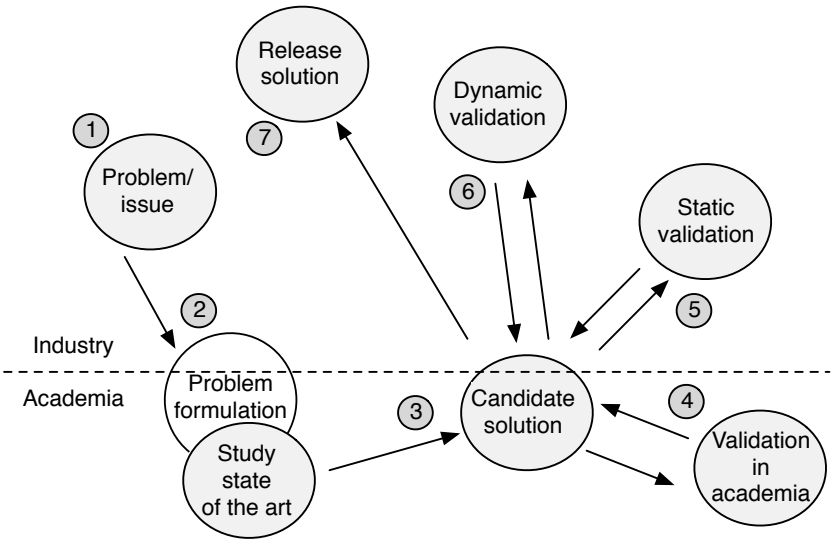


Figure 14: Overview of research approach

The QUPER model has been developed and evaluated in several stages, where several steps in Figure 14 have been repeated. For example, the candidate solution (step 3 in Figure 14) has been improved several times based on the results from different validations. Moreover, as the model has been improved, new validations (steps 5 and 6 in Figure 14) have been conducted to evaluate the new parts and improvements of the model. Below, the evolvement of the QUPER model is described.

Step 1, identify problems based on industry needs, implies that the research is based on problems and demands explicitly identified in industry. Paper IV presents the identified industry problems targeted by QUPER.

Step 2, formulate a research agenda and study the state-of-the-art. The research agenda was formulated in close cooperation with industry contact persons, and state-of-the-art was studied in order to see if solutions already existed (e.g., Section 2 in Paper IV).

Steps 3 and 5, formulate a candidate solution based on the understanding of the problem, research agenda, and state-of-the-art. In addition, an initial static validation (interviews) in industry was conducted. Paper XVIII presents the first version of the candidate solution and the results from the initial static validation.

Step 3, subsequent to the formulation and initial validation, a modification of the solution was conducted based on the previous validation and new development of the solution, see Paper XV.

Step 6, the first dynamic validation of one part (the benefit view, see e.g., Paper IV, Section 4) of the solution in industry was carried out as a small test to validate the real applicability (see Paper XVI).

Step 5, a second round of static validations was conducted in industry to validate the applicability of QUPER's cost view in an industrial environment (see Paper IV).

Step 6, a first dynamic validation of the entire solution in industry was carried out as a small test (see Paper V, which is an extended version of Paper VIII). The results of this dynamic validation gave feedback as to the solution's viability.

Steps 3 and 5, Paper IV presents the first complete version of the solution, which is based on all previous conducted validations. In addition, modifications of the solution based on new ideas and refinements were carried out in close collaboration with industry professionals. Static (interviews) validations were carried out in industry to validate the solution's applicability in an industry environment using real requirements.

Steps 4 and 6, validation in academia and dynamic validation. To facilitate technology diffusion from academia to industry, effective tools and tool support are important [135]. Furthermore, to transfer technology to practice, tool support and tool adaptation are important for the solution's scalability [75]. Paper VI presents a first prototype tool for QUPER. Moreover, an academic validation was carried out to evaluate the usability and workflow of the QUPER prototype tool. The academic validation provided valuable feedback, which was an important input to the finalization of the first version of the prototype tool. The results made it possible to catch some issues without using industry resources. Furthermore, a dynamic validation, in terms of a small test, was conducted to evaluate the tool's applicability in industry, and to enable easier adoption of the solution by practitioners and thereby to improve the technology transfer in practice.

Step 7, release the solution. As the solution matures by the previous validations, the solution is "released", i.e., available for implementation into industry practices. The available solution for implementation is presented in Paper IV.

3.7 Validity Threats

Even though the research in this thesis has been conducted with reliable and well-known strategies, and methods, the results should nevertheless be questioned and evaluated. The validity of the research demonstrates the trustworthiness of the results, i.e., how true the results are and that the results are not biased by the researchers' subjective opinion [156]. There are different ways of classifying validity threats in the literature. In general, four types of validity threats are discussed in empirical studies, for example, experiments [184] and case studies [185], [156]. Although Wohlin et al. [184] discuss threats to experiments, the classification scheme is similar to the case study threats in [185], [156]. In fact, three of the four types of threats are called the same, and have the same meaning. Although

the fourth type of validity threat is called reliability in [185], [156] and conclusion validity in [184], both are concerned with the ability of replicating the study and obtaining the same results. Furthermore, the validity threats discussed in Wohlin et al. [184] have been considered in several empirical studies which are not controlled experiments, e.g., in case studies [76], [134], and in reported surveys [77].

In this thesis, the four perspectives of validity and threats as presented in Wohlin et al. [184] are considered. The four perspectives can be summarized as follows (more detailed presentations of validity threats are available in [184], [185], [156], [147]):

- *Construct validity* is concerned with the relation between theories behind the research and the observations, i.e., choosing and collecting the right measures for the concepts being studied. The use of *multiple sources of evidence* and establish a *chain of evidence* may increase construct validity [185] and ensure that the result is an effect of the treatment.

- *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 [184]. 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 [185]. Internal validity is a large threat to case studies due to that industrial environment changes over time [185].

- *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.

- *Conclusion validity* arise from the ability to draw accurate conclusions, i.e. the reliability of the results [184]. 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 of reliability [185].

Table 8 provides an overview of validity threats observed throughout this thesis, stating and describing the threats with references to the concerned papers. Further details about the threats in the context of each individual paper is provided within the papers.

Evaluation apprehension is a threat related to the behavior of the subjects, more specifically, subjects being afraid of being evaluated. This is a threat in Paper I, II, IV, V, and VI, and was alleviated by the guarantee of anonymity as to

Table 8: An overview of selected validity threats

Threat	Description	Concerned papers
Evaluation apprehension	Subjects are afraid of being evaluated	I, II, IV, V, VI
Incorrect data	Incomplete and inaccurate data	I, II, III, IV, V, VI
Selection bias	Researcher biased in selecting subjects	I, II, IV, V, VI
Generalizability	Generalize the findings beyond the included companies	I, II, III, IV, V, VI
Confounding factors	Other factors affecting the outcome	IV, V, VI

all information divulged during the interviews, and the answer was only to be used by the researcher.

Since all studies (Paper I to Paper VI) are of empirical nature, *incorrect data* is a validity threat. In case of the interviews, the correct data was assured by recording the interviews (Paper IV, V, and VI) and taking records in the form of written extensive notes (Paper I and II). In addition, the researcher had the chance to validate the questions with the subjects lessening the chances of misunderstandings. The document analysis study (Paper III) is based on a requirements specification given to a sub-contractor of the case organization, which was fully accessible to the researcher.

A threat to validity in Paper I, II, IV, V, and VI is the selection process of subjects for the interviews. *Selection bias* is always present when subjects are not fully randomly sampled. A possible bias may be that only subjects that have a positive attitude towards QR and the QUPER model are selected. However, in Paper I, II, IV and VI subjects were selected based on their roles by a ”gate-keeper” at the different companies. In Paper V, the selection of subjects was conducted in cooperation with a manager at the case company.

The ability to *generalize* the results, i.e., in this thesis the applicability of the findings beyond the included companies, is a threat to validity in all papers (Paper I to Paper VI) as they are all qualitative studies. However, qualitative studies rarely attempt to generalize beyond the actual setting since it is more concerned with explaining and understanding the phenomena under study. Furthermore, the nature of qualitative designs makes it impossible to replicate since identical circumstances cannot be recreated. However, the fact that more than one company acknowledge most of the identified challenges in Paper I and II increases the possibility to generalize the results beyond these studies. The study in Paper III only covers one specific case company; therefore, the generalizability may be ques-

tioned. However, understanding the phenomena may help in understanding other cases and situations. For Paper IV, V, and VI, some of the problems introduced as motivation behind QUPER, could to some extent be general for organizations faced with developing embedded systems for markets. However, it is not possible to generalize the results from these evaluations based on two organizations, although from a perspective of the concepts and practical application of QUPER may give an overview of faced challenges when QUPER has been implemented.

Confounding factors are important when making inferences about root-cause relationships, which is the case for Paper IV to Paper VI in which inferences are made about the improvements of release planning for QR. The confounding factors cannot be ruled out as the studies were conducted in an uncontrolled industrial environment. A confounding effect could involve problems with learning how to use the QUPER model (Paper IV and V) and the QUPER prototype tool (Paper VI) when assessing its benefits. In addition, the use of very enthusiastic or skeptical subjects could be a confounding factor, but the selection bias is discussed above.

4 Results

The research in this thesis is presented in two parts (Part I and II), each containing three papers, which is illustrated in Figure 15. This section presents the main results of this thesis related to each paper. The discussed results are based on the conclusions form the results of the included papers. An overview of the results of the individual papers, linked to research questions and research gaps, is illustrated in Table 9.

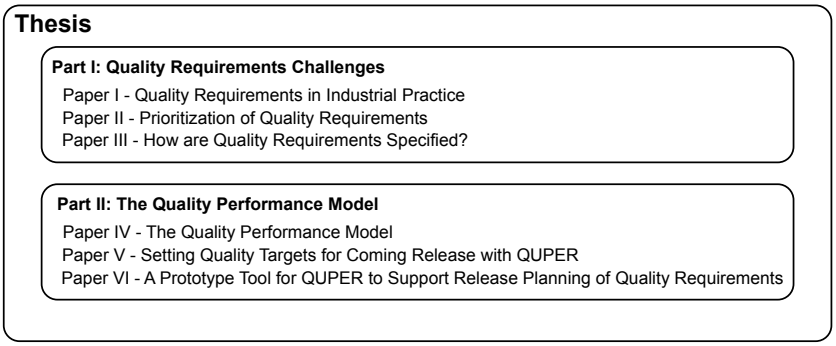


Figure 15: Overview of thesis

Table 9: Overview of results

Paper	RG	RQ	Main results
I	RG1	RQ1	The results suggest that QR are not taken into consideration during product planning and thus not included as hard requirements in the projects, making the realization of QR a reactive rather than proactive effort.
II	RG2	RQ2	In the studied cases, QR are by default seen as having a lower priority than FR, and only get attention in the prioritizing process if decision-makers are dedicated to invest specific time and resources on QR prioritization.
III	RG3	RQ3	The findings from this study suggest that methods for QR need to encompass many aspects to comprehensively support working with QR. Solely focusing on, e.g., quantification of QR might overlook important requirements since there are many QR in the studied specification where quantification is not appropriate.
IV	RG4	RQ4, RQ5	The results from this evaluation point to a relevant and feasible model that allows decision-makers to have more substance to the decisions of what level of quality to aim for in coming releases.
V	RG4	RQ5, RQ6	The results indicate that the QUPER model demonstrated usefulness in supporting early decision-making in, e.g. release planning of QR and architectural decisions related to QR.
VI	RG4	RQ5, RQ6	The prototype tool may provide a clear overview of the current market situation by the generated roadmaps, and to reach an alignment between practitioners, e.g., product managers and developers, of what level of quality is actually needed.

4.1 Paper I - Quality Requirements in Industrial Practice

Paper I aims at discovering challenges experienced by practitioners in market-driven software development. A qualitative survey was conducted at eleven embedded software developing companies. Paper I reports on findings from interviews with 22 practitioners, one product manager and one project manager from each company, involved in managing and handling QR. A number of challenging issues, both from a business-to-business and business-to-consumer perspective, as well as from a product and project perspective related to interdependencies, cost estimation, and dismissal of QR, were found. Some of the identified challenges, for example, interdependencies related to QR, are acknowledged by other studies in the literature.

Hence, some of the challenges discovered in Paper I have also been identified in related research. However, some challenges have not been discussed in other studies. For example, that *REQUIRES* (R1 requires R2 to function, but not vice versa) and *CVALUE* (R1 affects the value of R2 for a customer) are considered as the most common and important interdependency types to identify, which can be seen as interesting as most software engineering research on QR is performed on implementation level where *AND* and *OR* are the dominant interdependency types investigated. All practitioners discussed the impact of QR interdependencies on product development. Delays or exclusions of requirements may be the result of QR interdependencies; however, the practitioners indicate low extent of interdependency management due to the complexity of the task. Close to 1 out of 5 QR are dismissed from the project at some stage during development. One of the main reasons for dismissal of QR is that QR have lower priority than FR.

The sampling of companies was made with respect to differences between the companies since we wanted to discover an as broad spectrum of challenges as possible. However, it may be possible that other challenges would appear if other companies participated. Thus, the industrial QR challenges in Paper I is not a universal picture, although it has suited its purpose of increasing the understanding of QR and helping to find research areas in need of further investigation.

4.2 Paper II - Prioritization of Quality Requirements

Following the problem of not including QR as hard requirements in the project (Paper I), an effort was aimed to discover how QR are prioritized in industry, to understand the challenging problem of including QR into projects. Paper II reports the results of state-of-practice in prioritization of QR. A qualitative survey was conducted at eleven companies where 22 practitioners were interviewed.

The results show that ad-hoc is the dominant requirements prioritization method for QR, followed by numerical assignment. However, in most cases, QR are by default put into the not *critical* category, i.e., QR always have the lowest priority, and only get attention in the prioritization process if decision-makers are dedicated to invest specific time and resources on QR prioritization. The finding that

ad-hoc is the most common prioritization ”method” of QR is not in line with related research, which suggests that numerical assignment is the most common one. A likely explanation of why ad-hoc prioritization is used could be that there is a lack of useable and useful (scalable) prioritization techniques. Moreover, looking into what criteria are taken into account when prioritizing QR, the two most used criteria are: (1) no criterion, i.e., prioritize are based on ”gut feeling” and experience, and (2) customer input. These results contradicts some earlier work of what criteria to use, e.g., [22]. Furthermore, although the literature suggest that it is important to consider multiple criteria before deciding if a requirement should be implemented directly, later, or not at all, the results in Paper II show that only three (out of 22) practitioners use more than one criterion when prioritizing QR.

The reasons for not prioritizing QR was not related to the prioritization process itself, but rather how QR were treated in the overall requirements engineering process. The main challenges with prioritizing QR are, gathering (elicit) QR, lack of well specified QR, and lack of quantified QR. The indirect effects such as difficulties with elicitation of, specification of, and quantification of QR can be an issue affecting effective and efficient prioritization.

4.3 Paper III - How are Quality Requirements Specified?

Following the discovery that lack of elicitation of, and well specified and quantified QR were among the main challenges for difficulties of prioritizing QR (Paper II), a document study at a case company was initiated. In Paper III, an in-depth investigation of an industrial requirements specification is conducted. The purpose of Paper III is to investigate how QR are specified, in particular if QR are stated in a non-quantifiable manner.

The results show that 38% of 2,178 requirements are QR. Even though relatively many QR exist in the requirement specification, there are areas of improvements, e.g., few maintainability requirements exist. It is not possible to claim that all, or almost all QR are discovered at the case company; however, the fact that more than 800 QR are specified in the requirements specification suggests that many QR are discovered and elicited. This result is not in line with, neither the general opinion nor with related research (e.g., [30]) that QR are difficult to discover, or to discover them at all. In general, the discipline of specifying (writing) requirements at the case company is good. A requirement only consists of one requirement, i.e., two requirements are not written as one, with the only exception of mixing FR and QR. Moreover, 56% of all QR are quantified with a direct metrics. However, not all QR are suitable to be quantified, e.g., several security requirements. In addition, several QR refers to a certain standard where the quantification may be hidden in the standard.

Quality requirements interdependencies can cause problems if ignored, as found in Paper I. This makes it difficult to specify cross-cutting concerns. As a workaround, the case company has a separate section for performance requirements.

However, the problem with this solution is that not all requirements on a specific subject are specified at the same place in the requirement specification, which may lead to missing QR as the completeness could be more difficult to assess.

4.4 Paper IV - The Quality Performance Model

Paper IV presents the results of supporting release planning of QR through the presentation and validation of the first complete version of the QUPER (quality performance) model, including the detailed guidelines of how to apply QUPER in practice. QUPER was developed to support release planning and roadmapping of QR, more specifically with the goal to provide concepts for qualitative reasoning of orders of magnitude rather than precise mathematical formulas. The model was developed in close collaboration with a case company in the mobile handset domain. The version of QUPER presented in Paper IV is based on previous work (Paper IX, XI, XV, XVI, and XVIII) where different aspects of the model were evaluated.

To apply QUPER in practice, seven steps are envisioned. First, pinpoint any relevant QR and list them as candidate QR, then define which scale and measurement unit that can be used to express the level of quality of a selected QR. In step 3, is about identifying competing products' as well as own products' quality levels; while in the forth step, market expectations are defined in terms of breakpoints. Step 5 is about estimating the cost in terms of the values of cost barriers, and then candidate requirements (Step 6) are proposed, discussed, and decided. Finally, dependencies, in terms of how cost estimates are affected by other QR, are identified. In general, the results show that the QUPER model is easy to understand, that the detailed guidelines work in an industrial environment and the model does not take too much time to apply in practice. The detailed guidelines were found to be helpful due to easy steps to follow, and in particular the provided examples for each step. QUPER's last step, how to identify cost dependencies, was viewed as easy to follow, and at the same time detailed enough to be useful in practice.

The results from the validation indicate that QUPER is useful and applicable in an industrial setting, and that the model improves the decision-making process of release planning for quality requirements. In particular, the visualized roadmap with relations to the market and competitors provides more substance to decisions of what level of quality to aim for in the coming releases. In addition to the decision-making process, the results indicate that the QUPER model could improve the communication between people through a "common language" (QUPER's concepts) that is understood by everyone (that has used the model).

4.5 Paper V - Setting Quality Targets for Coming Releases with QUPER

Paper V presents one case of QUPER tailoring, implementation, and evaluation conducted at a case company in the electronic payment-processing domain. The main purpose of Paper V is to show how QUPER was tailored to fit a different organization than the case company in the mobile handset domain, and how the model performed in terms of usefulness and usability in an industrial environment with professionals using real QR.

The overall result indicates that QUPER is relevant in early decision-making process, e.g. release planning. The concepts of breakpoints, competitor analysis, cost barriers, and identification of own products quality level provides a greater understanding of the needed level of quality for the coming releases. In particular the visualization of the roadmap view, provides a clear picture of the current market situation, and what level of quality to aim for. The evaluation indicates, not only that QUPER is applicable and relevant in the selected domain of electronic payment-processing, extending beyond the previously investigated mobile handset domain, but also that QUPER is applicable to more QR than performance requirements,which where applied in previous case studies of the model. Moreover, the results show that the goal of QUPER to be domain-relevant by tailoring the model is feasible.

4.6 Paper VI - A Prototype Tool for QUPER to Support Release Planning of Quality Requirements

Based on the experiences from the two industry evaluations in Paper IV and V, a tool was developed with the purpose of making the process more efficient, to improve the possibility of visualization of the roadmap view, and to facilitate technology diffusion from academia to industry where tool support and tool adaptation are important for the model's scalability.

As the tool is a first prototype tool for QUPER, the first six steps of the QUPER model in Paper IV were implemented. The development of the QUPER prototype tool was carried out in close cooperation between academia and industry. The development of the tool was conducted in two steps. First, requirements for the tool was elicited from eleven practitioners at the case company where important input to the development of the QUPER prototype tool was discovered. Second, the first beta version of the QUPER prototype tool was evaluated in academia by usability tests.

The QUPER prototype tool has eight main features, (1) the tool provides a detailed step-by-step guide of how to use QUPER concepts, (2) a form for experienced users to add and edit QR, (3) a generation of the roadmap view, (4) the tool offers a set of options for managing features and QR, which are accessed through a hierarchical tree, (5) basic information about all existing versions of a particular

QR is displayed in an activity window together with the roadmap view, (6) save and load the information in the internal database to a text file using an XML-like format, (7) import features and QR from spreadsheet files to provide the practitioners with flexibility, and (8) a manual that is divided into two parts, one about the QUPER model and one part specifically about the tool.

The tool was used and evaluated at a case company. The evaluation shows that the QUPER prototype tool's generated roadmap provides a clear overview of a product's and competing products' level of quality, and what level of quality to aim for in the coming releases. In addition, the tool helps practitioners to use the QUPER model in a standardized way to avoid inconsistent usage of QUPER's concepts.

5 Synthesis

This section draws together the obtained results in order to provide answers to the research questions asked in this thesis.

- *RQ1: Which challenges related to quality requirements are experienced by practitioners in the market-driven software development industry?* In the study investigating quality requirements in industrial practice (Paper I) it was shown that many challenges were identified among the case companies. One challenge that emerged was interdependency management of quality requirements. In general, the results indicate a low extent of interdependency management. The identification of dependencies is a complex task and the potential number of dependencies may be very large. Therefore, the understanding of which interdependency types are considered most important (REQUIRES and CVALUE) may give an indication of which interdependencies to start to identify. This can be seen as rather interesting as most SE research on QR is performed on realization/implementation level where AND and OR are the dominant perspectives investigated. Another challenge is cost estimation of QR. The results revealed that one predominant problem in estimating QR was their propensity to impact larger parts of a system, and span over several (or almost all) FR. This is strongly related to the level of dismissal of QR late in projects. A third challenge is related to the late dismissal of QR. The reasons for late dismissal of QR are: (1) poor cost estimations, (2) lack of resources, and (3) that QR have lower priority than FR. However, the main challenge is that QR are not taken into consideration during project planning.

- *RQ2: How are quality requirements prioritized in the market-driven software development industry?* In Paper II, prioritization of QR were investigated. The findings show that ad-hoc is the dominant prioritization method

for quality requirements, followed by numerical assignment. Although numerical assignment is used to prioritize QR, in most cases, QR are by default out into the not critical category, i.e., QR always have lower priority than FR. This result supports the challenge that QR have lower priority than FR in RQ1. Reasons for not prioritizing QR were not related to the prioritization process itself, but rather how QR were treated in the overall requirements engineering process. The fact that companies use ad-hoc prioritization may be explained by several reasons. One reason could be that there is inadequate technology and knowledge transfer from research to industry. Also, a likely explanation could be that there is a lack of usable and useful (scalable) prioritization techniques.

- *RQ3: How are quality requirements specified in the market-driven software development industry?* Looking into how QR are specified in an industrial requirements specification (Paper III), the results reveal that all requirements in the specification were written in structured natural language without explicit specification of interdependencies across requirements. As QR typically crosscut a functional decomposition, the lack of referencing structure creates obstacles. For example, FR are sometimes repeated several times for each associated QR. Another example is having a separate section for crosscutting concerns, apart from the functional structure of the specification. This causes problems with getting an overview of QR, which may lead to deficiencies in completeness and even contradicting requirements discovered late in the process. The findings from this study suggest that methods for QR need to encompass many aspects to comprehensively support working with QR. Solely focusing on, for example, quantification of QR might overlook important requirements since there are many quality requirements in the studied specification where quantification is not appropriate.

- *RQ4: How can the specification of quality requirements be improved when setting future quality targets in a market-driven software development environment?* The improvement of the specification of QR future quality targets was investigated in Paper IV. In general, the results show that the QUPER model is easy to understand, that the detailed guidelines work in an industrial environment and the model does not take too much time to apply in practice. The detailed guidelines were found to be helpful due to easy steps to follow, and in particular the provided examples for each step. The results from the validation indicate that QUPER is useful and applicable in an industrial setting, and that the model improves the decision-making process. Hence, the complete version of the QUPER model improves the specification of quality requirements when setting future quality targets. Although all three views in the QUPER model help professionals when specifying future quality targets for QR, the results show that the roadmap view is the most important view of the QUPER model. The roadmap view provides a great visualization of

the market situation and an easy to understand overview, which improves the specification of quality requirements future quality targets.

- *RQ5: 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?* In all three studies (Paper IV, V, and VI), the results suggest that the QUPER model could improve the release planning process and understanding of QR, in particular with the visualization of the roadmap view. The central part of the improvement in the decision-making process is the roadmap view with the importance of relating the needed level of quality to the market and the competitors. The decisions about the needed level of quality will have a better substance compared to just presenting a metric of the quality level. Furthermore, in Paper VI, the findings suggest that the QUPER prototype tool could help in setting the right targets due to the visualization of the generated roadmap view.

- *RQ6: How can QUPER be tailored to suit industrial environments?* One goal of the QUPER model is for the model to be domain-relevant, i.e., it should be possible to combine the concepts of QUPER with an organization's existing practices by tailoring the model. Two studies (Paper V and VI) examined how the QUPER model could be tailored to suit organizations existing practices. In Paper V, the results show that it is possible, with a few modifications where the main addition to QUPER's steps was a competitor analysis, to adopt and tailor QUPER's generic concepts and guidelines (as described in Paper IV) to the case company. In Paper VI, a prototype tool was developed where one major objective is to evaluate how tool support of the QUPER model may improve the practical application of the model in an industrial context. The results indicate that the flexible workflow of the tool, i.e., not being forced to use all of QUPER's steps, makes it easier to adopt the QUEPR model to existing practices, even within the same organization where different projects work in different ways.

6 Research Agenda

This section describes how the research can be continued in the future. All six included papers have possibilities of further research, which are presented and discussed in relation to each paper.

6.1 FR1: Increasing Survey Sample with Focus on Diversity

Paper I and II take a first step towards a deeper understanding of QR challenges in industry. However, it would be valuable to add further interviews with people in

other organizations to provide a more comprehensive view of QR challenges. Although the sample in the conducted qualitative surveys is broad, organizations that use market-driven development and open source software would further increase the range of the sample. Moreover, agile development approaches are becoming more common in the software industry, and further experiences from agile development would be valuable.

The performed studies in Paper I and II focused on the product and project manager roles. It would be valuable to include, for example, developers to be able to understand QR challenges during the entire lifecycle. This could be conducted in relation to the three views of quality in the ISO/IEC 2502 standard, *internal*, *external*, and *quality in use*. The internal view of quality relates to the product view and reflects the perspectives of developers and designers, the external view of quality relates to the project view and reflects to the perspective of the project and product manager, while quality in use aligns to the business perspective meaning the customers and the users of the system.

6.2 FR2: Deeper Analysis of QR Metrics in Industry

Paper III takes a first step towards a deeper understanding of how QR are specified, and particular how QR are quantified in an industrial requirements specification. However, the investigation only use archival data, hence, to complement this study, an interview study with sub-domain experts would improve the understanding of the rationale behind the specifications of requirements. The impact of standards on the requirements practice is only briefly analyzed in Paper III, therefore, to further understand the impact of standards, interviews with practitioners are recommended.

6.3 FR3: Further Improvements of the QUPER Model

The QUPER model could be applied in additional organizations to examine its usefulness and limitations, in particular, in a dynamic validation of the version described in Paper IV. The use of the QUPER model's concepts shows promise, but the current state of QUPER has barely begun to investigate the area of release planning of QR. Although the QUPER model has been applied in two companies, organizations and projects with different characteristics need to be involved. Furthermore, it could be possible to investigate if software architecture evaluation methods, such as ATAM, may be used together with QUPER as input to the second cost barrier. The evaluations in this thesis indicate that the second cost barrier is related to new software architecture. Therefore, if methods such as ATAM could evaluate the current architecture, identify the current level of quality for a certain quality indicator the software architecture could generate, may make it easier to estimate QUPER's second cost barrier.

The QUPER prototype tool needs further improvements and evaluations. Improvements are needed, e.g., to include all steps of QUPER as presented in Paper IV, in particular the cost dependency step. Dependencies may have a major impact on the estimated cost for other features. The cost to improve the quality level for one feature may imply an improved level of quality for other features. This may lead to a change of other features cost barriers and which feature to select for the coming release. Therefore, it is important to implement the cost dependency step into the QUPER prototype tool. Moreover, to include a generated roadmap at each step in the detailed step-by-step guide is important. Further evaluations in industry in different domains where the long-term effect, in terms of benefits and challenges, of using the QUPER prototype tool needs to be investigated to validate its feasibility and scalability.

7 Conclusion

In order to help software product developing organizations to improve, and to solve real problems faced by professionals in an industrial environment, research should be based on industry needs and challenges. The solutions and results should be validated and evaluated in a non-simulated environment for their usefulness and scalability. The research presented in this thesis stemmed from a concrete need for supporting QR to enhance high-level decision-making related to release planning and roadmapping. The identification of QR challenges faced by professionals (Part I) was needed to figure out what, and how to improve the ability to make early estimates with adequate accuracy of QR. The QUPER model (Part II) describes how to support QR in the release planning process, and the validation and transfer to industry was a way of refining and evaluating the model and contributing to industry practice.

The overall contribution of this thesis is improving practice of supporting QR in release planning and roadmapping, and adding to a field of research with knowledge of state-of-practice to provide a focus of further research based on real industry needs.

The importance of QR needs to be acknowledged by practitioners, not only in theory, but also in practice. There seems to be a bespoke development mindset in the market-driven developing companies, where the immediate project gets a higher priority than the long-term evolution of the product. Having an extra function is often considered more valuable than higher quality. This may be due to lack of knowledge and difficulties in committing to QR, especially when there is pressure to deliver the product. However, the main problem is that QR are not taken into consideration during product planning and thus not included as hard requirements in the projects since QR are often by default seen as having a lower priority than FR, and only prioritized if time and resources are available once all FR have been implemented in the coming release, making the realization of QR a

reactive rather than proactive effort. Product management may thus not be able to plan and rely on QR to achieve competitive advantages.

The evaluation of the QUPER model indicates that QUPER is relevant in early decision-making process in relation to release planning and roadmapping. The concepts of breakpoints, competitor analysis, cost barriers, and identification of own products quality level provides a greater understanding of the needed level of quality for the coming releases. In particular the visualization of the roadmap view, provides a clear picture of the current market situation, and what level of quality to aim for. Moreover, the QUPER model can be used to support early discovery and quantified quality targets in relation to the market and users' expectations, which minimizes the risks of architectural failure and falling short of meeting users' real needs when design the software architecture. The results show that the goal of QUPER to be domain-relevant by tailoring the model is feasible, as the model has been tailored to suit organizations in both the mobile handset, and the electronic payment processing domains.

The addition of the QUPER prototype tool shows that the tool's generated roadmap provides a clear overview of a product's and competing products' level of quality, and what level of quality to aim for in the coming releases. The visualization of the roadmap may be good when arguing the importance of a QR's needed level of quality for coming releases. The QUPER prototype tool can help practitioners to use the QUPER model in a standardized way to avoid inconsistent usage of QUPER's concepts, which may be the case in a manual usage of QUPER.

Although several QR challenges faced by 22 professionals from 11 companies have been identified, it is not possible to claim that these challenges are completely representative of the population. Therefore, further interviews from other organizations may be needed to provide a more comprehensive view. In addition, QR challenges faced by other roles than product managers and project leaders would provide an understanding of QR challenges during the entire lifecycle. The initial results of the QUPER model and the prototype tool show great promise, but also give information about limitations on which future research can be based on. To fully understand and evaluate the potential benefits of the QUPER model, the model should be used in a project from its start until a product is launched to the market. Moreover, the cost dependency step needs further evaluations, in particular dynamic validations, and to be implemented in the QUPER prototype tool.

The overall goal of this thesis is to increase the awareness and understanding of quality requirements, and to find means for improving the ability to make early estimates of quality requirements, e.g., performance requirements, in order to enhance high-level decision-making related to release planning and roadmapping. The conception and gradual refinement of the QUPER model grew out of the collaboration between academia and industry. QUPER is aimed at addressing some of the challenges identified in cooperation with our industry partner, and challenges identified in the research. Initial results of QUPER show great promise, but also give us information about limitations on which future research can be based on.

BIBLIOGRAPHY

[1] A. Abran, J.W. Moore, P. Bourque, R. Dupuis, and L.L. Tripp. *SWEBOK: Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, 2004.

[2] M. van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Determination of the next release of a software product: an approach using integer linear programming. In *Proceedings of the 11th International Workshop Requirements Engineering: Foundation for Software Quality*, pages 119–124, 2005.

[3] M. van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Software product release planning through optimization of what-if analysis. *Information and Software Technology*, 50(1–2):101–111, 2008.

[4] A. Al-Emran and D. Pfahl. Operational planning, re-planning and risk analysis for software releases. In *Lecture Notes in Computer Science*, volume 4589, pages 315–329, 2007.

[5] A. Al-Emran, D. Pfahl, and G. Ruhe. A method for re-planning of software release planning using discrete-event simulation. *Software Process Improvement and Practice*, 13(1):19–33, 2008.

[6] A. Al-Emran, D. Pfahl, and G. Ruhe. Decision support for product release planning based on robustness analysis. In *Proceedings of the 18th IEEE International Requirements Engineering Conference*, pages 157–166, 2010.

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

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

[9] 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.

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

[11] J. Asundi, R. Kazman, and M. Klein. Using economic considerations to choose among architecture design alternatives. Technical report, Carnigie Mellon - Software Engineering Institute, 2001.

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

[13] A.J. Bagnall, V.J. Rayward-Smith, and I.M. Whittley. The next release problem. *Information and Software Technology*, 43(14):883–890, 2001.

[14] M.R. Barbacci, R. Ellison, A.J. Lattanze, J.A. Stafford, C.B. Weinstock, and W.G. Wood. Quality Attribute Workshops (QAWs). Technical report, Carnigie Mellon - Software Engineering Institute, 2003.

[15] 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.

[16] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 2003.

[17] K. Beck. *Extreme Programming Explained*. Addison-Wesley, 2005.

[18] W. Bekkers, M. Spruit, I. van de weerd, and S. Brinkkemper. A situational assessment method for software product management. In *Proceedings of the 18th European Conference on Information Systems*, pages 22–34, 2010.

[19] W. Bekkers, I. van de weerd, S. Brinkkemper, and A. Mahieu. The influence of situational factors in software product management: An empirical study. In *Proceedings of the Second International Workshop on Software Product Management*, pages 41–48, 2008.

[20] W. Bekkers, I. van de weerd, M. Spruit, and S. Brinkkemper. A framework for process improvement in software product management. In *Proceedings of the 17th European Conference on Systems, Software and Services Process Improvement*, pages 1–12, 2010.

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

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

[23] B. Berenbach, D.J. Paulish, J. Kazmeier, and A. Rudorfer. *Software & Systems Requirements Engineering: In Practice*. Pearson Education Inc., 2009.

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

[25] R. Berntsson Svensson, T. Gorschek, and B. Regnell. Quality requirements in practice: An interview study in requirements engineering for embedded systems. In *Proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 218–232, 2009.

[26] R. Berntsson Svensson, M. Höst, and B. Regnell. Managing quality requirements: A systematic review. In *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 261–268, 2010.

[27] E. Bjarnason, K. Wnuk, and B. Regnell. Overscoping: Reasons and consequences - a case study on decision making in software product management. In *Proceedings of the Fourth International Workshop on Software Product Management*, pages 30–39, 2010.

[28] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby. Cost models for future software life cycle processes: Cocomo 2.0. *Annals of Software Engineering*, 1(1):57–94, 1995.

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

[30] 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.

[31] J. Bosch. *Designing and Use of Software Architecture Adopting and Evolving a Product Line Approach*. Pearson Education, 2000.

[32] 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(1):13–38, 1999.

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

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

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

[36] 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 of the Fifth IEEE International Symposium on Requirements Engineering*, pages 84–91, 2001.

[37] E. Carmel and S. Becker. A process model for packaged software development. *IEEE Transaction on Engineering and Management*, 42(1):50–61, 1995.

[38] J. Carriere, R. Kazman, and I. Ozkaya. A cost-benefit framework for making architectural decisions in a business context. In *Proceedings of the 32nd international conference on software engineering*, pages 149–157, 2010.

[39] M.B. Chrissis, M. Konrad, and S. Shrum. *CMMI: Guidelines for Process Integration and Product Improvement*. Addison-Wesley, 2003.

[40] L. Chung and J.C.S do Prado Leite. On non-functional requirements in software engineering. In *Lecture Notes in Computer Science*, volume 5600, pages 363–379, 2009.

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

[42] J. Cleland-Huang. Toward improved traceability of non-functional requirements. In *Proceedings of the Third International Workshop on Traceability in Emerging Forms of Software Engineering*, pages 14–19, 2005.

[43] J. Cleland-Huang, C.K. Chang, and M. Christensen. Event-based traceability for managing evolutionary change. *IEEE Transactions on Software Engineering*, 29(9):796–810, 2003.

[44] J. Cleland-Huang, C.K.Chang, G. Sethi, K. Javvaji, H. Hu, and J. Xia. Automating speculative queries through event-based requirements traceability. In *Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*, pages 289–298, 2002.

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

[46] J. Cleland-Huang, R. Settimi, X. Zou, and P. Sole. Automated classification of non-functional requirements. *Requirements Engineering*, 12(2):103–120, 2007.

[47] CMMI Product Team. Capability Maturity Model Integration (CMMI): Version 1.1, 2002.

[48] J.M. Coakes and E.W. Coakes. Specification in context: Stakeholders, systems and modeling of conflict. *Requirements Engineering*, 5(2):103–113, 2000.

[49] A. Cockburn. *Agile Software Development*. Addison-Wesley, 2002.

[50] D. Condon. *Software Product Management - Managing Software Development from Idea to Product to Marketing to Sales*. Aspatore Books, 2002.

[51] R.G. Cooper. *Winning at New Products*. Basic Books, 2001.

[52] M.A. Cusomano and R.W. Selby. *Microsoft Secrets*. Simon and Schuster, 1995.

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

[54] 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.

[55] 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 of the 14th IEEE International Requirements Engineering Conference*, pages 176–185, 2006.

[56] A.M. Davis. *Software Requirements: Objects, Functions and States*. Prentice Hall, 1993.

[57] B. Deifel. A process model for requirements engineering of ccots. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications*, pages 316–320, 1999.

[58] M. Denne and J. Cleland-Huang. The incremental funding method: data-driven software development. *IEEE Software*, 21(3):39–47, 2004.

[59] Y. Dittrich, K. Rönkkö, J. Eriksson, C. Hansson, and O. Lindeberg. Cooperative method development, combining qualitative empirical research with technique, and process improvement. *Empirical Software Engineering*, 13(3):231–260, 2008.

[60] L. Dobrica and E. Niemela. A survey on software architecture analysis methods. *IEEE Transactions on Software Engineering*, 28(7):638–653, 2002.

[61] 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 of the 13th IEEE International Conference on Requirements Engineering*, pages 373–382, 2005.

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

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

[64] C. Ebert. The impacts of software product management. *The Journal of Systems and Software*, 80(6):850–861, 2007.

[65] C. Ebert and M. Smouts. Tricks and traps of initiating a product line concept in existing products. In *Proceedings of the 25th International Conference on Software Engineering*, pages 520–525, 2003.

[66] N.A. Ernst and J. Mylopoulos. On the perception of software quality requirements during the project lifecycle. In *Lecture Notes in Computer Science*, volume 6182, pages 143–157, 2010.

[67] A. Fink. *The survey handbook*. Sage Publications, 2003.

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

[69] G. Fitzgerald and N. Russo. The turnaround of the london ambulance service computer-aided despatch system. *European Journal of Information Systems*, 14(3):244–257, 2005.

[70] E. Folmer and J. Bosch. Architecting for usability: a survey. *The journal of systems and software*, 70(1–2):61–78, 2002.

[71] T. Gilb. *Competitive Engineering*. Elsevier Butterworth-Heinemann, 2005.

[72] R.L. Glass. The software-research crisis. *IEEE Software*, 11(6):42–47, 1994.

[73] M. Glinz. On non-functional requirements. In *Proceedings of the 15th IEEE International Requirements Engineering Conference*, pages 21–26, 2007.

[74] L. Gorchels. *The Product Manager's Handbook: The Complete Product Management Resource*. McGraw-Hill, 2006.

[75] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE Software*, 23(6):88–95, 2006.

[76] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. Industry evaluation of the requirements abstraction model. *Requirements Engineering*, 12(3):163–190, 2007.

[77] T. Gorschek, E. Tempero, and L. Angelis. A large-scale empirical study of practitioners' use of object-oriented concepts. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pages 115–124, 2010.

[78] T. Gorschek and C. Wohlin. Requirements abstraction model. *Requirements Engineering*, 11(1):79–101, 2006.

[79] D. Greer and G. Ruhe. Software release planning: An evolutionary and iterative approach. *Information and Software Technology*, 46(4):243–253, 2004.

[80] D. Gross and E. Yu. From non-functional requirements to design through patterns. *Requirements Engineering*, 6(1):18–36, 2001.

[81] 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.

[82] M. Höst, B. Regnell, J. Natt och Dag, J. Nedstam, and C. Nyberg. Exploring bottlenecks in market-driven requirements management process with discrete event simulation. *The Journal of Systems and Software*, 59(3):323–332, 2001.

[83] IEEE Std 1061-1998. IEEE Standard for a software quality metrics methodology, 1998.

[84] IEEE Std 1220-2005. IEEE Standard for Application and Management of the Systems Engineering Process, 2005.

[85] International Software Product Management Association. SPM framework, July 2011.

[86] ISO/IEC 12207:1997. Software life cycle process, 1997.

[87] ISO/IEC 15288:2002. Systems engineering - system life cycle processes, 2002.

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

[89] S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

[90] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley, 1999.

[91] S. Jantunen, L. Lehtola, C. Gause, R. Dumdum, and R.J. Barnes. The challenge of release planning: Visible but not seen? In *The Fifth International Workshop on Software Product Management*, 2011.

[92] S. Jantunen, K. Smolander, and D.C. Gause. How internationalization of a product changes requirements engineering activities: An exploratory study. In *Proceedings of the 15th IEEE International Requirements Engineering Conference*, pages 163–172, 2007.

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

[94] H-W. Jung. Optimizing value and cost in requirements analysis. *IEEE Software*, 15(4):74–78, 1998.

[95] 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.

[96] H. Kaindl, S. Brinkkemper, J.R. Jr Bubenko, B. Farbey, S.J. Greenspan, J.C. Sampaio do Prado Leite, N.R. Mead, J. Mylopoulos, and J. Siddiqi. Requirements engineering and technology transfer: Obstacles, incentives and improvement agenda. *Requirements Engineering*, 7(3):113–123, 2002.

[97] H. Kaiya, A. Osada, and K. Kaijiri. Identifying stakeholders and their preferences about nfr by comparing use case diagrams of several existing systems. In *Proceedings of the 12th IEEE International Conference on Requirements Engineering*, pages 112–121, 2004.

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

[99] J. Karlsson. Software requirements prioritizing. In *Proceedings of the Second IEEE International Conference on Requirements Engineering*, pages 110–116, 1996.

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

[101] J. Karlsson, S. Olsson, and K. Ryan. Improved practical support for large-scale requirements prioritising. *Requirements Engineering*, 2(1):51–60, 1997.

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

[103] 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.

[104] L. Karlsson, P. Berander, B. Regnell, and C. Wohlin. Requirements prioritization: An experiment on exhaustive pair-wise comparisons versus planning game partitioning. In *Proceedings of the Eight International Conference on Empirical Assessment in Software Engineering*, pages 145–154, 2004.

[105] 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(6):588–604, 2007.

[106] R. Kazman, G. Abowd, L. Bass, and P. Clements. Scenario-based analysis of software architecture. *IEEE Software*, 13(6):47–55, 1996.

[107] R. Kazman, J. Asundi, and M. Klein. Making architecture design decisions: An economic approach. Technical report, Carnigie Mellon - Software Engineering Institute, 2002.

[108] R. Kazman, M. Barbacci, M. Klein, and S.J. Carriere. Experience with performing architecture tradeoff analysis. In *Proceedings of the 21st International Conference on Software Engineering*, pages 54–63, 1999.

[109] 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.

[110] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transaction on Software Engineering*, 28(8):721–734, 2002.

[111] R.N. Kostoff and R.R. Schaller. Science and technology roadmaps. *IEEE Transaction on Engineering Management*, 48(2):132–143, 2001.

[112] G. Kotonya and I. Sommerville. *Requirements Engineering - Processes and Techniques*. John Wiley & Sons, 1998.

[113] A. van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the Fifth International Symposium on Requirements Engineering*, pages 249–261, 2001.

[114] A. van Lamsweerde. Goal-oriented requirements enginering: A roundtrip from research to practice. In *Proceedings of the 12th IEEE International Requirements Engineering Conference*, pages 4–7, 2004.

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

[116] D. Leffingwell and D. Widrig. *Managing Software Requirements - A Unified Approach*. Addison-Wesley, 2000.

[117] 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.

[118] L. Lehtola, M. Kauppinen, and S. Kujala. Requirements prioritization challenges in practice. In *Proceedings of the Fifth International Conference on Product Focused Software Process Improvement*, pages 497–508, 2004.

[119] L. Lehtola, M. Kauppinen, and S. Kujala. Linking business view to requirements engineering: Long-term product planning by roadmapping. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, pages 439–443, 2005.

[120] P. Loucopoulos and V. Karakostas. *System Requirements Engineering*. McGraw-Hill Book Company Europe, 1995.

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

[122] L.A. Macaulay. *Requirements Engineering*. Springer-Verlag, 1996.

[123] N. Maiden. What has requirements research ever done for us. *IEEE Software*, 22(4):104–105, 2005.

[124] S. Maurice, G. Ruhe, O. Saliu, and A. Ngo-The. *Value-Based Software Engineering*, chapter Decision Support for Value-Based Software Release Planning, pages 247–261. Springer, 2006.

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

[126] J. Mylopoulos, L. Chung, and B. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, 1992.

[127] C.J. Neill and P.A. Laplante. Requirements engineering: the state of the practice. *IEEE Software*, 20(6):40–45, 2003.

[128] A. Ngo-The and G. Ruhe. A systematic approach for solving the wicked problem of software release planning. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 12(1):95–108, 2008.

[129] L. Nguyen and P.A. Swatman. Managing the requirements engineering process. *Requirements Engineering*, 8(1):55–68, 2003.

[130] 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.

[131] B. Paech and T. Wetter. Rational quality requirements for medical software. In *Proceedings of the 30th International Conference on Software Engineering*, pages 633–638, 2008.

[132] M.C. Paulk, B. Curtis, M.B. Chrissis, and C.V. Weber. Capability maturity model integration for software: Version 1.1, 1993.

[133] D.A. Penny. An estimation-based management framework for enhancive maintenance in commercial software products. In *Proceedings of the International Conference on Software Maintenance*, pages 122–130, 2002.

[134] F. Pettersson, M. Ivarsson, and T. Gorschek. A practitioner's guide to light weight software process assessment and improvement planning. *Journal of Systems and Software*, 81(6):972–995, 2008.

[135] S.L. Pfleeger. Understanding and improving technology transfer in software engineering. *Journal of Systems and Software*, 47(2):111–124, 1999.

[136] S.L. Pfleeger. *Software Engineering - Theory and practice*. Prentice-Hall, 2001.

[137] B. Potter, J. Sinclair, and D. Till. *An Introduction to Formal Specification and Z*. Prentice Hall, 1996.

[138] C. Potts. Invented requirements and imagined customers: requirements engineering for off-the-shelf software. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, pages 128–130, 1995.

[139] Project Management Institute. *A guide to the project management body of knowledge (PMBOKGuide)*. PA: Project Management Institute, 2000.

[140] 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.

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

[142] B. Regnell and S. Brinkkemper. *Engineering and Managing Software Requirements*, chapter Market-Driven Requirements Engineering for Software Products, pages 287–308. Springer, 2005.

[143] 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.

[144] 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(1):51–62, 2001.

[145] 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 the First Swedish Conference on Software Engineering Research and Practice*, pages 49–52, 2001.

[146] S. Robertson and J. Robertson. *Mastering the Requirements Process*. ACM Press, 1999.

[147] C. Robson. *Real World Research*. Blackwell, 2002.

[148] S. Rochimah, W.M.N. Kadir, and A.H. Abdullah. An evaluation of traceability approaches to support software evolution. In *Proceedings of the 2007 International Conference on Software Engineering Advances*, 2007.

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

[150] W.W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the IEEE WESTCON*, pages 1–9, 1970.

[151] G. Ruhe. *Product release Planning - Methods, Tools and Applications*. CRC Press, 2010.

[152] G. Ruhe and D. Greer. Quantitative studies in software release planning under risk and resource constraints. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering*, pages 262–270, 2003.

[153] G. Ruhe and J. Momoh. Strategic release planning and evaluation of operational feasibility. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, page 313b, 2005.

[154] G. Ruhe and A. Ngo-The. Hybrid intelligence in software release planning. *International Journal of Hybrid Intelligent Systems*, 1(2):99–110, 2004.

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

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

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

[158] O. Saliu and G. Ruhe. Supporting software release planning decisions for evolving systems. In *Proceedings of the 29th Annual IEEE/NASA Software Engineering Workshop*, pages 14–26, 2005.

[159] P. Sawyer. Packaged software: Challenges for re. In *Proceedings of the Sixth International Workshop on Requirements Engineering: Foundations of Software Quality*, pages 137–142, 2000.

[160] P. Sawyer. Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems*, 9(1):47–58, 2000.

[161] P. Sawyer, I. Sommerville, and G. Kotonya. Improving market-driven re processes. In *Proceedings of the International Conference on Product Focused Software Process Improvement*, pages 222–236, 1999.

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

[163] K. Schmid. Scoping software product lines - an analysis of an emerging technology. In *Proceedings of the First conference on Software product lines : experience and research directions*, pages 513–532, 2000.

[164] K. Schmid. A comprehensive product line scoping approach and its validation. In *Proceedings of the 24th International Conference on Software Engineering*, pages 593–603, 2002.

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

[166] C. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transaction on Software Engineeirng*, 25(4):557–572, 1999.

[167] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transaction on Software Engineering*, 32(11):736–743, 1997.

[168] F. Shull, J. Singer, and D.I.K. Sjøberg. *Guide to advanced empirical software engineering*. Springer-Verlag, 2008.

[169] C.A. Singer. A requirements tutorial. quality systems and software requirements. Technical report, Special Report SR-NWT-002159, 1992.

[170] I. Sommerville. *Software Engineering*. Addison-Wesley, 2007.

[171] I. Sommerville and P. Sawyer. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, 1999.

[172] G. Stark, P. Oman, A. Skillicorn, and R. Ameele. An examination of the effects of requirements changes on software maintenance releases. *Journal of Software Maintenance: Research and Practice*, 11(5):293–309, 1999.

[173] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, and S.B. Saleem. A systematic review on strategic release planning models. *Information and Software Technology*, 52(3):237–248, 2010.

[174] H. Thayer. Software engineering: a tutorial. *Computer*, 35(4):68–73, 2002.

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

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

[177] 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 First International Workshop on Software Product Management*, pages 3–11, 2006.

[178] 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.

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

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

[181] K. Wnuk, R. Berntsson Svensson, and B. Regnell. Investigating upstream versus downstream decision-making in software product management. In *Proceedings of the Third International Workshop on Software Product Management*, pages 23–26, 2009.

[182] K. Wnuk, B. Regnell, and L. Karlsson. What happened to our features? visualization and understanding of scope change dynamics in a large-scale industrial setting. In *Proceedings of the 17th IEEE International Conference on Requirements Engineering*, pages 89–98, 2009.

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

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

[185] R.K. Yin. *Case study research: design and methods*. Sage Publications, 2003.

[186] E.S.K. Yu and J. Mylopoulos. Understanding "why" in software process modelling, analysis, and design. In *Proceedings of 16th international conference on software engineering*, pages 159–168, 1994.

QUALITY REQUIREMENTS CHALLENGES

PAPER I

QUALITY REQUIREMENTS IN INDUSTRIAL PRACTICE - AN EXTENDED INTERVIEW STUDY AT ELEVEN COMPANIES

Abstract

In order to create a successful software product and assure its quality, it is not enough to fulfill the functional requirements, it is also crucial to find the right balance among competing quality requirements (QR). An extended, previously piloted, interview study was performed to identify specific challenges associated with the selection, trade-off, and management of QR in industrial practice. Data was collected through semi-structured interviews with eleven product managers and eleven project leaders from eleven software companies. The contribution of this study is fourfold: First, it compares how QR are handled in two cases, companies working in business-to-business markets, and companies that are working in business-to-consumer markets. These two are also compared in terms of impact on the handling of QR. Second, it compares the perceptions and priorities of QR by product and project management respectively. Third, it includes an examination of the interdependencies among quality requirements perceived as most important by the practitioners. Fourth, it characterizes the selection and management of QR in down-stream development activities.

Richard Berntsson Svensson, Tony Gorschek, Björn Regnell, Richard Torkar, Ali Shahrokni, and Robert Feldt
IEEE Transaction on Software Engineering, 2011, in print

1 Introduction

As the role of software increases and becomes a substantial part of industrial and consumer products, the complexity also escalates, making the process area of requirements engineering (RE) central for success [28]. The characteristics of a product are determined by functionality, but also by the non-functional or quality aspects, a.k.a. quality requirements (QR), such as performance and usability [9].

To create a successful product and assure quality, it is not enough to fulfill the functional requirements. For example, even if the product works, it may be difficult to use, or too costly to maintain [12]. The importance of having a handle on QR can be seen as obvious; however, when it comes to customer satisfaction, end-users are often dissatisfied with software quality [24]. Therefore, QR play a critical role in software product development, and not dealing with QR may lead to more expensive software products and increased time-to-market [11]. The ability to develop a software product that meets customers' requirements, and offers high value to both the development company and the customer, increases the likelihood of market success substantially, thus QR play a central role and can be seen as a key competitive advantage [2], [4].

However, despite their importance, QR are often poorly understood, generally stated informally in a non-quantifiable manner, often contradicting, and difficult to validate [9], [22]. This is further aggravated in market-driven development, where the situation is even more complex [3], due to the large number of requirements stemming from multiple internal and external sources, and the continuous flow into the development organization [16], [18]. The challenges associated with QR have been addressed in part by other studies, see e.g. [25], [26], and [32]. However, none of these have primarily focused on QR.

This paper presents the results of an empirical study that includes data collected through in-depth interviews with twenty-two practitioners from eleven different companies in Sweden of which six are multinational. After ten interviews from five companies, a short paper [6] was presented at an international workshop. The study focuses on the elicitation, analysis and negotiation, management, and general handling of QR in industry. This exploratory study can be seen as a study of state-of-practice in industry, but also an investigation as to what extent state-of-the-art in research, in terms of methods and tools, has penetrated industry practice.

The study incorporates two main perspectives with regards to QR [15], through the study of companies with two distinctly different types of customers. Six of the companies mainly develop and sell products to other companies. For example, a company developing industry robots does not sell to traditional consumers, but rather to other companies that use the robots in their product development. These companies are denoted business-to-business (B2B) [39]. The second type of companies in the study develops products for the consumer market, for example mobile phones or laptops. These companies are denoted business-to-consumer (B2C) [39]

in the paper. The comparison between the two company categories B2B and B2C is the main focus of the paper; however, in each company we elicited information about QR handling from two roles central to decision making in relation to QR. The overall product responsibility is held by product managers (PM), responsible for the overall product and the selection and long-term planning of the product evolution and offering. The second role studied was that of the project perspective. Project leaders (PL) are responsible for managing and prioritizing within the realization phases. The choice to study both perspectives within each company was motivated by the intention of improving coverage and the further possibility to compare the views to identify possible conflicts [17], [36].

The remainder 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, and 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 in literature that include QR related challenges.

Laubars et al. published their field study on requirements modeling [32]. Their study included both customer-specific and market-driven projects. The challenges found by Laubars et al. include e.g. vaguely stated requirements, requirements misunderstandings, changing requirements, and difficulties with prioritization of requirements [32]. Moreover, two challenges related to performance requirements were identified, specification (rational not always obvious) and associating performance requirements with dataflow specifications. Later, a study on requirements engineering challenges in small and medium sized enterprises was presented by Kamsties et al. [25]. Similar to Laubars et al. [32], customer-specific and market-driven projects were represented. Kamsties et al. [25] agree to some extent with Laubars et al. [32] and identified unclear and incomplete requirements as challenges. Other identified challenges are: specification of graphical users interfaces and lack of traceability among requirements. Moreover, Kamsties et al. [25] found that requirements are too vague to test. A study that solely focuses on market-driven software development challenges was published by Karlsson et al. [26]. Challenges related to quality requirements were identified by Karlsson et al. [26], where the main problem was interdependencies related to QR. This problem is not only related to identifying existing interdependencies, but also to what extent QR affect each other and how to deal with it. Moreover, supporting release planning of QR caused problems for the companies. Berntsson Svensson et al. conducted a field study to discover and describe how QR are handled in industry [6]. The findings highlight three important challenges, how to get QR into the projects when functional requirements are prioritized, how to know when the quality level is good enough, and how to achieve testable QR.

Finally, there are a number of studies that are not focusing on challenges, but focus on requirements interdependencies [7], [9], [10], and classification and measurement of QR [22], [34]. For example, Carlshamre et al. identified a set of interdependency types that are present in industry [7], while Olsson et al. concluded that methods need to handle the diversity of QR [34].

The focus of the above mentioned studies, with the exception of Berntsson Svensson et al. [6], has not been primarily on QR, but QR-related 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. Even though Berntsson Svensson et al. [6] solely focused on QR, only five companies were included in the study with a primary focus on PM and PL perspectives. In this study we have extended the data to eleven companies, and, in addition, included challenges related to cost estimation of QR as well as the perspective of comparing B2B and B2C in the analysis, which presents new data from all eleven cases.

3 Research Methodology

The investigation presented in this paper was carried out using a qualitative research approach, namely in-depth semi-structured interviews [37]. Qualitative research aims to investigate and understand phenomena within its real life context [37]. 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 [37], [38]. The purpose of this study is to gain in-depth understanding of QR within market-driven embedded systems companies. Due to the potential richness and diversity of data that could be collected, semi-structured interviews would best meet the objectives of this study. Semi-structured interviews help to ensure common information on pre-determined areas is collected, but allow the interviewer to probe deeper where required.

We choose interviews over doing a large survey as the concept of QR are named and treated very differently in industry, what might be called a quality requirement in one place is simply adherence to product limitations in another [1]. For this reason it was important to have a presence when eliciting the data making it possible to elaborate on what we were looking for and compensate for these differences in culture and naming. Several times we had to put five to ten minutes of explanation[1] into what we were investigating before the interview subject understood and we could proceed. In addition, the interviewer had the chance to validate the questions with the interviewee lessening changes of misunderstandings. That is, the interviewer went back to the interviewee to validate the interviewers interpretation of the results to minimize misinterpretations and validate the results. The research questions in Table 1 provided a focus for the empirical investigation.

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

Table 1: Research Questions

Research Questions (QR = Quality Requirements)	**Example of answers for each RQ**
RQ1: What QR are considered most important, and are there any distinguishably characteristics in relation to customer type (B2B vs. B2C)?	The most important QR for the subject's product in their domain is performance requirement.
RQ2: What interdependencies between QR are present in the companies?	The existing interdependency types in the subject's product are *ICOST* and *REQUIRES*.
RQ2.1: What interdependencies are deemed most important, and how does this compare to previous studies?	The most important interdependency type for the subject to identify is *REQUIRES*.
RQ2.2: To what extent are interdependencies elicited, analyzed and documented?	According to the subject, no elicitation, analysis or documentation of interdependencies are performed in his/her project.
RQ3: How are cost estimations of QR performed, and what is the accuracy of these cost estimates?	In the subject's project, expert judgment is used to estimate the cost of QR.
RQ4: To what extent are QR dismissed from projects after project initiation?	In the subject's project, 20% of all QR are dismissed after project initiation.
RQ4.1: If QR are dismissed, is any consequence analysis performed pre- or post dismissal?	According to the subject, no new analysis is performed when QR are dismissed.
RQ4.2: Are QR specified in a measurable (quantifiable) manner?	According to the subject, QR are sometimes (some QR are, while others not) quantified in their project.

Berntsson Svensson and Aurum [5] discovered that there are differences in which factors are important for project/product success across industries. Therefore, it is important to investigate what quality aspects are considered most important across industries developing products for different customers (RQ1). In addition, Karlsson et al. discovered that it is possible to manage functional dependencies; however, a major problem is to deal with dynamic interdependencies, i.e. quality requirements, which influence a larger part of the functionality or other quality requirements of the system [26]. Therefore, it is important to understand interdependencies related to quality requirements (RQ2). Software cost estimates are the basis for project planning, bidding, and budgeting and are critical for project success [19]. In addition, Berntsson Svensson et al. [6] found that poor cost estimation is one main reason for QR being dismissed after project initiation, therefore, it is important to understand how cost estimations of QR are performed in industry (RQ3). Kamsties et al. found that requirements are often too vague to test [25], therefore, it is important to investigate if QR are quantified in industry (RQ4). Also, dismissal of QR from projects may have an impact on the predicted return of investment, as well as the cost for the customers (RQ4).

3.1 Research Design and Data Collection

The investigation can be divided into three phases:

Planning/Selection: The sampling strategy used was a combination of maximum variation sampling [35] and convenience sampling [35] within our industrial collaboration network. The researchers contacted a "gate-keeper" at each company who identified two subjects (one PM and one PL) that he/she thought were the most suitable and representative of the company to participate in this study. That is, the researchers did not influence the selection of subjects, nor did the researchers have any personal relationship to the subjects. Twenty-two participants at eleven software development companies participated. From each company, one product manager (PM) and one project leader (PL) from the same project were interviewed, resulting in twenty-two data points. The research instrument[2] used in this study, and in [6], was designed with respect to the different areas of interest and inspiration from [26].

All eleven companies develop embedded systems using a market-driven software development approach. The companies themselves vary in respect to size, type of products, type of customers, and application domain, a characterization (following the guidelines of [21]) can be see in Table 2 (more details are not revealed for confidentiality reasons) following the recommendations of [20]. The companies are divided into two main categories based on type of customers: (1) business-to-business [39] companies (B2B), for example, Company E develops control systems for other industry partners, and (2) business-to-consumer [39]

[2] http://serg.cs.lth.se/research/experiment_packages/quality_requirements

companies (B2C) for example, Company C develops products within the telecom domain for end-users.

Data collection: The study used a semi-structured interview strategy [37]. One interviewee and one interviewer attended all interviews. During the interviews, the purpose of the study and a general explanation of QR (see footnote two) were presented to the interviewee. Then, questions about the different areas of interests in relation to QR were discussed in detail. For all interviews, varying in length from 40 to 90 minutes, we took records in the form of written extensive notes in order to facilitate and improve the analysis process.

Analysis: The content analysis [37] involved marking and discussing interesting sections in the recorded notes. The first two authors examined the categories, first individually, and then together in a workshop setting. The category analysis included examination of the content from different perspectives and a search for explicitly stated or concealed pros and cons in relation to how QR are handled in industry. For all statistical tests on the quantitative data given by respondents, a non-parametric Wilcoxon rank sum test [40] with significance tested at the 0.05 level (unless otherwise stated) was performed. This was used as a method to judge between which sets of data there is any discernable difference and thus reduce the number of such possible differences that would otherwise be implied. The raw anonymized data (individual match encoding and ordering) used for the Wilcoxon rank sum tests are available upon request. The results from the analysis and significance levels are found in Section 4.

3.2 Validity

In this section, threats to validity in relation to the research design and data collection are discussed. We consider the four perspectives of validity and threats as presented in Wohlin et al. [40].

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 questions where the participants are asked to express their own opinions.

By collecting data from a wide range of sources on the topic, mono-operation bias [40] was avoided. The potential problem of evaluation apprehension [40] was alleviated by the guarantee of anonymity as to all information divulged during the interviews, and the answers was only to be used by the researcher, i.e. not be showed or used by any other participants, companies, or researcher. 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 propose new factors, i.e. some interdependency types may be missing. The quantitative data given by respondents is subjective since it is not based on any objective measurements;

Table 2: Company Characteristics

	Type of customer	***# Employees***	***Domain***	***Development process***	***# of Reqs in a typical project***	***% of QR in a typical project***
A	B2B	~100	Control systems	Incremental development	>1000	~10%
B	B2B	~3000	Telecom	Plan-driven	~7000	~10%
C	B2C	>5000	Telecom	Plan-driven	>20000	Unknown
D	B2C	325	Telecom	Agile-Scrum	~100 features	~10%
E	B2B	65	Control systems	Waterfall-Iterative	Differs	Differs
F	B2C	~700	Surveillance	Iterative	~250	~15%
G	B2C	~100	Consumer electronics	Plan-driven	~300	~5%
H	B2B	~700	Telecom	Agile-Scrum variant	~200	20%
I	B2B	~50	Security	Waterfall and Agile-Scrum	~100	~15%
J	B2B	~90	Control systems	Plan-driven	~100	~10%
K	B2C	280	Telecom	Waterfall-Iterative	~1000	~5%

there might be differences in how the questions were interpreted that renders the comparisons between QR invalid regardless of the statistical tests employed.

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 ensure that the interview instrument, including the posed questions, are of high quality to obtain highly reliable measures, several pilot studies were conducted, to avoid poor question and poor layout, prior to conducting the interviews.

Internal validity: These threats are related to issues that may affect the causal relationship between treatment and outcome. Threats to internal validity include instrumentation, maturation and selection threats. The potential problem of instrumentation threats was alleviated by developing the research instrument with close reference to literature relating to quality requirements, influenced by previously validated interview instrument [26], and a previously piloted interview study [6]. Moreover, keeping the interview session to 90 minutes, which was possible by collecting background information before the interview session started, alleviates maturation threats. Threat to selection bias is always present when study subjects are not fully randomly sampled. However, given that 11 different companies from different industrial networks and geographical locations are included, and interviewees were selected based on their roles by a "gate-keeper" at the companies, this threat has limited effect.

External validity: The external validity is concerned with the ability to generalize the results, i.e. in this case the applicability of the findings beyond the included companies. Qualitative studies rarely attempt to generalize beyond the actual setting since it is more concerned with characterizing, explaining and understanding the phenomena under study. The nature of qualitative designs also makes it impossible to replicate since identical circumstances cannot be recreated. However, the development of a theory can help in understanding other cases and situations. The fact that more than one participant and company acknowledge several of the discovered challenges increases the possibility of transferring the results to other situations. The large number of companies and contexts also contributes to generalizability. To avoid the interaction of selection and treatment, interviewees were selected according to their roles within the company by a "gate-keeper", hence the researchers did not select the subjects themselves. Moreover, companies were selected from different geographical locations.

4 Results and Analysis

The following four sub-sections present and discuss one research question each, corresponding to the research questions in Table 1.

4.1 Important Quality Aspects (RQ1)

In analyzing Research Question 1 (RQ1), this section examines the most important quality aspects, as illustrated in Figure 1 (note that Figure 1 only visualizes the ranking of QR and is not intended to visually compare B2B and B2C). In Figure 1 we can see the top ranked quality aspects that received at least 10 ranking points by the practitioners, i.e. any quality aspect that received less than 10 points is not displayed in Figure 1.

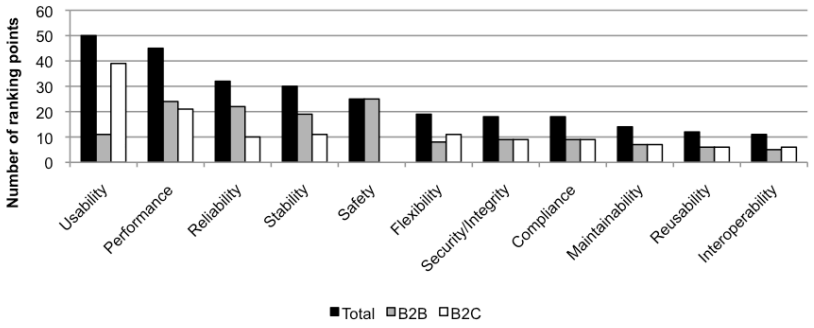


Figure 1: Importance of quality aspects

Based on Lauesen's comparison of ISO9126 and McCall quality factors [29], 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 aspect (our approach was not to impose preconceived definitions but to try to understand existing industrial practice and practitioners' own interpretations of QR). The interviewees gave five points to the most important, four points to the second most important and so on.

Looking at Figure 1, we see that usability (with 50 ranking points) is the highest prioritized quality aspect among our case organizations. In terms of importance, usability is followed by performance requirements (45 points), reliability (32 points), and stability (30 points). Performing the Wilcoxon rank sum test on the vectors of ranks gives statistically significant p-values of 0.05 for usability from stability and for performance from flexibility and down. However, there is no significant difference between the top ranked aspects of reliability, performance and usability.

Several interviewees explained that an unusable product will never sell. The importance of usability was further explained by one interviewee, *"usability is the main competitive advantage, and usability aspects influence the architecture as early decisions"*. One reason for the prioritization of performance, as explained

by several interviewees, is that *"performance requirements are always important since the software developed always act as one sub-system in a larger system"*.

The importance of usability and performance are inline with the findings in Berntsson Svensson et al. [6]. The reason why reliability was ranked as the third most important quality aspects was explained by one interviewee: *"the ability to function in extreme environment and to provide long term value is of major importance for our customer"*. The importance of reliability is confirmed in the study by Johansson et al. [23].

Stability was viewed as important because the customers never accept an unstable product according to one interviewee. Another interviewee explained the importance of having a product that does not have any defects visible to the customer. However, Leung found that the two most important types of QR are availability and accuracy [31], which is *not* inline with the result in this study. In addition, Leung found that performance requirements (time behavior) are only considered the fifth most important quality aspect. The difference between the studies may be explained by the focus, i.e. we focused on B2B and B2C in a market-driven developing context, while Leung focused on intranet applications [31].

Apart from prioritizing performance requirements as the second most important QR, B2B and B2C prioritized different quality aspects. B2B companies ranked safety (25 points), performance (24 points), and reliability (22 points) as the three most important quality aspects, while B2C ranked usability (39 points), performance (21 points), and stability (11 points) as the most important ones.

A Wilcoxon rank sum test showed a p-value of 0.01, showing that safety is more important to B2B than B2C. The importance of safety was explained by several interviewees, *"because we do not sell our products to consumers, instead we sell to other industry partners"*. One interviewee expanded the view by stating that it is important to make sure that no humans, material, or the environment are harmed.

One reason for the prioritization of usability for consumer products is, *"if the product is not usable we will not sell any products"*.

B2B companies uniquely identified safety and accuracy as important quality aspects. On the other hand, B2C uniquely identified portability. The difference in priority between B2B and B2C may not be a surprise as different customers are targeted. However the insight of B2B and B2C having different priorities is an important insight for e.g. researchers, as it enables focus on certain quality aspects in research depending on company type.

4.2 Interdependencies (RQ2)

Based on interviews with requirements engineers at two companies and a proposal, Carlshamre et al. identified a set of six different interdependency types that are present in industry [7]: **(1)** R_1 *AND* R_2: R_1 requires R_2 to function, and R_2 requires R_1 to function, **(2)** R_1 *REQUIRES* R_2: R_1 requires R_2 to function, but

not vice versa, **(3)** R_1 *TEMPORAL* R_2: Either R_1 has to be implemented before R_2 or vice versa, **(4)** R_1 *CVALUE* R_2: R_1 affects the value of R_2 for a customer, **(5)** R_1 *ICOST* R_2: R_1 affects the cost of implementing R_2, and **(6)** R_1 *OR* R_2: Only one of R_1, R_2 needs to be implemented.

The interviewees had the option of adding additional types of interdependencies, which resulted in one new type discovered during the interviews. The new interdependency type, as explained by the interviewee, is that only one of the requirements can be implemented in this project due to time restrictions (mainly including functional requirements (FR) instead of QR). We call this new interdependency type R_1 *TXOR* R_2, meaning R_1 for time-related exclusive or R_2.

All of the other six predefined interdependency types were used by the interviewees to characterize perceived interdependencies, both among different QR, and interdependencies between QR and FR, as illustrated in Table 3. However, Company I did not recognize/identify any of the interdependency types neither among QR, nor between QR and FR, while Company J did not recognize any of the types among QR.

In general, the most common interdependency types identified among QR were *REQUIRES*, *CVALUE* and *ICOST* (13 of 22, as illustrated in Table 4, which shows a summary view of the frequency of occurrence of the various interdependency types), while the least frequently used one was *TEMPORAL* (6 of 22). When the preferences were analyzed based on company type the findings show a difference of opinion.

B2B viewed *REQUIRES* (6 of 12) as the most common type, while B2C viewed *CVALUE* (9 of 10) as the most common one. When examining the most frequent identified interdependency types between QR and FR, *TEMPORAL* (15 of 22) was considered most common, while *OR* (7 of 22) was considered the least frequent one.

However, comparing company types, the findings show a difference. For B2B *TEMPORAL* (8 of 12) was viewed as the most common type, while *AND* and *OR* (4 of 12) were considered least frequent. On the other hand, for B2C *REQUIRES* (9 of 10) was considered most common, and *OR* (3 of 10) was viewed as least frequent by B2C.

In the study by Carlshamre et al., three of five case companies viewed value related (*ICOST* or *CVALUE*) interdependency types as the most common [7]. 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 exceptions of Company E, I, and J).

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. the focus was on requirements in general [7]. Moreover, we looked into interdependencies for the entire product, while Carlshamre et al. focused on dependencies among 20 selected requirements [7]. In addition, we have 11 case companies, compared to five in Carlshamre et al [7]. However, our results of a mix of value

Table 3: Existing Interdependency Types

	Existing interdependency types				**Most important types to identify**			
	QR to QR		**QR to FR**		**QR to QR**		**QR to FR**	
	PM	*PL*	*PM*	*PL*	*PM*	*PL*	*PM*	*PL*
Business-to-business companies (B2B)								
AND	B,H	H	A,B,H	H	H			
OR	B,E,H	B,H	B,E	B,H	E		E	
TEMPORAL	B,H	B,H	A,B,H,J	B,E,H,J		H	H,J	E,H,J
REQUIRES	A,B,H	B,E,H	A,B,H	B,H		B,E		B
CVALUE	A,B	B,H	A,B,H	B,H				
ICOST	A,B,H	B,H	A,B,H	B,H	A,B		A,B	
Business-to-consumer companies (B2C)								
AND	C,F,G	C,F,K	C,G,K	C,D,F,G				
OR	C,D	C,D,F,K	D	D,F		F		F
TEMPORAL		F,K	C,D,K	D,F,G,K				
REQUIRES	C,K	C,D,F,G,K	C,D,F,G	C,D,F,G,K	C	D,G,K	C,D	D,G,K
CVALUE	C,F,G,K	C,D,F,G,K	C,D,F,G	C,D,F,G	F,G,K	C	F,G,K	C
ICOST	C,D,K	C,D,F,G,K	C,D	C,D,F,G	D			

Table 4: Frequency of Occurrence of Interdependency Types

	AND	OR	TEMPORAL	REQUIRES	CVALUE	ICOST
Existing interdependency types						
QR to QR						
Total	9	11	6	13	13	13
B2B	3	5	4	6	4	5
B2C	6	6	2	7	9	8
QR to FR						
Total	11	7	15	14	13	11
B2B	4	4	8	5	5	5
B2C	7	3	7	9	8	6
Most important types to identify						
QR to QR						
Total	1	2	1	6	4	3
B2B	1	1	1	2	0	2
B2C	0	1	0	4	4	1
QR to FR						
Total	0	2	5	6	4	2
B2B	0	1	5	1	0	2
B2C	0	1	0	5	4	0

and functionality related types as the most common ones are inline with a study by Berntsson Svensson et al. [6].

The softgoal interdependency graph (SIG) [9], [10] 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 seven different interdependency types were present in the companies. Furthermore, the types *AND*, and *OR*, where viewed among the least common interdependency types among QR as well as between QR and FR.

RQ2.1: What interdependencies are deemed most important, and how does this compare to previous studies? According to the interviewees, the most important interdependency type to identify among QR was *REQUIRES*; however, B2B and B2C were not in agreement.

B2B companies considered *REQUIRES* and *ICOST* as the most important, while *REQUIRES* and *CVALUE* were prioritized by B2C. One interviewee explained that *ICOST* is important for B2B because *"the cost is lower for R2 if R1 is implemented, then we get a higher return of investment for that requirement"*. Only two interviewees identified *OR*, and only one identified *AND* as the most important interdependency types.

Interestingly, in identifying the most important interdependency type between QR and FR, the total result was identical to interdependency types among QR. On closer examination the result between B2B and B2C varies.

B2B prioritized *TEMPORAL*, but also uniquely identified *TEMPORAL* and *ICOST*. According to one interviewee, *TEMPORAL* is the most important type *"due to high cost and delay in the schedule if this does not work"*. One explanation of why *ICOST* was identified by industry products is, according to one interviewee, *"due to getting a lower cost when designing software from "cheap" hardware components"*.

B2C prioritized *REQUIRES*, but also uniquely identified *CVALUE*. One interviewee explained that *REQUIRES* is the most important interdependency type because *"functionality first, then the quality aspect of the functionality is relevant"*.

Surprisingly, *AND* was not viewed as the most important interdependency type between QR and FR by any of the interviewees. The low result for *AND*, and *OR* interdependency types as the most common and important ones raises the question how useful the SIG [9], [10] is in an industrial context.

It is not surprising that B2B and B2C have different views on interdependency priority. According to Carlshamre et al., value-related interdependencies are subjective; it may be difficult to state whether the cost exceeds the value for the customer [7].

Surprisingly, both among QR, and between 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., which found that *ICOST* and *CVALUE* were the most important types of interdependencies in market-driven developing companies, while *REQUIRES* was considered the most important in

	Elicited		Analyzed		Documented		Impact on product development
	PM	PL	PM	PL	PM	PL	
Business-to-business companies (B2B)							
A	Only during change request	No	In impact analysis	No	In the impact analysis	No	Add features and FR affects the performance
B	During implementation proposal	During the design phase	No	During the design phase	In the implementation proposal	Written down as bundled requirements in the design	Delayed or removed requirements
E	No	No	No	No	No	No	An incomplete product
H	Brainstorming, and in workshops and meetings	During workshops and meetings	Both external (customer) and internal (workshops)	Continuously during workshops and meetings	In different documents	In Excel, PowerPoint, sketches, and by using UML	Order of implementation
I	No	No	No	No	No	No	The design of the system
J	By each engineer and during workshops	No, due to cost	First individually, then in a workshop	Using historical data and workshops	In the requirements specification, for traceability	In the requirements specification, for traceability	Order of implementation of other QR
Business-to-consumer companies (B2C)							
C	No	No	No	No	No	No	Order of implementation, planning and the quality level
D	No	No	No	No	No	No	The maintainability of the system and the planning
F	No	During workshops	No	During workshops by grouping requirements	No	Sometimes, it is optional. Documented in a large html document	Order of implementation
G	During requirements breakdown	No	When breaking down requirements	Not for QR	In the requirements specification	No	Limits possible options
K	No	By the architect	As a consequence of prioritization	Of the architecture	By the design team	In the architecture	Order of implementation and the quality level

Table 5: Interdependency Management

bespoke (also known as contract development [29]) developing companies [7]. One interviewee 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? The interviews indicated a rather good agreement between the PM views and the views of the PL regarding whether or not interdependencies are elicited, analyzed and documented in the case organizations, as can be seen in Table 5.

In two (E, I) of the six B2B companies the PM and PL agree that the case organization does not carry out elicitation, analysis or documentation of interdependencies among QR, while the other four B2B companies (A, B, H, J) indicate activities where interdependencies are managed, some at late stages of design (A, B) and some during workshops and customer meetings (H, J).

For B2C companies it is only one out of five companies for which PM and PL agree that the case organization manage interdependencies to any large extent

(K), while a few indications of limited activities can be found at two other B2C organizations (F, G).

In total for as many as seven out of 11 case organizations, either both or one of PM and PL indicate low extent of interdependency management (A, E, J, C, D, F, G). One reason for this may be that dealing with interdependencies is a complex task and the number of potential dependencies may be very large, resulting in an ad-hoc approach to interdependency management where only a limited set of critical interdependencies are dealt with. This hypothesis is in line with the results from the study by Karlsson et al. [26]. Other explanations were discovered during the interviews. Several interviewees stated that QR are assumed and, therefore, not actively looked for. One interviewee explained that dependencies related to QR are not handled *"because we have a focus on functionality. We fulfill what our customers' want and do not reflect over QR"*.

Concerning the impact of interdependencies on product development, the interviewees stress several examples, as reported in Table 5. As many as five of the case organizations (H, J, C, F, K) report that the order of implementation is affected by interdependencies, and company B indicates that delays or exclusions of requirements may be the result of QR interdependencies, which also affects what is implemented. Furthermore, four of the companies (A, E, C, D) stated that the quality of the product is affected, e.g. performance and maintainability. Also the design of the system may be affected (I, G) in terms of limiting possible options.

4.3 Cost Estimations (RQ3)

Looking at how cost estimations are performed for QR the immediate response from all companies, and roles, was that no distinction was made between quality and functional requirements. Looking at Table 6, all of the companies stated that expert opinion was the predominant method used for estimation, which is inline with the results from Molokken and Jorgensen [33]. Some companies stated that historical data and previous experience of similar projects was used in an ad-hoc manner (E, J, F), mainly as an effect of the fact that the experts had done estimations before. Table 6 summarizes the perceived accuracy of estimations at the companies, for example, Company A the PM answered that in the worst case the estimates were off by 100%, but the normal case was about 25% off, and at best it was accurate. In general, PM estimates the cost to be slightly higher than PL in the normal case (p-value 0.040).

The spread between the companies regarding accuracy is quite large, from e.g. Company C where worst case is 80% off and best is 0% off, to Company B, which ranges from 300% to 50% at best.

Comparing B2B with B2C we can see the average and median for each group in Table 7. The values are calculated by splitting the difference between the roles in each company and excluding company G (in the cases where data is NA the available data is used). From the results we can see that the worst-case alternative

Table 6: Cost Estimation Accuracy and Method

	Role	Accuracy (percentage of wrong estimates)			Type of estimation
		Worst	*Normal*	*Best*	
Business-to-business companies (B2B)					
A	PM	100%	25%	0%	Experts using implementation proposal
	PL	NA	NA	NA	
B	PM	300%	80%	50%	Expert estimations
	PL	200%	25%	10%	
E	PM	75%	50%	25%	Expert estimations and previous projects
	PL	70%	30%	0%	
H	PM	100%	40%	0%	Previous experience and gut feeling
	PL	80%	50%	20%	
I	PM	200%	50%	25%	Expert estimations using related information
	PL	75%	0%	0%	
J	PM	40%	20%	20%	Historical data
	PL	40%	20%	20%	
Business-to-consumer companies (B2C)					
C	PM	NA	NA	NA	Expert estimations
	PL	80%	25%	0%	
D	PM	200%	40%	10%	Expert estimations
	PL	300%	50%	25%	
F	PM	NA	NA	NA	Expert estimations, gut feeling, historical data
	PL	50%	0%	0%	
G	PM	NA	NA	NA	NA
	PL	NA	NA	NA	
H	PM	80%	70%	60%	Expert estimations
	PL	50%	35%	15%	

Table 7: Cost Estimation Accuracy Averages per Group

Type		Worst	Normal	Best
B2B	AVG	115	35	14
	MED	95	33	13
B2C	AVG	55	31	14
	MED	58	35	9

is the largest divider between the groups (B2B: AVG 115, MED 95 and B2C: AVG 55, MED 58). That is, the B2B group in the worst case has much more inaccurate estimates than the B2C group. The difference between the groups is almost non-existent in relation to the normal or best scenario.

From the interviews one reason for this was that the B2B companies had much larger single development instances (projects), and a worst-case scenario in a large project would give a larger estimation inaccuracy than the worst case of a smaller (shorter) B2C project (no statistical significance is claimed).

The interviews revealed that one predominant problem in estimating QR was their propensity to impact larger parts of a system, and span over several (or almost all) FR. For example, realizing most features is influenced by (or influences) the QR of performance. This result is not inline with Molokken and Jorgensen who found the major reasons for inaccurate estimates of requirements are over-optimistic estimates and user changes or misunderstandings [33].

The complexity of QR demands deeper analysis and using the same amount of resources and methods to estimate QR can have detrimental effects. This is strongly related to the level of dismissal of QR late in projects, covered in the next section.

4.4 Dismissal of Quality Requirements (RQ4)

We asked the interviewees how often QR, specified and selected for inclusion in a project, were subsequently dismissed from projects during development (see Table 8). In average, 19% of all QR are dismissed; meaning almost every fifth QR planned and included in a project is dismissed at some stage prior to release. This result is inline with Berntsson Svensson et al., which found that 22.5% of all QR are dismissed at some stage [6]. When comparing B2B and B2C, the least amount of dismissed QR is slightly higher for B2B (4.5%) than for B2C (3%). In worst-case (Most in Table 8); in average, 41% of all QR are dismissed for B2B, while 31% in B2C.

According to the interviewees, there are three main trends of which types of QR that are more representative of the ones being dismissed.

First, there is no difference between types of QR; instead, other factors decide which ones are dismissed. One factor is the FR, one interviewee explained that the FR decides which QR that are dismissed because all QR that are related to the dismissed FR are also removed. A second factor according to one interviewee is that *"usually the QR that are most important for the customer are the hardest to understand, and QR we do not understand are dismissed"*.

Second, for B2C, performance requirements are more often dismissed due to the difficulties in proper estimation. Third, for B2B, QR that are not visible to the customer, such as maintainability and testability, are more often dismissed than other QR.

Table 8: Estimated Dismissal Rates of Quality Requirements

	Role	Dismissal rate			Reason for dismiss rate
		Least	*Avg.*	*Most*	
Business-to-business companies (B2B)					
A	PM	10%	15%	20%	Poor cost estimation, testing QR very late
	PL	0%	50%	90%	
B	PM	10%	20%	90%	Poor cost estimation, lack of resources
	PL	1%	5%	20%	
E	PM	0%	50%	100%	Poor cost estimation, lower priority than FR
	PL	10%	25%	50%	
H	PM	0%	2%	10%	Lower priority than FR
	PL	NA	NA	NA	
I	PM	0%	2%	10%	Lack of resources, QR not quantified
	PL	10%	15%	20%	
J	PM	<10%	0%	10%	Poor cost estimation
	PL	<10%	<10%	10%	
Business-to-consumer companies (B2C)					
C	PM	NA	NA	NA	Poor cost estimation, lack of resources, lower priority than FR
	PL	NA	NA	NA	
D	PM	0%	5%	10%	Issues we cannot affect, e.g. network capacity
	PL	0%	10%	20%	
F	PM	NA	NA	NA	Hardware constraints
	PL	0%	10%	20%	
G	PM	0%	10%	20%	Lack of resources, new product development
	PL	0%	30%	50%	
H	PM	20%	60%	90%	Poor cost estimation, lower priority than FR
	PL	2%	3%	5%	

Moreover, QR that are not considered important for B2B are more often dismissed, for example, usability is more removed than the prioritized performance, which is inline with the result in RQ1 (see Section 4.1). One interviewee explained, *"usability is more often dismissed because the damage for our customers is not that great. It is easier to sell a product with bad usability than with bad performance"*.

The result reveals three main reasons for dismissal of QR:

1. Poor cost estimations
2. Lack of resources
3. QR have lower priority than FR.

There was no major difference between B2B and B2C with regards to why QR are dismissed from projects.

Poor cost estimations is related to the difficulties to estimate the cost of QR that have a global impact on the system, which is inline with the result in RQ3 (Section 4.3). The difficulties of estimating the cost of QR are related to the complexity of QR, lack of knowledge, and understanding of how to manage QR in practice.

Lack of resources is related to prioritization of what is important to implement. One interviewee explained that their resources had more important tasks to do, and there is a lack of resources to optimize QR, therefore it is easier to dismiss them.

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. One interviewee explained, *"in general it is easier to dismiss QR than FR"*. However, this focus has implications on the system, as explained by one interviewee, *"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 pre- or post dismissal? In 82% of the companies it was revealed that there is a lack of communication between PM and PL. PM stated that there is a consequence analysis, while the PL stated that no consequence analysis is preformed.

In 55% of those companies (a mix of B2B and B2C), PM stated that a consequence analysis is only conducted if the customers are directly affected. The consequence analysis may include new prioritization of all requirements and new cost estimations, as explained by one interviewee, *"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 one 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 PL shared the view of the PM. All PL claimed that nothing happens when QR were dismissed from the projects. One explanation, which was qualified by one PL, is that *"we do not have time to re-analyze the consequence of QR, other things are more important"*.

Another explanation according to another PL, *"if we remove QR we can deliver on time"*. Only in two companies (Company J and K) was there an agreement between PM and PL. In Company J, a consequence analysis is performed in terms of new cost estimations, while in Company K no analysis is conducted.

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 is inline with a study by Lehtola and Kauppinen, which found that communication problems were a difficulty for understanding the importance of a requirement [30]. This might indicate a lack of estimation models/techniques for QR.

The result from RQ3 (Section 4.3) shows that the same cost estimation strategy is used for QR as for FR. Maybe QR needs to be estimated with a different strategy. 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.

RQ4.2: Are QR specified in a measurable (quantifiable) manner? Looking at Table 9, in 64% of all companies the view of how often QR are quantified differed between PM and PL. Interestingly, in 82% of all companies, the PL claimed that QR were quantified *sometimes*, while in 67% of these cases the PM views differed, stating *always* or *never*.

For B2B, PM and PL had different views in 83% of the companies, while only 40% of B2C had different views. In total, 36% of all interviewees claimed that QR are always quantified. However, when examining the result for B2B and B2C separately, there is a small difference.

For B2B, 33% of the interviewees stated that QR are always quantified, while 40% in B2C. Only in 36% of the companies (Company C, D, G, and H) an agreement between PM and PL could be observed. Interestingly, three of these companies (C, D, and G) are B2C.

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 [6], [13], [14], and [26].

It is surprising to note that both interviewees stating that QR are *never* quantified represent B2B. This is related to identified challenges with regards to QR and the perceived difficulties in achieving testable QR [6], which may explain why QR are not *always* quantified. Another possible explanation may be that some QR are more difficult to quantify than others, e.g. usability requirements are more difficult to quantify than performance requirements. Kamsties et al. found that the

Table 9: Quantification of Quality Requirements

Company	Role	Answer
Business-to-business companies (B2B)		
A	PM	Always
	PL	Sometimes
B	PM	Never
	PL	Sometimes
E	PM	Always
	PL	Sometimes
H	PM	Sometimes
	PL	Sometimes
I	PM	Always
	PL	Sometimes
J	PM	Never
	PL	Always
Business-to-consumer companies (B2C)		
C	PM	Always
	PL	Always
D	PM	Sometimes
	PL	Sometimes
F	PM	Always
	PL	Sometimes
G	PM	Sometimes
	PL	Sometimes
K	PM	Always
	PL	Sometimes

specification of usability requirements is a challenge [25]. Furthermore, another explanation may be that not all QR are suitable to be quantified, e.g. due to the nature of security requirements, many of them may not be suitable for quantification. In addition, as found in [34], many QR refers to different standards where the quantification part may be hidden.

In a study by Olsson et al., about half of the QR were found to be quantified which seems to confirm the findings [34]. However, one interesting observation that cannot 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.

5 Conclusions

In conclusion, this paper presents the results of an empirical study that examines QR in practice in eleven software companies. Data is collected from eleven PMs and eleven PLs at the companies, thus constituting 22 in-depth interviews in total.

In relation to RQ1, which QR are considered most important, the findings reveal that: (1) in general and especially for B2C, usability is deemed the most important QR, and (2) for B2B, safety is considered the most important aspect. The importance of identifying the needs of a particular company type before dealing with improvements of how to handle QR can of course be discussed.

One solution could be to develop ways (models, tools, techniques) to handle all types of QR for all types of industry. However, the rather immature level of QR handling in industry, in spite of research efforts, may indicate that a blanket solution might not be the best way to go. Rather aimed approaches to e.g. figuring out how companies can handle the most important (to them) QR could be an alternative.

What we can say is that not all QR are equally important for all company types. This insight of difference in priorities is important for e.g. researchers, as it enables focus on certain quality aspects in research depending on company type. These findings complement the findings of [5], [6], [23], which, in part, confirm this; however, it contradicts [31].

The findings for RQ2, interdependencies, show that: (1) *REQUIRES* and *CVALUE* are considered as the most common and important interdependency types to identify, (2) while B2B viewed *REQUIRES* as the most common and important one, B2C considered *CAVLUE*, and (3) *AND*, and *OR* were considered the least common and least important interdependency types. This can be seen as rather interesting as most software engineering research on QR is performed on realization/implementation level where *AND* and *OR* are the dominant perspectives investigated (e.g. architectures, implementation order etc.) [8], [10], [27]. Fewer studies are devoted to for example the value aspects which are conveniently pushed out of the scope of engineering [7].

Interdependencies can have a critical impact on product development in terms of e.g. planning, design and quality. Despite the importance to identify interdependencies, few of the companies actually manage to a large extent to effectively elicit, analyze and document interdependencies. The identification of dependencies is a complex task and the potential number of dependencies may be very large. Therefore, the understanding of which interdependency types are considered most important may give an indication to practitioners of which to start to identify.

For researchers, the knowledge of which interdependency types that exists, are the most common and important ones, provides a focus of what to include in potential new models/techniques for identifying, specifying and managing dependencies. These findings complement the findings of [6], [26], who to part confirms this; however, it contradicts [7], [9], [10]. The impact interdependencies of QR have on product development are uniquely reported by this paper.

In relation to RQ3, cost estimation of QR, the findings show that:

1. There is no distinction between FR and QR during cost estimation
2. Expert opinion is the predominant method for estimation
3. In worst case, B2B has much more inaccurate estimates than B2C.

Difficulties of estimating QR may be related to their large impact on the entire system and span over all most all FR, which makes it hard to estimate the cost. However, inaccurate estimates of QR have major effects on the entire system. Therefore, it is important for practitioners to understand that estimates of QR demand a deeper analysis, and using the same resources and methods for QR and FR may not provide accurate estimates.

For researchers, the inaccurate estimates of QR enable a focus on models/techniques with focus on QR for cost estimation. These findings complement the findings of [25] and [33], who to part confirms this; however, it contradicts [33] with regards to reasons for inaccurate estimates. Furthermore, this paper uniquely reports the perceived inaccuracy of QR estimates, and reasons for inaccurate estimates of QR.

The findings for RQ4, dismissal of QR, reveal that:

1. Close to 1 out of 5 of all QR are dismissed from the projects at some stage during development, with little or no consequence analysis performed
2. For B2C, performance requirements are more often dismissed due to the difficulties in proper estimation, while for B2B, QR that are not considered important, for example, usability is more often dismissed than performance requirements
3. Poor cost estimation and the fact that QR have lower priority than FR are the main reason for dismissal.

The importance of QR needs to be acknowledged by practitioners, not only in theory, but also in practice. Dismissal of QR may solve a short-term problem; however, in the long-term, which the results reveal, the value of the system and the competitive advantage may decrease. These findings complement the findings of [6], [13], [26]. Some QR are easier to specify and test than others, for example, performance requirements are easy to specify and test; however, they affect a large part of FR that may make them difficult to keep in the project. Other QR, for example, usability requirements are more difficult to specify and test. To lower the dismiss rate of QR, by improving the specification and quantification of the more difficult ones, it may be easier to keep them in the project instead of dismissing when running out of time.

In general, the results indicate that there might be a difference in relation to type of company (B2B or B2C). Furthermore, there seems to be a bespoke development mindset where the immediate project gets a higher priority than the long-term evolution of the product. Having an extra function is considered more valuable than higher quality. This contradicts the initial view (RQ1) where QR were labeled as critical. In addition, there seem to be difficulties in committing to QR, especially when there is pressure to deliver the product. However, the main problem is that QR are not taken into consideration during product planning and thus not included as hard requirements in the projects, making the realization of QR a reactive rather than proactive effort. Product management may thus not be able to plan and rely on QR to achieve competitive advantages.

Acknowledgment

This work was partly funded by VINNOVA (the Swedish Agency for Innovation Systems) within the MARS project and by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se). Furthermore, we would like to thank all of the participants and their companies who have helped in making the data collection possible for this research.

Bibliography

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

[2] 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.

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

[4] 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.

[5] R. Berntsson Svensson and A. Aurum. Successful software projects and products. In *Proceedings of the Fifth International Symposium on Empirical Software Engineering*, pages 144–153, 2006.

[6] R. Berntsson Svensson, T. Gorschek, and B. Regnell. Quality requirements in practice: An interview study in requirements engineering for embedded systems. In *Proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 218–232, 2009.

[7] 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 of the Fifth IEEE International Symposium on Requirements Engineering*, pages 84–91, 2001.

[8] L. Chung, B.A. Nixon, and E. Yu. Using non-functional requirements to systematically support change. In *Proceedings of the Second International Conference on Requirements Engineering*, pages 132–139, 1995.

[9] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

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

[11] 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.

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

[13] S. Fricker, T. Gorschek, and M. Glintz. Goal-oriented requirements communication in new product development. In *Second International Workshop on Software Product Management*, 2008.

[14] 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.

[15] T. Gorschek and A. Davis. Requirements engineering: In search of the dependent variables. *Information and Software Technology*, 50(1–2):67–75, 2008.

[16] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. Industry evaluation of the requirements abstraction model. *Requirements Engineering Journal*, 12(3):163–190, 2007.

[17] T. Gorschek and C. Wohlin. Identification of improvement issues using a lightweight triangulation approach. In *Proceedings of the European Software Process Improvement Conference*, pages VI.1–VI.14, 2003.

[18] T. Gorschek and C. Wohlin. Requirements abstraction model. *Requirements Engineering Journal*, 11(1):79–101, 2006.

[19] 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.

[20] M. Ivarsson and T. Gorschek. Technology transfer decision support in requirements engineering research: A systematic review of rej. *Requirements Engineering Journal*, 14(3):155–175, 2009.

[21] M. Ivarsson and T. Gorschek. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering*, 16(3):365–395, 2011.

[22] S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

[23] 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.

[24] 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.

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

[26] 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(6):588–604, 2007.

[27] R. Kazman, M. Barbacci, M. Klein, S.J. Carriere, and S.G. Woods. Experience with performing architecture tradeoff analysis. In *Proceedings of the 19th International Conference on Software Engineering*, pages 54–63, 1999.

[28] 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.

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

[30] 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.

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

[32] M. Lubars, C. Potts, and C. Richter. A review of the state of the practice in requirements modeling. In *Proceedings of the First IEEE International Symposium on Requirements Engineering*, pages 2–14, 1993.

[33] K. Molokken and M. Jørgensen. A review of software surveys on software effort estimation. In *Proceedings of the International Symposium on Empirical Software Engineering*, pages 223–230, 2003.

[34] 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.

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

[36] F. Pettersson, M. Ivarsson, and T. Gorschek. A practitioner's guide to light weight software process assessment and improvement planning. *Journal of Systems and Software*, 8(16):972–995, 2008.

[37] C. Robson. *Real World Research*. Blackwell, 2002.

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

[39] E. Turban, D. King, J.K. Lee, and D. Viehland. *Electronic Commerce: A Managerial Approach*. Prentice Hall, 2006.

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

PAPER II

PRIORITIZATION OF QUALITY REQUIREMENTS: STATE OF PRACTICE IN ELEVEN COMPANIES

Abstract

Requirements prioritization is recognized as an important but challenging activity in software product development. For a product to be successful, it is crucial to find the right balance among competing quality requirements. Although literature offers many methods for requirements prioritization, the research on prioritization of quality requirements is limited. This study identifies how quality requirements are prioritized in practice at 11 successful companies developing software intensive systems. We found that ad-hoc prioritization and priority grouping of requirements are the dominant methods for prioritizing quality requirements. The results also show that it is common to use customer input as criteria for prioritization but absence of any criteria was also common. The results suggests that quality requirements by default have a lower priority than functional requirements, and that they only get attention in the prioritizing process if decision-makers are dedicated to invest specific time and resources on QR prioritization. The results of this study may help future research on quality requirements to focus investigations on industry-relevant issues.

Richard Berntsson Svensson, Tony Gorschek, Björn Regnell, Richard Torkar, Ali Shahrokni, Robert Feldt, and Aybüke Aurum
19th IEEE International Requirements Engineering Conference (RE), Trento, Italy, 29 August – 2 September 2011

1 Introduction

Requirements engineering is a decision-centric process [1], and decision support plays an important role in enabling the delivery of value to stakeholders [22]. Hence, decision support is crucial in achieving value to stakeholders. This is further aggravated in market-driven incremental development, where the situation is even more complex [2], due to that 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 [10].

To deliver business value, a key issue is to decide what to develop; therefore, it is important to make trade-offs between different requirements and stakeholders [27]. Requirements prioritization is an important part in requirements negotiation and release planning [22], [27].

For a product to be successful, it is not enough to fulfill the functional requirements (FR), it is crucial, and challenging, to find the right balance among competing quality requirements (QR). Although literature offers many methods for requirements prioritization, the research on prioritization of QR is limited [5], [12].

This paper presents the results of an empirical study that includes data collected through in-depth interviews with 22 practitioners from 11 different companies. The study focuses on prioritization of QR in industry. This exploratory study can be seen as a study of state-of-practice in industry, but also an investigation of how state-of-the-art in research, in terms of methods and techniques, has penetrated industrial practice.

The study incorporates two main perspectives with regards to prioritization of QR [8], through the study of two roles central to decision making in relation to QR. First, the product perspective is studied through the role of the Product Manager (PM), responsible for the overall product perspective and the selection of the overall planning of the product evolution and proposition offering. Second, the project perspective is studied through the role of the Project Leader (PL), responsible for managing and prioritizing requirements within the realization phases.

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

2 Related work

The quality of a software product is often considered as the ability of the product to satisfy customer and user needs. To increase the chance of a successful product, it is important to find, select, and plan the right releases with suitable requirements [3]. If the ”wrong” requirements are selected and implemented in the product, users may resist buying the product [3]. Decision-makers often face

the challenge of having more requirements than are possible to implement given different constraints, such as time, cost, and other scarce resources. Therefore, it is crucial to distinguish the important requirements from the less important ones to maximize the overall business value [28]. To find the most important requirements that add most value to business, several requirements prioritization approaches are introduced in the literature. A selection of prioritization techniques from literature is summarized in the following subsection.

2.1 Requirements Prioritization Techniques

Numerical Assignment (Grouping) is, according to [3], [21], the most traditional and common prioritization technique. Numerical assignment is based on grouping requirements into different categories, where three groups are common in practice [30]. According to [31], using categories like high, medium, and low may confuse the stakeholders since different stakeholders may have different views of what, e.g. high and medium means.

The pair-wise comparison technique suggested by Karlsson [15] is based on the Analytical Hierarchy Process (AHP) [29]. In this technique, pairs of requirements are compared according to their importance. The comparisons provide an understanding of each requirements share of the total value.

Cost-value approach [16] is a prioritization technique based on AHP. The cost-value approach uses a two-dimension graph that displays the requirements value against its cost. AHP is used from a customer and user perspective to assess the value of each requirement, followed by an assessment of the requirements cost from an implementation perspective. The next step is to plot these into a cost-value diagram, which is used to analyze and discuss the requirements.

Cumulative voting ($100-Dollar Test) is a straightforward prioritization technique where stakeholders are given a fictitious $100 to distribute on requirements. When the money has been distributed, the requirements are ranked so that the highest total reflects the most important requirement, the next highest is the next-most important, etc. on a ration scale.

Ranking is a technique based on an ordinal scale where requirements cannot be tied in ranking, which means the most important requirement is ranked first, and the least important ranked last, on an ordinal scale. It is not possible to see the relative priority difference among the requirements. There are a variety of ways to rank the requirements, e.g. using bubble sort or binary search tree [17].

2.2 Empirical Studies of Requirements Prioritization

So far, to the best of our knowledge, we know of no empirical studies of with specific focus on how QR prioritization is conducted in practice [5]. In this section, we therefore describe a selection of empirical studies based on prioritization of requirements in general.

Karlsson conducted an empirical comparison of the pair-wise comparison technique and numeral assignment technique [15]. Five participants applied the techniques on 14 requirements. Karlsson found that the relative prioritization by pair-wise comparison and that the judging of a requirements relative importance to the other tends to be more accurate than assigning absolute numbers.

Karlsson et al. conducted a self-experiment to compare six prioritization techniques [17]. All three authors prioritized 13 QR using each technique. Karlsson et al. concluded that the AHP was the most promising technique due to providing the most trust worthy results, and it includes a consistency check [17]. However, scalability was identified as a main problem with AHP.

In 2007, Karlsson et al. conducted an experiment of comparing tool-supported pair-wise comparison with planning game (PG) [19]. The results show that tool supported pair-wise comparison was less time consuming than PG. While PG seemed to be more difficult to use, its results were found to be slightly more accurate; however, the differences were not statistically significant.

Lethola and Kauppinen conducted an experiment with industry practitioners to evaluate two requirements prioritization methods, pair-wise comparisons and Wiegers' method [21]. The results indicate that prioritization methods may have limited ability to support decision-making in market-driven product development. Moreover, Lethola and Kauppinen identified which prioritization methods are present in the studied companies [21]. The existing methods are: mutual cost-value analysis, modified Kano model, and evaluating aspects affecting to priorities. In addition, product and project level prioritization practices were identified. At project level, grouping requirements, negotiation, and impact validation were used, while at product level, priority lists and open-ended multirelease planning were used.

Herrmann and Paech conducted two quantitative experiments with students to evaluate two requirements prioritization techniques, risk estimation and ranking [13]. The results highlight challenges of risk estimation and what is important during practical requirements prioritization based on risk estimation.

The focus of the above mentioned studies, with the exception of Lethola and Kaupinnen [21], has not been primarily on how QR are prioritized in industry, but focus on the evaluation and comparison of different requirements prioritization techniques. This paper presents a study with the primary focus on how QR are prioritized in industry. Even though Lethola and Kaupinnen [21] identified which requirements prioritization techniques are used in industry, their focus was not on QR.

3 Research Methodology

The study was carried out using a qualitative research approach [26]. A qualitative research approach is useful when the purpose is to explore an area of interest, and

Table 1: Research Questions

Research Questions
RQ1: How is QR prioritization performed?
RQ2: What criteria are used when QR are prioritized?
RQ3: How does QR prioritization at product and project levels differ?

when the aim is to improve the understanding of phenomena. In addition, qualitative research is directed primarily at collecting and analyzing data with the aim of achieving information depth rather than breadth [6] in an inductive way [7]. The purpose of this study is to gain in-depth understanding of the nature of QR prioritization within market-driven embedded systems companies. The aim is also to provide a basis for future research. Due to the explorative nature of the study, a qualitative approach has been considered suitable. Furthermore, due to the potential richness and diversity of data that could be collected, semi-structured interviews would best meet the objectives of this study. Semi-structured interviews help to ensure that common information on pre-determined areas is collected, but allow the interviewer to probe deeper where required. In addition, by using semi-structured interviews instead of using a large survey, the interviewer had the chance to validate the questions with the interviewee lessening changes of misunderstandings. The research questions in Table 1 provided a focus for the empirical investigation.

3.1 Research Design and Data Collection

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 analyzed. The investigation can be divided into three phases:

Planning/Selection: The first phase of the study involved brainstorming and planning meetings to design the study and to identify different areas of interests. The sampling strategy used was a combination of maximum variation sampling and convenience sampling within our industrial collaboration network [23]. A ”gate-keeper” at each company identified two subjects that he/she thought were the most suitable and representative of the company to participate in this study. Eleven software development companies participated in the study, and from each company, one product manager (PM) and one project leader (PL) from the same project were interviewed, resulting in 22 data points. The interview instrument was designed with respect to the different areas of interest and inspiration from [18]. To test the interview instrument[1], two pilot interviews were conducted prior to the industry study.

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

All eleven companies develop embedded systems using a market-driven software development approach. The included companies vary in respect to size, type of product, and application domain, a rudimentary characterization (following the guidelines of [14]) can be see in Table 2 (more details are not revealed for confidentiality reasons).

Data Collection: The study used a semi-structured interview strategy [26]. One interviewee and one interviewer attended all interviews. 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 [26] involved creating categories where interesting parts from the interviews were marked and discussed. The first two authors examined the categories, first individually and then together in a workshop setting. The category analysis included examination of the content from different perspectives and a search for explicitly stated or concealed pros and cons in relation to prioritization of QR in industry. The results from the analysis are found in Section 4.

3.2 Validity

In this section, threats to validity in relation to the research design and data collection are discussed. We consider the four perspectives of validity and threats as presented in Wohlin et al. [32].

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 [32] was avoided by collecting data from a wide range of sources on the topic of the study. The potential problem of evaluation apprehension [32] was alleviated by the guarantee of anonymity as to all information divulged during the interviews, and the answers was only to be used by the researcher, i.e. not be showed or used by any other participants, companies, or researcher.

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. In order to obtain highly reliable measures, the interview instrument, including posed questions, two pilot studies were conducted prior to conducting the interviews.

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 potential problem of instrumentation threats was alleviated by developing the research instrument with close reference to literature relating to QR, influenced by previously vali-

Table 2: Company Characteristics

	# Employees	***Domain***	***Development process***	***# of Reqs in a typical project***	***% of QR in a typical project***
A	∼100	Control systems	Incremental development	>1000	∼10%
B	∼3000	Telecom	Plan-driven	∼7000	∼10%
C	>5000	Telecom	Plan-driven	>20000	Unknown
D	325	Telecom	Agile-Scrum	∼100 features	∼10%
E	65	Control systems	Waterfall-Iterative	Differs	Differs
F	∼700	Surveillance	Iterative	∼250	∼15%
G	∼100	Consumer electronics	Plan-driven	∼300	∼5%
H	∼700	Telecom	Agile-Scrum variant	∼200	20%
I	∼50	Security	Waterfall and Agile-Scrum	∼100	∼15%
J	∼90	Control systems	Plan-driven	∼100	∼10%
K	280	Telecom	Waterfall - Iterative	∼1000	∼5%

dated interview instrument [18]. In addition, maturation threats were alleviated by reducing the duration of interview sessions by collecting background information before the interview, and by keeping the interview session to 90 minutes. Selection bias is always present when subjects are not fully randomly sampled. However, interviewees were selected based on their roles by a "gate-keeper" at 11 different companies from different geographical locations, which limited the effect of this threat.

External validity: The external validity is concerned with the ability to generalize the results, i.e. in this case the applicability of the findings beyond the included companies. Qualitative studies rarely attempt to generalize beyond the actual setting since it is more concerned with explaining and understanding the phenomena under study. The nature of qualitative designs also makes it impossible to replicate since identical circumstances cannot be recreated.

However, understanding the phenomena may help in understanding other cases and situations. The fact that more than one company acknowledges most of the identified challenges increases the possibility to generalize the results beyond this study. The large number of companies and contexts also contributes to generalizability. To avoid the interaction of selection and treatment, interviewees were selected according to their roles within the company by a "gate-keeper", and companies were selected from different geographical locations.

4 Results and Analysis

The following three sub-sections present and discuss one research question each, corresponding to the research questions in Table 1.

4.1 Prioritization of Quality Requirements (RQ1)

In analyzing Research Question 1 (RQ1), this section examines how QR are prioritized in industry. Table 3 shows what prioritization method was used at the 11 companies (note that Table 3 displays the answer for both PM and PL, i.e. if only one method is shown for one company, both PM and PL gave the same answer).

Looking at Table 3, we see that ad-hoc (14 out of 22) is the dominant requirements prioritization method for QR among our case organizations. Ad-hoc is followed by numerical assignment (6 out of 22), pair-wise comparisons (1 out of 22), and key performance indicators (1 out of 22). When we asked the interviewees how QR were prioritized, six answered that ranking (see Section 2.1) was used (illustrated as numerical assignment in Table 3). However, when the interviewees described how they prioritize QR, the description of the method was similar to numerical assignment (see Section 2.1). This interpretation of ranking differs from the one formulated in literature. This leads to a possible mismatch between the established academic interpretation of ranking and the industrial interpretation of it.

Table 3: Prioritization Methods

Company	***Prioritization method(s)***
A	Numerical assignment
B	Ad-hoc, Numerical assignment
C	Ad-hoc, Pair-wise comparisons
D	Ad-hoc
E	Ad-hoc
F	Numerical assignment
G	Ad-hoc, Numerical assignment
H	Ad-hoc
I	Ad-hoc
J	Ad-hoc
K	Key performance indicators, Ad-hoc

The finding that ad-hoc is the most common prioritization ”method” of QR is not in line with [3], which stated that numerical assignment is the most common prioritization technique. The unstructured prioritization of QR warrants the question if there is any difference of prioritization of QR between mature as opposed to new products, e.g., unstructured prioritization of QR for new products may be expected as the quality may not be well understood for that product, while mature products may involve a more structured prioritization technique. This is however not the most likely case since the focus of this study is how prioritization of QR is performed. If different techniques and methods had been used for mature and new products, the practitioners would probably have mentioned this. Another question lies in prioritization over time, i.e., prioritization of improved quality over the entire lifecycle for the product as opposed to prioritization in the beginning of the project. There may be such differences, but these were not discovered by this investigation as it only reflects on the method used up-front, not specifically how the QR changes over time. Life-cycle priority change analysis is an interesting matter for further research.

Interestingly, while [13], [15], [17], [19], [21] have conducted experiments to evaluate which prioritization technique is more promising than others. Out of all the methods tested, only pair-wise comparisons (1 out of 22 in Table 3) is used at our studied companies.

One possible explanation for this discrepancy may be explained by the focus, i.e. we focused solely on prioritization of QR, while [13], [15], [17], [19], [21] evaluated prioritization techniques for prioritization of requirements in general. However, as many as 14 out of 22 interviewees indicate that FR are prioritized using the same method as for QR. One explanation may be that time to market is important, and to use a more complex prioritization method takes too long, which

was explained by several interviewees. Another possible explanation may be related to customer input. Several interviewees stated that a few important customers are selected to prioritize all requirements for the supplier, despite using a market-driven development approach. No further elaborations were given. Scalability of methods can also be a factor of not using the techniques, which is especially evident with pair-wise comparisons [20].

Ad-hoc prioritization: In total for as many as nine out of eleven case organizations, either both or one of PM and PL indicate that QR are prioritized in an ad-hoc fashion. There are different "approaches" at the studied companies of how QR are prioritized in an ad-hoc fashion: (1) customer input, (2) marketing department, and (3) "gut feeling" (based on the practitioners experience). In three case organizations it is the customer who prioritizes all QR, either direct (provides a list of prioritized requirements to the developing company) or indirect (the PM or PL makes a decision, based on his/her gut feeling/experience, if a particular requirement may be important to their customers). One interviewee explained, *"if a quality requirement is important to the customer, it is a high priority. If it is not important to the customer, it has a low priority, meaning it will not be implemented"*. Another interviewee explained, *"In Scrum, we have a direct prioritization by the customers. If quality requirements are not added and prioritized by the customers, we will not improve the quality of the products"*.

In some case companies, the marketing department handles the prioritization of all requirements, including QR. However, the software department has the "right" to change the list of priorities based on their experience and believes of what level of quality may be expected by the market and potential customers.

In as many as four companies, the PM or PL's gut feeling is the prioritization method. One interviewee explained, *"some people [PM and PL] think quality requirements are important, while others think release time is more important"*. That is, it is up to the PM or PL in charge of a certain product/project to prioritize what is important. Another interviewee further explained, *"some quality requirements (security) are always important for everyone, while others (e.g. usability) are down prioritized by some managers, but some managers consider them important"*. Another factor that affects the decision maker's gut feeling is internal and external stakeholders. The stakeholder who "screams" the loudest is heard, i.e. their preferences are given the highest priority.

Numerical assignment: In literature, numerical assignment is described as putting requirements into different categories based on their importance, e.g. using categories like high, medium and low (see Section 2.1). All interviewees, using numerical assignment to prioritize QR, stated that three categories are used when they prioritize QR, which is in line with [30]. Two different sets of categories are used at our studied companies, 1, 2, or 3 (where 1 is the highest priority), or *critical*, *medium*, and *not critical*, which is in line with the results in [15]. However, in three out of four case companies that use numerical assignment, all QR are by default put into the not critical category, i.e. QR always have lower priority than

all FR. One interviewee explained, *"once we have implemented all functional requirements, then we look into quality requirements. If, and only if there is time and resources available, then we will prioritize our quality requirements and improve the quality of the product"*. In a study by Berntsson Svensson et al, one challenge in managing QR was to get QR into projects when FR are prioritized [4], which is in line with the results in this study.

In the fourth company, QR are by default added to the critical category of requirements. The reason is due to the company's competitors. According to one interviewee, the only way to differentiate from their competitors is to have a product with the highest level of quality, having a competitive advantage.

Prioritization in agile development: Racheva et al. conducted a literature review on agile requirements prioritization methods that can be found in the literature [24]. The result is a list of 15 different requirements prioritization methods; however, none of the found methods in Racheva et al. [24] are used by any of the case organizations (D, H, I) that use Scrum in our study. According to all six interviewees, QR are prioritized in an ad-hoc fashion. One possible explanation to the differences may be explained by the focus, i.e. we focused solely on prioritization of QR. However, the main reason for not using any prioritization method is, according to several interviewees, that *"agile makes projects the focus and not the system, or long term view. This is very detrimental for quality requirements as you do not see the big picture or look beyond your own release"*.

General discussion on prioritization challenges: Looking at how QR are prioritized, most of the studied companies do not use a specific method, while four companies use numerical assignment. However, the ones using numerical assignment do not prioritize QR, instead, all QR are either put into the critical or not critical category by default. We asked the interviewees why prioritization of QR is challenging, or why QR are by default put into a preselected category. Surprisingly, not a single challenge or reason was related to the prioritization itself. However, this result is in line with the results in Lethola and Kauppinen [21]. Their results indicate that the challenges in requirements prioritization are other than to order a list with a set of requirements. In our study, the reasons for difficulties in prioritizing QR are:

- Elicitation of QR
- Lack of well specified QR
- Quantify QR
- What is good enough?
- Knowledge about QR

Elicitation of QR is related to difficulties in identifying the important QR. However, even if the important QR have been elicited, another challenge lies

in specifying QR, which affects the prioritization process. One interviewee explained, *"if the quality requirements are well described, they are as easy as functional requirements to prioritize"*. The importance of writing understandable requirements is confirmed in a study by Karlsson et al. [18]. To be able to prioritize QR, quantified (measurable) QR seem to be of major importance. If the managers do not know which level of quality is expected, they cannot prioritize them. Even if QR are quantified, what is good enough asked one interviewee. The interviewee further explained, *"we must be able to understand what the market needs, and to be able to understand the value and the affects level of quality has on the products, but we do not have that experience"*. There seems to be a lack of understanding and knowledge about managing and handling QR, not only in prioritization, but also from elicitation to understanding market needs. All of these challenges affect the prioritization process of QR, which was confirmed by several interviewees.

Several companies have stated that the main challenge with QR is in the elicitation, specification, quantification, and knowledge about QR, and not primarily in the prioritization of them. This is however relevant as the handling of QR is a chain of events, from elicitation down thorough specification and prioritization. This paper focuses on the latter. Even though many companies have inadequately specified (and elicited) QR all companies have QR in one shape or another. Thus, the prioritization of them, in relation to each other and in relation to FR, is important. The fact that companies use ad-hoc prioritization may be explained by several reasons. For example, ad-hoc prioritization may be adequate for the task in most cases in our studied companies. This is however probably not the case as the dismissal rate of QR during project development is 22.5% [4]. Another reason could be that there is inadequate technology and knowledge transfer [9] from research to industry. Also, a likely explanation could be that there is a lack of usable and useful (scalable) prioritization techniques.

General discussion on comparison between functional and quality requirements prioritization: We believe that the main difference in prioritization of QR and FR is that QR have the potential of being measured with a sliding value on a continuous scale rather than being either included or excluded. The quality level is thus typically not viewed as either good or bad, but rather as something with different shades of goodness on a sliding scale, thus adding another dimension to consider in the prioritization process. However, although many prioritization techniques, e.g., the cost-value approach [16], are mainly used for FR, specific quality targets can of course be included as discrete objects of prioritization in these techniques. This may be one reason to why several subjects stated that the same prioritization methods are used for both FR and QR. The prioritization of what level of quality is needed may be viewed by the subjects as specification and quantification, which may explain why these issues were considered the main challenges of prioritization of QR.

Table 4: Criteria used in QR prioritization

Company	*Role*	*Criteria*
A	PM	Cost, Return of investment
	PL	No criterion
B	PM	Customer input
	PL	Value
C	PM	Value
	PL	Value
D	PM	Cost
	PL	Cost
E	PM	No criterion
	PL	No criterion
F	PM	No criterion
	PL	Value,Customer input
G	PM	No criterion
	PL	Importance of quality requirement
H	PM	Customer input
	PL	No criterion
I	PM	Customer input
	PL	Customer input
J	PM	No criterion
	PL	Customer input
K	PM	No criterion
	PL	Cost/benefit

4.2 Used Criteria when QR are prioritized (RQ2)

We asked the interviewees what criteria are taken into account when prioritizing quality requirements. Looking at Table 4, the only criteria mentioned by more than one practitioner were: no criterion (7 out of 22), customer input (6 out of 22), value (4 out of 22), and cost (3 out of 22).

Looking at Figure 1 and 2, almost the same criteria are used when prioritizing QR by numerical assignment or in an ad-hoc fashion. In both prioritization methods, value estimates, cost estimates, and customer input are taken into account. However, when examining how many interviewees stated they used a certain criterion, there was a difference between the two prioritization methods. Only one (the number within parenthesis in Figure 2) interviewee uses customer input when putting QR into different priority groups, while as many as five (see Figure 1) interviewees prioritizing QR in an ad-hoc way took the customer's input into account. One explanation may be, if the customer provides input to, or actually prioritizes

the requirements, there is no need for the company to prioritize the QR themselves, which is strongly related to the description of how QR are prioritized in Section 4.1.

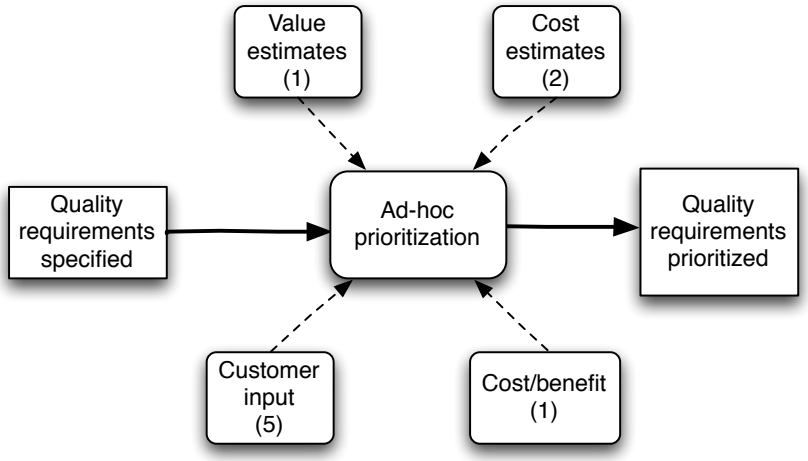


Figure 1: Criteria in ad-hoc prioritization

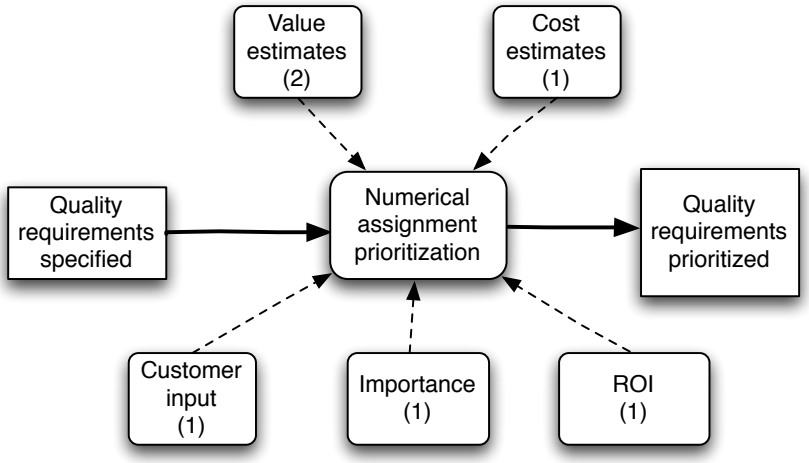


Figure 2: Criteria when prioritizing by numerical assignment

Table 5: Frequency of occurrence of criteria

Criteria	*Total*	*Ad-hoc*	*Numerical Assignment*
No criterion	8	5	2
Customer input	6	5	1
Value	4	1	2
Cost	3	2	1
ROI	1	0	1
Cost/benefit	1	1	0
Importance	1	0	1

In the study by Berander and Andrews [3], six aspects (criteria) of prioritization are presented: importance, penalty, cost, time, risk, and volatility. Our results show that only two (importance and cost) of these six criteria are taken into account when prioritizing QR at our studied companies. According to [3], in practice, it is important to consider multiple criteria, like cost and value, before deciding if a requirement should be implemented directly, later, or not at all. This is not in line with the results in our study, which show only three practitioners use more than one criterion when prioritizing QR.

Although criteria like customer input, cost and value estimations are taken into account when prioritizing QR, in general, the most common criterion is actually having none.

This is illustrated in Table 5 (note that the *total* column in Table 5 includes criteria used in other prioritization methods than ad-hoc and numerical assignment).

Instead of calculating different estimates where the input value is based on ”gut feeling”, most managers at our studied companies prioritize QR based on a ”gut feeling” combined with their own experience. In the study by Lethola and Kauppinen, the results indicate difficulties for practitioners to estimate cost and value for requirements [21], which is in line with the findings of our study.

The top two criteria in ad-hoc prioritization are, to have no specific or explicit criterion defined, and customer input, while when numerical assignment is used, the top two are, no specific or explicit criterion defined, and value estimates. Numerical assignment uniquely uses importance and ROI when prioritizing QR. On the other hand, ad-hoc prioritization uniquely takes cost/benefit analysis into account.

Interestingly, only one interviewee of 22 uses cost/benefit criteria when prioritizing QR. The low result for cost/benefit raises the question how useful prioritization techniques for QR like the cost-value approach [16] are in an industrial context. Furthermore, Herrmann and Daneva found 240 papers on requirements prioritization based on cost and benefit estimation [12]. One may ask if researchers are focusing on the correct set of prioritization criteria given the results.

Surprisingly, not a single one (out of six interviewees) of the practitioner from any of the three case organizations in our study estimates value of each requirement and take this into account when prioritizing QR. No further elaboration was given. This result is not in line with [24], [25], who state that the estimation of business value of each requirement is deemed important. The three criteria taken into account in the prioritization process among our case organizations are: (1) customer input (3 out of 6), cost estimation (2 out of 6), and no specific or explicit criterion defined (1 out of 6). In [25], the practitioners agreed that the developers are active in the requirements prioritization process, which is not in line with our findings. Our results show, in two of three companies, the customers prioritize the requirements at product management level, while only one PL indicated that the customers prioritize their requirements on project level.

4.3 Difference between Product and Project Level (RQ3)

In analyzing Research Question 3 (RQ3), this section examines the difference of prioritization of QR at product and project level respectively, as illustrated in Table 6.

In general, ad-hoc and numerical assignment are the two most common prioritization techniques when prioritizing QR at both product and project level. However, 73% of all PM prioritize in an ad-hoc fashion, while only 55% of the PL. Moreover, 18% of the PM put requirements in to priority groups (numerical assignment), while 36% of the PL used numerical assignment to prioritize QR. This result is partly in line with [21], who found that, at project level prioritization practices, the practitioners put requirement into groups according to their importance. However, in [21], grouping requirements was not present at product management level prioritization practices, which is not in line with our results. The difference between the studies may be explained by the focus, i.e. we focused solely on QR, while in Lethola and Kauppinen the focus was on requirements in general [21]. Moreover, we only focused on companies working with embedded systems, while Lethola and Kauppinen have a mix of companies working with embedded, interactive, and software systems [21]. In addition, we have 11 case companies compared to seven in Lethola and Kauppinen [21].

Looking at Table 6, the most common criterion for requirements prioritization at product level is no criterion, while at project level, no criterion, customer input, and value are equally common among our case organizations. When comparing product and project level, no criterion is used when prioritizing QR in 45% of the companies at product level, while only 27% at project level. This is a substantial difference and seems to indicate a much larger criterion focus in projects compared to pre-project.

Product level prioritization: In general, at product management level, the PM prioritizes, mainly based on his/her gut feeling and experience (no criterion in Table 6), QR at a high abstraction level. In general, QR are prioritized early on

Table 6: Frequency of occurrence at product and project level

	Product level	***Project level***
Prioritization Method		
Ad-hoc	8	6
Numerical assignment	2	4
Pair-wise comparisons	0	1
Key performance indicators	1	0
Criteria		
No criterion	5	3
Customer input	3	3
Cost	2	1
Value	1	3
ROI	1	0
Cost/benefit	0	1
Importance	0	1

in the development process where important information, such as used criteria for prioritization, are specified in a document, which in most of the case organizations is an Excel sheet. Interestingly, although the PM is responsible for all projects related to his/her product, the PM do not have an overall picture of how QR affects different projects and other parts of the system. However, QR are prioritized for several projects by the PM.

Project level prioritization: In general, at project level prioritization, a ”list from above” (from PM) of prioritized QR is given to the PL. Each area (project) should prioritize QR at a lower level of abstraction (the QR are broken down from high to low abstraction level); however, QR may affect the entire system and most, if not all FR. Despite the knowledge of the affect QR may have, none of the PL had an understanding of the affects. In most of the case organizations, QR were not broken down to a lower abstraction level, hence not prioritized, apart from being assigned to the lowest priority group. The main reason for not breaking down QR is because *”it requires the most skilled managers and developers at project level, which we do not have access to”*, according to one interviewee. Lethola and Kauppinen found that negotiation took place in project meetings (at project level), especially when the project group could not implement all the prioritized requirements [21], which is not in line with our findings. If QR cannot be fulfilled, there is no negotiation process. Instead, the PL use a tool supported system to change the priority of the QR, and add a short explanation, which in most cases is related to time restrictions and available recourses.

5 Conclusions

This paper presents the result of an empirical study that examines how QR are prioritized in practice at eleven software companies. Data was collected from 11 PMs and 11 PLs, constituting 22 in-depth interviews in total. We do not claim, in any way, that this paper represents an empirical study that is completely representative of the population. There may of course be companies not part of this study that have different ways of working with prioritization of quality requirements. However, it is relevant to observe that among our 11 case companies, many are market leaders in their respective domain.

In relation to RQ1 that asks how QR are prioritized in industry, the findings reveal that: (1) ad-hoc and grouping (numerical assignment) requirements are the dominant methods for prioritizing QR, (2) although numerical assignment is used frequently in QR prioritization at our studied companies, QR are by default often considered to have the lowest priority, and (3) the reasons for not prioritizing QR was not related to the prioritization process itself, but rather how QR were treated in the overall requirements engineering process.

For researchers, a deeper understanding of QR prioritization in industrial practice provides a focus of what to improve/include in supporting techniques for prioritizing QR. Further, a question that is seldom addressed is how sophisticated the methods actually need to be, and what is desirable according to the practitioners? Many techniques and methods that are developed for requirements prioritization are complex, e.g. techniques based on AHP, but as the results show, complex techniques are seldom used in industrial practice. Instead, simple techniques, such as numerical assignment and ad-hoc prioritization, are used in practice. These findings complement the findings of [3], [21].

The findings for RQ2, regarding criteria used for prioritization, we see:

- It is most common to have no specific or explicit criterion defined when prioritizing QR
- Input from customer is more dominant in ad-hoc prioritization than when using numerical assignment
- Only one interviewee mentioned the use of cost/benefit analysis when prioritizing QR

The result that cost and value estimates are not used as input to the prioritization process of QR may be related to their large impact on the entire system, which make the estimates more difficult to take into account. This result is in line with the findings of [21].

For researchers, the understanding of what criteria are used when prioritizing QR enable a focus on what is needed by practitioners. These findings contradict the findings of [24], [25] with regards to the importance of value estimates when QR are prioritized.

The findings for RQ3, regarding the difference of prioritizing QR at product and project level, reveal that:

- Ad-hoc prioritization is more common at product level compared to project level
- 45% of product managers use no specific criterion when prioritizing QR
- At project level, it is difficult to break down QR to a low enough abstraction level

Difficulties in prioritizing QR at project level may be related to the difficulties of breaking down QR to a lower abstraction level. Therefore, it is important for practitioners to understand, and requires knowledge of how to transform QR to the right abstraction level. The importance of having requirements at the right level of abstraction is in line with [10], [11].

The findings of this paper suggest that there seems to be a lack of knowledge about managing QR. However, the main problem seems to be that QR are by default seen as having a lower priority than FR, and only prioritized if time and resources are available once all FR have been implemented in the coming release. Hence, product management may not be utilizing QR to achieve a competitive advantage, but focusing on FR. Moreover, several of the organizations in this study use the same prioritization method for prioritizing both QR and FR, even though they are very different in nature.

For researchers, the knowledge of the challenges associated with QR prioritization in industry is central. Further, the realization that indirect effects such as difficulties with abstraction level, elicitation of, and specification of, QR can be an issue affecting effective and efficient prioritization, but can also give new avenues of research. Instead of focusing on weather one prioritization technique or method is better than another, the focus could be on improving the practitioners' knowledge and understanding of managing QR as such. It would also be interesting to study how priorities of QR change over the whole product evolution life-cycle.

Acknowledgment

This work was partly funded by VINNOVA (the Swedish Agency for Innovation Systems) within the MARS project and by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se). Furthermore, we would like to thank all of the participants and their companies who have helped in making the data collection possible for this research.

Bibliography

[1] A. Aurum and C. Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945–954, 2003.

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

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

[4] R. Berntsson Svensson, T. Gorschek, and B. Regnell. Quality requirements in practice: An interview study in requirements engineering for embedded systems. In *Proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 218–232, 2009.

[5] R. Berntsson Svensson, M. Höst, and B. Regnell. Managing quality requirements: A systematic review. In *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 261–268, 2010.

[6] L. Blaxter, C. Hughes, and M. Tight. *How to Research*. Open University Press, 2001.

[7] B. Fitzgerald. An empirical investigation into the adoption of systems development methodologies. *Information and Management*, 34(6):317–328, 1998.

[8] T. Gorschek and A. Davis. Requirements engineering: In search of the dependent variables. *Information and Software Technology*, 50(1–2):67–75, 2008.

[9] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE Software*, 23(6):88–95, 2006.

[10] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. Industry evaluation of the requirements abstraction model. *Requirements Engineering Journal*, 12(3):163–190, 2007.

[11] T. Gorschek and C. Wohlin. Requirements abstraction model. *Requirements Engineering Journal*, 11(1):79–101, 2006.

[12] A. Herrmann and M. Daneva. Requirements prioritization based on benefit and cost prediction: An agenda for future research. In *Proceedings of the 16th IEEE International Requirements Engineering Conference*, pages 125–134, 2008.

[13] A. Herrmann and B. Paech. Practical challenges of requirements prioritization based on risk estimation. *Empirical Software Engineering*, 14(6):644–684, 2009.

[14] M. Ivarsson and T. Gorschek. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering*, 16(3):365–395, 2011.

[15] J. Karlsson. Software requirements prioritizing. In *Proceedings of the Second IEEE International Conference on Requirements Engineering*, pages 110–116, 1996.

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

[17] 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.

[18] 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(6):588–604, 2007.

[19] L. Karlsson, T. Thelin, B. Regnell, P. Berander, and C. Wohlin. Pair-wise comparisons versus planning game partitioning - experiments on requirements prioritization techniques. *Empirical Software Engineering*, 12(1):3–33, 2007.

[20] L. Lehtola and M. Kauppinen. Empirical evaluation of two requirements prioritization methods in product development projects. In *Proceedings of the European Software Process Improvement Conference*, pages 161–170, 2004.

[21] 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.

[22] A. Ngo-The and G. Ruhe. *Engineering and Managing Software Requirements*, chapter Decision support in requirements engineering, pages 267–286. Springer, 2005.

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

[24] Z. Racheva, M. Daneva, and L. Buglione. Supporting the dynamic reprioritization of requirements in agile development of software products. In *Proceedings of the Second International Workshop on Software Product Management*, 2008.

[25] Z. Racheva, M. Daneva, K. Sikkel, R. Wieringa, and A. Herrmann. Do we know enough about requirements prioritization in agile projects: Insight from a case study. In *Proceedings of the 18th IEEE International Requirements Engineering Conference*, pages 147–1156, 2010.

[26] C. Robson. *Real World Research*. Blackwell, 2002.

[27] G. Ruhe. Software engineering decision support - a new paradigm for learning software organizations. In *Proceedings of the Fourth International Workshop on Advances in Learning Software Organization*, pages 104–113, 2002.

[28] G. Ruhe, A. Eberlein, and D. Pfahl. Quantitative winwin - a new method for decision support in requirements negotiation. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, pages 159–166, 2002.

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

[30] I. Sommerville and P. Sawyer. *Requirements engineering - A good practice guide*. John Wiley and Sons, 1997.

[31] R. Wiegers. *Software Requirements*. Microsoft Press, 1999.

[32] 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

HOW ARE QUALITY REQUIREMENTS SPECIFIED? A DOCUMENT ANALYSIS CASE STUDY

Abstract

This paper analyses a sub-contractor specification in the mobile handset domain. The objective is to understand how quality requirements are specified and which types of requirements exist in a requirements specification from industry. The case study is performed in the mobile handset domain, where a requirements specification was analyzed by codifying and characterizing the pertaining requirements. The requirements specification is written in structured natural language with unique identifiers for the requirements. Of the 2,178 requirements, 827 (38%) are quality requirements. Of the quality requirements, 56% are quantified, i.e., having a direct metric in the requirement. The variation across the different sub-domains within the requirements specification is large. The findings from this study suggest that methods for quality requirements need to encompass many aspects to comprehensively support working with quality requirements. Solely focusing on, for example, quantification of quality requirements might overlook important requirements since there are many quality requirements in the studied specification where quantification is not appropriate.

Richard Berntsson Svensson, Thomas Olsson, and Björn Regnell

Submitted to *Information and Software Technology*

1 Introduction

Software has become a substantial part of both industrial and consumer products, and as a consequence, the complexity of the software has escalated. Hence, requirements engineering (RE) is a cornerstone in software development, and central for success [30]. A software product's characteristics are not only determined by the functional requirements (FR), but also quality requirements (also called non-functional requirements) [12]. A FR specifies *what* the system should perform, while quality requirements (QR) specify how well it should be performed [12], for example, *"it shall not take longer than 1 second to open the web browser application"*.

To increase the chance of market success, it is important not only to develop a software product that meets customers' requirements and expectations, but also offers high value for the software development company as well as for the customers, hence QR are a key concern throughout the software lifecycle [17], [18], and can be seen as a key competitive advantage [3]. However, despite the importance of QR, it is generally acknowledged that QR are difficult to capture and specify. Several studies, e.g., [5], [10], [12], [11], [19], [25], [29] have identified challenges of QR as: difficult to gather, often poorly understood, general stated informally in a non-quantifiable manner, where should QR document, and difficulties to get attention for QR.

If methods for managing QR are immature and unusable (scalable) in industrial practice [6], a first step towards developing more useful methods for QR, and thereby improving industry practice, is to understand in more detail the problems faced in industry. The importance of well specified and quantified QR have been recognized in the literature. For example, Berntsson Svensson et al. discovered that the difficulties in prioritizing QR are related to, e.g., well specified and quantified QR [6], while Jacobs reported that the introduction of a new method with focus on QR and quantification of QR, enabled the test cases to be based on measurements instead of being untestable [25].

However, to the best of our knowledge, no study has actually looked into a requirements specification in industry to analyze how QR are specified, in particular how QR are quantified, and which types of requirements exists in a requirements specification from industry. This paper presents the results of a case study that includes data collected through a requirements specification that consists of 2,178 requirements from a market-driven development case company. After an early analysis of the requirements specification, a short paper [33] was presented at a workshop. This paper extends our previous report on preliminary findings [33] with more in-depth description of the requirements specification and account of research methodology, as well as a more thorough analysis, discussion, conclusions, and examples of requirements. The study focuses on understanding QR and how they are specified, in particular how metrics are used in an industrial requirements specification within a market-driven embedded system company. The goal is *not*

to test a specific theory or treatment, but to understand a specific phenomenon, namely the requirements specification of QR in a real world industrial situation. Hence, the research approach is open-ended, exploratory qualitative research [38].

The remainder of this paper is organized as follows. In Section 2, the background and related work are presented. The case company is presented in Section 3, while the research methodology is described in Section 4. Section 5 presents the results and relates the findings to previous studies, and Section 6 gives a summary of the main conclusions.

2 Related Work

Research in the area of QR has concentrated on modeling and representation of QR. However, research related to specification, classification, and measurement of QR are also introduced in the literature.

Borg et al. investigated the management of QR in two development organizations [10]. The results show that QR are discovered too late, or not discovered at all; difficulties in prioritization of QR; and difficulties to estimate cost and measures of QR. In another study, Grimshaw and Draper conducted four case studies with an attempt to focus on the QR determination process and improve the understanding [20]. Grimshaw and Draper found that QR are often overlooked, methodologies for QR do not help in the elicitation process, and there is a lack of consensus about quality requirements [20]. Lubars et al. conducted a field study on requirements modeling and found that the rationale of performance requirements is not always obvious, and that usability requirements should not be documented in the requirements specification [32]. In another survey, Kamsties et al. found that requirements are too vague to test, and challenges related to specification of usability requirements were identified [28]. In Berntsson Svensson et al., three important challenges of QR were highlighted: (1) how to get QR into the projects when FR are prioritized, (2) how to know when the quality level is good enough, and (3) how to achieve testable quality requirements [4]. Moreover, in another study by Berntsson Svensson et al., the results show that QR are sometimes specified in a quantifiable manner [5].

In the literature there are several suggestions of how QR should be elicited. Cysneiros and Leite argue that QR should not be dealt within the scope of FR because QR require a more detailed reasoning [14]. On the other hand, Doerr et al. argue that the elicitation of QR, FR, and the architecture must be intertwined because the refinement of QR is not possible without detailed FR and architecture [16]. In addition, Hassenzhal et al. argue that it is important to gather different aspects such as QR, design approach, and the relationships among them to ensure a basic understanding of the design problem [21].

Several studies have addressed the perceived importance of different types of QR. Johansson et al. found that reliability was identified by a multitude of stake-

holders to be the most important QR [26], which was also identified as the most important QR for intranet applications by Leung [31]. Sibisi and Waveren [39] reported functionality as the most important one for two projects, while [15] identified usability requirements as the top QR type. In Berntsson Svensson et al., types of QR were studied from two perspectives, business-to-business (B2B) and business-to-consumer (B2C) [5]. The results show that safety and performance requirements are the two most important QR for B2B companies, while usability and performance requirements were the most important once for B2C companies.

In the literature, only two methods for specifying measurable QR have been empirically evaluated [7], the Gilb style method [25] and the QUPER model [8]. Jacobs introduced and evaluated the Gilb style method at a case company [25]. To make QR measurable, concepts such as scale (the unit in which the requirement should be measured) and meter (how the measurement will be performed) were used. The method puts a focus on QR and a common understanding of QR was considered as crucial. By using the concept of meter, it was found that test cases were already defined during the RE phase. The QUPER model [8] has two main concepts: *breakpoints* and *barriers*. A breakpoint is an important aspect of the non-linear relation between quality and benefit, while barriers represent an interesting aspect of the non-linear relation between quality and cost. The two concepts of breakpoints and barriers provide three views: (1) the *benefit view*, (2) the *cost view*, and (3) the *roadmap view*. Quality indicators are identified to measure the aspects of quality of interests, where a level of benefit offered by competitors are looked at. This information is used to plan needed level of quality for future release of the software product.

Al-Kilidar et al. evaluated the ISO/IEC 9126 [24] standard in terms of its ability to quantify and measure the quality attributes of a software design [1]. The results show that the ”common language” proposed by ISO/IEC9126 did not have a standard interpretation. The authors argue that ISO/IEC 9126, in its present form, does not achieve any of its objectives. Moreover, Berntsson Svensson et al. found that there may be a possible mismatch between the established academic interpretation of quality characteristics of ISO/IEC9126 and the industrial interpretation of it [4].

3 Case Company Description

The case study is conducted at a case company that develops software and hardware for the mobile handset market. The case company has more than 5,000 employees and develops their products, about 20-40 unique mobile phone models each year, for a global and competitive market, where several millions of phones are sold each year. The individual products are developed on a common platform using a product line approach [34], hence, QR are mainly specified for the platform instead of individual products. The case company has several consecutive releases

of a platform (a common code base of the product line) where each of them is the basis for one or more products that reuse the platform's functionality and qualities. The case company has two types of platform releases, a *major* and a *minor* release. A major release has a lead-time between two and three years from start to launch, and the focus is on functionality growth and quality improvements of the product portfolio. Minor platform releases usually focus on the platform's adaptations to different products. Various sub-contractors develop parts of the platform, while the case company itself develops others.

This case study investigates a requirement specification, which is described in the following section, given to a sub-contractor of the case company. This particular sub-contractor provides mobile platform technology for integration into mobile products. However, the sub-contractor does not only provide mobile platforms to the case company, they provide platforms to the case company's competitors as well.

3.1 The Requirements Specification

The sub-contractor specification contains requirements in different sub-domains, which in practice can be seen as a collection of several independent specifications for different sub-domains. The different sub-domains, which range from being very hardware centric to pure software sub-domains with several experts in each sub-domain and little overlap of the expertise across sub-domains, are presented in Appendix. Hence, the different sub-domains can more or less be viewed as independent specifications, written by different practitioners at different point in time and with different ways of specifying requirements. In total, the requirements specification contains 2,178 requirements, including both hardware and software requirements, as well as functional and quality requirements. The requirements specification is written in English using natural language where a typical requirement consists of 1-5 sentences, and the requirements specification is structured hierarchically. The QR range from being pure hardware to pure software related. The requirements specification is focused on enabling technologies rather than end-user requirements as the specification in question is for the core platform, and not end-user applications. The requirements specification is reused over time, where new requirements have been added, while obsolete requirements have been removed.

The sub-contractor uses the specification as the basis for a statement of compliance in the negotiation process with the case company. The specification has been used over a longer period of time for several generations of platforms. Hence, the requirements have been reviewed and used extensively over the years and across several releases. Furthermore, sub-domain experts, usually between 2-10 experts in each sub-domain, write the different requirements that are associated with each sub-domain.

Table 1: Research questions

Research Questions (RQ)
RQ1: How are quality requirements distributed in a requirements specification?
RQ2: How are quality requirements specified, especially how are they quantified?
RQ3: What different types of quality requirements exist in a requirements specification?

4 Research Methodology

The investigation presented in this paper was carried out using a qualitative research approach, namely a case study [37]. Qualitative research aims to investigate and understand phenomena within its real life context [36]. 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 [36], [37]. The purpose if this study is to gain in-depth understanding of how QR are specified, in particular, how metrics are used in an industrial requirements specification within market-driven embedded system companies. Due to the focus of this study, an exploratory case study methodology was chosen since case studies are an in-depth investigation of phenomena on a specific case. In addition, in the case of an exploratory case study, little knowledge about the phenomena is available; hence the study aims at identifying propositions and hypothesis, which can be used in forthcoming confirmative research and empirical studies, such as case studies, of QR. Our approach was to explore how QR are specified without preconceived hypotheses aiming for an unbiased understanding of the case [38]. The research questions in Table 1 provided the focus for the empirical investigation.

As the focus of this study is how QR are specified, a requirements specification (archival data [37]) was analyzed in depth, also called content analysis [36]. A content analysis is an unobtrusive study of an artifact, where the analysis of the content is a quantified codification of the artifact [36], [38]. One advantage of using archival data is that access to the authors of the requirements specification is not needed. Moreover, content analysis is unobtrusive since we are not influencing a practitioner, which is a common risk in interviews. Hence, the analysis in this study is based on the requirements specification alone, along with the experience of the researchers.

4.1 Data Collection and Analysis

Since an exploratory methodology [35] was used in this research, no pre-defined codes were used during data collection and analysis. At the start of the analysis,

much focus and attention were given to the development of a good codification of the requirements. We started out without a defined set of metrics since we did not want to limit the coding in the beginning. Therefore, the first steps of the coding were used to come up with a set of suitable metrics. The coding of requirements was conducted in four steps. In the following, each step is described in detail.

1. **A preliminary coding was performed to identify metrics of interest to be coded in more detail.**
   An overall coding of the entire set of requirements was performed. The goal was to have a first coding of the requirements into classes of functional and quality requirements, and to explore a detailed classification. In this step, all requirements were considered, not just QR. In the sub-sequent steps, the effort is focused on the quality 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, so called observer triangulation [36].

3. **Detailed coding, initial iteration.**
   After the identification of which metrics to collect in the second step, a more detailed coding of sub-domains, scales and characteristics of the QR followed. During the first iteration, the main goal was to get a first understanding to the QR. The requirements coding was performed (by all authors) of different parts of the requirements specification in a random manner. Then, the emerging codes and data were analyzed to derive a consistent and reliable codification. However, not all QR were coded in this step, as the purpose of step 3 was to derive a suitable and consistent coding of the detailed classification. The coding of all requirements was conducted in step 4. Similar to step 2, observer triangulation was used to limit the influences of the individual researcher on the requirements specification.

4. **Final detailed coding.**
   The purpose of step 4, the final detailed coding iteration, was to code all of the existing QR in the requirements specification. This coding provides the final result presented in Section 5. As in steps 2 and 3, observer triangulation was used make sure the codes were consolidated and consistent before finalizing the last step.

In all four steps, the coding was performed in parallel by all three authors. In addition to the consolidating in the last three steps to ensure consistent and reliable

results, there was an overlap of the coding among the three authors. As the four steps describe, the codes and the metrics were built up as the study progressed, and instead of ”forcing” a requirement into a category and avoiding coding an aspect previously not perceived, the coding scheme was updated to ensure that as many relevant suitable metrics and codes as possible were discovered. Step 2 was mainly a learning step for the authors where much time was spent on gaining a common understanding, and a good basis for the continuing of the study.

In the data analysis phase, content analysis [36] and descriptive statistics [37], [40] were used to identify patterns and interesting phenomena, and complemented with examples of requirements to provide further illustrations and background. Given the amount of data and number of ways to dissect it, the analysis was performed iteratively over a period of time.

4.2 Validity

In this section, threats to validity in relation to the research design and data collection are discussed. We consider the four perspectives of validity and threats as presented in Wohlin et al. [40].

Construct validity: The construct validity is concerned with the relation between theories behind the research and the observations, i.e., choosing and collecting the right measures for the concepts being studied. Correct data is a validity threat, in this case whether the requirements specification data is accurately coded (classifying requirements into FR and QR), and that the requirements are up-to-date. As reported in Section 4.1, several researchers performed the coding of requirements in parallel. Most requirements are coded by at least two persons, in order to achieve a high reliability. Furthermore, the evolutionary manner in which the specification is used indicates that the quality of the requirements specification is good, and that the requirements are up-to-date. Otherwise, the requirements specification would not be reused by the case company, which increases the construct validity. Lastly, concerns about the probability that the identified theory is likely to explain the observations are mainly addressed through peer reviews of the result of this and other manuscripts presenting the result.

Conclusion validity: Threats to conclusion validity arise from the ability to draw accurate conclusions i.e. the reliability of the results. Moreover, conclusion validity is related to the repeatability of the study, such as the data collection procedures. To draw accurate conclusions relates to how the data in the case study in question is interpreted, which is dependent on the collected data. Hence, if the description is not valid, the interpretation cannot be valid. This issue is addressed by having a mix of researchers (a practitioner, a senior researcher and a junior researcher). Also, observations are discussed thoroughly and all results explained by providing relevant examples, thereby justifying the interpretation. The threat concerned with the repetition or replication of the study, and in particular that the same

result would be found if re-doing the study, is mitigated by providing a detailed description of the process of coding the requirements.

Internal validity: These threats are 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. One threat in this study is the reliability of the data, and if the data is accurately collected and coded. This threat is mitigated by observer triangulation. Three researchers have independently coded overlapping parts of the requirements specification. Both the overlaps and the uniquely coded parts are reviewed to ensure that the metrics are collected correctly and the codes are accurate and reliable. Also, an audit trail (research notes) was kept, to enable the researchers review not only the outcome but also the process that lead to the outcome.

External validity: The external validity is concerned with the ability to generalize the results, i.e. in this case the applicability of the findings beyond the included company. One threat to external validity in this study is that only one case has been studied. Thus, the context and the case have been described in detail, which supports the generalization of the problems identified. Furthermore, qualitative studies rarely attempt to generalize beyond the setting; it is more concerned with explaining and understanding the phenomena under study. In addition, qualitative studies are impossible to replicate since identical circumstances cannot be recreated. However, developing a theory may help in understanding other cases.

5 Results and Analysis

The following three sub-sections present and discuss one research question each, corresponding to the research questions in Table 1.

5.1 Distribution of Quality Requirements (RQ1)

The requirements specification was analyzed and coded in detail, and the emerging codes (types and characteristics), which are illustrated in Figure 1, are a result of a long process. In Figure 1, we see that a *requirement* (R) can be one of the types *functional* (FR) or *quality* (QR). Since the focus of this study is QR, FR are not further broken down or analyzed.

A requirement has two characteristics, *sub-domain* and *standard* (see Figure 1). Sub-domain is a grouping of requirements into sub-domains of applications, which are detailed in Appendix, for example, network access and multimedia. The characteristic *standard* is a tagging whether or not the requirement is directly, or indirectly referring to a specific standard. For example, to the 3GPP standard, which is commonly used in the mobile handset domain, or to a multimedia standard such as video encoding.

A quality requirement has an ISO 9126 characteristic. This characteristic refers to the standard ISO 9126 [24] and is a mapping of the quality requirements to

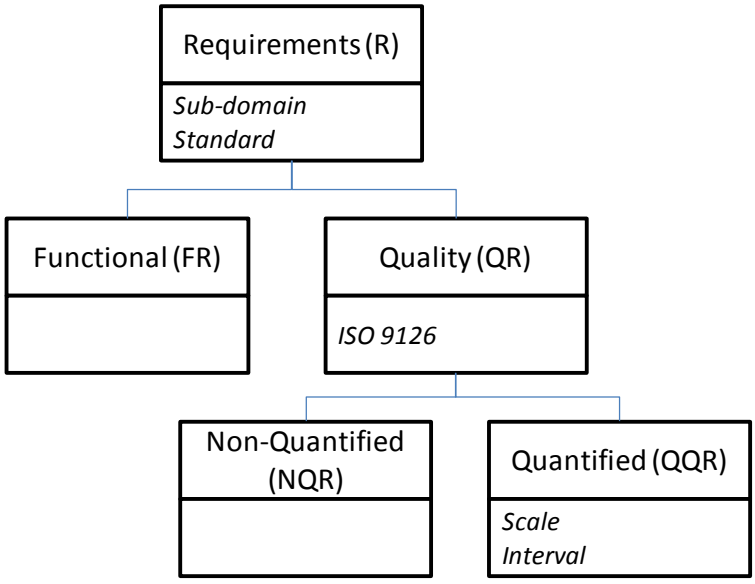


Figure 1: Emerging codes with characteristics

the ISO 9126 standard characteristics (Figure 7, in Section 5.3, shows the ISO 9126 characteristics found in the requirements specification). Although the ISO 9126 standard has been replaced by ISO 25030 [9], ISO 9126 was chosen as the characteristics because of three reasons, (1) the quality model of ISO 25030 is based on the ISO 9126, (2) ISO 9126 is more widespread in industry, and most importantly, (3) the ISO 9126 standard is currently used at the case company.

A quality requirement is further detailed into two types, *quantified quality requirements* (QQR) and *non-quantified quality requirements* (NQR). The QQR type is a quality requirement with a direct quantification within the requirement, while a NQR is a quality requirement without metrics. A QQR has two characteristics:

- *Scale* - whether on a discrete or a continuous scale. Memories, for example, are only available on a 2-multiple scale, e.g., 32 or 64 MB. Typically, 23 MB of memory do not exist. Hence, memory size is a discrete scale. On the other hand, response time is typically on a continuous scale, for example, 4.3 seconds or 22 milliseconds.

- *Interval* - whether the metric is specified as one value, or as an interval (one-sided or double-sided). A QR such as ”Support for encoding frame rate of 15 fps.” does not specify an interval; it is an absolute value. On the other hand, ”The platform shall support an online 90 degree image frame rate at

Table 2: Distribution of requirements across the sub-domains

Sub-domain	FR & QR	FR	QR	Grand total	Total QR (FR & QR + QR)
Architecture	6	199	59	264	65
Audio	21	57	37	115	58
Camera	10	13	12	35	22
Connectivity	25	196	33	254	58
Display	1	19	16	36	17
HW architecture	0	1	9	10	9
IMS	14	22	3	39	17
Industrialization	2	40	21	63	23
Java	24	16	8	48	32
Memory	3	69	32	104	35
Messaging	2	23	3	28	5
Mobile TV	7	32	4	43	11
Multimedia	32	181	14	227	46
Network	7	181	20	208	27
Positioning	2	31	9	42	11
Power	6	23	27	56	33
Radio	125	185	137	447	262
Security	2	7	81	90	83
UI	1	21	3	25	4
Video telephony	4	35	5	44	9
Grand Total	294	1351	533	2178	827

minimum 15 fps." is a QQR with a one-sided interval, and "The frame rate change shall be variable between 15 and five (5) fps." is a double-sided interval.

In total, the requirements specification contains 2,178 requirements distributed over 20 sub-domains, where the number of requirements per sub-domain varies from 10 to 447, which is illustrated in Table 2 (FR & QR is a requirement that have both a functional, as well as a quality aspect, see Section 5.2).

Figure 2 displays the distribution between FR and QR in the requirements specification. The majority of the requirements are FR (62%), while 38% of the requirements contain as aspect of quality, thus coded as QR.

Looking at the distribution of types of requirements across the different sub-domains, the results show that no sub-domain completely lacks QR (see Figure 3). The median is that 35% of the requirements in each sub-domain are QR; however it varies across the subdomains. The sub-domain Network has the least percentage

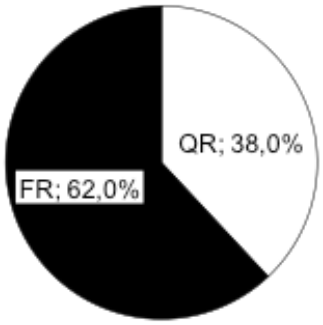


Figure 2: Total distribution of requirements types

(13%) of QR, while the Security sub-domain has the highest percentage of QR (92%).

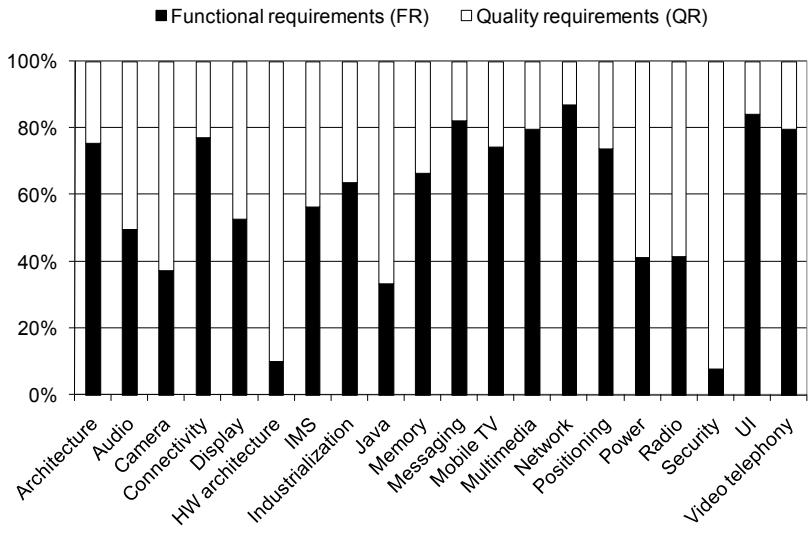


Figure 3: Distribution of FR and QR across the sub-domains

An interesting finding from the codification process is that the use of standards

is common. Standards are found in as many as 38% (in 313 out of 827 QR) of the requirements. An example of a how requirement with a reference to a standard can be specified:

Example 1 *Support for AMR-WB.*

The support for a specific codec is a FR; however, the standard may contain several other requirements, including both FR and QR. Another example of how a requirement can be specified:

Example 2 *Support for H.263 Profile 0, Level 10.*

In Example 2, a specific quality level is explicitly pointed out; hence it is a QQR. In addition, the support for H.263 is specified in another (functional) requirement. A third example of how a requirement (NQR) with a reference to a standard can be specified:

Example 3 *The platform shall be R99 compliant.*

Although the requirement in Example 3 is only one line, the R99 standard is huge and has large implications on the product. The sub-domains that use most standard references are (see Table 2), Messaging (80%), IMS (76%), and Memory (74%), while the sub-domains with least references to a standard are, HW architecture (0%), Audio (0%), Industrialization (0%), and UI (0%).

Discussion: In the analyzed requirements specification, 38% of all requirements (827 out of 2178 requirements) are QR, but the variation across the sub-domains is large. The variation is related to number of QR (from 4 to 262 QR), how QR are specified, and the types of QR that exists. The differences among the sub-domains may be explained by, (1) the technically differences between the sub-domains, ranging from having a main emphasis on hardware, mixed hardware and software, to mainly focusing on software, (2) level of maturity, i.e., for how long requirements for a particular sub-domain have been present in the software product, (3) available resources in terms of practitioners working with requirements, and (4) that some sub-domains may be more critical from a quality viewpoint and more critical to important stakeholders than other sub-domains. Despite the differences across the sub-domains, it seems unlikely that it stems solely from their nature. It seems as if there is a methodological problem as well, with an insufficient support for working with QR. Especially since several sub-domains are deficient on QR, or have a low quantification penetration for QR that should be quantified.

The result that some sub-domains have many QR, while others scarcely any suggests that the priority between FR and QR varies, which is similar to the results in [5]. In our results, 827 QR (38% of all requirements) have been discovered and specified, which suggests that many QR can, and have been elicited and specified. This result is neither inline with the results in [10] where QR were found difficult

Table 3: Use of standard references across the sub-domains

Sub-domain	No standard	Standard	Total QR
Architecture	54	11	65
Audio	58	0	58
Camera	18	4	22
Connectivity	18	40	58
Display	6	11	17
HW architecture	9	0	9
IMS	4	13	17
Industrialization	23	0	23
Java	11	21	32
Memory	9	26	35
Messaging	1	4	5
Mobile TV	9	2	11
Multimedia	32	14	46
Network	9	18	27
Positioning	4	7	11
Power	31	2	33
Radio	167	95	262
Security	43	40	83
UI	4	0	4
Video telephony	4	5	9
Grand Total	514	313	827

to discover, if discovered at all, nor in line with [20] that reports that QR are often overlooked.

Looking into the representation of QR, although the majority of the QR are separated from FR, which is partly in line with [11] who states that QR are usually separated from FR, 36% of all QR (294 out of 827) are requirements with both a functional aspect, as well as a quality aspect (see Example 5). Moreover, according to [11] and to the IEEE Standard 830 - Recommended Practice for Software Requirements Specifications [23], QR are listed separately under different sections in the requirements specification, which is not in line with our results where most QR and FR are grouped into sub-domains based on different areas. In addition, the result that 36% of all QR are a mix of FR and QR suggests that the elicitation of QR and FR needs to be intertwined. This is in line with the elicitation processes in [14] and [16], but not in line with Hassenzhal et al [21] who argues that QR, design approaches, and their relationships should be gathered together, i.e., FR and QR should not be dealt with within the same scope.

The use of standards specifically, but also how much information, such as identification number, QR type, rationale, and originator, a requirement should be comprised of are proposed by some authors, for example [35]. In practice, it is not possible to specify every single detail. However, hiding requirements in standards, or relying on implicit domain knowledge might be risky. It does; however, not appear to change how requirements are written in the analyzed requirements specification. It is not possible to identify any correlation between quantified QR and QR that refers to a standard, or between standards and how requirements are written.

5.2 Specification of Quality Requirements (RQ2)

The requirements specification is written in structured natural language where requirements are organized by the use of heading hierarchies. In addition, all requirements are numbered with a unique ID. In general, a requirement only contains a single requirement, i.e., two requirements are not written as one. The only exception is the mixing of functional and quality aspects within one requirement. Of all requirements (both FR and QR) in the requirements specification, 14% are a mix of FR and QR, while 24% are ”pure” QR. A mixed requirement has both the functional part as well as a quality part included, for example:

Example 4 *(Bluetooth) Support for multi-link. Clarification: X*[1] *simultaneous links.* ***(FR and QQR)***

In Example 4, the functional part is multi-link, while the quality aspect is the number of simultaneous links. Although this requirement is classified as QR, there

[1] Actual numbers will not be entered, for confidentiality reasons.

is a functional aspect, which makes the requirement a mixed requirement. However, typically, there are never multiple QR within one requirement, mixed or not. Figure 4 shows the distribution of mixed FR and QR across the sub-domains.

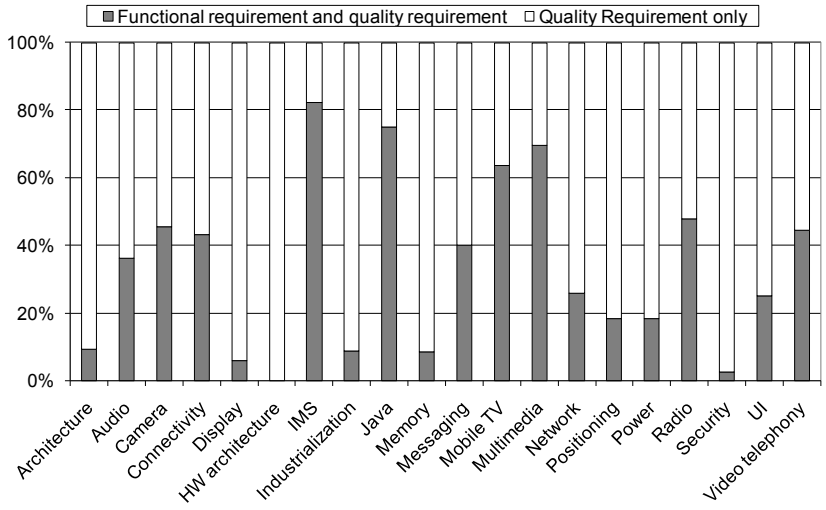


Figure 4: Distribution of mixed (FR and QR) across the sub-domains

Although two requirements are never written as one requirement, it is not uncommon; however, to have multiple quality level for one FR, for example:

Example 5 *"The platform shall receive uncompressed data and shall compress and save the data to desired JPEG size. Clarification: It shall maximum XX s/megapixel to accomplish the whole process for a YX M camera resolution."* ***(FR and QQR)***
"The platform shall receive uncompressed data and shall compress and save the data to desired JPEG size. Clarification: It shall take maximum XY s/megapixel to accomplish the whole process for a YY M camera resolution." ***(FR and QQR)***

In addition, another way of specifying requirements in the requirements specification is to write FR and QR separated:

Example 6 *(Mobile TV) Support for Time Shift (playback with delay).* ***(FR)***
The limit for the time shift buffer is available memory. ***(NQR)***

In Example 6, it has already been specified that there should be a time buffer, thus the NQR specifies the quality level for the time buffer. A third way of specifying a QR to a particular sub-domain is to use the requirements specification's

heading hierarchy. In these instances, the FR is not repeated, but there is one QR for each level of quality. For example:

Example 7
Section 1.2.3. Audio A/D
Support for stereo A/D at 8 kHz. ***(QQR)***
Support for mono A/D at 16 kHz. ***(QQR)***

Example 7 shows the most common way of specifying QR, i.e., by the use of the heading hierarchy within the requirements specification. By specifying QR in this way, it is made implicit for what the QR refers to.

When looking at the distribution of quantified and non-quantified QR, we see that 56% of all QR are quantified with a direct metric (QQR). However, the variance across the sub-domains is large, which is illustrated in Figure 5.

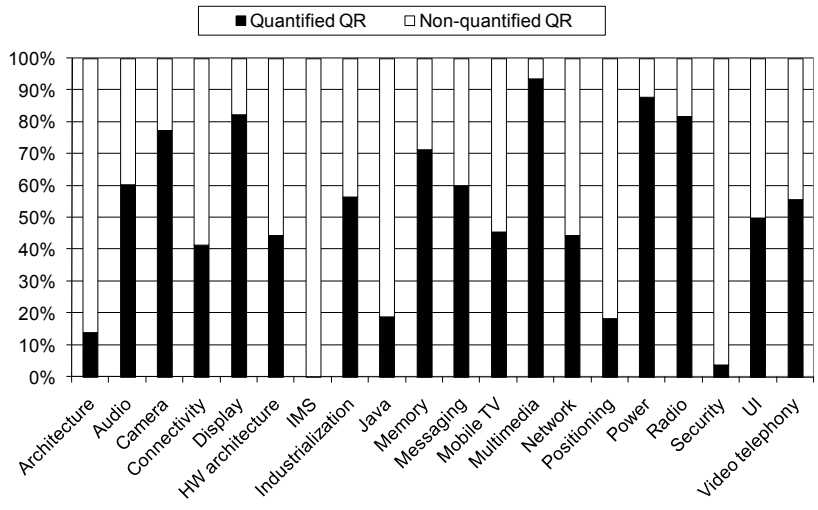


Figure 5: Distribution of QQR and NQR across the sub-domains

In Table 4, the results show that the IMS sub-domain has no QQR at all, and Security has a small portion of QR that are quantified.

One reason why IMS has no QQR may be due to many QR with a reference to a standard, which may hide QQR. The Security sub-domain has many specified QR that are not suitable to be quantified, for example:

Example 8 *Support for X.509 certificates with MD2_RSA signature.*

Furthermore, looking into the distribution of QQR over ISO 9126 characteristics (see Figure 6), eight ISO 9126 characteristics have no QQR, e.g., Replaceability (sub characteristic of Portability) and Understandability (sub characteristic

Table 4: Distribution of NQR and QQR across the sub-domains

Sub-domain	NQR	QQR	Total QR
Architecture	56	9	65
Audio	23	35	58
Camera	5	17	22
Connectivity	34	24	58
Display	3	14	17
HW architecture	5	4	9
IMS	17	0	17
Industrialization	10	13	23
Java	26	6	32
Memory	10	25	35
Messaging	23	3	5
Mobile TV	6	5	11
Multimedia	3	43	46
Network	15	12	27
Positioning	9	2	11
Power	4	29	33
Radio	48	214	262
Security	80	3	83
UI	2	2	4
Video telephony	4	5	9
Grand Total	362	465	827

of Usability), while Security (sub characteristic of Functionality) has few QQR. Among the six main characteristics of the ISO 9126 standard, Efficiency and Reliability have many QQR, while Portability only has a few. Among the sub-domains, the two sub-domains with most QQR are Multimedia and Power (see Example 9 for an example of a QQR).

Example 9 *Support for 2500mA charge current.*

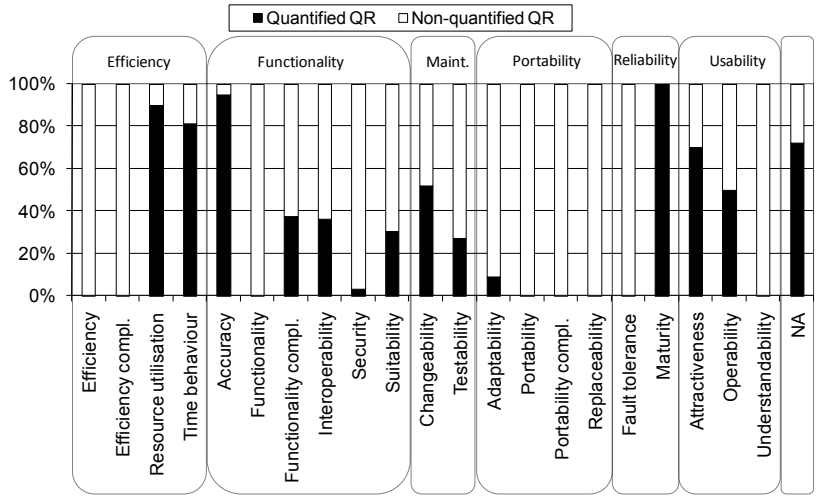


Figure 6: QQR and NQR distribution over ISO 9126 characteristics ("NA" represents requirements with no corresponding ISO 9126 characteristic)

QQR can either be specified using a single absolute value (Example 10) or specified using an interval, single (Example 11) or double-sided (Example 12), for example:

Example 10 *Support for stereo D/A at 8 kHz.(Absolute, no interval, discrete scale)*

Example 11 *The maximum delay from call answer is pressed to opened audio paths is XY ms. (One-sided interval, continuous scale)*

Example 12 *It shall be possible to dedicate a hostbuffer in RAM that is configurable between X to Y MB for HDD. (Double-sided interval, discrete scale)*

Looking into how QQR are specified, 57% are specified with an absolute value (like Example 10), 36% with a one-sided interval (like Example 11), while only 7% with a double-sided interval (like Example 12). The sub-domains with most

QQR using an interval are Industrialization and Power. On the other hand, the sub-domains Java and UI only have QQR with absolute values. As many as 77% of the QQR are quantified using a continuous scale; however, there are sub-domains that mainly use a discrete scale to quantify QR, Display (86% of the QQR are on a continuous scale), Multimedia (81%), and Video telephony (80%).

Discussion: In the literature, several authors [2], [13], [22] have used QR with structured requirements representation notation, for example, combining QR with use cases and misuse cases. However, not a single use case, or misuse case were present in the analyzed requirements specification. Instead, all of the requirements in the specification are written in natural language with a heading hierarchy where all requirements are numbered. However, linking or references outside the heading hierarchy is seldom found. This leads to two problematic situations:

- For mixed FR and QR, the functional part is sometimes repeated several times for each quality level, which leads to redundant text and updating problems.
- It is difficult to get an overview of QR associated with a FR and also if QR are related.

The former is a compromise of readability and maintainability of the specification. It is usually easier to read the FR and QR together, but it does tend to clutter the specification by repeating requirements. The latter puts constraints on how relationships between requirements can be expressed in a usable manner. The mixing of QR and FR may be explained by a lack of understanding that the quality part can be considered to be a requirement on its own.

The interdependencies among requirements can cause problems if ignored [5], which makes it difficult to specify crosscutting concerns. Without a structure for specifying interdependencies among requirements, cross-functional aspects might be difficult to specify. As a work-around, there is a separate section on performance in the analyzed requirements specification, which is suggested by [19], [23] for example:

Example 13
2. Performance
...
2.3 FM-Radio
...
2.3.4 FM-radio Record
Listening to FM-radio with Bluetooth headset and record from the radio at the same time.
Use case extension: handling an incoming call (e.g. MP3 ringtone) without stopping the recording.

The problem with this work-around is that not all requirements on a specific subject, e.g. the FM-radio in Example 13, are specified at the same place in the specification. Some requirements on the FM-radio are in the FM-radio section and some are found in the Performance section. The lack of a good overview of all QR related to a FR might lead to missing QR, as the completeness may be more difficult to assess. For QR, the problem becomes even more severe compared to interdependencies between FR since QR are typically crosscutting and affects other QR as well as FR, both in a positive and negative way. By explicitly documenting interdependencies between QR, it becomes easier to see through the crosscutting aspects [5]. In the analyzed requirements specification, both implicit interdependencies exist and cause problems as well as an inadequate structure forcing cumbersome explicit handling, e.g. through repetition of requirements or deep hierarchies. However, it still needs to be proven to be cost-effective before explicit handling of interdependencies can be said to be a general recommendation in industry.

It is sometimes suggested that all QR should be quantified, e.g., in ISO/IEC 9126 [24]. In the analyzed requirements specification, 56% (465 out of 827) of all QR are quantified. The results show that the quantification is given without an interval and a rationale, which is in line with [28]. Without a rationale and an interval, the developers are left guessing why a specified level of quality was chosen, and how to handle even a small deviation from the target when testing even with quantified QR. Furthermore, our study shows that there is a lack of information, such as intervals and rationales, to deduct when the quality level is reached, suggesting that the observations by Berntsson Svensson et al. [4] can be confirmed in this case study. Moreover, Berntsson Svensson et al. found that quality requirements are sometimes specified in a quantifiable manner [5], which is in line with the findings in presented in this paper. Although many QR are quantified in the analyzed requirements specification, several QR are not quantified, and likely should not be quantified. Based on the findings, there are many examples in this specification where quantification is not appropriate. Security, for example, is one sub-domain (and ISO 9126 characteristic) that sticks out with a low number of QQR (see Figure 6). Also areas such as portability and maintainability have few QQR, suggesting that NQR are also a relevant part of the process and the specification. Therefore, focusing solely on quantifying QR, as in the Gilb style method [25] and the QUPER model [8], is inadequate if applied at the case company. However, the variations in the quantified QR in our results suggest that using a structured and systematic method to achieve quantification could improve the specification of the QR that should be quantified.

There are large differences in how the quantified QR are written. It does not always make sense to quantify QR by using an interval (one-sided or double-sided), e.g. sampling frequency. For memories, for example, only specific sizes exist. It is not possible have 35MB of memory, it is either 32 or 64MB. Therefore, depending on the nature of the specific QQR, the quantification will be different. How

QQR are specified impacts other parts of the development process in addition to the specification. For example, when prioritizing how much memory there should be in a product in the mobile handset domain (more memory means higher cost) the practitioners are faced with a discrete scale. Moreover, during requirements testing, the use of interval quantification can be vital to assess the outcome of the test case. It is rarely the case that, e.g., 4.2 seconds, is the only acceptable outcome. Perhaps anything between 4 and 5 seconds is ok. Hence, using intervals in the QQR clarifies how to interpret the test case outcome. Furthermore, intervals can provide a direction on when, for example, to stop improving the performance. For example, if the target is less than 10 seconds and the current performance are at 1 second, it is unlikely that further improvements will improve the return on investment. Although better performance is in general better, improving performance beyond certain levels will not increase the value for the end customer [8]. By adding a double-sided QQR, it is clearer when further improvements will not increase the market value.

5.3 Existing Types of Quality Requirements (RQ3)

The domain-specific coding developed for this study contains five types of requirements (see Figure 1), 20 sub-domains (see Appendix), and information about the quantification of QR (see Section 5.1). The sub-domains of the specific coding are comparable to the ISO 9126 sub-characteristic, though they may be on different dimensions. Hence, they are complementary, as opposed to competing. To assign an ISO 9126 characteristic and sub-characteristic to a QR requires both understanding of the QR as well as the ISO 9126 characteristic. The main problem experienced with the ISO 9126 coding is that often several of the ISO 9126 characteristics are candidates. In the studied requirements specification, ten ISO 9126 sub-characteristics are not present (see Table 5). In addition, an ISO 9126 sub-characteristic could not be determined for some QR. Instead, only the top-level characteristic is assigned.

Figure 7 shows which ISO 9126 characteristics and sub-characteristics that are present in the different sub-domains, while Table 6 shows the distribution of QQR and NQR across the ISO/IEC 9126 sub-characteristics. Figure 7 and Table 6 show that 39% of the QR are in the Efficiency characteristic and 35% in Functionality. In total, 11% of the QR could not be assigned to any ISO 9126 characteristic and is marked ”N/A” in Figure 7 and Table 6.

Looking at Table 5 and Figure 7, it is not surprising that all sub-characteristics of the Efficiency characteristic are present in the requirement specification, since efficiency is central to the mobile phone. It is a small-embedded system, with an ever increasing amount of uses, pushing the boundaries for that the hardware can deliver. Also, being a telecommunications domain, following standards are central, a major part of the Functionality characteristic. A surprising finding was the lack of maintainability requirements, particular since the development methodol-

Table 5: ISO 9126 characteristics not present in the requirements specification

ISO 9126 Characteristic	ISO 9126 sub-characteristic
Maintainability	Analyzability
	Stability
	Maintenance compliance
Portability	Replaceability
	Installability
	Co-existence
Reliability	Recoverability
	Reliability compliance
Usability	Learnability
	Usability compliance

ISO 9126 Characteristics	ISO 9126 sub characteristics	Architecture	Audio	Camera	Connectivity	Display	HW architecture	IMS	Industrialization	Java	Memory	Messaging	Mobile TV	Multimedia	Network	Positioning	Power	Radio	Security	UI	Video telephony	Grand Total
Efficiency	Efficiency																	4			1	5
	Efficiency compliance														1							1
	Resource utilization	3	1	1	1				1	1	3		1		1		20	45				78
	Time behavior	8	5	8	17	1			9	1	14	1	5	30	5	1	1	130			3	239
Functionality	Accuracy		18			6										2	4	9				39
	Functionality												2					2				4
	Functionality compliance		1	2	11	5		5		19	14	2	2	12	5	2		4			2	86
	Interoperability		7		2													2				11
	Security	1			14			10	1			1				1			65			93
	Suitability	14	4		3	1				5		1			4	3	4		14			53
Maintainability	Changeability	5	2	11		3		2	1	1	2				1		1					29
	Testability	7							11				1		4		1	2				26
NA	NA	16					9								4		1	62	1			93
Portability	Adaptability	5	10			1					1				1	2	1	1	1			23
	Portability														1							1
	Portability compliance				3																	3
	Replaceability	3			1					2									1			7
Reliability	Fault tolerance	3									1											4
	Maturity																	1				1
Usability	Attractiveness		4		5															1		10
	Operability		6		1					3				4						3	3	20
	Understandability																		1			1
Grand Total		65	58	22	58	17	9	17	23	32	35	5	11	46	27	11	33	262	83	4	9	827

Figure 7: ISO 9126 to sub-domain mapping (an empty field means no ISO characteristic found in that sub-domain), ”NA” representing requirements with no corresponding ISO 9126 characteristic

ogy at the case company follows a platform principle; hence reuse of the same platform for many products.

Discussion: To assess a requirements specification for quality of the QR, a detailed analysis is needed. As seen in Sections 5.1 and 5.2, there are large differences across the requirements specification. Therefore, a standard set of metrics is likely to be too general to be useful, not only for practitioners, but also

Table 6: Distribution of QQR and NQR across ISO/IEC 9126 characteristics

ISO/IEC 9126 sub-characteristics	**QQR**	**NQR**	**QR**
Efficiency	0	5	5
Efficiency compliance	0	1	1
Resource utilization	70	8	78
Time behavior	194	45	239
Efficiency Total	264	59	232
Accuracy	37	2	39
Functionality	0	4	4
Functionality compliance	32	54	86
Interoperability	4	7	11
Security	3	90	93
Suitability	16	37	53
Functionality Total	92	194	286
Changeability	15	14	29
Testability	7	19	26
Maintainability Total	22	33	55
NA	67	26	93
NA Total	67	26	93
Adaptability	2	21	23
Portability	0	1	1
Portability compliance	0	3	3
Replaceability	0	7	7
Portability Total	2	32	34
Fault tolerance	0	4	4
Maturity	1	0	1
Reliability Total	1	4	5
Attractiveness	7	3	10
Operability	10	10	20
Understandability	0	1	1
Usability Total	17	14	31
Grand Total	465	362	827

for researchers. Using a generic standard such as the ISO/IEC 9126 does have drawbacks. As the requirements specification is not written with the standard in mind, many QR can be classified as several characteristics and sub-characteristics. Hence, for the sake of the coding, the characteristic that all authors felt was the most appropriate was chosen. This indicates that a tailored model may be more appropriate than the use of a standard model, which is also found in [1], [16], and [27]. Furthermore, the "common language" proposed by ISO/IEC 9126 do not have a standard interpretation, hence, ISO/IEC 9126 in its present form does not achieve any of its objectives [2], which is in line with the results in this study. Moreover, Berntsson Svensson et al. found that there may be a possible mismatch between the established academic interpretation of quality characteristics of ISO/IEC9126 and the industrial interpretation of it [4]. Especially, our results indicate that there is a need to tailor the breakdown of quality attributes to the domain in question. Although a domain-specific method for coding QR requires an initial tailoring to be useful, once the coding scheme is defined, the method can be reasonably reliable and efficient.

Looking into the types of QR that are present in the requirements specification, the results show that efficiency requirements (323 of 827) are the most specified QR type in the requirements specification, followed by functionality (286 of 827) requirements, which is not in line with the findings in de la Vera et al. [15] who found that performance requirements are the third most frequent specified QR type. The most frequent QR type in [15] are usability followed by maintainability.

Among the sub-characteristics in Table 6 (excluding the NA category), the most frequently specified QR types are: (1) Time behavior (239 of 827), (2) Security (93 out of 827), and (3) Functionality compliance (86 out of 827). The frequency of specified time behavior and security requirements implies that these are the most important types of QR to specify, which is not in line with the results in Johansson et al. [26] who found that reliability is the most important QR. In addition, only five reliability requirements are present in the analyzed requirements specification. Moreover, Leung found that the two most important types of QR are availability and accuracy [31], which is not in line with our results. In addition, Leung found that performance requirements (time behavior) are only considered the fifth most important quality aspect. The importance of performance requirements is in line with the findings for B2B companies in Berntsson Svensson et al. [5], which may be explained by the focus of the study. The case company's requirements specification in our study is in the B2B domain, and [5] is the only study to analyze the importance of QR based on type of customers. In addition, usability requirements were not considered as important for B2B companies in [8], which is in line with our results.

6 Conclusions

In conclusion, this paper presents the results of an empirical study that examines how QR are specified in industrial practice at a case company in the mobile handset domain. Data is collected from a requirements specification written in structured natural language that contains 2,178 requirements, whereof 827 (38%) are QR.

In relation to RQ1, how QR are distributed in a requirements specification, the findings reveal that there is a large variation across the different sub-domains. The variation is related to number of QR (ranging from 4 to 262 QR), existing QR types, and how QR are specified, e.g., number of NQR, QQR, and use of standard references. This variation can to some part be explained by the characteristics of the different sub-domains. However, the results indicate that there is a lack of a systematic method for QR, especially since some areas have deficiencies when it comes to QR.

Although relatively many QR exists in the requirements specification, there are areas of improvement. For example, there are very few maintainability requirements in the requirements specification. As the case study company employs a platform development approach, maintainability is a key factor in keeping high quality and reducing effort of using the same platform for several products. It may be, though, that maintainability requirements are hidden in other quality requirements such as portability and functionality. Standards are commonly used in all types of requirements in the case company's requirements specification. However, the phenomena is not well understood when it comes to problems and implications of hiding requirements within standards.

The findings for RQ2, quantification of QR, show that 56% (465 out of 827) of all QR are quantified. This and other variations across the sub-domains suggest that methods for QR need to be able to cope with a variety of QR types. Solely focusing on, for example, quantifying QR might overlook important requirements.

The specification is written in structured natural language without explicit specification of interdependencies across requirements. As QR typically crosscut a functional decomposition, the lack of referencing structure creates obstacles. For example, FR are sometimes repeated several times for each associated QR. Another example is having a separate section for crosscutting concerns, apart from the functional structure of the specification. This causes problems with getting an overview of QR, which may lead to deficiencies in completeness and even contradicting requirements, discovered late in the process. In addition, this may lead to a maintainability problem, as requirements on one subject are spread out in the specification with little or no support for finding them in the specification. Hence, practitioners are reliant on the human factor to find the interdependencies across requirements. By improving the structure for specifying relationships across requirements, many positive effects could potentially be seen.

In relation to RQ3, existing types of QR in the requirements specification, the findings reveal that performance requirements are the most frequently specified

types of QR in the requirements specification, which is not surprising considering the developed software products (small embedded system) at the case company. In addition, the several of the characteristics and sub-characteristics of ISO/IEC 9126 does not exist in the requirements specification, and as many as 11% of all QR could not be classified as any of the characteristics in the ISO/IEC 9126 standard. Using standard methods such as ISO/IEC 9126 may be difficult as it fails to incorporate domain specific aspects relevant for a successful approach. Moreover, the use of ISO/IEC 9126 was more time consuming and less reliable than the tailored codes. Therefore, 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 QR. This impacts all areas of RE, starting with elicitation and analysis to specification and validation.

To complement this study, an interview study with sub-domain experts would improve the understanding of the rationale behind the specifications of requirements. The impact of standards on the requirements practice is only briefly analyzed in this study. Therefore, to further understand the impact of standards, interviews with practitioners are recommended.

Writing requirements in a structured natural language form is commonly used in industry. Despite many years of traceability research and research on dependencies across requirements, the state of practice still struggle with complex interdependencies among requirements. Therefore, it would be interesting to analyze requirements specifications focused on explicit and implicit interdependencies among requirements in general, and QR in particular. This can be performed as a document analysis study and complemented with interviews to get a comprehensive understanding of the problems. It is also important when evaluating methodologies for dependencies among requirements to keep in mind the return of investment. It is not obvious that adding explicit dependencies will be beneficial when analyzing the complete development process, since maintainability problems may occur.

Acknowledgment

This work was partly funded by VINNOVA (the Swedish Agency for Innovation Systems) within the MARS project and by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se). Furthermore, we would like to thank all of the participants and their companies who have helped in making the data collection possible for this research.

Appendix

Architecture	Architecture includes requirements on the architecture as such, e.g. API requirements or componentization of software.
Audio	Requirements related recording and playback of audio are found in this sub-domain, e.g. sampling rate or number of speakers.
Camera	Camera includes requirements on the camera and its interfaces, e.g. resolution support and encoding of pictures.
Connectivity	Requirements related to local connectivity, e.g. USB or Bluetooth^{TM}, as opposed to connections to the mobile phone network.
Display	Display includes requirements on e.g. color depth, resolution or number of displays.
HW architecture	Hardware requirements or mechanical requirements, e.g. component height or pin compatibility, are associated with the HW architecture sub-domain.
IMS	The IP Multimedia Subsystem (IMS) is a framework for delivering Internet Protocol (IP) multimedia services to mobile devices, such as voice or chat applications.
Industrialization	Industrialization includes requirements related to production of devices in factory, such as time to download software to the devices and test log requirements.
Java	Java includes requirements on which Java APIs to support, such as JSR-135, and on behavior of java applications, such as memory requirements.
Memory	Requirements on memories, e.g. RAM bit order or flash memories error handling, found in the Memory sub-domain.
Messaging	Requirements on e.g. SMS or email, are found in the Messaging sub-domain.
Mobile TV	Mobile TV includes requirements on Mobile-TV enables, such as broadcast standard and recording requirements.
Multimedia	Multimedia application requirements, such as encoding and decoding standard and bit rate of video, are part of the Multimedia sub-domain.
Network	Requirements related to GSM or UMTS network access, both circuit-switched as well as packet-switched, are part of the Network sub-domain.
Positioning	Positioning related requirements, e.g. GPS and emergency location services, are found in the Positioning sub-domain.
Power	Power includes requirements on charging, batteries, etc.
Radio	Everything surrounding radio access protocols, such as sensitivity and frequencies, are part of the Radio sub-domain.
Security	Security includes requirements on Security, such as encryption and identification of users.
UI	UI include requirements in, for example, input, such as simultaneous key presses, and output, such as mechanical feedback.
Video telephony	Video telephony related requirements, such as resolution on a video call and protocols, are part of the Video Telephony sub-domain.

Bibliography

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

[2] I. Alexander. Misuse cases help to elicit non-functional requirements. *Computing & Control Engineering Journal*, 14(1):40–45, 2003.

[3] 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.

[4] R. Berntsson Svensson, T. Gorschek, and B. Regnell. Quality requirements in practice: An interview study in requirements engineering for embedded systems. In *Proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 218–232, 2009.

[5] R. Berntsson Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, and R. Feldt. Quality requirements in industrial practice - an extended interview study at eleven companies. *IEEE Transaction on Software Engineering*, 2011. In print.

[6] R. Berntsson Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, R. Feldt, and A. Aurum. Prioritization of quality requirements: State of practice in eleven companies. In *Proceedings of the 19th International Requirements Engineering Conference*, pages 69–78, 2011.

[7] R. Berntsson Svensson, M. Höst, and B. Regnell. Managing quality requirements: A systematic review. In *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 261–268, 2010.

[8] R. Berntsson Svensson, Y. Sprockel, B. Regnell, and S. Brinkkemper. Setting quality targets for coming releases with quper - an industrial case study. *Requirements Engineering*, 2011. In print.

[9] J. Boegh. A new standard for quality requirements. *IEEE Software*, 25(2):57–63, 2008.

[10] 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.

[11] L. Chung and J.C.S do Prado Leite. On non-functional requirements in software engineering. In *Lecture Notes in Computer Science*, volume 5600, pages 363–379, 2009.

[12] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[13] L. Chung and S. Supakkul. Representing nfrs and frs: A goal-oriented and use case driven approach. In *Lecture Notes in Computer Science*, volume 3647, pages 29–41, 2006.

[14] L.M. Cysneiros, J.C.S. do Prado Leite, and J. de Melo Sabat Neto. A framework for integrating non-functional requirements into conceptual models. *Requirements Engineering*, 6(2):97–115, 2001.

[15] J. de la Vara, K. Wnuk, R. Berntsson Svensson, J Sanchez, and B. Regnell. An empirical study on the importance of quality requirements in industry. In *Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering*, pages 438–443, 2011.

[16] 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 of the 13th IEEE International Conference on Requirements Engineering*, pages 373–382, 2005.

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

[18] N.A. Ernst and J. Mylopoulos. On the perception of software quality requirements during the project lifecycle. In *Lecture Notes in Computer Science*, volume 6182, pages 143–157, 2010.

[19] M. Glinz. On non-functional requirements. In *Proceedings of the 15th IEEE International Requirements Engineering Conference*, pages 21–26, 2007.

[20] D.J. Grimshaw and G.W. Draper. Non-functional requirements analysis: Deficiencies in structured methods. *Information and Software Technology*, 43(11):629–634, 2001.

[21] 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.

[22] A. 21. Herrmann and B. Paech. Moqare: misuse-oriented quality requirements engineering. *Requirements Engineering*, 13(1):73–86, 2008.

[23] IEEE. IEEE Recommended Practice for Software Requirements Specifications. IEEE Std. 830-1998, 1998.

[24] ISO/IEC 9126-2001(E). Software Engineering - Product Quality - Part 1: Quality Model, 2001.

[25] S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

[26] 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.

[27] 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.

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

[29] 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(6):588–604, 2007.

[30] 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.

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

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

[33] 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.

[34] C. Pohl, G.Böckle, and F.J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

[35] S. Robertson and J. Robertson. *The Volere requirements process, Mastering the Requirements Process*. Addison-Wesley, 1999.

[36] C. Robson. *Real World Research*. Blackwell, 2002.

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

[38] C. Seaman. Qualitative methods in empirical studies of software engineering. *IEEE Transaction on Software Engineeirng*, 25(4):557–572, 1999.

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

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

THE QUALITY PERFORMANCE MODEL

PAPER IV

VALIDATION OF THE QUALITY PERFORMANCE MODEL: SUPPORTING RELEASE PLANNING OF QUALITY REQUIREMENTS

Abstract

In a competitive open market, as experienced by market-driven software product developing organizations, it is important to plan the product's releases with time-to-market in mind. For a software product to be successful in the market, the software product needs to be released to the market at the right time, and with higher level of quality than its competitors' products. Hence, quality requirements can be seen as a key competitive advantage. This results in challenges related to setting the right quality target in relation to future market demands and competitors' products. When is the quality level a competitive advantage? This situation was identified as a direct need in industry. A model, with three views and a detailed guideline of how to apply the model in practice, was developed in response to the industrial need. The model aims at supporting high-level decision-making, e.g., release planning of quality requirements. The model was validated at a case company in the mobile handset domain with 11 professionals using real quality requirements. The results from the industrial validation point to a relevant and feasible model that is applicable in an industrial environment. The model will allow decision-makers, e.g., product managers, to have more substance to the decisions of what level of quality to aim for in the coming releases.

Richard Berntsson Svensson, Björn Regnell, and Jane Cleland-Huang
Submitted to *IEEE Transaction on Software Engineering*

1 Introduction

Market-driven incremental product development and delivery (release) is becoming more common in the software industry [27]. One goal of market-driven incremental product development is to develop a product with high customer value by delivering an "optimal" subset of requirements in a certain release. In addition, to increase the chances of market success, the software product needs to be released to the market at the right time with higher level of quality than the competitors' products [3]. The decisions (a.k.a. software release planning) about what features and quality at what point in time have to be taken, making these decisions a major determinant of the success of a product [31], [22]. Moreover, lack of good release planning may result in unsatisfied customers [31].

Software release planning is the process of deciding which features should be included in which release [2]. Software release planning is closely associated with product management and requirements engineering decisions. Software release planning is conducted at two levels, strategic and operational [2]. In strategic release planning (SRP), managers prioritize and assign features to releases to meet technical and resource constraints to achieve maximum customer satisfaction. Operational release planning (ORP) takes place after the strategic release planning process where the focus is on one particular release. ORP focuses on assigning developers to tasks to develop the selected features for the particular release. The idea of SRP is to select *what* features and requirements a release should contain, *when* it should be released (time) and at what cost this should be achieved.

An especially challenging problem for an organization that develops software-intensive incremental products offered to a market is to set the right quality target in relation to future market demands and competitor products. When is the quality level good enough? When is the quality level a competitive advantage? In the literature, there are several approaches that provide support for SRP and prioritization, e.g. Release Planning Prototype [9], EVOLVE [13], and release planning through optimization and what-if analysis [1]. These methods use generic algorithms to resolve the release planning issue. Although it may be possible to define release planning as a mathematical optimization problem, it may not be worthwhile to apply complex mathematics or advanced computational algorithms to achieve "optimal", if the input data to the optimization process is highly uncertain. To the best of our knowledge, few studies have looked into SRP and prioritization of quality requirements (QR), despite the fact that QR are of major importance for market success [2], [14]. According to [30], only two SRP methods address quality constraints. The quantitative Win-Win model [26] addresses effort and time constraints, but not the quality level of QR. Arguably, then, the only method to address quality and cost constraints of QR is the QUPER model [30].

The QUPER model, (QUality PERformance model), was developed to support release planning and roadmapping of QR, more specifically with the goal to provide concepts for qualitative reasoning of orders of magnitude rather than precise

mathematical formulas. QUPER was developed at a case company in the mobile handset domain [23], while abstract generic guidelines of how to apply QUPER in practice were developed in cooperation between academia and industry [21].

This paper is based upon previous work published in [6], [7], [8], [21], [23] where different aspects of the QUPER model were introduced. We are now in a position to present a complete version of the QUPER model. This paper adds the following contributions to our previous investigations of QUPER:

- The added step of how to incorporate cost dependencies between QR.
- The added step of defining scale and unit, for example, if a performance requirement is added, should the quality level be measured in milliseconds or seconds?
- A detailed step-by-step practical guideline of how to apply QUPER in practice, with an illustration of a QR from the first to the last step.
- A complete overview of QUPER's steps and workflow.
- A new validation of the complete version of the QUPER model with 11 professionals at a case company to evaluate QUPER's usefulness and applicability using the detailed guidelines in a non-simulated environment using real quality requirements.

The remainder of this paper is organized as follows. Section 2 offers an overview of related work, while background and motivation is presented in Section 3. Section 4 offers an introduction and exemplification of the QUPER model. Section 5 presents how QUPER was evaluated at the case company, and lessons learned are discussed. Limitations of the study are discussed in Section 6, while Section 7 gives a summary of the main conclusions.

2 Related Work

The process of selecting requirements for a certain release of a software product is called release planning, where the primary goal is to maximize the expected value of a product release. Several approaches and strategies have been proposed to resolve issues related to requirements selection and prioritization. In this section, a selection of release planning methods is presented: EVOLVE [13], EVOLVE* [28], F-EVOLVE [19], Release Planner Prototype [9], and a method for software release planning using optimization of what-if analysis [1].

EVOLVE [13] is an evolutionary and iterative method that looks ahead for more than one release. The method balances stakeholders' conflicting opinions to achieve the highest degree of satisfaction with the available resources. EVOLVE* [28] is a hybrid intelligent framework with the objective to create synergy between

computational intelligence and the knowledge and experience of human experts. Software product release planning through optimization and what-if analysis [1] is a method that applies mathematical programming to provide a solution for the next release problem. The release planner prototype [9] 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.

All of these release planning methods use generic algorithms and linear programming to resolve the release planning issue. Input data such as requirements value, requirements cost, resources, stakeholder importance, and budget constraints are used to come up with an "optimal" release. In addition, analysis of requirements dependencies is present in these methods. However, none of the methods address quality constraints. According to Svahnberg et al., only two strategic release planning methods address quality constraints [30]. The quantitative Win-Win [26] addresses effort and time constraints, but not the quality level of quality requirements. The only method to address quality and cost constraints of QR is the QUPER model [30]. For a more comprehensive analysis of 24 strategic release planning methods, we refer to Svahnberg et al. [30].

Prioritization of requirements is often conducted prior, or as a part of the release planning process. Several prioritization techniques and cost-benefit models are introduced in the literature. The contributions in this area include: Kano [15], Planguage [11], a cost-value approach [17] based on the analytical hierarchical process (AHP) [29], and quality function deployment (QFD) [16]. Planguage has roadmap related concepts such as past, record, and trend in templates for quality requirements. QUPER could be used together with the planguage method to express breakpoints, barriers, and targets related to, for example, express competing products in different market segments.

The cost-value approach [17] uses a two-dimensional graph that displays the requirements value against its cost. AHP [29] is used from a customer and user perspective to assess the value of each requirement, followed by an assessment of the requirements cost from an implementation perspective. The next step is to plot these into a cost-value diagram, which is used to analyze and discuss the requirements. This approach, supporting trade-off analysis and is mainly used for functional requirements. Quality requirements can be included as objects of prioritization in AHP, but as discrete objects are compared against each other, the relation to a sliding scale of quality is not explicitly addressed. The QUPER model thus goes further by introducing a third dimension related to the continuous (or a set of discrete points on a scale) nature of quality attributes. There are potential strategies for combining QUPER with AHP-based approaches, e.g. by comparing breakpoints of different use cases.

Similar to QUPER, Kano et al. [15] developed a model for evaluating patterns of quality. The evaluation is based on customer's satisfaction with specific quality attributes. Kano's model explains the relationship between customer satisfaction and the degree of achievement of a specific quality attribute in a two-dimension

graph. Kano's approach views quality as non-linear. However, Kano's approach does not include a cost dimension. Moreover, Kano's model is not related to roadmapping, benefit breakpoints, or cost barriers to indicate important aspects of quality relations.

QFD [16] is a comprehensive, customer and user oriented approach to product development. The QFD process starts by organizing the project, including the formation of a cross-functional team, followed by the establishment of relationships among requirements and then prioritization. To fully implement QFD, customers and users need to be visible; however, not all market-driven projects have access to customers and users. Furthermore, QFD measures quality attributes using a scale where no clear distinctions between the values are provided. While QFD may require a complete change of current practices, QUPER is a simple reference model to be used in combination with current practices to support communication of quality attributes.

3 Background and Motivation

The development of QUPER was prompted by the faced challenges of rapid technology development in combination with increasing market demands on expanding product portfolios targeting a wide scope of different capabilities and price ranges in the mobile device industry [24]. Moving towards rapidly changing market requirements and environmental regulations has urged dramatic changes in software companies for future economic survival. Moreover, global competition forces companies to become more competitive and responsive to consumers and market developments, and creating value for software companies is more important than ever before.

Software quality is not only defined by the relevant perspectives, but also by the context in which it exists [18]. For example, just as each line of cars has a target market, software quality must be planned to allow a development company to meet its business objectives. Less than perfect software quality may be ideal [32], but deciding what is good enough can only be decided in a given business context [18]. Thus, the tough question to answer is 'when is the quality level good enough'? There is no silver bullet to this question; it must consider the context in which it is being asked.

The need for a supporting model for handling and working with quality requirements in this context was also explicitly identified during an investigation of the cross-company requirements engineering process between two case companies [24]. Furthermore, the companies explicitly stated the importance of having a handle on QR. The importance of having a handle on QR has been confirmed in Berntsson Svensson et al. [4].

3.1 Requirements Decision Assessment Results

During the requirements decision assessment conducted at the cross-company requirements engineering process between two case companies [24], in total 16 requirements compliance (decision) issues were identified and formulated (see [24] for detailed information). The QUPER model was primarily built on five of the 16 issues in [24]. The five issues are presented and described below:

- Issue-1: *Transient inclusion:* The assessment findings identified a typical reason behind *transient inclusion* is that a certain requirement may have been excluded due to technology development or market considerations. Therefore, being responsive to technology and market changes is considered crucial.

- Issue-2: *De-scoped to next:* The assessment findings identified several reasons for a requirement to be *de-scoped to next* release, including: (1) the customer has changed the requirement's priority due to, e.g. market changes; and (2) unrealistic cost estimates, thus not possible to implement within available resources. Due to great uncertainties in the market and cost estimations, improved techniques for cost estimations may minimize these uncertainties.

- Issue-3: *Change inclusion* is a typical change request with the result of an accepted change request. Hence, the requirement is allowed to change the plans of the on-going development. Several reasons for change requests were identified between the two cross-companies, including late discovery of (1) system performance problems, and (2) specification quality problems. The specification quality problems resulted in misaligned interpretation of what should be delivered.

- Issue-4: *Change reconsidered* scenarios result from rejected changes that are reconsidered during on-going development; however, these scenarios are a very uncommon. One reason behind change reconsidered is reconsidered estimations, e.g. product market value, due to a changing environment and uncertain predictions of future market needs.

- Issue-5: *Proprietary solution chosen.* One reason to make a proprietary solution is due to competitive advantage (in relation to competing product vendors). Even if the company offers a solution of a certain feature, it may not be as advanced or as high performing as the competing products.

3.2 Motivation

Two main factors motivated the creation and evolvement of QUPER: (1) a direct need identified in industry (see Section 3.1) and (2) a suitable model was not

found in the literature, i.e. a model for supporting release planning of quality requirements (see Section 2).

Regarding the industry need (1) there was an expressed interest from the two cross-companies to improve the way of working with quality requirements towards the needs of the market. The actual need for this type of model has become even more apparent after the initial development of QUPER. A different organization in a different domain than the mobile handset showed an interest in applying QUPER to their organization [8], due to which they experienced similar challenges as the two cross-companies.

Looking at the state of art, there is a research being conducted in the area of strategic release planning in a market-driven development situation. Although there is an identified need in industry to support QR in release planning (described in Section 3.1), and it is important to have a handle on QR [4], there is a lack of an appropriate model (2). Offering support for release planning of QR prompted the effort to develop QUPER in a generic way for organizations faced with certain issues, rather than tailoring the model towards on organization.

3.3 Evolvement of the Quality Performance Model

The QUPER model was developed in several stages, in close collaboration with industry, where the process can be described by the technology transfer model's step 1-3 and 5-6 [12], as illustrated in Figure 1.

Subsequent to (1) the initial identified improvement areas in industry (see Section 3.1), (2) problem formulation and studying the research field and domain (see Section 2), and (3) formulation of a candidate solution in cooperation with industry (see [23]), QUPER went through four (two *static*, e.g., interviews, and two *dynamic*, e.g., pilot projects and controlled small tests) major validations. The *static validation* (step 5 in Figure 1) steps involved brainstorming and interview sessions with selected practitioners that are involved in/responsible for release planning of quality requirements. The *dynamic validations* (step 6 in Figure 1) consisted of using QUPER for a real live release planning effort. Each round of the validation processes was used to refine the model in terms of contents, structure, as well as testing issues relating to usability and usefulness.

First, (5) a *static validation* with four practitioners was conducted to evaluate QUPER's benefit view [21], followed by a (6) *dynamic validation* of the benefit view where four practitioners applied the practical guidelines of the benefit view for 3 months in a non-simulated environment in real projects using real requirement [6]. Then, a second (5) *static validation* with eight practitioners was conducted to evaluate QUPER's cost view [7]. Finally, a second (6) *dynamic validation* was conducted of QUPER's cost and benefit view where four practitioners applied to concepts in real projects using real requirements [8].

The version of QUPER presented in this article (see Section 4), i.e. the model that evolved as a result of the validations, is the first version that includes all

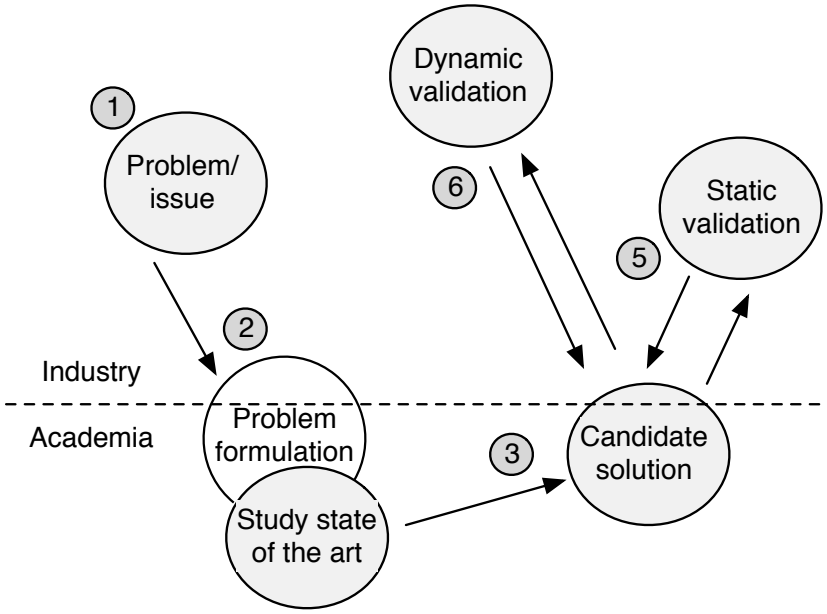


Figure 1: Overview of QUPER evolvement [12]

steps and the detailed guidelines of how to apply QUPER in practice. QUPER is mainly aimed towards establishing support for discussion and decision-making in upstream requirements engineering related to, e.g. release planning and roadmapping.

4 QUPER Structure and Guidelines

This section describes QUPER's structure and the supporting guidelines of applying QUPER in practice developed as part of it.

The new contributions of this paper (in Section 4) are: (1) the added step of how to identify cost dependencies between QR (Section 4.7), (2) the added step of defining scale and unit (Section quper:step2), (3) the detailed step-by-step practical guidelines with an illustration of a QR (Sections 4.1-4.7), and (4) a complete overview of QUPER's steps and the workflow of QUPER, which is shown in Appendix (parts of the Appendix have been published previously in [4]).

The reason for adding the cost dependency step is because dependencies may have a major impact on the estimated cost for other QR. The cost to improve the quality level for one QR may imply an improved level of quality for other QR.

This may lead to a change of other QR cost barriers and which QR to select for the coming release. Therefore, it is important incorporate a cost dependency step in the QUPER model.

The QUPER model aims to support the ability to make early estimates with adequate accuracy of QR that are input to discussion and high-level decision-making related to, e.g. release planning and roadmapping. One objective of QUPER is to define a prioritization model that includes a third dimension related to quality, as a complement to cost and value that are used in prioritization of functional requirements [17].

The basis for the construction of QUPER and its three (benefit, cost, and roadmap) views is the concepts of *breakpoints* and *barriers*. The first view, the benefit view (Figure 2), illustrates the relation between quality and benefit in terms of three breakpoints. The *utility breakpoint* marks the shift from useless to useful quality, while the *differentiation breakpoint* marks the shift from useful to competitive quality (which only a few competitors' products reach). The *saturation breakpoint* marks the shift from competitive to excessive quality.

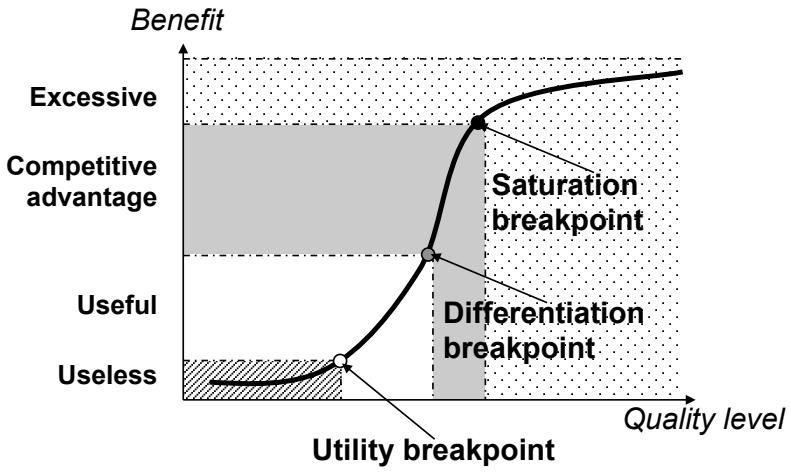


Figure 2: The benefit view [23]

The cost view (Figure 3) illustrates the relation between quality and cost in terms of *cost barriers*. A cost 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.

The roadmap view (Figure 4) combines the benefit and cost view by positioning the breakpoints and cost barriers on the same scale, which enables a visualiza-

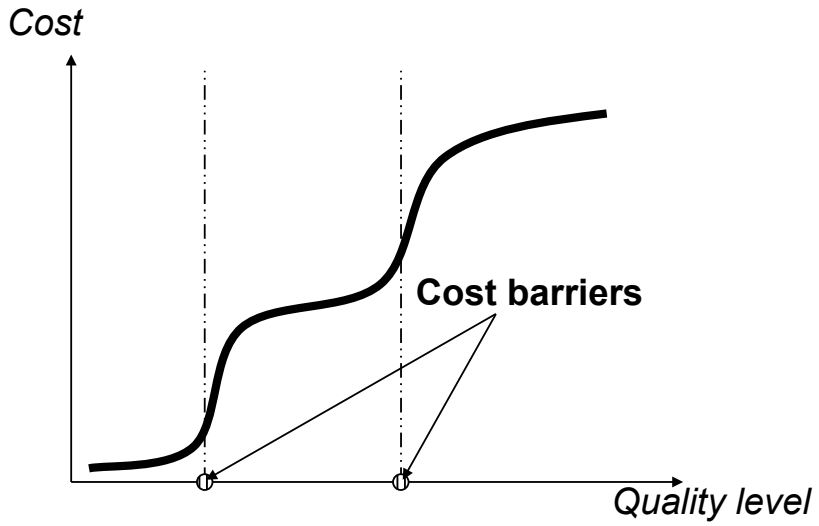


Figure 3: The cost view [23]

tion of the current situation. To support release planning and roadmapping, this view incorporate targets for coming releases.

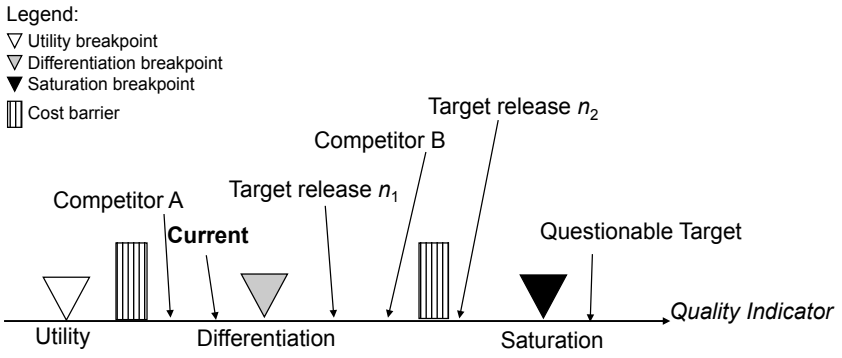


Figure 4: The roadmap view [23]

The version of QUPER presented in this section is based on the one developed in close collaboration with Sony Ericsson Moblie Communication AB (SEMC), see Section 5.2 for company description, and is an example of how QUPER can

look like. However, the presented version is intended as a starting point for any other organization wanting to tailor QUPER for their specific organization and products. The examples (to preserve confidentiality, all examples throughout this article are fictitious) in this section are important when working with QUPER as a mean to support training and practical use of QUPER for supporting release planning of QR.

To apply QUPER in practice, seven steps are envisioned, as can be seen in Appendix.

In the first step (Identify candidate QR) it is enough to pinpoint any relevant quality requirements and list them as candidate QR. The second step (Define scale and unit) is about defining which scale and measurement unit that can be used to express the level of quality of a selected QR. The third step (Identify reference levels) is centered around identifying competing as well as own products' quality levels to further calibrate the estimates, while in the forth step (Elicit quality breakpoints) market expectations are defined in terms of breakpoints. Step five (Estimate cost barriers) is about estimating the cost in terms of the values of cost barriers, and then candidate requirements (Step 6 - Set candidate requirements) are proposed, discussed, and decided. Finally, in step seven (Identify cost dependencies), dependencies, in terms of how cost estimates are affected by other QR, are identified. Each of these steps is described in further detail below.

4.1 Step 1 - Identify candidate QR

When defining quality requirements, it is important to consider relevant features, market segment, competitor, and hardware platform capability. Once such feature has been identified, the consequences for the particular quality requirement should be consider, for example:

- Different mobile phones offered to different market segments may have different requirements on image quality
- A competitor may recently have released a mobile phone with better gaming performance changing the perception of gaming quality
- Today's hardware is not the same as tomorrows, features may run much faster
- Users' evolving expectations, expects better performance in the latest mobile phones

If several QR have been identified, it may not be useful to apply QUPER's steps on all of them. Quality requirements where QUPER may not be relevant include, for example:

- Quality requirements that refers to a certain standard

- Quality requirements where a certain level of quality is always the same, e.g. in mobile TV where 28 frames per second is standard

It may be wise to prioritize the top-n (n might be decided to be, e.g. 10) QR using a defined criterion of "importance" (e.g. using simple priority grouping or based on expert judgment), in order to focus the QUPER efforts on the most important QR.

Figure 5 illustrates an example of the considered quality requirement *Time shift buffer size* for the feature Mobile TV Time Shift.

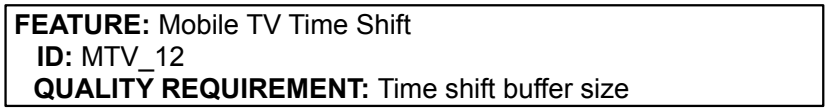


Figure 5: Illustration of considered feature and its quality requirement

4.2 Step 2 - Define scale and unit

For selected (most important, e.g. top 10 from step 1) QR, define a scale and a measurement unit that can be used to express the level of quality of QR. A scale can for example be *"time"* and the measurement unit can be *"minutes"*. Figure 6 shows the defined scale and unit for *Time shift buffer size*.

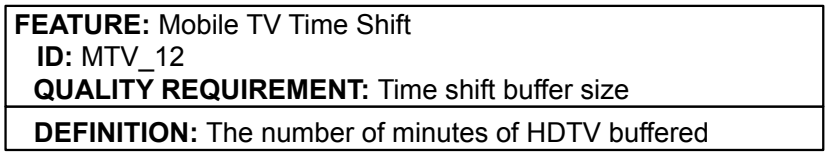


Figure 6: Illustration of defined QR with scale and unit

4.3 Step 3 - Identify reference levels

For each quality requirement, it is useful to identify reference levels based on actual products. Reference levels can be based on competing as well as own products (Qref). Estimates can be given in three forms, depending on how the potential uncertainty in the estimates should be captured:

- Point estimates including a single figure, e.g. 3 minutes

- Interval estimates including a [min, max] interval, e.g. 3-4 minutes
- Triangle distribution estimates including a three-tuple of [low bund, most probable, high bound] figures that show the estimated probability distribution, e.g. low: 3 minutes, high: 5 minutes, probable 4 minutes.

Although three forms of estimates can be given, point estimates was the most common form in previous case studies [6]. The reference levels further calibrate the estimates and provide objective measures to relate the QR to. Figure 7 illustrates added reference levels for *Time shift buffer size*.

FEATURE: Mobile TV Time Shift
ID: MTV_12
QUALITY REQUIREMENT: Time shift buffer size

DEFINITION: The number of minutes of HDTV buffered

REFERENCE LEVELS
PRODUCT: Competitor X **LEVEL:** 20 min
PRODUCT: Own product Y **LEVEL:** 40 min
PRODUCT: Competitor Z **LEVEL:** 160 min

Figure 7: Illustration of added reference levels

4.4 Step 4 - Elicit quality breakpoints

When all reference levels have been identified, for each quality requirement, ***the market expectations*** should be defined in terms of the values of quality breakpoints.

First, determine the utility breakpoint, which is the lowest acceptable value ***on the market*** for a given segment.

How to judge what is lowest acceptable value:

- Is it possible to sell this feature at this quality? If not, then below utility
- Will this quality generate a too high return rate? If yes, then below utility

Then, determine the saturation breakpoint, representing quality levels that are clearly considered excessive ***by the market***.

How to judge what is excessive quality:

- Over this breakpoint will not sell any more products
- Over this breakpoint will not give any market advantages

- Will enhance the user experience

Finally, the differentiation breakpoint somewhere between utility and saturation is determined. Values above this quality level gives market advantage compared to the current products of your competitors.

How to judge differentiation quality:

- The quality will be better than competitors
- The quality can be used in marketing the product

Similar to step 3 (Identify reference levels), estimates can be given in three forms; however, point estimates are the preferred form (see Step3 for more details).

Figure 8 shows the identified quality breakpoints for *Time shift buffer size*, while Figure 9 illustrates the roadmap view of *Time shift buffer size* with reference levels and the three (utility, differentiation, and saturation) quality breakpoints.

FEATURE: Mobile TV Time Shift
ID: MTV_12
QUALITY REQUIREMENT: Time shift buffer size

DEFINITION: The number of minutes of HDTV buffered

REFERENCE LEVELS
PRODUCT: Competitor X **LEVEL:** 20 min
PRODUCT: Own product Y **LEVEL:** 40 min
PRODUCT: Competitor Z **LEVEL:** 160 min

QUALITY BREAKPOINTS
UTILITY: 15 min **RATIONALE:** all products are able
SATURATION: 200 min **RATIONALE:** films are shorter
DIFFERENTIATION: 50 min **RATIONALE:** high price point

Figure 8: An illustration of quality breakpoints have been defined

4.5 Step 5 - Estimate cost barriers

When market expectations have been identified, for each quality requirement, estimate the cost in terms of the values of cost barriers (CB). To identify the cost barrier, practitioners with good domain and architectural knowledge may be needed. If possible, identify similar quality requirements' cost barriers from previous projects and use as input.

Although it is possible to identify and estimate one, two, or several cost barriers for each QR, the recommended number of cost barriers is two. The first cost barrier

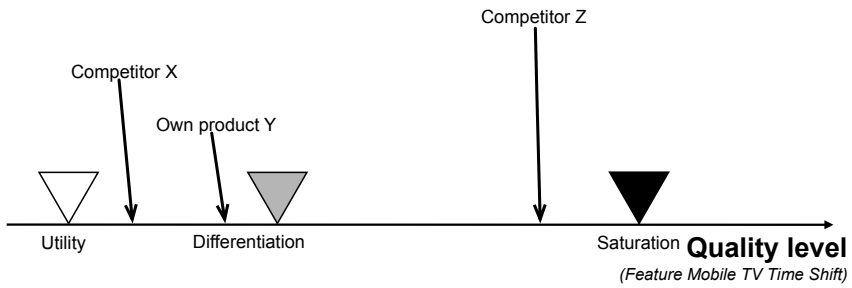


Figure 9: Illustration of reference levels and quality breakpoints merged into the roadmap view

is mainly related to software changes, while a second cost barrier is mainly related to new hardware components, or affects the entire software architecture.

First, estimate the first CB in terms of cost (C1) and at what quality level (Q1) where an increase in quality has a high cost penalty.

How to identify the first cost barrier:

- Q1: May relate to software changes, for example, requires a change in one or a few parts of the architecture, extensive optimization of code, or a major re-work of the code
- Q1: May only affect your own and/or closely related projects' code/architecture
- C1: Represents the cost penalty of raising the quality level from the current quality level (Qref) to Q1

Then, estimate the second CB in terms of cost (C2) and at what quality level (Q2) where an increase in quality has a high cost penalty.

How to identify the second cost barrier:

- Q2: May affect major (if not all) parts of the entire products' architecture
- Q2: The hardware's physical constraints may be used as Q2
- Q2: May require major infrastructure (e.g. code optimization) changes in several projects
- C2: Represents the cost penalty given that the C1 investment has been made, when raising the quality from Q1 to Q2

In Figure 10, cost barriers have been identified for *Time shift buffer size*, while Figure 11 illustrates the added cost barriers in the roadmap view.

FEATURE: Mobile TV Time Shift
ID: MTV_12
QUALITY REQUIREMENT: Time shift buffer size

DEFINITION: The number of minutes of HDTV buffered

REFERENCE LEVELS
PRODUCT: Competitor X **LEVEL:** 20 min
PRODUCT: Own product Y **LEVEL:** 40 min
PRODUCT: Competitor Z **LEVEL:** 160 min

QUALITY BREAKPOINTS
UTILITY: 15 min **RATIONALE:** all products are able
SATURATION: 200 min **RATIONALE:** films are shorter
DIFFERENTIATION: 50 min **RATIONALE:** high price point

BARRIER
Qref: 40 min
Q1: 90 min **RATIONALE:** new SW architecture needed
C1: 4 weeks
Q2: 180 min **RATIONALE:** new HW component needed
C2: 24 weeks

Figure 10: Illustration of feature with cost barriers

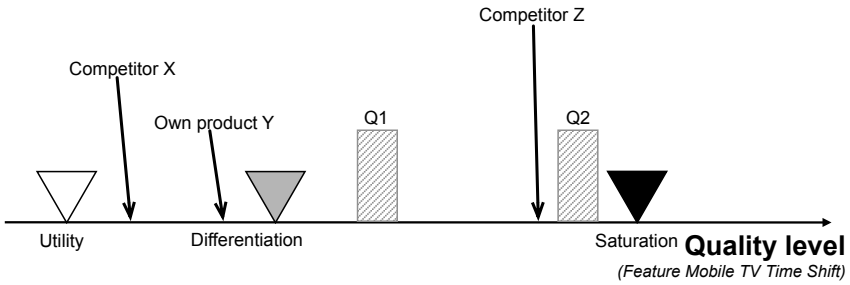


Figure 11: Illustration of quality breakpoints, reference levels, cost barriers merged into a roadmap view

4.6 Step 6 - Set candidate requirements

Now, make estimations, propose candidate requirements, discuss and decide actual requirements for coming releases, where estimates can be given in three forms (as in Step 3):

- Point estimates including a single figure
- Interval estimates including a [min, max] interval
- Triangle distribution estimates including a three-tuple of [low bund, most probable, high bound] figures that show the estimated probability distribution

One way to specify a requirements quality interval is by using both a *Good* and a *Stretch* target. The actual quality requirement is the interval that is specified by the two *targets*. It is possible to define the requirement interval in the following ways:

- With both a Good target and a Stretch target
- With *only* a Stretch target, which means the highest value is specified
- With *only* a Good target, which means the lowest accepted value is specified

Figure 12 shows the identified target, as an interval using *Good* and *Stretch*, for *Time shift buffer size*. Figure 13 illustrates the roadmap view for *Time shift buffer size* when the actual requirement (level of quality) for the next release is defined.

4.7 Step 7 - Identify cost dependencies

If cost dependencies among quality requirements are considered important to identify for cost estimations, then, for each top-n QR, identify which modules (architectural components/parts) that needs to be changed if that QR is to be improved beyond the ”next” breakpoint (either utility or differentiation depending on its current position).

How to identify potential dependencies:

- If two (or more) QR affect the same architectural part(s), they may be dependent on each other.
- Identify dependencies by already existing dependency tools/models, e.g. by a traceability tool or a Feature Dependency Model.

When potential cost dependencies among the top-n QR have been identified, for each top-n QR: (1) list which other top-n QR that are easier/cheaper to improve if this QR is improved, and (2) list which other top-n QR that are more difficult/-expensive to improve if this QR is improved.

Then, an expert subjectively (based on experience and ”gut feeling”) select m QR (e.g. the ones that will be implemented, the most important QR to improve the level quality) that is a subset of the top-n QR (m<=n) and set a quality level target for each of these m QR that seem to provide a reasonable cost increase.

FEATURE: Mobile TV Time Shift
ID: MTV_12
QUALITY REQUIREMENT: Time shift buffer size

DEFINITION: The number of minutes of HDTV buffered

REFERENCE LEVELS
PRODUCT: Competitor X **LEVEL:** 20 min
PRODUCT: Own product Y **LEVEL:** 40 min
PRODUCT: Competitor Z **LEVEL:** 160 min

QUALITY BREAKPOINTS
UTILITY: 15 min **RATIONALE:** all products are able
SATURATION: 200 min **RATIONALE:** films are shorter
DIFFERENTIATION: 50 min **RATIONALE:** high price point

BARRIER
Qref: 40 min
Q1: 90 min **RATIONALE:** new SW architecture needed
C1: 4 weeks
Q2: 180 min **RATIONALE:** new HW component needed
C2: 24 weeks

TARGET
GOOD: 80 min **RATIONALE:** will beat most
STRETCH: 90 min **RATIONALE:** if SW architecture is feasible

Figure 12: Illustration of feature with targets

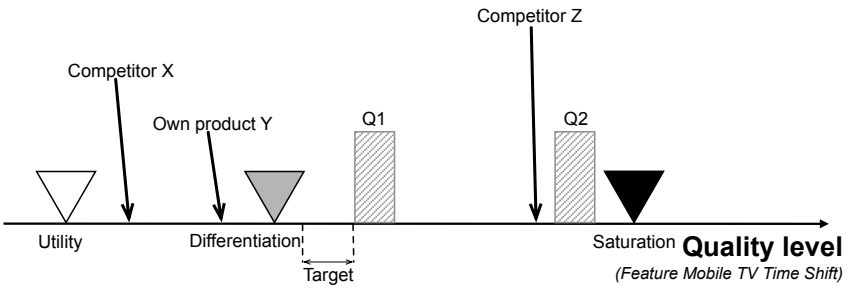


Figure 13: Illustration of quality breakpoints, reference levels, cost barriers and target merged into a roadmap view

Then, for this set of m QR; make an effort estimation in weeks or months informed by the above, by first making individual effort estimates of each m QR given that all of the targets are implemented by subjectively taking into account the ”synergies” and the ”counter working” in step 5, and the sum all up to a complete effort for the m QR.

Finally, if the total effort is too high or too low compared to available resources then change the subset in a ”smart way” (this new candidate set is derived subjectively based on ”gut feeling” and the experience of the expert) to arrive at another ”better” effort estimate.

5 QUPER Validation

This section offers a summary of prior validations of parts of the QUPER model, a presentation of the validation methodology of QUPER (as described in Section 4) as it was validated in industry against an organization faced with the issues described in Section 3.1. Finally, the lessons learned from the new validation are presented and discussed.

Although the QUPER model has been evaluated in prior evaluations (see Sections 3.3 and 5.1), the important step of cost dependencies has not been evaluated, neither by itself nor in combinations with all of QUPER's steps. Furthermore, the detailed guidelines of how to apply QUPER in practice (Sections 4.1-4.7) have not been evaluated by professionals in an industrial setting to evaluate its usefulness and applicability in an industrial setting. In addition, the detailed guidelines have not been evaluated in combination with QUPER's steps. Therefore, to investigate the usefulness and applicability of the complete QUPER model using the detailed guidelines in an industrial setting using real QR, a new validation is needed.

5.1 Prior Validations

The QUPER model has been developed and evaluated in several stages, as described in Section 3.3. The prior evaluations have evaluated different parts of the QUPER model.

First, the model was developed based on industry needs, and an initial evaluation of the model with six practitioners was conducted at one case company [23]. Then, the benefit view was tested using real requirements in real projects [6]. The cost view was evaluated by eight practitioners [7], followed by an evaluation of QUPER's benefit and cost views [8]. However, the cost dependency step (see Section 4.7) has not been evaluated in any of the previous evaluations.

The QUPER model has matured over the several industrial evaluations and improvements have been made to the model to be useful in an industrial setting. In addition, steps have been added to the model, and the detailed guidelines (see Sections 4.1-4.7) have evolved and been refined due to the industrial evaluations.

In the following sub-sections, the first novel evaluation of the complete QUPER model (as described in Section 4) at a case company (see company description in Section 5.2) and the lessons learned are presented.

5.2 Case Company Description

The case company has more than 9,000 employees and develops embedded systems for a global market using a product line approach [20]. The company's requirements database consists of more than 20,000 requirements where approximately 25% of the requirements are quality requirements. The company has several consecutive releases of a platform (a common code base of the product line) where each of them is the basis for one or more products that reuse the platform's functionality and qualities. The case company has two types of platform releases, a major and a minor release. A major release has a lead-time between two and three years from start to launch, and the focus is on functionality growth and quality improvements of the product portfolio. Minor platform releases usually focus on the platform's adaptations to different products. The company uses a stage-gate model with several increments, where Milestones (MSs) are used for controlling and monitoring the project progress. There are four milestones (MS1-MS4) for requirements management and design before the implementation starts, and three milestones (MS5-MS7) for the implementation and maintenance phase.

5.3 Validation Methodology

As the QUPER model was completed, it was set to be evaluated in an industrial environment using real quality requirements. The evaluation of the complete QUPER model was carried out using a qualitative research approach and in-depth semi-structured interviews [25] and self-administrated questionnaires [10], [25].

The first step was to plan the study and how to evaluate the QUPER model at the case company. The interview instrument (see Table 1) was design with inspiration from [8], while the self-administrated questionnaire (see Table 2) was inspired by [5]. The self-administrated questionnaire used a seven-point Likert scale, representing levels of agreement from "strongly disagree" to "strongly agree". To test the interview instrument and the questionnaire, two pilot interviews were conducted to improve the instruments prior to the industry validation. The selection of practitioners for participating in the evaluation was conducted in cooperation with two managers at the case company. Eleven practitioners, representing different roles and areas were chosen. The roles chosen are: 4 product managers, 2 project managers, 1 software architect, 1 test manager, 1 head of software quality, and 2 senior software engineers.

The second step involved applying the QUPER model in practice. The practitioners received the detailed practical guidelines, of how to use the model, to follow the steps using real quality requirements from their projects. The variation

Table 1: The Interview Instrument

Questions about the QUPER model
What is your general view of using QUPER?
What was helpful compared to the previous way of working?
Was it easier to coordinate the decision process?
What were the challenges in applying QUPER?
Do you think the estimates (targets) will be more accurate with QUPER?
Can the use of QUPER improve the decision-making process?
Final question
Is there anything else you would like to add that we have not mentioned?

Table 2: The Questionnaire

ID	**Questions**
Q1	QUPER is easy to understand
Q2	QUPER's guidelines work in an industrial setting
Q3	QUPER improves the understanding of needed level of quality
Q4	QUPER improves the understanding of QR
Q5	QUPER improves the decision-making process, e.g. release planning, of QR
Q6	QUPER's benefit view is helpful when specifying QR
Q7	It is difficult to identify the breakpoints
Q8	QUPER's cost view is helpful when specifying QR
Q9	It is difficult to identify the cost barriers
Q10	QUPER's roadmap view is helpful when specifying QR
Q11	Applying QUPER takes too much time to be useful

of how many QR each practitioner applied QUEPR's steps to range from a few QR up to 20. The main goal of the second step is to achieve an understanding of the QUPER's usefulness and applicability in an industrial environment.

The third step was carried out using semi-structured interviews [25] in the offices of the practitioners and lasted between 40 and 60 minutes each. During the interviews, the purpose of the evaluation was explained. Then, the practitioners answered the self-administrated questionnaire, followed by questions (from the interview instrument) about applying the complete QUPER model in practice, which was discussed in detail.

We took records in the form of audio recording and transcribed the recordings in order to facilitate and improve the analysis process. The collected data was analyzed using content analysis [25]. The content analysis involved marking and discussing interesting sections of the transcripts. The first author examined the sections from different perspectives for explicitly stated or concealed pros and cons in relation to the usefulness and applicability of the model.

5.4 Lessons learned

Below, lessons learned and the results from the self-administrated questionnaire are discussed. The results from the self-administrated questionnaire are presented in Table 3, while the median value of each question is shown in Table 4.

Ease of use

In general, the practitioners agree that the QUPER model is easy to understand (Q1 in Table 4), that the detailed guidelines work in an industrial environment (Q2), and the model does not take too much time to apply in practice (Q11).

During the interviews, several practitioners explained that the detailed guidelines (sections 4.1-4.7) are very helpful due to easy steps to follow, and in particular the provided examples (see Figures 5-13) for each step. Moreover, the steps in the detailed guidelines have about enough information, not too much or too little to be applicable in industry. Several practitioners stressed another important issue in relation to QUPER's applicability in industry, all steps are not mandatory to use (see Appendix). According the practitioners, if they are "forced" to go through all steps, some people may be too scared to use the model. One practitioner explained further, *"a model cannot be too big or too complicated, it must be a 'light model' to be applicable in industry, which QUPER fulfills"*. In addition, the steps in the detailed guidelines were seen as following a logical order when applied to QR.

Although the practitioners viewed QUPER as easy to use and understand, there were two main concerns about the detailed guidelines. First, a need for more examples, in particular of other QR than performance requirements, e.g., usability requirements. One practitioner asked, how do you specify a usability requirement using the QUPER model when the usability is not related to performance require-

Table 3: Distributions of questionnaire answers

ID	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Q1	0	0	0	2	3	5	1
Q2	0	0	0	0	1	8	2
Q3	0	0	0	0	3	5	3
Q4	0	0	0	1	5	3	2
Q5	0	0	0	5	0	5	1
Q6	0	0	0	2	4	4	1
Q7	0	2	3	3	1	2	0
Q8	0	0	0	5	1	4	1
Q9	0	0	1	4	2	3	1
Q10	0	0	0	0	0	5	6
Q11	1	5	1	3	1	0	0

Table 4: Questionnaire median value per question

ID	Median
Q1	Agree
Q2	Agree
Q3	Agree
Q4	Slightly agree
Q5	Agree
Q6	Slightly agree
Q7	Neutral
Q8	Slightly agree
Q9	Slightly agree
Q10	Strongly agree
Q11	Disagree

ments? The second main concern was related to inconsistent usage of the model. The practitioners believed that some people may use the concepts of the QUPER model in different ways, and a special concern was related to that higher quality is sometimes related to higher value, while other times a lower value means higher quality.

Importance of the three views

In Table 3, the results show that the roadmap view is the most important view of the QUPER model (Q10 in Table 4). In addition, the benefit view may be helpful when specifying QR (Q6 in Table 4), while the cost view is the least important (Q8) of the three views. One explanation of why the roadmap view is seen as the most important view was discovered during the interviews, the information from both the benefit and cost view is visualized in the roadmap view. Hence, the other views are not seen as important.

Several practitioners explained that the roadmap view provides a great visualization of the market situation and it provides an easy to understand overview. One subject further explained, *"the roadmap view gives you a real visualization of the reality"*.

In Table 4, the results show that the identification of breakpoints in the benefit view is viewed as neither difficult, nor easy (Q7). The reason may be explained by the different approaches of identifying the breakpoints. During the interviews, four different approaches of how to identify the breakpoints were discovered:

- Using their own subjective estimate, i.e., the practitioner has an understanding, based on his/her experience and "gut feeling", of the estimates for the

breakpoints.

- To perform several new tests of the competitors' products level of quality, and use these values as input when estimating the breakpoints.
- If these tests (as described above) have all ready been performed, it is easy to access a database with this information.
- To use advanced and extensive market analysis techniques to identify the breakpoints.

The cost view was viewed as the least important among the three views, which is related to the perceived difficulties on estimating the cost of requirements according to the practitioners. Several practitioners explained that cost estimation, in general, is always difficult regardless if it is for functional requirements or for QR. The difficulties lie in the ability to estimate the cost and map that cost to a real value, i.e., not only using cost estimations for resources planning, but actually estimate the actual cost of implementing QR. However, most of the practitioners believe that the accuracy of the cost estimates in QUPER's cost view would be as accurate as any other feature/requirement's cost estimation. This may explain why the practitioners viewed it slightly difficult to estimate the cost barriers (Q9 in Table 4). In addition, one practitioner explained, to estimate a cost barrier, an extensive estimation analysis work may be needed, which will be time consuming and therefore not useable in practice. However, the practitioner believed that practitioners that do cost estimations in their everyday work would find it easier to estimate the cost barriers.

Applicability of the cost dependency step

The last step in the QUPER model, identify cost dependencies (see Section 4.7), was viewed as easy to follow, and at the same time detailed enough to be useful in practice. The detailed guidelines provided the practitioners with a good enough understanding of potential dependencies between QR. According to several practitioners, the detailed guidelines for the cost dependency step are similar to their approach of dealing with dependencies between features. One practitioner believed that this step might be difficult to follow and apply for some practitioners; however, according to the practitioner, the QUPER model would still be useful even if everyone does not use this step.

Supporting release planning

In general, all practitioners agreed that QUPER improves the understanding of QR (Q3 in Table 4), and that the model would improve the decision-making process in, e.g., release planning of QR (Q5 in Table 4). In addition, the roadmap view is

seen as the central part of the improvement in the decision-making process (Q10 in Table 4).

During the interviews, the practitioners explained the importance of the roadmap view. The roadmap view provides the decision-makers with an overview, which is a good basis for discussions of which quality level to aim for in the coming releases. One practitioner further explained, it is easier to understand the thought behind, and the need for a certain level of quality when it is presented on the roadmap view since it is related to the market and the competitors.

The importance of relating the needed level of quality to the market and the competitors was expressed by several of the interviewed practitioners. One practitioner explained, *"the relation to the market and our competitors is very important for our 'selling features' since we will have a better understanding if we are market leaders or not"*. Furthermore, the decisions about the needed level of quality will have a better substance compared to just presenting a metric of the quality level.

In addition to the decision-making process, the practitioners believe that the QUPER model could improve the communication between the people. For example, the concepts of QUPER provide them with a "common language" that everybody (that has used QUPER) understands and make sure they are talking about the same things.

Although this first evaluation of the complete QUPER model shows promising results, the practitioners had a few concerns. First, there may be difficulties to convince others at the case company to use the model. It is easier to just decide the level of quality out of the blue instead of learning a new model and follow a set of guidelines. Some of the practitioners suggested to have a workshop to teach the QUPER model to the employees of the case company where a "QUPER expert" should be present at the first time.

Second, according to one practitioner, it is important to choose the right QR to apply QUPER. The QUPER model cannot be applied to all QR, e.g., certain QR must have a specific level of quality to fulfill a certificate or a standard.

Third, as several practitioners stated, to fully understand and evaluate the improvements of the decision-making process, the QUPER model should be used in a project from the start of a project until the product is launched to the market. However, the lead-time for projects at the case company is between two and three years.

6 Limitations

As for all empirical studies, there are limitations and threats to the validity. One threat is related to confounding factors, which are important when making inferences about root-cause relationships. In this paper, inferences are made about the improvements of release planning for QR. The confounding factors cannot be

ruled out as the study was conducted in an uncontrolled industrial environment. The use of very enthusiastic or skeptical subjects could be a confounding factor. In this study, several of the subjects have been involved during the entire, or part of the evolvement of the QUPER model. Hence, they may have a positive attitude towards the model from the beginning. To minimize this threat, several subjects that had not been part of the evolvement of the model were included in the sample size. Moreover, the case company is rather immature when it comes to handling QR. No process, or method of dealing with QR are used at the case company, which may influence the positive view of QUPER as it may be better to have any process than none at all.

Another threat is external validity, i.e., the ability to generalize the results beyond the included case company. Although the case company is large and develops technically complex embedded systems, it cannot be taken as a representative for all types of large companies developing embedded systems. Hence, the results should be interpreted with some caution. However, qualitative studies rarely attempt to generalize beyond the actual setting since it is more concerned with explaining and understanding the phenomena under study. Some of the problems introduced as motivation behind the conception of QUPER, to some extent could be general for organization faced with developing embedded products for a market. In addition, from a perspective of the concepts and practical application of QUPER as described in this paper can give an overview of the challenges facing companies that may implement QUPER.

Finally, further evaluations in industry where the long-term effect, in terms of benefits and challenges, of using QUPER needs to be investigated to validate its feasibility and scalability.

7 Conclusions

The QUPER model was developed in response to needs identified in industry and the lack of an appropriate model in the literature to address these needs. The goal of the QUPER model was to offer decision-makers, e.g., product managers, a model for supporting release planning of quality requirements for market-driven software product developing companies.

This paper presents the first complete version of the QUPER model, including the detailed guidelines of how to apply QUPER in practice. The goal of the model is to be useful in industry by being simple and robust, yet relevant to high-level decision-making such as release planning.

The basis for the construction of QUPER and its three (benefit, cost, and roadmap) views is the concepts of breakpoints and barriers. The benefit view includes three breakpoints; the cost view with cost barriers indicates steep increases of cost for relevant quality; while the roadmap view combines the benefit and cost

view into an ordinal scale where competitors' products and future targets for releases can be discussed.

As part of QUPER's development, evolvement, and refinement, parts of the model has been validated in a series of steps in prior industry validation in this article. During these prior validations, QUPER has matured, and improvements have been made. In this article, the complete version of QUPER was validated in industry at one case company with 11 industry professionals using real quality requirements. The validation was performed to assure QUPER's applicability in industry, and that the model is useful for decision-makers.

During the validation, the results show that QUPER is useful and applicable in an industrial setting, and the validation results indicate that the model improves the decision-making process of release planning for quality requirements. In particular, the visualized roadmap with relations to the market and competitors provides more substance to decisions of what level of quality to aim for in the coming releases.

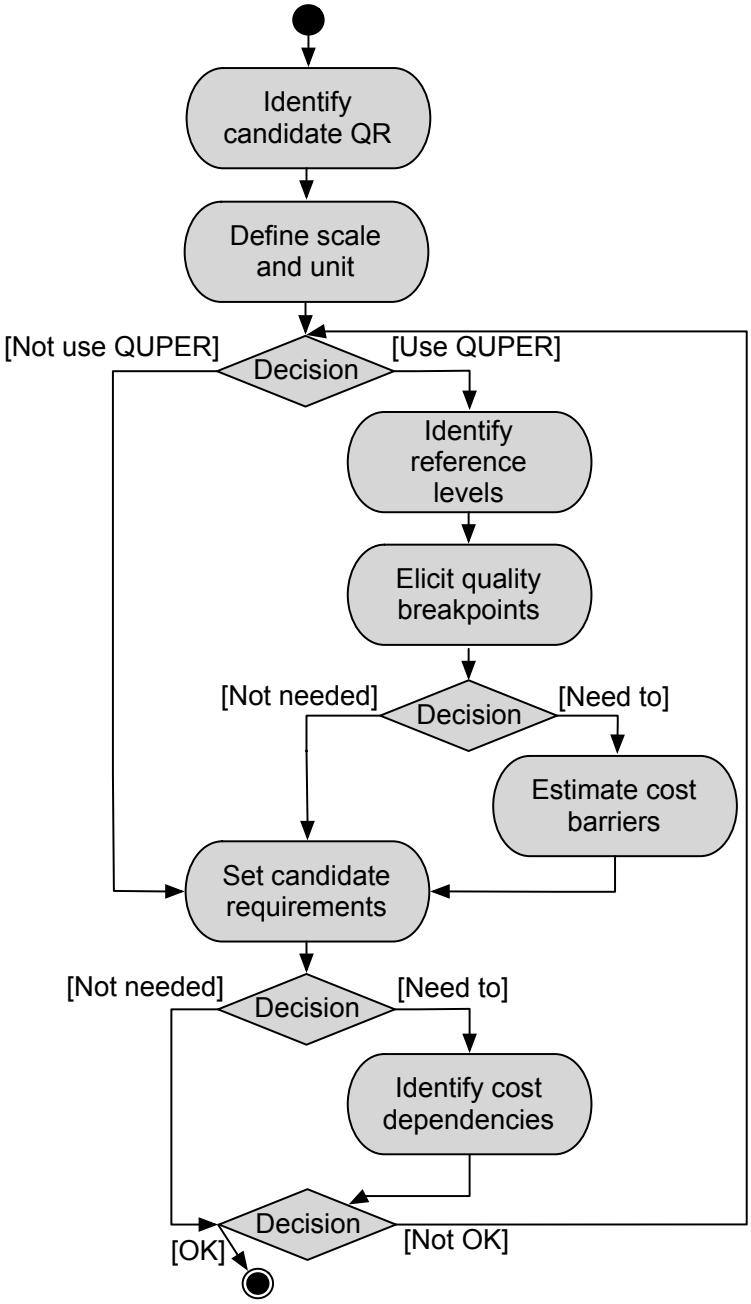
As stated earlier, QUPER presented in this article was developed with decision-makers in mind, particular product managers, supporting them in release planning of quality requirements. The initial results of QUPER show great promise, but also give information about limitations on which future research can be based on.

The next phase that will be undertaken in the validation and evolvement of QUPER, as described in this article, is further evaluations in industry in different domains where the long-term effects, in terms of benefits and challenges, of using QUPER need to be investigated to fully validate its feasibility and scalability.

Acknowledgment

This work was partly funded by VINNOVA (the Swedish Agency for Innovation Systems) within the MARS project and by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se). Furthermore, we would like to thank all of the participants and their companies who have helped in making the data collection possible for this research.

Appendix


Identify candidate QR
Define scale and unit
[Not use QUPER]
Decision
[Use QUPER]
Identify reference levels
Elicit quality breakpoints
[Not needed]
Decision
[Need to]
Estimate cost barriers
Set candidate requirements
[Not needed]
Decision
[Need to]
Identify cost dependencies
Decision
[OK]
[Not OK]

Bibliography

[1] M. van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Software product release planning through optimization of what-if analysis. *Information and Software Technology*, 50(1–2):101–111, 2008.

[2] A. Al-Emran, D. Pfahl, and G. Ruhe. Decision support for product release planning based on robustness analysis. In *Proceedings of the 18th IEEE International Requirements Engineering Conference*, pages 157–166, 2010.

[3] 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.

[4] R. Berntsson Svensson, T. Gorschek, B. Regnell, R. Torkar, A. Shahrokni, and R. Feldt. Quality requirements in industrial practice - an extended interview study at eleven companies. *IEEE Transaction on Software Engineering*, page in print, 2011.

[5] R. Berntsson Svensson, P. Lindberg Parker, and B. Regnell. A prototype tool for quper to support release planning of quality requirements. In *Fifth International Workshop on Software Product Management*, 2011.

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

[7] R. Berntsson Svensson, B. Regnell, and A. Aurum. Towards modeling guidelines for capturing the cost of improving software product quality in release planning. In *Second proceeding: Short papers, Doctoral Symposium and Workshops of the 11th international conference on product focused software process improvements*, pages 24–27, 2010.

[8] R. Berntsson Svensson, Y. Sprockel, B. Regnell, and S. Brinkkemper. Cost and benefit analysis of quality requirements in competetive software product management. In *Proceedings of the fourth International Workshop on Software Product Management*, pages 40–48, 2010.

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

[10] A. Fink. *The survey handbook*. Sage Publications, 2003.

[11] T. Gilb. *Competitive Engineering*. Elsevier Butterworth-Heinemann, 2005.

[12] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE Software*, 23(6):88–95, 2006.

[13] D. Greer and G. Ruhe. Software release planning: an evolutionary and iterative approach. *Information and software technology*, 46(4):243–253, 2004.

[14] S. Jacobs. Introducing measurable quality requirements: a case study. In *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering*, pages 172–179, 1999.

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

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

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

[18] B. Kitchenham and S. Pfleeger. Software quality: The elusive target. *IEEE Software*, 13(1):12–21, 1996.

[19] S. Maurice, G. Ruhe, O. Saliu, and A. Ngo-The. *Value-Based Software Engineering*, chapter Decision Support for Value-Based Software Release Planning, pages 247–261. Springer, 2006.

[20] C. Pohl, G.Böckle, and F.J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

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

[22] B. Regnell and S. Brinkkemper. *Engineering and Managing Software Requirements*, chapter Market-Driven Requirements Engineering for Software Products, pages 287–308. Springer, 2005.

[23] 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 *Proceedings of the 13th working conference on requirements engineering: foundation for software quality*, pages 277–291, 2007.

[24] B. Regnell, H. Olsson, and S. Mossberg. Assessing requirements compliance in system platform subcontracting. In *Proceedings of the Seventh International Conference on Product Focused Software Development and Process Improvement*, pages 362–376, 2006.

[25] C. Robson. *Real World Research*. Blackwell, 2002.

[26] G. Ruhe, A. Eberlein, and D. Pfahl. Trade-off analysis for requirements selection. *International journal of software engineering and knowledge engineering*, 13(4):345–366, 2003.

[27] G. Ruhe and D. Greer. Quantitative studies in software release planning under risk and resource constraints. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering*, pages 262–270, 2003.

[28] G. Ruhe and A. Ngo-The. Hybrid intelligence in software release planning. *International Journal of Hybrid Intelligent Systems*, 1(2):99–110, 2004.

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

[30] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, and S.B. Saleem. A systematic review on strategic release planning models. *Information and Software Technology*, 52(3):237–248, 2010.

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

[32] E. Youdon. When good enough is best. *IEEE Software*, 12(3):79–81, 1995.

PAPER V

SETTING QUALITY TARGETS FOR COMING RELEASES WITH QUPER - AN INDUSTRIAL CASE STUDY

Abstract

Quality requirements play a critical role in driving architectural design and are an important issue in software development. Therefore, quality requirements need to be considered, specified, and quantified early during system analysis and not later in the development phase in an ad-hoc fashion. This paper presents the quality performance model that estimates quality targets in relation to market expectations as a basis for the architecting of quality requirements. The purpose of the model is to provide concepts for qualitative reasoning of quality levels in the decision-making of setting actual targets of quality requirements for coming releases of the product. The quality performance model is evaluated at one case company, using a market-driven development approach, in the electronic payment-processing domain. The results show that the model is useful for supporting early decision-making in, e.g., release planning of quality requirements.

Richard Berntsson Svensson, Yuri Sprockel, Björn Regnell, and Sjaak Brinkkemper
Requirements Engineering Journal, 2011, in print

1 Introduction

Quality requirements (QR) describe important constraints upon the development and behavior of a software system. Hence, quality is one of the most important issues in software development [1]. The ability of a software-intensive system to meet a set of QR is to a large degree depending on the software architecture (SA) [8], and the SA thus constraints the achievement of various QR [2]. Consequently, QR are a driving force for architectural design [2]. Therefore, QR should be considered and specified as early as possible during system analysis [9]. Despite their importance, QR are often discovered late in the development process in an ad-hoc fashion, which may lead to problems such as architectural solutions failing to take into account critical QR, and systems may fall short of meeting users' real needs [9]. Therefore, one of the most critical tasks for a software architect is to create a design that can meet the QR that are vital to the success of the software product [10].

A challenging problem for an organization that develops software-intensive products offered to a market is to set the right quality target in relation to future market demands and competitor products. When is the quality level good enough? When is the quality level a competitive advantage? The decided level of relevant quality attributes that a future product release should achieve may have a major impact on the SA, since quality cannot be added to the system afterwards; it has to be built into the system and its architecture from the beginning [30].

This paper presents one case study of tailoring, implementation, and the most important evaluation of a method called QUality PERformance (QUPER) [23] that complements existing methods, such as ATAM [18], SAAM [17], and QASAR [6], with estimations of benefit and cost of quality targets in relation to market expectations as a basis for the architecting of QR. The QUPER model has the specific goal to provide concepts for qualitative reasoning on the orders of magnitude of quality levels in the decision-making that needs to take place when a software organization is setting the actual targets of QR for the coming releases of the product. The QUPER model is aimed to facilitate the elicitation, specification, quantification, and prioritization of QR. The main objectives of the case study are two: firstly, to investigate how the model performs in terms of usability and usefulness when combining all three views, and secondly, to understand how QUPER can be tailored to fit in a domain different from the previously studied domain (the mobile handset domain) [23], [3], [21], [4]. The novel contribution of this paper is the first dynamic validation to include all the three views of QUPER. Furthermore, this paper presents examples of one *transaction processing* and one *upgradability* QR to illustrate the process of QUPER. This paper extends our previous report on preliminary findings [5], with more in-depth description of the evaluated QUPER guidelines and account of research methodology, a more detailed explanation of the tailoring of the QUPER activities and real data on quality targets from the case, as well as further analysis, discussion, and lessons learned.

The remainder of this paper is organized as follows. In Sect. 2, the background and related work are presented. Section 3 introduces the case company where QUPER has been used. The concept of model tailoring and the implementation of QUPER at the company are presented in Sect. 4. The research methodology is described in Sect. 5, and Sect. 6 presents the results. Section 7 gives a summary of the main conclusions.

2 Background and related work

The development of QUPER was performed in close collaboration with industry. The process can be described by the technology transfer model's step one, two, three, five, and six [12], which is illustrated in Fig. 1.

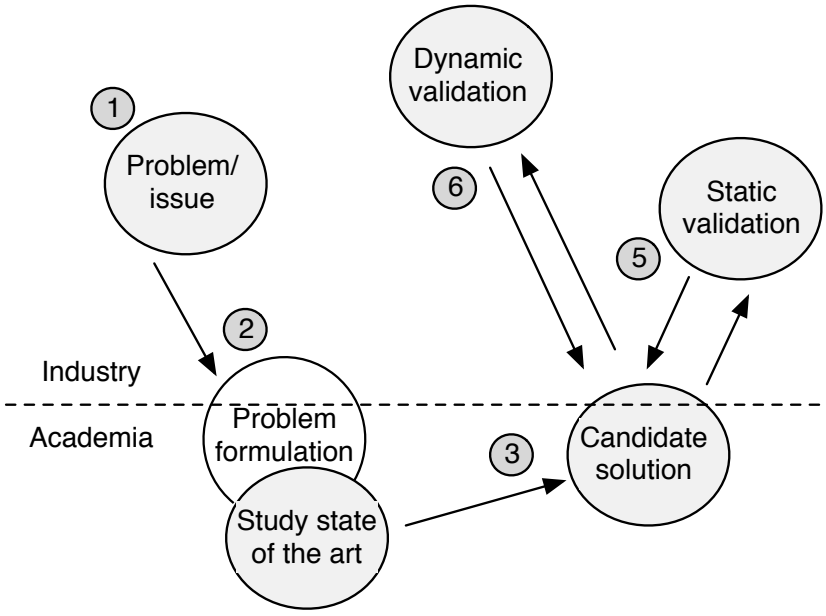


Figure 1: Overview of QUPER development

Industry needs and possibilities for improvements were investigated by focusing on the interface between the two case companies and the cross-company requirements engineering process [24]. The investigation results acted as a basis for model development [23]. QUPER was validated in several incremental steps through static validation (for example, interviews) and dynamic validation (for example, pilot projects and controlled small tests) [3], [5], [21], [4]. Each round of

the validation process was used to refine the model in terms of contents, structure, as well as test issues relating to usability and usefulness.

The evaluation presented in this paper aims to scope the success of the first dynamic validation in the form of implementation of QUPER's all three views in industry. The industry trails allow us to validate the models' usability and usefulness in a non-simulated environment. Feedback obtained from this evaluation will be used to further refine the model.

2.1 Introduction to QUPER

The QUPER model aims to support the ability to make early estimates with adequate accuracy of QR that are input to discussion and high-level decision-making in upstream requirements engineering related to, for example, roadmapping, release planning, and scoping. More details about roadmapping in market-driven requirements engineering can be found in Regnell and Brinkkemper [22]. One of QUPER's main objectives is to define a model for specification, quantification, and prioritization of QR that includes a third dimension related to quality, which may be used as input for the architecting process of QR, as a complement to cost and value that are used in prioritization of functional requirements [16].

Two hypotheses were used as reference when QUPER was developed. First, *quality is continuous*. Rather than being either included or excluded, quality requirements are assumed of having the potential of being measured on a continuous scale, i.e., a quality requirement is typically viewed as different shades of quality on a sliding scale. Second, *quality is nonlinear*, i.e., for quality requirements, such as performance; two questions regarding changes in quality level may be relevant:

1. Would slightly better performance be significantly more valuable from a market perspective?

2. Would significantly better performance be just slightly more expensive to implement?

The change in quality level may result in nonlinear changes to both cost and benefit, which is of interest to release planning and roadmapping. Aspects such as release targets, end-user experience, and business opportunities must be taken into account. At the same time, the feasibility with the evolving system architecture and available development resources must be considered.

Based on the results in [24] and from discussions with domain practitioners, three goals were selected as a guide to the QUPER model:

- *Robust to uncertainties*. The relations between quality requirements, their market value, and the implementation cost may be complex and thus difficult to estimate with high accuracy.

- *Easy to use*. QUPER should be easy to learn, remember, understand, and use by practitioners by only including a few concepts. This goal is inline with Pfleeger's recommendation for technology transfer from academia to industry [20].
- *Domain-relevant.* QUPER should be possible to combine with existing practice and possible to tailor to a particular domain. Adaption and tailoring are often prerequisites to successful technology transfer from academia to industry [12].

The basis for the construction of the QUPER model is the concepts of *breakpoints* and *barriers*. A breakpoint is an important aspect of the nonlinear relation between quality and benefit, for example, when a product's start-up time shifts from normal expectations to outperforming most competitors. Barriers represent an interesting aspect of the nonlinear relation between quality and cost, for example, achieving better performance may require an expensive rebuild of the architecture. The two concepts of breakpoints and barriers are the basis for QUPER's three views: (1) the *benefit view*, (2) the *cost view*, and (3) the *roadmap view*.

The benefit view (Fig. 2) illustrates the relation between quality and benefit in terms of three breakpoints. A breakpoint indicates a change in benefit level with respect to users' perception of quality and market value. The first breakpoint, *utility breakpoint*, marks the border between useless and useful quality. Useless quality means that a product is not accepted in the market, and users do not recognize its value. After passing the utility breakpoint, a product starts to become useful and thus have a market value. The second breakpoint, *differentiation breakpoint*, marks the shift from useful to competitive quality, which only a few of the competitors' products reach. The third breakpoint, *saturation breakpoint*, marks the shift from competitive to *excessive* quality; that is, quality levels beyond this breakpoint have no practical impact on the benefit for the considered context.

The cost view (Fig. 3) illustrates the relation between quality and cost in terms of foreseen *cost barriers*.

For a specific quality aspect in a specific context, we approximate the quality-cost relation to have two different steepness ranges. A cost 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 may be when comparatively inexpensive software optimization might produce high gains of performance. A typical cost barrier may be that an increase in quality is not feasible without a large reconstruction of the product's architecture.

The roadmap view (Fig. 4) combines the two previous views (benefit and cost views) by positioning the breakpoints and cost barriers on the same scale, which enables a visualization of breakpoints and cost barriers in relation to a product's current quality level and the competing products' quality. To support roadmapping, the roadmap view incorporates targets for coming releases.

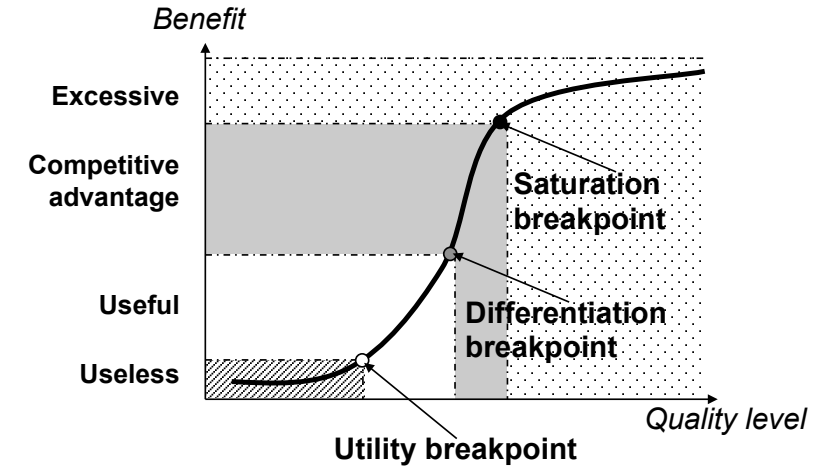


Figure 2: The QUPER benefit view

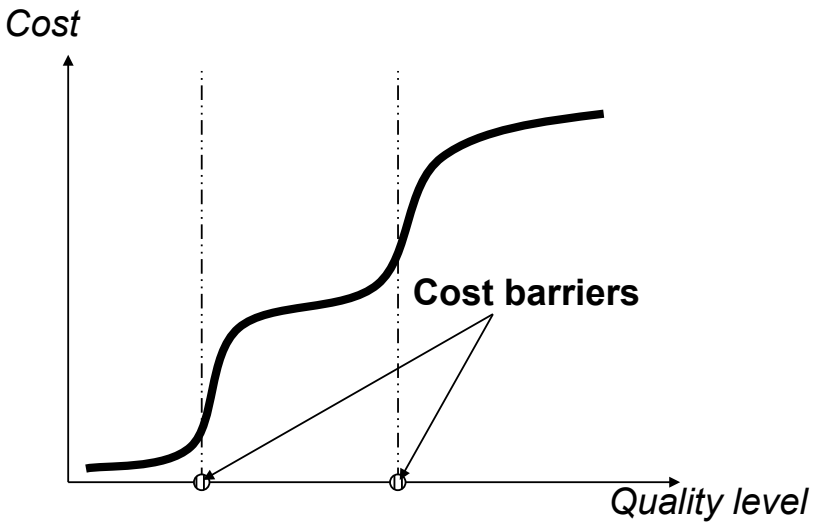


Figure 3: The QUPER cost view

The QUPER model generally aims to avoid making complete predictions of the inherently difficult relations between a product's benefit, cost, and quality. In-

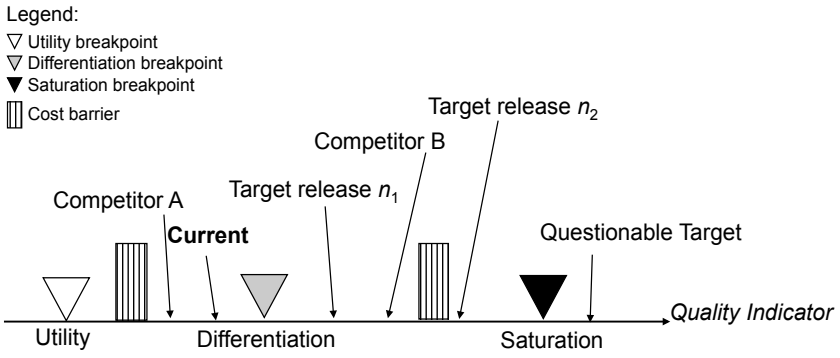


Figure 4: The QUPER roadmap view

stead, QUPER simplifies the problem by finding reasonable good predictions of a limited set of breakpoints and cost barriers (with ranges to indicate error margins, if appropriate). In all three views, the quality levels on the horizontal axes are measured by quality indicators, which may be specific to different entities such as feature, use case, domain, and market segment. Every quality indicator must be considered carefully to find special cases. The definition of these quality indicators is the main issue in tailoring QUPER for a certain domain or product.

QUPER guidelines

QUPER as presented in Sect. 2.1 is generic in nature. The practical application of QUPER's benefit view in [3], cost view in [4], and benefit and cost view in [5] are adopted prior to the model being set into operation at the companies. To apply QUPER in practice, the following steps are envisioned:

1. Define quality aspects.
2. Estimate your product's current quality level (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 the closest cost barrier in terms of cost (C1), and at what quality level where an increase in quality level (Q1) has a high-cost penalty.
5. Estimate the second cost barrier in terms of cost (C2), and at what quality level (Q2) where an increase in quality has a high-cost penalty.

6. 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 the particular indicator; for example, different products offered to different market segments have different requirements (levels of quality) for a quality aspect, or a competitor might recently have released a product with better performance, thereby changing users' perception of performance. Moreover, today's hardware is not the same as tomorrows. This has implications for performance requirements, as software features may run faster. In addition, users' evolving expectations might influence quality targets as users except better performance in the latest products.

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

In step 3, for each quality indicator, define the current market expectations in terms of the values of the three breakpoints in Fig. 2. 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 this breakpoint give market advantages.

In step 4, estimate the closest cost barrier (CB1) in terms of cost (C1) and at what quality level (Q1) where an increase in quality has a high-cost penalty. In order to identify and estimate CB1, the quality level (Q1) may be related to software changes, for example, requires a change in one or a few parts of the architecture, extensive optimization of the code, or a major rework of the code. Moreover, Q1 may only affect your own and/or closely related projects' code/architecture. It is important to note that C1 represents the penalty of raising the quality level from the current product's (Qref) quality level.

In step 5, estimate the second cost barrier (CB2) in terms of cost (C2), and at what quality level (Q2) where an increase in quality has a high-cost penalty. In order to identify and estimate CB2, the following three inputs may be used, Q2 may affect major (if not all) parts of the entire product's architecture, the hardware's physical constraints may be used as Q2, and major infrastructure (e.g. code optimization) changes in several projects may be required. C2 represents the cost penalty given that the C1 (in step four) investment has been made, when raising the quality from Q1 to Q2. Furthermore, the quality level for each cost barrier (Q1 and Q2) is one estimate, while the cost (C1 and C2) is another estimate.

In step 6, the final step, candidate targets are requirements with potential quality commitments. Different quality indicators may have a different number of relevant targets. The actual quality requirement is an interval that is specified by two targets, min target (the lowest acceptable quality for this quality indictor) and max target (the highest needed quality). Actual targets for the upcoming release or

future releases are chosen from the defined candidate targets. Figure 5 shows an example of a feature (quality requirement) of the collected and estimated information from all six steps (to preserve confidentiality, the feature, and the estimates in Fig. 5 are fictitious).

- **Step 1 - Description**
  - *Quality indicator:* Time to play music [seconds]
  - *Quality type:* Performance
  - *Definition:* Measured from player invoke button pressed until music is played using 2 GB memory stick type X with 100 tracks with average duration of 3 min
- **Step 2 - Current reference products**
  - *Competitor Product X:* 4 seconds
  - *Competitor Product Y:* 2 seconds
  - *Own Product Z (Qref):* 3 seconds
- **Step 3 – Current market expectations**
  - *Utility breakpoint:* 5 seconds
  - *Differentiation breakpoint:* 1.5 seconds
  - *Saturation breakpoint:* 0.2 seconds
- **Step 4 – Estimate the closest cost barrier (CB1)**
  - *Q1:* 2 seconds
  - *C1:* 4 weeks
- **Step 5 – Estimate the second cost barrier (CB2)**
  - *Q2:* 1 second
  - *C2:* 24 weeks
- **Step 6 – Candidate targets**
  - *Min target:* 2 seconds – This target is possible without a new architecture, but needs some software optimization.
  - *Max target:* 1 second – If we create a new architecture, this target (which is better than differentiation) will be easy to reach. Users might require this level of quality within 2 years.

Figure 5: An example of a QR of QUPER's six steps

2.2 Related work

In this section, work on addressing QR at the architecture level is presented. Different software architecture design methods for evaluating a SA quality are presented, followed by release planning methods for selection and prioritization of requirements (including QR). Finally, other requirements prioritization techniques are described.

The software engineering research community has defined various software architecture design methods [18], [17], [6], for example, the scenario-based architecture analysis method (SAAM) [17]. SAAM uses scenarios for evaluating quality attributes and is applied to a final version of the SA but prior to the detailed design. Another method is the architecture trade-off analysis method (ATAM) [18], which is developed to find trade-offs among quality attributes that affect each other at the architecture level. Moreover, the quality attribute-oriented software architecture design method (QASAR) [6], is a method for software architecture design that

employs explicit assessment of and design for QR of a software system. However, these design methods start with the design of the software architecture based on the functional requirements specified in the requirements specification, where most QR are typically not explicitly defined at this stage [10]; that is, requirements should be elicited, specified, quantified, and prioritized before the methods can be applied.

Several approaches and strategies have been proposed to resolve issues related to requirements selection and prioritization [29]. In this section, a selection of release planning methods is presented: EVOLVE [13], Release Planner Prototype [7], and a method for software release planning using optimization of what-if analysis [31]. All of these three methods use generic algorithms to resolve the release planning issue. In addition, analysis of requirements dependencies is present in these methods. However, none of the methods address quality constraints. According to Svahnberg et al. [29], only two strategic release planning methods address quality constraints. The quantitative Win-Win [26] addresses effort and time constraints, but not the quality level of quality requirements. The only method to address quality and cost constraints of QR is the QUPER model [29]. For a more comprehensive analysis of 24 strategic release planning methods, we refer to [29].

Several prioritization techniques and cost-benefit analysis models are introduced in literature. The contributions in this area include: Planguage [11], Kano [14], and quality function deployment (QFD) [15], and a cost-benefit approach [16] based on the analytical hierarchical process (AHP) [28]. Planguage has roadmap related concepts such as past, record, and trend for quality requirements. QUPER could be used together with Planguage to, e.g., express competing products in different market segments. Kano's model views quality relationships as nonlinear between customer satisfaction and the degree of achievement in a two-dimension graph. However, Kano's model does not include a cost dimension. QFD 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. QFD measures QR using a scale where no clear distinctions between the values are provided. The cost-value approach, Karlsson and Ryan [16], prioritize requirements based on the AHP. Their approach is mainly used for functional requirements; however, QR can of course be included as objects of prioritization in AHP.

3 Case company

The case company employs more than 250 employees, has more than 120,000 customers' and business partners, and operates in the electronic payment-processing market. The case company is an international organization that specializes in payment terminals, transaction processing, and development of saving- and customer-card systems. In order to structure and regulate the electronic payment market,

third-party organizations coordinate and set standards for the market, and how organizations are allowed to operate with other parties, e.g., banks and acquirers. Moreover, the third parties define a majority of the functionalities and requirements, which leaves little room for the case company to differentiate itself from its competitors. Hence, quality requirements play an important part.

At the moment, the case company does not have any defined quality standard to adhere to in relation to product requirements. First, product requirements are defined at a high abstraction level with input from the market. Then, the requirements are specified in more details for the software development team. It is the responsibility of the development team to estimate the realization costs and to communicate this with the management. Then, management decides, in collaboration with the development team, what requirements should be included in the upcoming release, and what requirements are planned for future releases. After a realization acceptance test is performed, a pilot test is conducted to test the requirements.

Since the case company is growing internationally, there is a higher focus on identification and gathering of quality requirements. In order to monitor the quality levels, a more structured approach to deal with quality requirements is needed. However, at the moment, when a software upgrade is about to be released, quality requirements are dealt with in an ad-hoc approach, unlike the product requirements. One of the case company's larger software upgrade releases for 2010 is the first release where quality requirements are elaborated.

Daily challenges faced by the case company include, how can we measure the quality of our products? When is the quality level good enough, and how much would it cost us? The case company does realize that quality requirements need to be planned and decided in the early decision-making process (such as release planning), before the initiation of the software development process.

3.1 System architecture

Figure 6 illustrates a general, high-level, view of the case company's architectural design of their product (due to confidentiality, no detailed description of the architectural design is presented).

The architecture presents different types of electronic payment solution (EPS) of which the customer chooses one that fits its business best. The EPS in use is connected through an interface to the software management system, which is operated by the case company. This software management system uses the connection with the EPS mainly for software upgrade, trouble shooting, and overall monitoring. Trouble shooting and overall monitoring was minimal and not considered reliable, i.e., only connection details were obtainable and whether the EPS was operating properly or not. This was due to the immatureness and instability of the software management system. Another important responsibility, yet not as challenging, the software management system has is to forward data from the customer

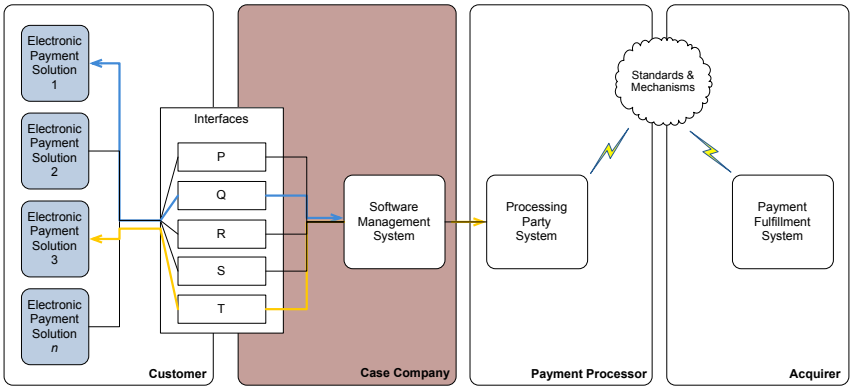


Figure 6: Simplified system architecture

to the payment processor. The Processing Party System is responsible for communication and provides instructions to the Acquirer. This communication happens through a series of obliged standards and mechanisms that has been imposed by the third-party organizations. This insures a controlled and secured environment to transmit delicate data. Finally, the Payment Fulfillment System is responsible for finalizing the actual payment, which has financial consequences for the customer and the end-user.

Figure 6 illustrates the architectural components that are influenced by QR, i.e., EPS, interfaces, the Software Management System, and the Processing Party System. The two colored lines in Fig. 6 represent two different types of QR that have been used in this case study (see Sect. 5.2). The blue line represents upgradeability (upgradability is a used term at the case company, but the description of upgradability is similar to perfective maintenance [19]); the Software Management System uses the connection through the interfaces to upgrade the EPS with the new software and the EPS reports back. The yellow line represents Transaction Processing (performance); the EPS sends data through the interfaces to the case company, which in turn forwards the data to the Payment Processor. The payment processor reports back to the EPS. Some instructions are provided directly to the Acquirer, and others are performed on a scheduled basis.

In this case study, upgradability refers to the capability to upgrade the running system on the EPS (see Fig. 6) with new software. A scenario may be where the case company has a new version of the software or when the third-party organizations push for, e.g., new standards. The quality indicators (see Table 1, Sect. 6.1) of upgradability indicate the upgrading of the system in use, whereas the quality indicator Download Success Rate is measured by the success rate of upgrades in percentage; for example, upgrade version B12.34 for EPS1 via interface P has a success rate of 67%, while the success rate via interface Q is 52% (these are

fictitious examples).

4 Model tailoring

QUPER as presented in Sect. 2.1 is generic in nature; therefore, an adaption of the six steps in Sect. 2.1 needs to be addressed prior to the model being set into operation in an organization. This can be seen as a tailoring of QUPER to fit a specific organization. The tailoring of QUPER was conducted in meetings between a researcher and practitioners of the case organization (see Sect. 5.2). The tailoring process resulted in three additions to QUPER's steps and one modification, which are illustrated in a Process-Deliverable Diagram in Fig. 9 in "Appendix."

In QUPER's first step, prioritization of selected QR was added. The case company operates in a competitive domain and gives priority to market requirements and customers' wishes. Once all QR were identified, the case company prioritized six QR (see Sects. 5.1 and 5.2) to indicate which QR the case company would like to apply QUPER for. The second step of QUPER includes estimating quality levels of competitor products. In the case company's domain, gathering competing products' quality level is difficult due to a dense market with only three competitors. To gather competing products' quality level, a competitor analysis was conducted together with the Sales & Marketing Department. The results of how the competitor analysis was performed are presented in Sect. 6.1.

A modification of QUPER's steps four and five with regard to estimate cost barriers was needed. The case company had difficulties in estimating two cost barriers, in particular the second cost barrier. The practitioners thought of the meaning of the first cost barrier and realized, for the case company, it is enough to estimate one cost barrier. In QUPER's last step, estimate candidate targets; a direct cost was added. The direct cost, relating to the development cost, is an estimate of reaching the defined targets.

5 Evaluation design

The evaluation in this paper was carried out using a qualitative research approach, namely action research [25] and in-depth semi-structured interviews [25]. Qualitative research aims to investigate and understand phenomena within its real-life context and is useful when the purpose is to explore an area of interests [25]. In addition, qualitative research aims to improve the understanding of phenomena [25], [27]. Action research aims to improve practice, the understanding of practitioners, and the situation in which the practice took place [25]. Hence, we chose action research to test QUPER in a live development situation (referred to dynamic validation hereafter). The idea was to use QUPER for real requirements in order to validate its components and to investigate whether QUPER was appli-

cable to a real development situation, which involved validating all three views of QUPER.

In action research, a researcher enters the project where tasks are performed by using the researchers method. Action research comprises of four steps [25]:

1. Plan how current practice can be improved.
2. Implement the plan.
3. Observe the effects.
4. Reflection.

After step four (reflection), the researcher evaluates the performance of the used method and draw conclusions. The evaluation of QUPER's performance was conducted by using semi-structured interviews (referred to static validation hereafter). Due to the potential richness and diversity of data that could be collected, semi-structured interviews would best meet the objectives of the evaluation. Semi-structured interviews help to ensure common information on predetermined areas is collected but allow the interviewer to probe deeper where required.

The evaluation was divided into three phases, where the first phase (planning) relates to the first step in action research. The second phase (workshop) relates to the second and third steps in action research, while the third phase (interviews) relates to the last step (step four) in action research. Each of the three phases is detailed below.

5.1 Phase 1: planning

In the planning phase, brainstorming sessions, meetings to plan the study, and how to apply QUPER at the case company were conducted. In a meeting with the case company, six QR such as upgradability, usability, efficiency, and reliability were identified for applying QUPER. The interview instrument [5] was designed with inspiration from [3]. To test the interview instrument, a pilot interview was conducted to improve the instrument prior to the static validation. The selection of interview subjects for participating in the evaluations was conducted in cooperation with a manager at the case company. Four interview subjects representing different roles were chosen. The roles chosen are as follows: two product managers, one product quality manager, and one acceptance manager.

5.2 Phase 2: workshop

This phase consists of two steps: presentations and applying QUPER in practice.

Presentations The theory and concepts of QUPER and practical application of the model were presented in several steps. A first presentation was held for a product manager where an introduction of QUPER was presented. After the

presentation, the product manager showed an interest in QUPER. Therefore, a new presentation of QUPER and its practical application was given to several managers at different departments. After the second presentation, planning meetings of how to apply QUPER at the case company using the generic guidelines presented in Sect. 2.1 were conducted. The adaption of the generic guidelines is presented in Sect. 4. As a result of the planning meetings, a third and final presentation of QUPER and model tailoring was presented to selected managers and experts, which were selected based on their roles and expertise by the local manager.

Applying QUPER in practice The main goal of the workshop is to achieve an understanding of how to use QUPER on real requirements in coming projects. During the application of QUPER, the second author provided help and feedback of how to apply QUPER on their requirements to the practitioners. Due to limited time to the dynamic validation and absence of vital information, the case company decided to apply QUPER on two QR: one upgradability and one performance requirement (see Sect. 6.1).

5.3 Phase 3: interviews

The interview phase was conducted in two steps: data collection and data analysis.

Data collection The static validation uses a semi-structured interview [25] approach where all interviews were carried out by one interviewer and one interviewee. During the interviews, the purpose of the study was first presented and followed by questions about applying QUPER in practice. Questions about applying QUPER in practice were discussed in detail. For all interviews, varying in length from 35 to 60 min, we took records in the form of audio recording and transcribed the recordings in order to facilitate and improve the analysis process.

Data analysis The content analysis [25] involved marking and discussing interesting sections in the transcripts. The first author examined the categories from different perspectives and searched for explicitly stated or concealed pros and cons in relation to applying QUPER in practice. The results from the analysis are found in Sect. 6.

5.4 Validity threats

In this section, threats to validity in relation to the research design and data collection are discussed. We consider the four perspectives of validity and threats as presented in Wohlin et al. [32].

Conclusion validity

Threats to conclusion validity arise from the ability to draw accurate conclusions. The interviews were done in one uninterrupted work session. Thus, answers were not influenced by internal discussions. The sampling strategy used for the dynamic

validation may pose a threat to the validity of the investigation. The subjects may not be totally representative of the role they represent at the company. The main assurance that this misrepresentation is minimized is the fact that the subjects were selected in cooperation with a senior manager with extensive knowledge and experience concerning the development process and the personnel at the company. To ensure that the interview instruments, including the posed questions, are of high quality to obtain highly reliable measures, several pilot studies were conducted to avoid poor question and poor layout, prior to conducting the interviews.

Internal validity

These threats are related to issues that may affect the causal relationship between treatment and outcome. As the evaluation of the QUPER model was conducted with different subjects, they were called upon to voice their opinions and views regarding the release planning practices with regards to the implementation of QUPER. The potential problem of instrumentation threats was alleviated by developing the research instrument with close reference to literature relating to quality requirements, influenced by previously validated and piloted interview instrument [10], [30], [3]. Moreover, keeping the interview session to 60 min, which was possible by collecting background information before the interview session started, alleviates maturation threats [6].

External validity

The external validity is concerned with the ability to generalize the results, i.e., in this case, the applicability of QUPER in industry at companies other than the case company. Qualitative studies rarely attempt to generalize beyond the actual setting since it is more concerned with characterizing, explaining, and understanding the phenomena under study. The nature of qualitative designs also makes it impossible to replicate since identical circumstances cannot be recreated. However, as some of the problems introduced as motivation behind the conception of QUPER, to some extent could be general for organization faced with developing embedded products for a market.

However, it is not possible to generalize the results from this evaluation; although from a transferability perspective, the concepts and practical application of QUPER in this study and in [23], [3], [5], [21], [4] can provide an overview of faced challenges when QUPER has been implemented.

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. By collecting data from four subjects representing four

Table 1: Overview of features

Feature	Quality Indicator	Measurement	Interface				
Upgradability	Time	Time measured in minutes	P	Q	R	S	T
	Download success rate	Success rate	P	Q	R	S	T
Transaction Processing	Speed A	Time measured in seconds	P	Q	R	S	T
	Speed B	Time measured in seconds	P	Q	R	S	T

different roles on the topic, mono-operation bias [32] was avoided. The potential problem of evaluation apprehension [32] was alleviated by the guarantee of anonymity as to all information divulged during the interviews and the answers was only to be used by the researcher, i.e., not be showed or used by any other participants, companies, or researcher.

6 Evaluation results

The evaluation results are presented in the four following subsections; general view of QUPER and the first two steps of applying QUPER in practice (Sect. 2.1), the benefit view, the cost view, and the roadmap view.

6.1 General view of QUPER

The subjects identified two, one upgradability and one transaction processing feature (QR), to apply and use the concepts of QUPER for the evaluation. For each of the features, two quality indicators (QI) were identified, and for each QI, five different interfaces were used, which is illustrated in Table 1 (to preserve confidentiality, the names in Table 1 are fictitious); that is, for each quality indicator, QUPER's steps 2–6 were performed five times (once for each interface). Throughout the evaluation results of applying QUPER at the case company, two fictitious examples of quality indicators, "Download success rate" and "Speed B" will be used to illustrate the process of QUPER.

The case company's previous process of estimating and quantifying QR was based on gut feeling. One subject explained there was never a market approach. Instead, the practitioners estimated and quantified the QR based on what they would

find acceptable. The requirements were tested later in the process (after deliverance and implementation), but only on workability and not if certain goals (level of quality) were achieved. One subject explained, we tested more on the functionality part and not other aspects such as QR. The introduction of QUPER helped the practitioners to become more aware of QR and their position on the market in relation to their competitors. This was further explained by one subject, *"the QUPER way of thinking is essential because it forces you to know where you stand on the market, what you want to achieve, and how much it would cost you to get there."* Due to the introduction of QUPER, the company constantly thinks of, measures, and defines its position on the market, which, according to the subjects, improves the early decision-making process by raising discussions.

One of QUPER's three goals (Sect. 2.1) is an easy to use model, i.e., easy to learn, remember, understand, and use by practitioners. All subjects agreed that it is easy to identify which QR is applicable to apply and use the concepts of QUPER for. Moreover, most of the subjects viewed QUPER as understandable, not very easy, but definitely not difficult. However, one subject found QUPER's concepts of breakpoints and cost barriers difficult to understand. The subject explained, *"for me, it was the terms (terminology). We are not used to such terms or way of working... it takes some time, mostly because it is completely new to us"*.

The second step in QUPER's practical guidelines is to identify reference levels by gathering information about the competitors. With only three competitors in a relatively dense market, it is not desirable to give away information that the competitors may use. Hence, it was a challenge to conduct a competitor analysis. One subject explained the situation, *"it is difficult for us to know where the competitors are, we know just about, but we would need to put in far more energy in order to gain more information and then determine where we are positioned."* More structured information about the competitors would be needed in order to complete the steps according to the subjects.

Another subject stated that the Marketing and Sales Department calls its competitors, as private individuals that are interested in their products, to collect sales information, e.g., offer-prices and the process. One subject missed the opportunity to receive important information about its competitors from the customer feedback. The customers give little input about the competitors.

To be able to gather any information about the competitors' current level of quality, with regards to the selected quality requirements, a short questionnaire was developed together with the Marketing and Sales Department. The second author called all three of the competitors to conduct a competitor analysis in a discrete manner. Since sales-employees gave the answers to technical questions, the second author decided to conduct the competitor analysis a second time to confirm the answers. In addition, according to the case company's experts, some of the collected quality levels were viewed as unrealistic. The second competitor analysis was conducted by using the same questionnaire as in the first round, but with a different sales-employee that answered the questions the first time; names

Table 2: Speed B breakpoints in seconds

Breakpoints	**Speed B (per interface)**				
	P	Q	R	S	T
Utility	30	23	3	20	15
Differentiation	18	20	1.5	12	8
Saturation	10	5	1	10	5

Table 3: Download success rate breakpoints in percentage

Breakpoints	**Download success rate (per interface)**				
	P	Q	R	S	T
Utility	60	60	60	40	50
Differentiation	90	90	90	75	90
Saturation	100	100	100	80	95

of the sales-employees were noted.

A general challenge of applying QUPER was identified. Moving from no method/process of dealing with QR to a method with new terminology, according to the subjects, it was difficult to apply a new model and its terminology. The main reason for this was that the subjects were used to estimate quality requirements based on gut feeling, that is, no process or common terminology was used in the pervious ”process”. Working with QR in a more structured way was completely new to the subjects.

6.2 Benefit view

Tables 2 and 3 show the identified breakpoints for the quality indicators Speed B and Download success rate. When estimating the three breakpoints, a combination of expertise in the area, the latest tests, and years of domain knowledge, and experience of the system were used. The utility breakpoints are mostly defined by third-party organizations, which all actors in the electronic payment-processing market must adhere to. However, even if the estimated value marks the shift useless to useful quality as defined by the third-party organizations, it may not be acceptable from a customer point of view. Therefore, the case company sometimes used their experience and extensive domain knowledge to estimate the utility breakpoint. Collecting and organizing customer feedback is an important input to the estimations of saturations breakpoints. One subject explained the importance of having a high accuracy of the estimates because *”we want to earn more money on our contracts, so the better the quality management, the more profit we have”*.

The uncertainties of the estimated breakpoints range from reasonable to quite some doubts. This is mainly due to the fact that the case company had never actively upgraded their products remotely. The estimates are based on small tests, which did not represent the actual numbers in the 'real world'. In addition, the subjects stated that the uncertainties in estimates are also related to lack of gathering and using information about their own product's level of quality, i.e., the existing information was not reliable. However, one subject stated, if we can apply it correctly and measure, then we would get a better prediction.

In general, all subjects viewed the saturation breakpoint as the most important one. An example was provided by one subject, *"for instance, we were going to invest more in improved quality [for a certain QR], but for the customers, it is less interesting if it is actually one second faster, while the investment was €40,000. That is what the graph [the benefit view] explains."* Moreover, the benefit view was viewed as a great template to use for discussion with management. If management wants to improve a certain quality requirement, there is no need to write a full story because the benefit view show where the market is and when to stop improving the quality (the saturation breakpoint).

The subjects identified three main challenges:

1. Difficult to estimate the saturation breakpoint.
2. Difficult to estimate the differentiation breakpoint.
3. Difficult to apply QUPER's steps within the case company.

The saturation breakpoint was experienced as the most challenged breakpoint to estimate, it was difficult to know at which level the quality was seen as excessive. When do you know that you are 'over the top' asked one subject? Moreover, another subject questioned who should decide what is excessive quality.

The differentiation breakpoint was identified as a difficult breakpoint to estimate. Two main reasons were provided. The first reason is related to the difficulties to gather information about the competitors, while the second reason is closely related to the lack of feedback from the main customers.

The third challenge is the applicability of QUPER's steps within the case company. One subject explained that there are no hard figures to test, which makes it difficult to estimate a value for a breakpoint. In addition, no customer research is conducted.

6.3 Cost view

Tables 4 and 5 illustrates the estimated cost barriers (CB) for Speed B and Download success rate, where Qref represents the current level of quality for each quality indicator. According to QUPER's guidelines (Sect. 2.1), two cost barriers should be estimated. However, in the case company, as explained in Sect. 4, only the

Table 4: Speed B cost barrier

Interface	Qref	Q1	Q2	CB
P	35	24	24	15% of the total software optimization budget
Q	15	15	15	
R	3	3	3	
S	15	15	15	
T	3–15	10	10	

Table 5: Download success rate cost barrier

Interface	Qref	Q1	Q2	CB
P	85	90	95	10% of the total software optimization budget
Q	95	90	95	
R	89	90	95	
S	40	90	95	
T	50	90	95	

first cost barrier is estimated (Tables 4, 5 show the quality level (Q1 and Q2) for both CB, but the quality level is estimated to be the same). The estimated cost for the cost barriers is measured through percentage of the total software optimization budget, e.g., for Speed B, the cost to improve all interfaces (to reach Q1 and Q2) is estimated to be 15% of the total software optimization budget. To estimate the cost barrier, the subjects' related, as much as possible, to products where they have a direct influence since not all products are always accessible for the case company. The subjects used market expectations and their own knowledge and experience as input to the estimates. One subject explained there are values that must be adhered to in the market segment; however, important QR for the customers must also have a dedicated part of the improvement budget.

The subjects viewed the first cost barrier as a reachable improvement. Moreover, the first cost barrier, according to the subjects, is related to software optimization. The subjects believe the second cost barrier should be seen as a differentiator from the competitors, and that the cost is related to the cost of designing new software architecture. One subject stated, *"CB2 is more a blank page, how would we do it if we were to do it [estimate the second cost barrier] all over again, probably a new architecture"*. The estimates for the cost view are more uncertain than the estimates for the benefit view according to the subjects. Two reasons were discovered, first, lack of reliable and historical data to support the estimates of the cost

barriers. Second, the subjects' lack of in-depth knowledge and experience of cost estimations of QR had an impact on the uncertain estimates.

In general, all subjects agreed that the cost view is very useful in their situation. The mapping of cost to potential quality level improvements in an early phase, i.e., before the development starts, is very important. One subject explained, *"This is the most important for me. Thinking in advance which budget, what is most important, what do we want, and what does it [the system in terms of quality level] have to deliver"*. Another subject compared with the previous process of handling QR, before the competitors quality level was unknown, it was not known how much we needed to improve, and therefore, impossible to estimate the cost. The subject explained, with QUPER it is possible to understand where our product stands in relation to the competitors, and *"we have an idea where the differentiation breakpoint is, so we can give better estimations on how much we should improve and if we find it worth it"*.

6.4 Roadmap view

Tables 6 and 7 show the targets, an interval between good and stretch target, for Speed B and Download success rate. In general, the subjects appreciated the use of intervals to quantify QR. However, intervals should not be used for all QR, some QR are better quantified by an absolute value. In addition, as explained in Sect. 4, the case company added a direct cost in the last step. The reason is to estimate the cost (direct cost in Tables 6, 7) to reach the target, which is the cost to achieve the targets for all five interfaces. The direct cost in Tables 6 and 7 is calculated by:

(Number of months needed)
x (number of needed employees)
x (working hours per month)
= numbers of needed man-hours

(Number of needed man-hours) x (cost per hour)
= direct cost

Figures 7 and 8 illustrates the visualization of all collected information from QUPER's six steps for the QR Speed B for interface T and Download success rate for interface T.

One subject stated that he/she already uses the mindset of QUPER for functional requirements. The subject explained feature A is a functionality that some of their competitors have, but the case company is missing. To stress the importance of implementing this feature, the subject created a presentation for the board to explain where the competitors are, and where the case company is, and required investment to surpass the competitors. According to the subject, the introduction

Table 6: Speed B targets and direct cost

Interface	Qref	Good target	Stretch target	Direct cost
P	35	30	22	€430,080
Q	15	12	8	
R	3	2	2	
S	15	12	10	
T	3–15	7	5	

Table 7: Download success rate targets and direct cost

Interface	Qref	Good target	Stretch target	Direct cost
P	85	90	100	€273,360
Q	95	90	100	
R	89	90	100	
S	40	80	95	
T	50	80	95	

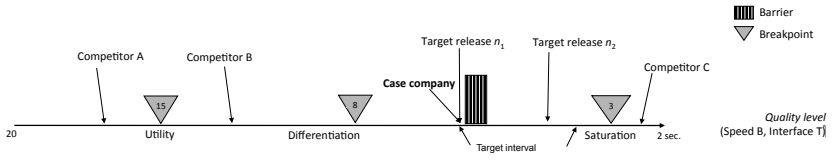


Figure 7: Roadmap view of Speed B

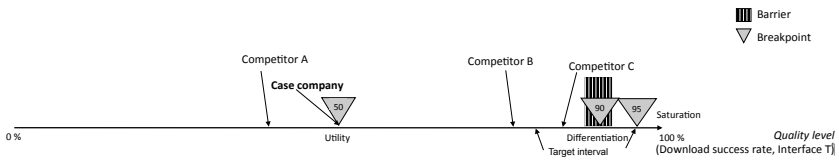


Figure 8: Roadmap view of Download success rate

of QUPER helps the case company in making these decisions for both functional requirements and QR.

The roadmap view was viewed as the most essential and influencing part of QUPER by the subjects. The visualization of the roadmap view and the support for early decision-making, the constant thinking of where to aim, and how much of the budget should be reserved to achieve it, triggers for improvement. One subject explained, *"when you show two competitors that are in front of [the case company] in a picture, then that is our trigger to improve"*. Moreover, another subject stated that the *"roadmap view depicts clearly where you stand and your competitors... it visualizes and people are sensitive for that, it stays with you and you remember it"*.

6.5 Lessons learnt

Five main lessons learnt were identified during the evaluation of the QUPER model in this study:

1. Time to adopt and apply QUPER in practice.
2. Tailoring of QUPER.
3. Applicability of QUPER in more than one domain.
4. Applicability on upgradability and performance requirements.
5. Improved decision-making.

Time to adopt and apply QUPER in practice

One of QUPER's goals is that the model should be easy to learn, remember, and use by practitioners. Within three months, it was possible to introduce, apply, and evaluate QUPER at the case company, without any direct support. The practitioners at the case company that were involved in the introduction of QUPER spent between 10 and 13 h each, that is, in performing QUPER related activities. However, these hours do not only include performing QUPER's six steps, but it also includes introduction of QUPER, presentations, and general meetings, brainstorming and planning meetings, results and evaluation meetings, and the tailoring of QUPER.

Tailoring of QUPER

A third goal with QUPER is for the model to be domain-relevant, i.e., it should be possible to combine the concepts of QUPER with existing practices by tailoring the model. It was possible, with a few additions and one modification, to adopt and tailor QUPER's generic concepts and guidelines (Sect. 2.1) to the case company. The main addition to QUPER's steps was competitor analysis. In order to identify competitive advantage and identify the differentiation breakpoint, competitors' level of quality is essential. If an organization does not have a process or

an approach for gathering information about its competitors, one approach could be to call its competitors in a discrete manner through anonymous inquiries to, e.g., the competitors sales and support services, which was a strategy applied by the case company. Furthermore, a direct cost was added to the candidate target step. Instead of quantifying QR by using QUPER and then, in a later phase of the development process, estimate the cost, the case company identified a possibility to estimate the cost of achieving the specified target directly in QUPER's steps. One main benefit of adding the direct cost was the ability to discuss the targets in release planning and know its cost, which may have an influence on the specified targets for coming releases.

Applicability of QUPER in more than one domain

In previous evaluation of QUPER [3], [4], the model has only been evaluated in the mobile handset domain. In this evaluation, the QUPER model was evaluated in a different domain, namely the electronic payment-processing domain. The results from the two domains are similar. The view of an improved release planning process by using QUPER and importance of a rich understanding of the market as input for release planning and early decision-making is inline with the results in Berntsson Svensson et al. [3]. In addition, similar to the experiences in the mobile handset domain [3], the main benefit of the breakpoints was the saturation breakpoint, where the quality level changes from competitive to excessive quality. Moreover, the challenges of the identification of the values for the differentiation and saturation breakpoints are the same as in [3]. However, the reasons differ, while the case company in this study had problems with gathering competitor information to be able to calibrate the differentiation breakpoint, in [3], the mobile handset company's problems was related to when to stop calibrating those breakpoints.

Similar to the practical guidelines of QUPER's cost view in [4], the subjects believe that the first cost barrier is related to software optimization, while the second cost barrier may be related to new investments in larger architectural evolution steps. Furthermore, the first cost barrier is viewed as easier to estimate than the second cost barrier. The reason is that the first cost barrier is closer to the current quality level than the second one, which is inline with the results in [4].

Applicability on upgradability and performance requirements

In general, the subjects did not experience any major differences between applying QUPER for upgradability or performance requirements. The subjects believe that performance requirements are easier to measure; however, if the success rate of upgradability requirements is specified correctly, it is as easy to measure. One difference between the two types of QR was identified; upgradability requirements have more environmental variables/uncertainties than performance requirements. However, as stated by one subject, *"if we are relating to QUPER, then QUPER*

helps us with upgradability requirements because of its complexity". This result may indicate that the QUPER model is applicable to more QR beyond performance requirements that where applied in previous case studies [3], [4].

Improved decision-making

In general, QUPER does not only help in creating a more aligned view of quality requirements, but also to use one method to measure all quality requirements. All subjects confirmed that QUPER would support and coordinate the early decision-making process, e.g., release planning. The subjects agreed that the roadmap view provides a clear and understandable representation of the market as a basis for decisions. Particular information about targets, what the estimated cost for reaching the targets is, competitors and your own product's level of quality, and the identified saturation breakpoint was viewed to be of major help in release planning. To be able to discover and quantify quality targets in relation to market expectations early during system analysis may help software architects to build in the markets' and users' expected level of quality into the system and its architecture from the beginning, Hence, minimizing the risk of architectural failure and falling short of meeting users' real needs.

Before the case company used QUPER, quality requirements were dealt with in an ad-hoc approach. The use of QUPER improved the decision-making of specifying and quantifying quality requirements, that is, QUPER made it clear to the case company where (what level of quality) they are in relation to market expectations and competitors, what level of quality they should aim for, and what the aimed quality level will cost the case company; for example, when the roadmap view of download success rate (Fig. 8) showed two competitors in front of the case company, it was easier, due to more details and a better understanding of the market situation, to make a decision of what level of quality is needed for the next release.

7 Conclusions

This paper has introduced one case study of tailoring, implementation, and evaluation QUPER to provide estimations of benefit and cost of quality targets in relation to market expectations as a basis for the architecting of quality requirements. Industry professionals evaluated the QUPER model in the electronic payment-processing domain in real projects, using real requirements.

First, the overall result indicates that QUPER is relevant in early decision-making process, e.g., release planning. The concepts of breakpoints, competitor analysis, cost barriers, and identification of own products quality level provides a greater understanding of the needed level of quality for the coming releases. The combination of all three views, in particular the visualization of the roadmap view, provides a clear picture of the current market situation, and what level of quality

to aim for. Second, the QUPER model can be used to support early discovery and quantified quality targets in relation to the market and users' expectations, which minimizes the risks of architectural failure and falling short of meeting users' real needs when design the software architecture.

The evaluation indicates, not only that QUPER is applicable and relevant in the selected domain of electronic payment processing, but also extending beyond the previously investigated mobile handset domain. Moreover, the results show that the goal of QUPER to be domain-relevant by tailoring the model is feasible. The case company successfully adopted QUPER's generic concepts and guidelines by adding four steps, where the additional step of competitor analysis was the main and most important addition. In addition, the evaluation indicates that QUPER may be applied to other quality requirements than performance requirements.

The subjects identified two main challenges by using QUPER. First, difficulties to identify and specify the values for the differentiation and saturation breakpoints. Second, determining an improvement where a significant investment is required, i.e., identifying the cost barriers and their values.

This paper has provided a first validation of the usefulness of combining QUPER's all three views. Additional work is still required to improve the QUPER model. First, dependencies may have a major impact on the estimated cost for other features. The cost to improve the quality level for one feature may imply an improved level of quality for other features. This can lead to a change of other features cost barriers and which feature to select for the coming release. Therefore, it is important to find heuristics to efficiently support and incorporate cost dependencies into the QUPER model. Second, Pfleeger highlights the importance of effective tools and tool support to facilitate technology diffusion [20]. To enable easier adoption of QUPER by practitioners, a QUPER tool support will be developed. Third, investigate if software architecture evaluation methods, such as ATAM, may be used together with QUPER as input to the second cost barrier. The subjects believe the second cost barrier is related to new software architecture. Therefore, if methods, such as ATAM, could evaluate the current architecture, identify the current level of quality for a certain quality indicator the SA could generate, may make it easier to estimate QUPER's second cost barrier.

Appendix

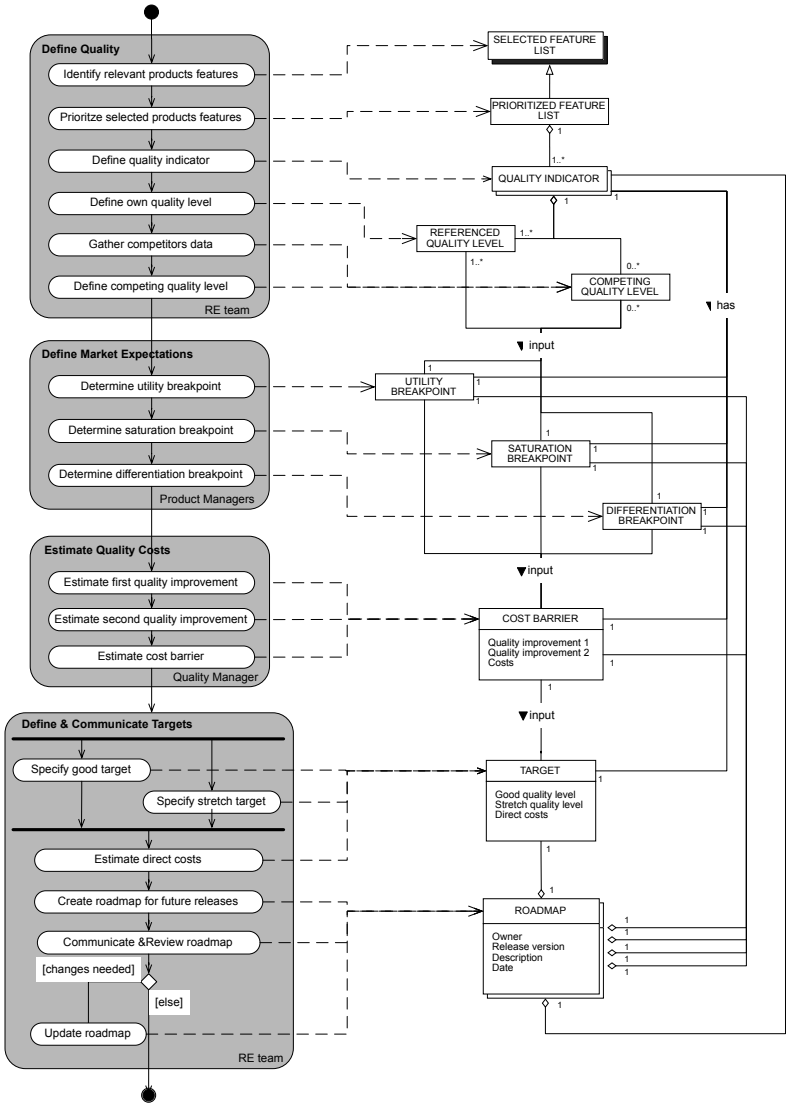


Figure 9: The QUPER activities and data tailored to the specific case

Bibliography

[1] M. Ali-Babar and R. Capilla. Capturing and using quality attributes knowledge in software architecture evaluation process. In *Proceedings of the First international workshop on managing knowledge*, pages 56–62, 2008.

[2] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 2003.

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

[4] R. Berntsson Svensson, B. Regnell, and A. Aurum. Towards modeling guidelines for capturing the cost of improving software product quality in release planning. In *Second proceeding: Short papers, Doctoral Symposium and Workshops of the 11th international conference on product focused software process improvements*, pages 24–27, 2010.

[5] R. Berntsson Svensson, Y. Sprockel, B. Regnell, and S. Brinkkemper. Cost and benefit analysis of quality requirements in competetive software product management. In *Proceedings of the Fourth International Workshop on Software Product Management*, pages 40–48, 2010.

[6] J. Bosch. *Designing and Use of Software Architecture Adopting and Evolving a Product Line Approach*. Pearson Education, 2000.

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

[8] J. Carriere, R. Kazman, and I. Ozkaya. A cost-benefit framework for making architectural decisions in a business context. In *Proceedings of the 32nd international conference on software engineering*, pages 149–157, 2010.

[9] J. Cleland-Huang, R. Settimi, X. Zou, and P. Sole. Automated classification of non-functional requirements. *Requirements Engineering*, 12(2):103–120, 2007.

[10] E. Folmer and J. Bosch. Architecting for usability: a survey. *The journal of systems and software*, 70(1–2):61–78, 2002.

[11] T. Gilb. *Competitive Engineering*. Elsevier Butterworth-Heinemann, 2005.

[12] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE Software*, 23(6):88–95, 2006.

[13] D. Greer and G. Ruhe. Software release planning: an evolutionary and iterative approach. *Information and software technology*, 46(4):243–253, 2004.

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

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

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

[17] R. Kazman, G. Abowd, L. Bass, and P. Clements. Scenario-based analysis of software architecture. *IEEE Software*, 13(6):47–55, 1996.

[18] R. Kazman, M. Barbacci, M. Klein, and S.J. Carriere. Experience with performing architecture tradeoff analysis. In *Proceedings of the 21st international conference on software engineering*, pages 54–63, 1999.

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

[20] S.L. Pfleeger. Understanding and improving technology transfer in software engineering. *Journal of Systems and Software*, 47(2):111–124, 1999.

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

[22] B. Regnell and S. Brinkkemper. *Engineering and Managing Software Requirements*, chapter Market-Driven Requirements Engineering for Software Products, pages 287–308. Springer, 2005.

[23] 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 *Proceedings of the 13th working conference on requirements engineering: foundation for software quality*, pages 277–291, 2007.

[24] B. Regnell, H.O. Olsson, and S. Mossberg. Assessing requirements compliance scenarios in system platform subcontracting. In *Proceedings of the Seventh international conference on product focused software process improvements*, pages 362–376, 2006.

[25] C. Robson. *Real World Research*. Blackwell, 2002.

[26] G. Ruhe, A. Eberlein, and D. Pfahl. Trade-off analysis for requirements selection. *International journal of software engineering and knowledge engineering*, 13(4):345–366, 2003.

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

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

[29] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, and S.B. Saleem. A systematic review on strategic release planning models. *Information and Software Technology*, 52(3):237–248, 2010.

[30] M. Svahnberg, C. Wohlin, L. Lundberg, and M. Mattsson. A method for understanding quality attributes in software architecture structures. In *Proceedings of the 14th international conference on software engineering and knowledge engineering*, pages 819–826, 2002.

[31] M. van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Software product release planning through optimization of what-if analysis. *Information and Software Technology*, 50(1–2):101–111, 2008.

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

PAPER VI

A PROTOTYPE TOOL FOR QUPER TO SUPPORT RELEASE PLANNING OF QUALITY REQUIREMENTS

Abstract

Release planning plays an important role for the success of a software product vendor that develops software-intensive incremental products. It is important that the software product is released to the market at the right time, and offers higher quality than the competitors. However, an especially challenging problem for a software product vendor is to set the right quality target in relation to future market demands and competitor products. In this paper, we present a prototype for QUPER, a special-purpose tool for supporting release planning of quality requirements. The applicability of the QUPER prototype tool is demonstrated through academic and industrial evaluations. The study showed that the tool provides a clear overview of the current market situation by the generated roadmaps, and to reach an alignment between practitioners, e.g., product managers and developers, of what level of quality is actually needed.

Richard Berntsson Svensson, Pontus Lindberg Parker, and Björn Regnell
Fifth International Workshop on Software Product Management (IWSPM), Trento, Italy, 30 August, 2011.

1 Introduction

Market-driven product development and delivery is increasingly common in the software industry [29]. Deciding what requirements to include into a product is not a trivial task, and only a sub-set of the requirements may be included, while others often are postponed to a later point in time [12], [29]. One major goal of market-driven product development is to develop a product with high customer value. Moreover, to increase the chance of market success, it is important that the software product is released to the market at the right time, and it is often of high priority that it offers a better quality than the competitors' products [2]. Hence, release planning is a major determinant of the success of a product [20], [29], and lack of good release planning practices may result in unsatisfied customers [29].

Release planning is the process of deciding what features should be included in the next releases, and what features should be included in which release [29]. In literature, there are several models and tools that provide support for release planning, for example, Release Planning Prototype [6], EVOLVE [12], and the ReleasePlanner^{TM} tool [23]. However, none of these models and tools has explicit support for release planning of Quality Requirements (QR), also known as non-functional requirements, including requirements on e.g. performance, capacity, usability, reliability, maintainability, etc. [13].

To create a successful product and assure quality, it is not enough to fulfill the functional requirements. Quality requirements describe important constraints upon the development and behavior of a software system. Therefore, QR play a critical role in software product development [9]. A challenging problem for an organization that develops market-driven products is to set the right quality targets in relation to future market demands and competitor products. Therefore, we have developed the QUality PERformance model (QUPER) [21] to support release planning of QR with the goal to provide concepts for qualitative reasoning of orders of magnitude of important quality attributes. QUPER has been evaluated in two companies in two different domains, namely the mobile handset [3], [4] and electronic payment processing domains [5]. However, the development and evaluations of the QUPER model are conducted based on manual usage of the QUPER model, i.e., adding information about QR in, e.g., a database or spreadsheet without any visualization (see Section 4.2).

To facilitate technology diffusion from academia to industry, effective tools and tool support are important [17]. Furthermore, to transfer technology in practice, tool support and tool adaptation are important for the model's scalability, and to incorporate training efforts in future model implementations [11].

In this paper, we present a first version of the QUPER prototype tool, a tool that we have developed based on the QUPER model and input elicited from practitioners. A main objective of the paper is to enable easier adoption of the QUPER model by practitioners and thereby to improve the technology transfer in practice. Another major objective is to evaluate how tool support of the QUPER model may

improve the practical application of the model in an industrial context. The development and evaluation of the tool have been conducted in close cooperation between academia and industry. The main contribution of this work is two-fold: (1) the tool prototype for QUPER based on requirements elicitation with several types of stakeholders from an industrial partner using the QUPER method, and (2) a two-step tool evaluation with lessons learned for further development and application of the QUPER tool support and the underlying method.

The remainder of this paper is organized as follows. Section 2 offers an overview of the basic concepts of the QUPER model and related work, while the case company is described in Section 3. Section 4 presents how the QUPER prototype tool was developed and its main features. Section 5 describes the design of the industrial evaluation and the results of the tool evaluation. Section 6 gives a summary of the main conclusions.

2 Background and Related Work

In this section, the basic concepts of the QUPER model are described. Also related work of release planning and tools for QR are presented.

2.1 The Quality Performance Model

The aim of the QUPER method is to support the ability to make early estimates of quality with adequate accuracy in order to provide QR that can act as adequate input to discussion and high-level decision-making in upstream requirements engineering related to, for example, release planning and roadmapping.

QUPER is based on two hypotheses:

1. *Quality is continuous.* Rather than being either included or excluded, QR are assumed of having the potential of being measured on a continuous scale, i.e. a QR is typically viewed as different shades of quality on a sliding scale.

2. *Quality is non-linear.* For QR, such as performance, a change in a software product's quality level may result in non-linear changes to both cost and benefit.

The basis for the construction of the QUPER model is the concepts of breakpoints and barriers.

A *breakpoint* is an important aspect of the non-linear relation between quality and benefit, for example, when a product's start-up time shifts from normal expectations to outperforming most competitors. A breakpoint indicates a change of benefit level with respect to users' perception of quality and market value. The first breakpoint, the *utility breakpoint*, marks the border between useless and useful quality. The second breakpoint, *differentiation breakpoint*, marks the shift from

useful to competitive quality, which only a few of the competing products on the market reach. The third breakpoint, *saturation breakpoint*, marks the shift from competitive to excessive quality. That is, quality levels beyond this breakpoint have no practical impact on the benefit for the considered context.

A *barrier* represent an interesting aspect of the non-linear relation between quality and cost, for example, achieving better performance may require an expensive rebuild of the architecture. For a specific quality aspect in a specific context, we approximate the quality-cost relation to have two different steepness ranges. A cost 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.

The roadmap view (see Section 4.2) positions the breakpoints and cost barriers on the same scale, which enables a visualization of breakpoints and cost barriers in relation to a product's current quality level and the competing products' quality (reference levels). To support release planning, the roadmap view incorporates targets for coming releases.

The QUPER model generally aims to avoid making complete predictions of the inherently difficult relations between a product's benefit, cost, and quality. Instead, QUPER aims to simplify the problem by finding reasonably good predictions of a limited set of breakpoints and cost barriers.

For a complete and detailed description of the QUPER model, we refer to [5], [19], [21].

2.2 Related Work

Several approaches and strategies have been proposed to resolve issues related to release planning. In this section, a selection of release planning methods and tools is presented: EVOLVE [12], EVOLVE* [25], F-EVOLVE [14], Release Planner Prototype [6], and a method for software release planning using optimization of what-if analysis [1]. All of these methods use linear programming and/or generic algorithms to resolve the optimization aspect of release planning. Input data such as requirements value, requirements cost, resources, stakeholder importance, and budget constraints are used to come up with an optimal or near-optimal selection of features with respect to trade-offs between aspects such as cost, value and resource allocation. In addition, analysis of requirements dependencies is present in these methods. However, none of the methods address quality constraints and the support of communication and estimation of desired quality levels. According to Svahnberg et al., only two strategic release planning methods address quality constraints [27]. The quantitative Win-Win [24] addresses effort and time constraints, but not the quality level of quality requirements. The only method to address quality and cost constraints of QR is the QUPER model. For a more comprehensive analysis of 24 strategic release planning methods, we refer to Svahnberg et al. [27].

ReleasePlanner^{TM} [23] is a release planning tool that goes beyond the prioritization of features. ReleasePlannerTM takes a comprehensive approach of release planning by including optimization and considering available resources.

According to Ruhe, traditional release planning favors delivery of functional requirements, while the view of quality aspects, such as performance and reliability, is missing in the released products [23]. One approach to include quality aspects in release planning is to use EVOLVE II and generate alternatives for cost devoted to functional versus quality requirements [23]. The first alternative devotes, e.g., 100% of the resources to functionality, the second 90% to functionality and 10% to quality, while the fifth alternative equally divides the resources between functionality and quality. Although this approach includes the cost for QR in release planning, what level of quality the next release should have on a continuous quality scale for specific quality aspects is not considered. There are potential strategies for combining QUPER with EVOLVE II, e.g., by using QUPER in the decision process of needed level of quality and use this as an input to EVOVLE II for resource allocation. Such combinations are out of scope of the presented study and may be objects of further studies.

Prioritization of requirements is often conducted prior, or as a part of the release planning process. Several prioritization techniques and cost-benefit models are introduced in the literature, for example Planguage [10]. Planguage has roadmap related concepts such as past, record, and trend in templates for QR. QUPER could be used together with the planguage method to express breakpoints, barriers, and targets related to, for example, express competing products in different market segments.

Finally, there are a number of tools that are not focusing on the problem of release planning, but focus on QR. For example, NFR-assistant [28] and SA^3 [26] that are based on the NFR Framework [7] using a goal-oriented approach for linking customers' wishes to solutions and rationales, while Cleland-Huang et al. describes a technique for automating the detection and classification of QR scattered across both structured and unstructured documents [8]. However, these approaches do not address the decision-making on setting quality targets for future software releases, which is a major goal of QUPER.

3 Case Company Description

The QUPER prototype tool presented subsequently has been developed in close cooperation with a case company that develops software and hardware for the mobile handset market. The case company has more than 5,000 employees and develops their products for a global, competitive market using a product line approach [18]. The company's requirements database consists of more than 20,000 requirements where approximately 25% of the requirements are quality requirements [15]. The company has several consecutive releases of a platform (a com-

mon code base of the product line) where each of them is the basis for one or more products that reuse the platform's functionality and qualities. The case company has two types of platform releases, a major and a minor release. A major release has a lead-time between two and three years from start to launch, and the focus is on functionality growth and quality improvements of the product portfolio. Minor platform releases usually focus on the platform's adaptations to different products. The company uses a stage-gate model with several increments, where milestones are used for controlling and monitoring the project progress. There are four milestones for requirements management and design before the implementation starts, and three milestones for the implementation and maintenance phase.

4 QUPER Prototype Tool

The QUPER prototype tool is designed for supporting release planning of QR. The tool is operated through a Graphical User Interface (GUI) that consists of three parts, (1) a menu, (2) a hierarchical tree structure to the left (see Figure 3), and (3) an activity window to the right (see Figure 3). The menu and hierarchical the tree structure is always visible for the user through out the work flow, while the activity window displays different information depending on the action of the users.

The purpose of the QUPER prototype tool is to provide practitioners, mainly product and project managers, with support for the usage of the QUPER model in an industrial environment.

4.1 Development of QUPER prototype tool

The development of the first version of the QUPER prototype tool, as presented in Section 4.2, was carried out in close cooperation between academia and industry. The first version is developed with the basic functionalities that allow a practitioner, e.g., product managers, to work with the basic concepts of the QUPER model (as presented in Section 2.1). The development of the QUPER prototype tool consists of the following two steps.

Step 1 - Requirements Elicitation: The requirements elicitation step used interviewing [13] as the elicitation technique. Eleven subjects from the case company were chosen to give a rich picture of the domain. One interviewee and one interviewer attended all interviews, which varied between 40 and 60 minutes. Transcripts of all interviews were made in order to facilitate and improve the elicitation process.

The main findings, and input to the development of the QUPER prototype tool from the elicitation interviews were the following high-level quality goals and priorities:

Table 1: Results from close-ended questions

Questions	**strongly disagree <-> strongly agree**						
1. The tool is easy to understand	1	0	1	0	1	2	0
2. The guide took too much time to use	0	0	3	1	0	1	0
3. The guide provides enough information	1	0	1	1	1	0	1
4. The quality form took too much time	2	2	1	0	0	0	0
5. The manual and guide are difficult to find	2	0	1	2	0	0	0

- *Usability: comprehensive screens*, i.e., being able to manage all input data without having to jump between several windows.
- *Usability: flexible workflow*, i.e. not being forced to add all information; it should be possible to generate a roadmap view without the prerequisite to first add, e.g., the cost barriers.
- *Interoperability: export to standard picture file formats* for inclusion in presentations to other managers and staff using other applications.
- *Interoperability: interchange with spreadsheet programs*, i.e., an import and export function of spreadsheets were considered important since many subjects already have requirements stored in spreadsheets.
- *Prioritization: focus on the roadmap view*, the subjects considered the roadmap view to be the most important view of the QUPER model to have in the tool.

Step 2 - Academic Evaluation: Once a beta version of the QUPER prototype tool had been developed, usability tests [13] were performed in academia. The main goal of the usability tests was to achieve an understanding of the usability and the workflow of the QUPER prototype tool. In addition, finding bugs and inconsistencies was an important part of the tests. Five subjects, four PhD students in Software Engineering, and one business administration staff at Lund University participated in the usability tests, which lasted for about 30 minutes.

Each subject was provided with a set of instructions, including data about two quality requirements. In addition, a questionnaire with both close-ended and open-ended questions was provided to the subjects. The close-ended questions used a seven-point Likert scale, representing levels of agreement from ”strongly disagree” to ”strongly agree”. The result from the close-ended questions is illustrated in Table 1.

The main feedback from the open-ended questions include:

Table 2: QUPER prototype tool features

ID	Feature
1	A detailed guide to help users to use the tool for the first time
2	A form for experienced users to add and edit quality requirements
3	Generation of roadmap for each quality requirement
4	Data management of features and quality requirements
5	Version control of quality requirements
6	Data serialization
7	Spreadsheet interoperability
8	Manual about the QUPER model and the prototype tool

- *Inconsistencies* between the inputs fields and the guide and manual.
- *Presentation of roadmap view*, overlapping text in the roadmap view.
- *Bugs and missing features in the tool*, e.g. the QUPER prototype tool crashed in several situations.

The usability tests provided valuable feedback, which was an important input to the finalization of the first version of the QUPER prototype tool, which is described in the following section.

4.2 Functionality of QUPER prototype tool

We have identified eight main features of the QUPER prototype tool, which are listed in Table 2. In Figures 2, and 3, we indicate the GUI location of access to these features. Each feature is briefly described below.

Detailed guide. The tool provides a detailed step-by-step guide (see number 1 in Figure 3), which is based on [5], for, e.g., users that are new to QUPER and its concepts. Through a series of steps, explanations, and an example, the user is guided through the steps of the QUPER model while entering relevant data about a QR in the fields at the same time. While some of the steps are mandatory, others are optional, e.g., entering information about cost barriers is optional, which is illustrated in Figure 1.

Form for adding QR. The prototype tool has two forms for adding and editing information for features and QR. The feature part is primarily used to link QR to a particular feature. The QR form, which is illustrated in Figure 2, see number 2, contains fields for all of QUPER's concepts (see Section 2.1). The fields have been divided into seven small forms.

First, the top left form is for basic information, *id*, *definition*, and *state* about the QR. To the right of the basic form is a field for information about what scale

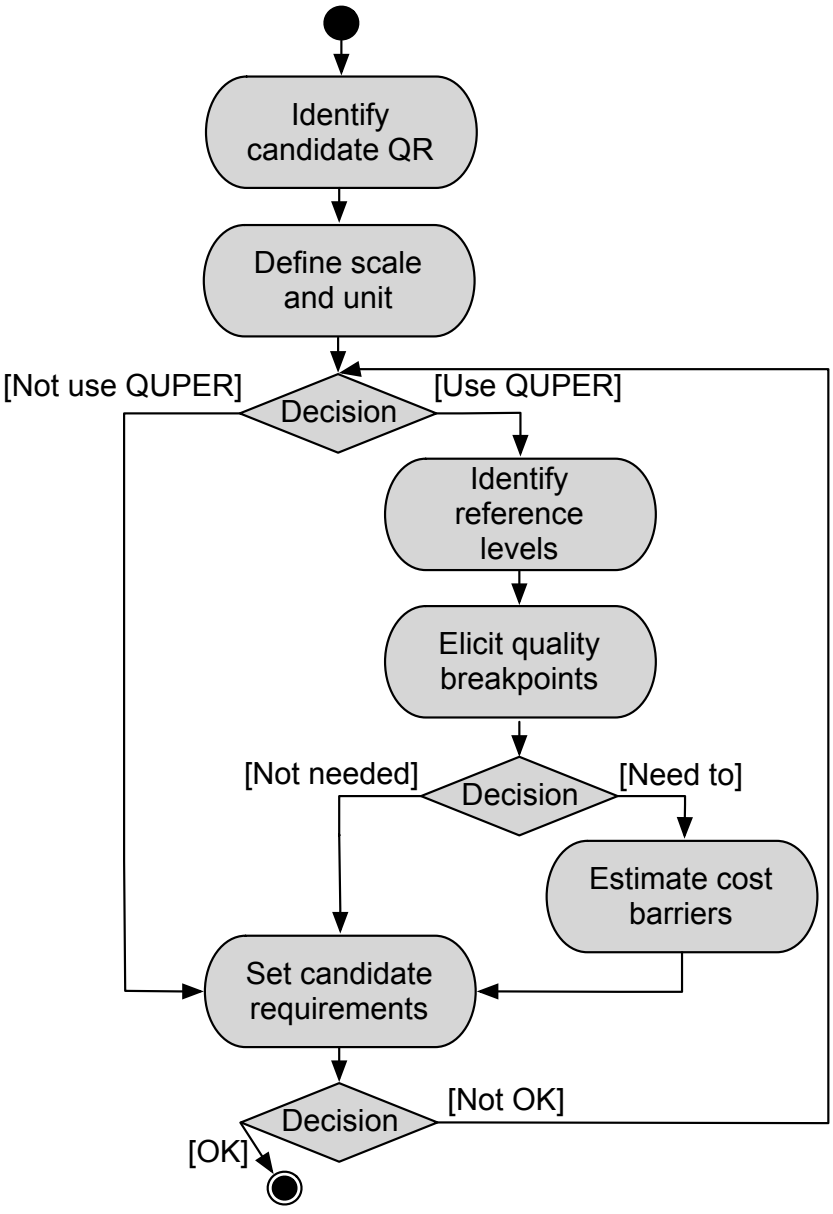


Figure 1: Overview of the QUPER prototype tool workflow

of the quality level should be used. For example, if a performance requirement is added, should the quality level be measured in milliseconds or seconds? Besides the unit of scale, the user can chose if a higher value means better or worse quality. For example, if the QR ”time delay” is added, lower value means higher quality, while for a QR such as ”battery life”, higher value means higher quality. In the generated roadmap view, the right side will always view higher quality, regardless if it is a higher or lower value.

Four (identify reference levels, elicit breakpoints, estimate cost barriers, and define targets) of the remaining five forms are related to the basic concepts of the QUPER model, which can be added into the generated roadmap. The four forms use a tabbed format to allow for several, e.g., reference levels and targets, without taking extra space in the activity window.

The form for the breakpoints is fixed to three tabs, one for each breakpoint, *utility*, *differentiation* and *saturation*. Each breakpoint tab has fields for a rationale and option of three different estimation techniques (all forms with estimations have the same choice).

Estimates can be given in three forms, depending on how the potential uncertainty in the estimates should be captured:

- Point estimates including a single figure, e.g. 3 minutes.
- Interval estimates including a [min, max] interval, e.g. 3-4 minutes.
- Triangle distribution estimates including a three-tuple of [low bund, most probable, high bound] figures that show the estimated probability distribution, e.g. low: 3 minutes, high: 5 minutes, probable: 4 minutes.

The user can choose which of the three estimation techniques to use. The choice will activate and deactivate the three fields according to which ones are relevant for the selected technique. The point estimate thus only has a ”most probable” value, the interval estimate has a ”lower bound” and a ”top bound”, while the triangle estimate has all three.

The forms for reference levels, barriers, and targets use tabs as well; however, the user is allowed to create new tabs and delete old ones. For reference levels, each tab contains a field for the product name and fields for estimations, while the forms for barriers and targets contain fields for a short description, a rationale, and estimations for the quality level.

The last form, at the bottom left corner in Figure 2, is for the scale for the cost (for barriers). This form only contains a field for entering the unit of the cost, e.g. 3.0 weeks. When applicable, a roadmap will be generated and displayed on top of the form (not shown in Figure 2 due to space related issues, but it is the same roadmap as shown in Figure 3).

Finally, to help the user, in addition to the detailed guide and manual, each form has a tooltip help area in terms of a question mark. When the mouse is positioned over any of the question marks, short help text is shown.

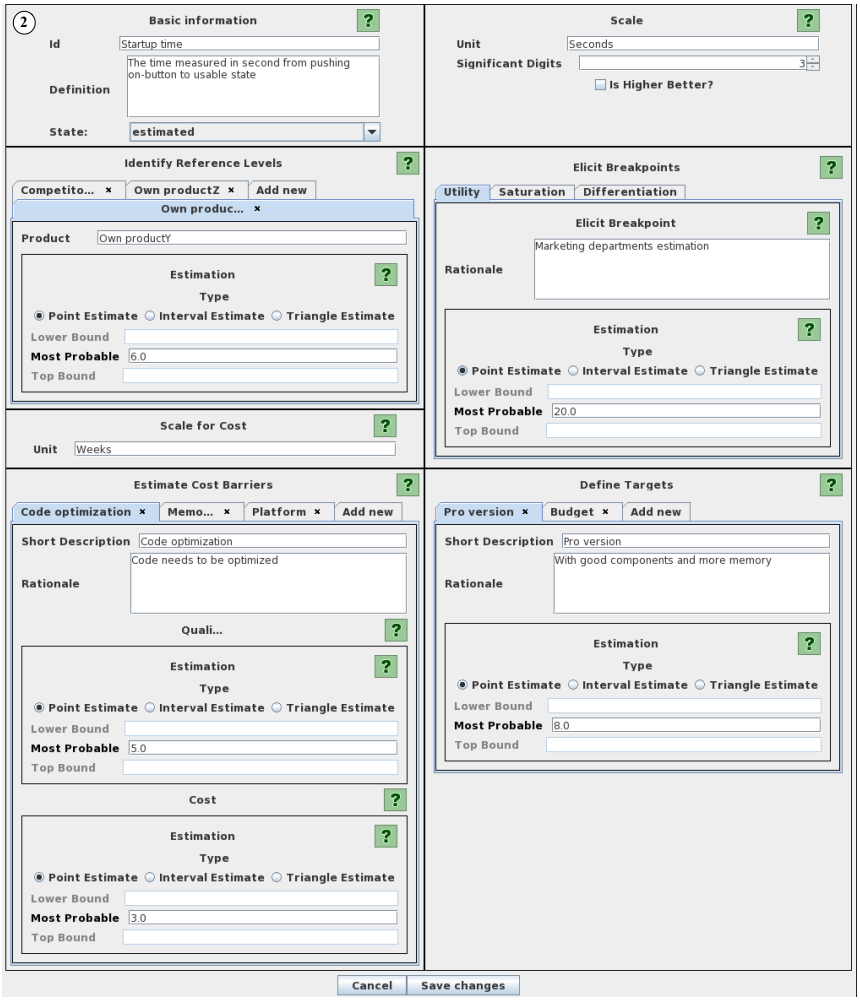


Figure 2: User interface: editing QR information

Generation of roadmap. The most important feature of the QUPER prototype tool is the automatic generation of the roadmap view. The roadmap view combines the estimates of breakpoints, barriers, reference levels, and targets for a visual representation on the same scale, which is illustrated in Figure 3, see number 3.

The breakpoints and barriers are separated from reference levels and targets with different symbols, as defined in the QUPER model [21]. Each marker has the estimated quality level displayed below. Above the marker, a description is shown. In addition, the roadmap view shows the name of the QR as a title and the unit for

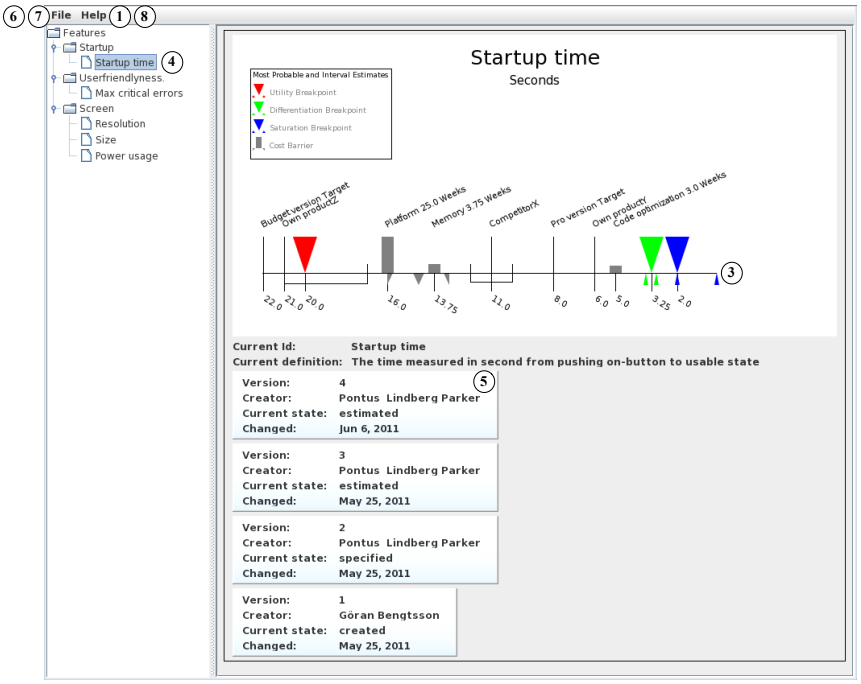


Figure 3: QUPER prototype tool user interface and features

the scale as a subtitle. On the top left corner, a legend depicting the symbols for the breakpoints and barriers is shown.

The roadmap views can be viewed in several places in the tool, when viewing the version history of a QR (see Figure 3), when editing a previously created QR, and after the last step in the detailed guide. At any time the roadmap view is displayed. Clicking on the roadmap gives the user the option to save the roadmap in a Portable Network Graphics (png) format. This feature is important for managers as the roadmap may be used in presentations when discussing what level of quality a feature should have in the coming releases.

Data management. The prototype tool offers a set of options for managing features and QR, which are accessed through the hierarchical tree on the left side in the GUI. First, there is a root node called ”Features”, which is illustrated in Figure 3, see number 4. The root node provides one action by right clicking on ”Features”, which is the option of creating new features. The next level in the tree structure contains the added features in the QUPER prototype tool. These features information can be viewed in the activity window by selecting them. The user has the option of right clicking on a certain feature to create and add a QR for the selected feature. Finally, by selecting a QR, the version history of the QR and the

latest roadmap view will be displayed in the activity window.

Version control. When selecting a QR in the tree structure, basic information about all existing versions of the particular QR is displayed in the activity window together with the roadmap view (if data for generation of roadmap view exists), which is illustrated in Figure 3, see number 5. A new version of a QR is saved in the database when the user click on the "save changes" button in the form (see Figure 3). When the user clicks on a particular version from the list, the form will be opened and displays all the information about the QR.

Data serialization. The prototype tool has the functionality to save and load the information in the internal database to a text file using an XML-like format, which can be used to save the data between different session and for sharing with practitioners, which is shown in Figure 3, see number 6. The XML like files may also be used to separate data, e.g., for different products or market segments. The QUPER prototype tool automatically saves the current database, and provides a backup feature that saves the current database to a new unique file with a single click.

Spreadsheet interoperability. The prototype tool is able to import features and QR from spreadsheet files to provide the practitioners with flexibility. To be able to import a spreadsheet file, the tool requires a certain format of the spreadsheet file. The file must consist of two sheets, one in which features are entered and one for QR. In the File menu (Figure 3, see number 7), the tool has a feature to generate a spreadsheet file with the required format, including predefined columns. To import features and QR to the prototype tool, this is done in the File menu.

Manual. The QUPER prototype tool has a manual that is accessible from the help menu, see number 8 in Figure 3. The manual is divided into two major parts, one about the QUPER model and one part specifically about the tool. The part about the QUPER model contains the same information as the detailed guide for adding QR.

5 Industrial Evaluation

In this section, we present the design of the industrial evaluation and the results.

5.1 Research design and data collection

The evaluation of the QUPER prototype tool was carried out in cooperation between academia and industry. The evaluation was carried out using an action research approach. Action research involves the improvement of practice, the understanding of practitioners, and the situation in which the practice takes place [22]. In this evaluation, we are involved in improving the use of the manual approach of the QUPER model at the case organization. In addition, we aim to improve the understanding of how practitioners may use QUPER better and faster in its envi-

ronment where the practice takes place. The general objectives of the research are to evaluate:

- the QUPER prototype tool in an industrial setting,
- what value the QUPER prototype tool may bring to release planning, and
- how the QUPER prototype tool may improve the technology transfer from academia to industry.

Five practitioners, three product managers, one project manager, and one test manager participated in the tool evaluation. The evaluation consists of the following four steps:

Planning: The first step 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, one pilot interview was carried out to adapt and improve the instrument. A summary of the used interview instrument is presented in Table 3. The sampling strategy used was a combination of maximum variation sampling [16] and convenience sampling [16].

Using the tool in real projects: During the application of the QUPER prototype tool on real requirements, the first author provided the practitioners with a set of instructions of how to use the tool. The main goal is to achieve an understanding of how to use the QUPER prototype tool on real requirements. The evaluations of the prototype tool were conducted between one and two weeks after the tool and instructions were introduced to four of the practitioners at the case company. The reason for the time delay was to provide the practitioners with time to use the tool on real requirements. One of the practitioners did not have time to evaluate the tool for a few weeks. Instead, the first and second authors performed a live demonstration of the QUPER prototype tool, followed by an interview.

Data collection: The study uses a semi-structured interview strategy [22]. In three interviews, one interviewee and one interviewer attended, while in two interviews one interviewee and two interviewers were present. During the interviews, the purpose of the study was presented to the interviewee. Then, questions about the different areas of interests in relation to the QUPER prototype tool were discussed in detail. For all interviews, varying in length from 30 to 50 minutes, we took records in the form of written extensive notes in order to facilitate and improve the analysis process.

Analysis: The content analysis [22] involved creating categories where interesting parts from the extensive notes were added and discussed. The first author examined the categories from different perspectives and search for explicitly stated or concealed pros and cons in relation to the QUPER prototype tool to support release planning of QR. The results from the analysis are presented in the following section.

Table 3: The interview instrument

General questions about the QUPER Prototype tool
What is your general view of using the tool?
What is helpful compared to working with QUPER manually?
Questions about the tool
Does the guide provide enough information to understand how to use QUPER?
What is your opinion about the forms of modeling QR in tool?
What is your view of the generated roadmap view?
How could the workflow be improved?
Would the tool be useful in industry?
Final questions
What are your biggest concerns about the tool?
Did you find any bugs?
Is there anything else you would like to add that we have not mentioned?

5.2 Results

In general, all subjects agreed that the QUPER prototype tool would help in the important shift of focus from FR to QR by providing a clear and understandable representation of the market (competing products) as a basis for QR. The importance of understanding the market is in line with the results in [3], [5].

In general, the subjects believed that the detailed guide is informative and the example makes it easy to understand and to add the information about QR while learning about QUPER and its concepts. However, one subject felt that the explaining text could be perceived as irritating while another subject believed it is better to have too much information (while learning) than too little. One subject explained, it is hard to complain about having too much information, instead, you could decided not to use the detailed guide and use the form (see Table 2) for experienced users when adding a QR.

The QUPER prototype tool provides the users with three different estimation techniques (see Section 4.2). All subjects believed that point estimates is by far the most important technique and will dominate the usage in practice, particular when estimating reference levels and breakpoints. However, for estimating cost barriers, the opinion among the subjects differs. Two subjects thought point estimates is most useful, while three preferred to estimate the cost barriers by intervals. This result is not in line with [5], where the results revealed the importance to incorporate support (e.g., interval estimates) for the uncertainty of cost estimates. When setting the target for a QR for the next release, all subjects, except one (preferred point estimates), expected interval estimates to be the dominant technique in prac-

tice. Although, none of the subjects believed that the triangle estimates would be used in practice, they did not see any reason to remove the option of using it.

Although the roadmap view may support release planning of QR, the case company is rather immature when it comes to QR, stated one subject. Previous attempts with models related to QR failed due to misunderstandings among the staff of how to use the model. The QUPER prototype tool would help the practitioners using QUPER in a consistent way, which was confirmed by one subject, *"a tool [the QUPER prototype tool] would standardize the process of creating QR. It would help in avoiding inconsistent usage of QUPER's concepts"*.

Supporting release planning of QR: All subjects confirmed that the QUPER prototype tool would support and coordinate the early decision-making process, e.g. release planning, of QR. One subject argued that the tool could help in setting the right targets due to the visualization of the roadmap view, but also in the specification and quantification of QR. In addition, several subjects stated that the clear overview of their own product's and the competing products' level of quality is an important input in the decision-making process. The clear visualization of the roadmap view may be good when arguing the importance of a QR, e.g., why do we continuously have lower quality than our competitors? One subject explained, *"it is a good picture to display when arguing why a certain target has been set"*. Four subjects compared the use of spreadsheets and the roadmap view visualization in such discussion. Two interviews explained that it is difficult to understand information presented in the form of a spreadsheet, visualization is important in these situations and the prototype tool provides you with a visual representation. The importance of a rich understanding of the market as input to release planning and early decision-making is in line with the results in [3], [5].

Improvement suggestions: When asked about how to prototype tool could be improved, all subjects pointed out the importance of having the roadmap with added information at each step in the detailed guide. One subject explained, *"the purpose of the tool is to provide support in the process of setting targets, but when we get the support, we have already set the target"*. Several subjects preferred to have a figure (similar to Figure 1) in the tool to illustrate the workflow of the QUPER prototype tool, both to see where they currently are, as well as being able to navigate between the different steps by clicking in the figure.

6 Conclusions and Future Work

This paper introduces the first version of the QUPER prototype tool and a validation of the tool in a case company to evaluate the tool's usefulness in an industrial setting. The QUPER prototype tool was developed in close cooperation between academia and industry, and five industry professionals evaluated the QUPER prototype tool using real requirements.

First, the overall results indicate that the QUPER prototype tool's automatically generated roadmap provides a clear overview of a product's and competing products' level of quality, and what level of quality to aim for in the coming releases, which is an important input in the early decision-making process, e.g., in release planning. The clear visualization of the roadmap may be good when arguing the importance of a QR's needed level of quality for coming releases. Second, the QUPER prototype tool is viewed to be a help to the practitioners in release planning of QR. Third, the QUPER prototype tool can help practitioners to use the QUPER model in a standardized way to avoid inconsistent usage of QUPER's concepts, which may be the case in a manual usage of QUPER. One main improvement of the tool was identified, namely the importance of having a roadmap that illustrates newly entered information, e.g., breakpoints and reference levels, about a QR after each step in the detailed guide, instead of only generating the roadmap once the user has walked through all steps.

We cannot make broad generalizations of these results as we undertook a small study with only five practitioners at one case company. However, some of the concepts and issues behind the development and evaluation behind the QUPER prototype tool could, to some extent be general for organizations providing products to open markets competing with quality aspects.

Based on this study, the observed findings provides important feedback regarding the usability and usefulness of the QUPER prototype tool, while enabling further improvements of the tool. In addition, the QUPER prototype tool needs further evaluations in industry in different domains and by more practitioners using more than a few quality requirements to validate its feasibility and scalability.

Acknowledgment

This work was partly funded by VINNOVA (the Swedish Agency for Innovation Systems) within the MARS project and by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se). Furthermore, we would like to thank all of the participants and their companies who have helped in making the data collection possible for this research.

Bibliography

[1] M. van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Software product release planning through optimization of what-if analysis. *Information and Software Technology*, 50(1–2):101–111, 2008.

[2] 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.

[3] R. Berntsson Svensson, T. Olsson, and B. Regnell. Introducing support for release planning of quality requirements - an industrial evaluation of the quper model. In *Proceedings of the Second International Workshop on Software Product Management*, 2008.

[4] R. Berntsson Svensson, B. Regnell, and A. Aurum. Towards modeling guidelines for capturing the cost of improving software product quality in release planning. In *Second proceeding: Short papers, Doctoral Symposium and Workshops of the 11th international conference on product focused software process improvements*, pages 24–27, 2010.

[5] R. Berntsson Svensson, Y. Sprockel, B. Regnell, and S. Brinkkemper. Cost and benefit analysis of quality requirements in competetive software product management. In *Proceedings of the Fourth International Workshop on Software Product Management*, pages 40–48, 2010.

[6] 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.

[7] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[8] J. Cleland-Huang, R. Settimi, X. Zou, and P. Sole. Automated classification of non-functional requirements. *Requirements Engineering*, 12(2):103–120, 2007.

[9] 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.

[10] T. Gilb. *Competitive Engineering*. Elsevier Butterworth-Heinemann, 2005.

[11] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE Software*, 23(6):88–95, 2006.

[12] D. Greer and G. Ruhe. Software release planning: an evolutionary and iterative approach. *Information and software technology*, 46(4):243–253, 2004.

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

[14] S. Maurice, G. Ruhe, O. Saliu, and A. Ngo-The. *Value-Based Software Engineering*, chapter Decision Support for Value-Based Software Release Planning, pages 247–261. Springer, 2006.

[15] 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.

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

[17] S.L. Pfleeger. Understanding and improving technology transfer in software engineering. *Journal of Systems and Software*, 47(2):111–124, 1999.

[18] C. Pohl, G.Böckle, and F.J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

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

[20] B. Regnell and S. Brinkkemper. *Engineering and Managing Software Requirements*, chapter Market-Driven Requirements Engineering for Software Products, pages 287–308. Springer, 2005.

[21] 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 *Proceedings of the 13th working conference on requirements engineering: foundation for software quality*, pages 277–291, 2007.

[22] C. Robson. *Real World Research*. Blackwell, 2002.

[23] G. Ruhe. *Product release Planning - Methods, Tools and Applications*. CRC Press, 2010.

[24] G. Ruhe, A. Eberlein, and D. Pfahl. Trade-off analysis for requirements selection. *International journal of software engineering and knowledge engineering*, 13(4):345–366, 2003.

[25] G. Ruhe and A. Ngo-The. Hybrid intelligence in software release planning. *International Journal of Hybrid Intelligent Systems*, 1(2):99–110, 2004.

[26] N. Subramanian and L. Chung. Sa3 - a tool for supporting adaptable software architecture generation for embedded systems. *Computer Standards and Interfaces*, 25(3):283–290, 2003.

[27] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, and S.B. Saleem. A systematic review on strategic release planning models. *Information and Software Technology*, 52(3):237–248, 2010.

[28] Q. Tran and L. Chung. Nfr-assistant: Tool support for achieving quality. In *Proceedings of the IEEE Symposium on Application-Specific Systems and Software Engineering and Technology*, pages 284–289, 1999.

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