A Case Study Evaluation of the Guideline-Supported QUPER Model for Elicitation of Quality Requirements

Berntsson Svensson, Richard; Regnell, Björn

Published in:
Requirements Engineering: Foundation for Software Quality/Lecture Notes in Computer Science

DOI:
10.1007/978-3-319-16101-3_15

2015

A Case Study Evaluation of the Guideline-Supported QUPER Model for Elicitation of Quality Requirements

Richard Berntsson Svensson[1](✉) and Björn Regnell[2]

[1] Department of Computer Science and Engineering,
Chalmers | University of Gothenburg, Gothenburg, Sweden
richard@cse.gu.se
[2] Department of Computer Science, Lund University, Lund, Sweden
bjorn.regnell@cs.lth.se

Abstract. [**Context & motivation**] For market-driven software product developing organizations operating on a competitive open market, it is important to plan the product's releases so that they can reach the market as early as possible with a competitive level of quality compared to its competitors' products. Hence, quality requirements can be seen as a key competitive advantage. The QUPER model was developed with the aim to support high-level decision-making in release planning of quality requirements. [**Question/problem**] As a follow up on previous studies on QUPER, this study investigates: What are practitioners' views on the utilities of QUPER extended with guidelines including domain-specific examples? [**Principal ideas/results**] In the presented case study, a set of detailed guidelines of how to apply QUPER in practice, including how to handle cost dependencies between quality requirements, was evaluated at a case company in the mobile handset domain with 24 professionals using real quality requirements. [**Contribution**] The results point to the importance of having concrete guidelines combined with instructive examples from real practice, while it is not always obvious for a practitioner to transfer cost-dependency examples into the domains that are different from the example domain. The transferability of guidelines and examples to support methodology adoption is an interesting issue for further research.

Keywords: Software engineering · Requirements engineering · Release planning · QUPER · Quality requirements · Empirical case study

1 Introduction

In market-driven software product development, 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 [20]. Release planning, the process of deciding which features and quality level should be

S.A. Fricker and K. Schneider (Eds.): REFSQ 2015, LNCS 9013, pp. 230–246, 2015.
DOI: 10.1007/978-3-319-16101-3_15

included in which release [3], which is both a cognitively and computationally difficult problem [20], is classified as a wicked problem [9] since different kinds of uncertainty 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' [20].

An especially challenging problem for organizations developing 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? Several methods and approaches supporting strategic release planning are reported in the literature. For example, Release Planning Prototype [10] and EVOLVE [15]. These techniques use generic algorithms to resolve the release planning issue. Using generic algorithms may not be worthwhile if the input data to the process is highly uncertain.

To the best of our knowledge, only few studies have looked into strategic release planning of quality requirements (QR), despite their importance for market success [4], [16]. According to the survey by Svahberg et al. [32], only two strategic release planning methods address quality constraints: The quantitative Win-Win model [28] addresses effort and time constraints, but not the quality level of QR, while the only method to address quality and cost constraints of QR is the QUPER model [32].

This paper is based upon previous work published in [6], [8], [22], [24] where different aspects of the QUPER model were introduced. This paper adds the following contribution to our previous investigations of QUPER, (1) the detailed practical guidelines of how to apply QUPER in practice, with an illustration of a QR, (2) the added step of how to incorporate cost dependencies between QR., and (3) Two new evaluations of the complete version of the QUPER model with 11 professionals in the first evaluation and 13 professionals in the second evaluation at a case company to evaluate QUPER's applicability using the detailed guidelines with real QR.

The remainder of this paper is organized as follows. Section 2 offers an overview of related work, while background and motivation are 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 in Section 6. Limitations of the study are discussed in Section 7, while Section 8 gives a summary of the main conclusions.

2 Related Work

There are several release planning methods in the literature, varying from informal approaches such as planning games in agile development [11] to more rigorous and formal methods as described in [15], [29], [31]. Svahnberg et al. identified 24 methods for strategic release planning, where 10 methods are extensions of others, thus 14 original methods were identified [32]. Of the 24 identified methods, 16 are related to the EVOLVE-family [15], [19], [29]. In addition to the

EVOLVE-family, other release planning methods include, e.g. software product release planning through optimization and what-if analysis [2], combining optimized value and cost with requirements interdependencies [9], and an approach using linear programming [1].

In Saliu and Ruhe [31], 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'.

In 'traditional' release planning, FR are favored, while quality aspects, such as performance and reliability, are missing in related products [27]. One approach to include quality aspects in release planning is to use EVOLVE II to generate alternatives for cost devoted to functional versus quality requirements [27]. For example, one alternative devotes 100% of the resources to functionality, while a second alternative devotes 90% to functionality and 10% to 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. It may be possible to combine QUPER with EVOLVE II by using QUPER to decide the needed level of quality and then use this as an input to EVOLVE II for resource allocation. However, such combinations are out of scope of this study and may be objects of further studies.

Software quality is not only defined by the relevant perspectives, but also by the context in which it exists [17]. 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 [33], but deciding what is good enough can only be decided in a given business context [17]. Thus, the tough question to answer is 'when is the quality level good enough'? This question is one of the motives behind the development of QUPER, and the study of its applicability in a real-world context with domain-specific examples to illustrate guidelines.

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 [25]. 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.

The need for a supporting model for handling and working with QR in this context was explicitly identified during an investigation of the cross-company requirements engineering (RE) process between two case companies [25]. Furthermore, the companies explicitly stated the importance of having a handle on QR, which has been confirmed by Berntsson Svensson et al. [5].

Two main factors motivated the creation and evolvement of QUPER: (1) a direct need identified in industry and (2) a suitable model was not found in the literature, i.e. a model for supporting release planning of QR (see Section 2).

Regarding the industry need, there was an expressed interest from the two cross-companies to improve the way of working with QR 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 research being conducted in the area of release planning in a market-driven development situation. Although there is an identified need in industry to support QR in release planning, and it is important to have a handle on QR [5], there is a lack of an appropriate model. 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.

4 QUPER Guidelines with Examples

This section describes the guideline-supported QUPER model for elicitation of QR. The main new contribution of this paper is the practical guidelines (see Steps 1-7 below) with an illustration of a QR when applying QUPER in practice. Moreover, the detailed guidelines include an added step of how to identify cost dependencies between QR (see Step 7 below). For a more detailed description of the QUPER model, see e.g. [24], [22]. An overview of the detailed guidelines are described in the following sub-sections, while a more detailed and complete description of the guidelines are available in [21].

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.

Step 1 - Identify candidate QR. When defining QR, 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 QR 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

Step 2 - Define scale and unit. For the selected 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"*.

Step 3 - Identify reference levels. For each QR, 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.

The reference levels further calibrate the estimates and provide objective measures to relate the QR to. Figure 1 illustrates added reference levels for the QR *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

Fig. 1. Illustration of added reference levels

Step 4 - Elicit quality breakpoints. When all reference levels have been identified, for each QR, ***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 Step 3 for more details).

Figure 2 shows the identified quality breakpoints for *Time shift buffer size*.

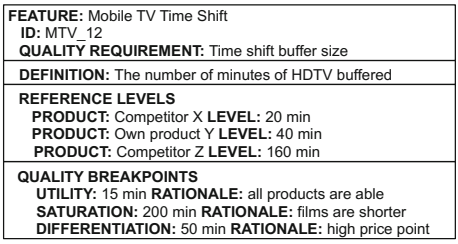

Fig. 2. An illustration of quality breakpoints have been defined

Step 5 - Estimate cost barriers. When market expectations have been identified, for each QR, estimate the cost in terms of the values of cost barriers (CB). To identify the CB, practitioners with good domain and architectural knowledge may be needed. If possible, identify similar quality requirements' CB from previous projects and use as input. Although it is possible to identify and estimate one, two, or several CB for each QR, the recommended number of CB is two. The first CB is mainly related to software changes, while a second CB 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 3, cost barriers have been identified for *Time shift buffer size*.

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 (see Step 3). One way to specify a requirements quality interval is by using both a *Good* and a *Stretch* target. The actual QR 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 4 shows the identified target, as an interval using *Good* and *Stretch*, 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

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

Fig. 3. Illustration of feature with cost barriers

Step 7 - Identify cost dependencies. If cost dependencies among QR 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

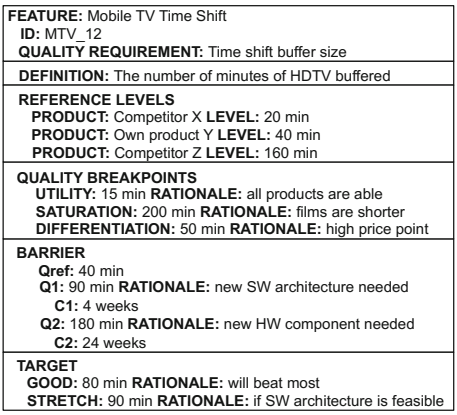

Fig. 4. Illustration of feature with targets

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 ($m \leq n$) and set a quality level target for each of these m QR that seem to provide a reasonable cost increase.

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 Case Study Evaluation

This section describes the evaluation methodology of QUPER as it was evaluated in industry with a case company. The evaluation is guided by this research question:

> RQ: What are practitioners' views on the utilities of QUPER extended with guidelines including domain-specific examples?

5.1 Case Company Description

The development of the detailed practical guidelines of how to apply QUPER in practice in this paper was developed and evaluated at a large company operating in a market-driven RE context using a product line approach. The company has about 5,000 employees and develops embedded systems for a global market. A typical project has around 60-80 newly added features, from which 700-1000 system requirements are produced. The company has a very large and complex requirements legacy database with requirements at different abstraction levels in orders of 20,000 requirements, which makes it an example of a very large-scale RE context [23]. About 25% of the system and legacy requirements are QR, i.e. either 'pure' QR [7] or a requirement that has both functional and quality aspects mixed [7]. A typical project at this company lasts for about 2 years and is implemented by 20-25 teams with about 40-80 developers per team.

5.2 Evaluation Methodology

Two evaluations of the complete QUPER model was carried out using a qualitative research approach, namely in-depth semi-structured interviews [26] and self-administrated questionnaires [13], [26]. Each of the two evaluations are described in detail below.

First evaluation *Planning/Selection:* 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 [6]. The self-administrated questionnaire used a seven-point Likert scale, representing levels of agreement from 'strongly disagree' to 'strongly agree'. When conducting research using self-administrated questionnaires, it is possible to test the internal consistency [12] (a type of reliability) and the shared variance through, e.g. a factor analysis [14]. However, these tests are dependent on how many responses obtained per item (over-determination of factors). In this case we have a very low variable/factor ratio (only 24 responses for 11 factors), therefore we believe that such an analysis is not feasible in this case[12], [18]

To test the interview instrument and the questionnaire, two pilot interviews were conducted to improve the instruments prior to the industry evaluation. The two pilot studies led to improved wording of a number of questions, one question was removed due to that it would give the same answer as another question, and two questions were completely rewritten in accordance to feedback.

The selection of practitioners for participating in the first evaluation was conducted in cooperation with two managers at the case company. The two managers identified the subjects 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. 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.

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

Q	**Do you agree that...?**
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

Applying QUPER in practice: The second step involved applying the QUPER model in practice. The practitioners received the detailed practical guidelines to follow the steps using real QR from their projects. The variation of how many QR each practitioner applied QUEPR's steps to range from a few QR up to 20 (the actual QR are not revealed for confidentiality reasons). The main goal of the second step was to achieve an understanding of the detailed practical guidelines usefulness and applicability in an industrial environment.

Data collection: The third step was carried out using semi-structured interviews [26] 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 written extensive notes in order to facilitate and improve the analysis process. 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.

Analysis: The collected data was analyzed using content analysis [26]. The content analysis involved marking and discussing interesting sections of the transcripts. The first author examined the sections individually. 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 the usefulness and applicability of the model. For the self-administrated questionnaire data given by the subjects, descriptive statistic was performed.

Second evaluation *Planning/Selection:* A second evaluation of the QUPER model was conducted with 13 new practitioners. The researchers contacted a "gate-keeper" at the case company who identified the subjects that he/she thought were the most suitable and representative of the company to participate in this study. The roles chosen are: 6 product managers, 3 project manager, 2 senior software engineers, 1 test manager, and 1 software developer. We continued with the sampling strategy developed in the first evaluation.

Applying QUPER in practice: The second step, similar to the first evaluation, involved applying QUPER in practice. We followed the same structure as in the first evaluation, i.e. the practitioners received detailed practical guidelines of how to use the model. These guidelines were used when applying QUPER to real QR from the practitioners real projects. The number of applied QR varies from 2 to 8 (the actual QR are not revealed for confidentiality reasons).

*Data collection:*The semi-structured interview approach was continued. The interviews varied between 50-65 minutes in length. Extensive written notes were taken in the same manner as in the first evaluation.

*Analysis:*Since we sought after a comprehensive view of the complete data set, the data from the first evaluation was analyzed together with the data from the second evaluation. In the final analysis we used the four categories (Sections 6.1 - 6.4) that emerged in the first evaluation. The extensive notes from the entire data set were analyzed by the first author were interesting quotations were marked with one or more of the four categories. For the analysis, all related note quotations for each category were complied and printed into a readable format. The results from the analysis are found in Section 6.

6 Lesson Learned

Below, lessons learned and the results from the self-administrated questionnaire are discussed. The results from the self-administrated questionnaire are shown in Table 3.

Table 3. Distributions of questionnaire answers

ID	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Q1	0	0	0	4	7	11	2
Q2	0	0	0	0	3	16	5
Q3	0	0	0	0	6	12	6
Q4	0	0	0	2	11	7	4
Q5	0	0	0	10	0	10	4
Q6	0	0	0	4	9	9	2
Q7	0	4	7	7	2	4	0
Q8	0	0	0	11	2	9	2
Q9	0	0	2	9	5	6	2
Q10	0	0	0	0	0	11	13
Q11	2	12	2	6	2	0	0

6.1 Ease of Use

In general, the practitioners agree that the QUPER model is easy to understand (Q1 in Table 3), 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 (Section 4) are very helpful due to easy steps to follow, and in particular the provided examples (see Figures 1-4) 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. 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 requirements? 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.

6.2 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 3). In addition, the benefit view may be helpful when specifying QR (Q6 in Table 3), 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.

In Table 3, 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: (1) 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, (2) to perform several new tests of the competitors' products level of quality, and use these values as input when estimating the breakpoints, (3) if these tests (as described above) have all ready been performed, it is easy to access a database with this information, and (4) 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 FR 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. This may explain why the practitioners viewed it slightly difficult to estimate the cost barriers (Q9 in Table 3). 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.

6.3 Applicability of the Cost Dependency Step

The cost dependency step in the QUPER model (see Section 4), 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. However, one practitioner believed that this step might be difficult to follow and apply for some practitioners.

6.4 Supporting Release Planning

In general, all practitioners agreed that QUPER improves the understanding of QR (Q3 in Table 3), and that the model would improve the decision-making

process in, e.g., release planning of QR (Q5 in Table 3). In addition, the roadmap view is seen as the central part of the improvement in the decision-making process (Q10 in Table 3).

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.

7 Limitations

Threats to validity are outlined and discussed based on the classification by Runeson et al. [30].

One threat is related to the selection process of subjects for interviews (construct validity). 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, the subjects were selected based on their role and experience of using QUPER by a "gate-keeper" at the case company. Moreover, the use of very enthusiastic or skeptical subjects could be a threat. 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. In addition, the presence of a researcher may influence the behavior of the subjects, more specifically, subjects being afraid of being evaluated. This threat was alleviated by the guarantee of anonymity as too all information divulged during the interviews, and the answers were only to be used by the researchers.

Since this study is of empirical nature, incorrect data (internal validity) is a validity threat. In case of the interviews, taking records in form of written extensive notes assured the correct data. In addition, the researchers had the chance to validate the questions and answers with the subjects lessening the chances of misunderstandings. The reliability of the study relates to whether the same outcome could be expected with another set of researchers. To increase the reliability of this study, a systematic and documented researcher process has been applied where a trace of evidence has been retained for each analysis step. The traceability back to each source of evidence is documented.

The ability to generalize the results beyond the actual study (external validity) is a threat to validity. 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, 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 the companies where QUPER has been implemented.

8 Conclusions

This paper presents the first complete version of the QUPER model, including the detailed guidelines of how to apply QUPER in practice. 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 [6], [8], [22], [24]. During these prior validations, QUPER has matured, and improvements have been made. In this paper, the complete version of QUPER was evaluated in industry at one case company with 24 industry professionals using real QR.

The results point to the importance of having concrete guidelines combined with instructive examples from real practice, while it is not always obvious for a practitioner to transfer cost-dependency examples into the domains that are different from the example domain.

Future work includes evaluations in industry in different domains where the long-term effects of using QUPER need to be investigated to fully validate its feasibility and scalability. Furthermore, to replicate this empirical study in the same domain, but in different companies to compare the usefulness and applicability

of the QUPER model is an interesting future work. Moreover, the transferability of guidelines and examples to support methodology adoption, and the use of analogy-based estimations are interesting issues for further research.

References

1. van den Akker, M., Brinkkemper, S., Diepen, G., Versendaal, J.: Determination of the next release of a software product: an approach using integer linear programming. In: Proc. of the 11th International Workshop on Requirements Engineering Foundation for Software Quality, pp. 119–124 (2005)
2. van den Akker, M., Brinkkemper, S., Diepen, G., Versendaal, J.: Software Product Release Planning through Optimization and What-if Analysis. Information and Software Technology **50**, 101–111 (2008)
3. Al-Emran, A., Pfahl, D., Ruhe, G.: Decision support for product release planning based on robustness analysis. In: Proc. of the 18th IEEE International Requirements Engineering Conference, pp. 157–166, September-October 2010
4. Barney, S., Aurum, A., Wohlin, C.: A product management challenge: Creating software product value through requirements selection. Journal of Systems Architecture **54**, 576–593 (2008)
5. Berntsson Svensson, R., Gorschek, T., Regnell, B., Torkar, R., Shahrokni, A., Feldt, R.: Quality Requirements in Industrial Practice - an extended interview study at eleven companies. IEEE Transaction on Software Engineering **38**, 935 (2012)
6. Berntsson Svensson, R., Lindberg Parker, P., Regnell, B.: A Prototype tool for QUPER to support release planning of quality equirements. In: Proc. of the 5th International Workshop on Software Product Management (IWSPM 2011), August 2011
7. Berntsson Svensson, R., Olsson, T., Regnell, B.: An investigation of how quality requirements are specified in industrial practice. Information and Software Technology **55**(7), 1224–1236 (2013)
8. Berntsson Svensson, R., Sprockel, Y., Regnell, B., Brinkkemper, S.: Cost and benefit analysis of quality requirements in competitive software product management: A case study. In: Proc. of the 4th International Workshop on Software Product Management, pp. 40–48. IEEE Compt. Soc., September 2010
9. Carlshamre, P.: Release planning in market-driven software product development: Provoking an understanding. Requirements Engineering **7**, 139–151 (2002)
10. Carlshamre, P., Regnell, B.: Requirements lifecycle management and release planning in market-driven requirements engineering. In: Proc. on the 11th International Workshop on Database and Expert Systems Applications, pp. 961–965. IEEE Comput. Soc., September 2000
11. Cockburn, A.: Agile Software Development. Addison-Wesley (2002)
12. Cronbach, L.: Coefficient alpha and the internal structure of tests. Psychometrika **16**(3), 297–334 (1951)
13. Fink, A.: The survey handbook. Sage Publications (2003)
14. Fabrigar, L.R., Wegener, D.T.: Exploratory Factor Analysis. OUP, USA (2012)
15. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Information and Software Technology **46**, 243–253 (2004)
16. Jacobs, S.: Introducing measurable quality requirements: A case study. In: Proc. of the 4th IEEE International Symp. on Requirements Engineering (ISRE 1999), pp. 172–179. IEEE CS Press, June 1999

17. Kitchenham, B., Pfleeger, S.: Software quality: The elusive target. IEEE Software **13**, 12–21 (1996)
18. MacCallum, R.C., Widaman, K.F., Zhang, S., Hong, S.: Sample size in factor analysis. Psychological Method. **4**, 84–99 (1999)
19. Maurice, S., Ruhe, G., Saliu, O., Ngo-The, A.: Decision support for value-based software release planning. In: Biffl, S., Aurum, A., Boehm, B., Erdogan, H., Grunbacher, P. (eds.) Value-Based Software Engineering, pp. 247–261. Springer, Berlin (2006)
20. Ngo-The, A., Ruhe, G.: A systematic approach for solving the wicked problem of software release planning. Soft Computing - A Fusion of Foundations, Methodologies and Applications 12, 95–108 (2008)
21. QUPER model. http://www.quper.org
22. Regnell, B., Berntsson Svensson, R., Olsson, T.: Supporting Roadmapping of Quality Requirements. IEEE Software **25**, 42–47 (2008)
23. Regnell, B., Svensson, R.B., Wnuk, K.: Can we beat the complexity of very large-scale requirements engineering? In: Rolland, C. (ed.) REFSQ 2008. LNCS, vol. 5025, pp. 123–128. Springer, Heidelberg (2008)
24. Regnell, B., Höst, M., Berntsson Svensson, R.: A quality performance model for cost-benefit analysis of non-functional requirements applied to the mobile handset domain. In: Sawyer, P., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 277–291. Springer, Heidelberg (2007)
25. Regnell, B., Olsson, H.O., Mossberg, S.: Assessing requirements compliance scenarios in system platform subcontracting. In: Münch, J., Vierimaa, M. (eds.) PROFES 2006. LNCS, vol. 4034, pp. 362–376. Springer, Heidelberg (2006)
26. Robson, C.: Real World Research. Blackwell, Oxford (2002)
27. Ruhe, G.: Product release planning - Methods, Tools and Applications. CRC Press (2010)
28. Ruhe, G., Greer, D.: Quantitative studies in software re-lease planning under risk and resource constraints. In: Proceedings of the International Symposium on Empirical Software Engineering (ISESE), pp. 262–271. IEEE, Los Alamitos (2003)
29. Ruhe, G., Ngo-The, A.: Hybrid Intelligence in Software Release Planning. International Journal of Hybrid Intelligent Systems **1**, 99–110 (2004)
30. Runeson, P., Höst, M., Rainer, A., Regnell, L: Case study research in software engineering - guidelines and examples. Wiley (2012)
31. Saliu, O., Ruhe, G.: Supporting software release planning decisions for evolving systems. In: Proc. of the 29th Annual IEEE/NASA Software Engineering Workshop, pp. 14–26 (2005)
32. Svahnberg, M., Gorschek, T., Feldt, R., Torkar, R., Saleem, S.B.: A systematic review on strategic release planning models. Information and Software Technology **52**, 237–248 (2010)
33. Youdon, E.: When good enough is best. IEEE Software **12**, 79–81 (1995)