Software Product Quality: Ensuring a Common Goal

Sebastian Barney, Claes Wohlin

School of Engineering
Blekinge Institute of Technology
sebastian.barney@bth.se, claes.wohlin@bth.se

Abstract: Software qualities are in many cases tacit and hard to measure. Thus, there is a potential risk that they get lower priority than deadlines, cost and functionality. This paper presents a method to evaluate the priority of software qualities in an industrial context. The method is applied in a case study, where the ISO 9126 model for software quality is combined with Theory-W to create a process for evaluating the alignment between success-critical stakeholder groups in the area of software product quality. The results of an exploratory case study using this tool is then presented and discussed. It is shown that the method provides valuable information about software qualities.

1. Introduction

Software quality forms an important part of a product offering. But what qualities are valuable is a very context dependant problem, changing with the product and perspective you bring. Maximising the value of a product's quality involves reconciling any conflicts between the key stakeholder groups. However, there is always a risk that qualities get a lower priority than delivery date, cost and functionality. This risk comes from the fact the qualities are most difficult to measure in relation to delivery date, cost and functionality. The balance between delivery time, cost, scope and quality is discussed as part of XP [3].

This obvious risk forms the starting point of the research presented in this paper. The main objective is to understand priorities of software qualities in an industrial context. The paper makes two main contributions. First of all, a method for analysing qualities in an industrial organisation is presented. Second, the method is applied and the paper presents an industrial study exploring the alignment priority given to various software qualities between the groups involved in the software development process.

Asking what defines an adequate level of quality in a software system is a highly context dependent question [14]. Software quality affects more than just the user of the software and each group involved with a software product brings its own perspective on quality [14]. Customers, developers, product managers, project managers and testers can all value the same qualities of the same product in different ways. Looking at the software supporting a social networking site, one could reasonably expect that customers would value usability higher than the other groups, developers value maintainability due to the dynamic nature of the product and

management values efficiency due to the scale and resources required of the application. But ultimately these value stances need to be reconciled.

The qualities each group values will also change depending on the product. Functionality is becoming increasingly important for mobile phones, reliability is more important in financial and medical domains, and portability for web-based applications.

Understanding both (a) the groups impacted by a software product and (b) the value provided to each group by the various software product qualities is useful information for companies developing software. As any development project will have time, resource and financial constraints, this information will allow the development effort to be focused in the most critical areas.

The remainder of the paper is outlined as follows. Section 2 presents some background in terms of related work, objectives and the context of the case study. In Section 3, the method for studying qualities with respect to priorities and management in an industrial context is presented. The results of the case study are presented in Section 4. In Section 5, the applicability of the method and the findings from the case study are discussed. Finally, Section 6 presents the conclusions.

2. Background

There are four variables that need to be controlled in the software development process according to XP – cost, time, quality and scope [3]. Further there is an axiom that states that external forces can set at most three of these variables with the remainder being set by the development team.

Time, cost and scope can all operate within acceptable ranges, but quality is a terrible control variable as it only allows very short term gains at a very high cost to all parties involved [3]. That said, quality does not need to be perfect [17] but development is much simpler when the success-critical stakeholders agree on what action should be taken [7].

2.1. What is Quality

The definitions of quality are both many and conflicting, even when only examining the topic in relation to software engineering. Looking across different disciplines it is possible to see a complex multifaceted concept of quality that can be described from five different perspectives [10]:

- The *transcendental perspective* defines quality as something that can be recognized but not defined in advance.
- The *user perspective* defines quality as fit for purpose.
- The *manufacturing perspective* defines quality as conformance to specification.
- The *product view* defines quality in terms of essential characteristics of the product in question.
- The *value-based view* defines quality in terms of the amount a customer is willing to pay for it.

By far the most common perspectives taken in the software development industry are that of the user and manufacturer. [12,14]. However, there is an increasing body of literature that recognises the importance of taking advantage of all of the perspectives involved in software development. Theory-W states that success requires all of the success-critical stakeholders to compromise [7], while requirement specification reading techniques that take advantage different perspectives have been found to catch 35% more defects than non-directed alternatives [2,5], and value-based software engineering now recognises the value brought by different perspectives into the development process [14].

Software quality is not only defined by the relevant perspectives, but also by the context in which it exists [14]. 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 can in fact be ideal [17], but deciding how much less than perfect can only be decided in a given business context [14].

2.2. Quality Models for Software Development

Numerous models have been developed to support software quality. Examples of these models include McCall quality model, Boehm's quality model, Dromey's quality model and ISO 9126.

McCall's quality model is the first of the modern software product quality models [14]. The model uses a hierarchy of factors, criteria and metrics to address internal and external product quality. Eleven factors define an external or user perspective of quality. Each of these factors is linked to between two and five of 23 criteria that define an internal or development perspective of quality. Further metrics are associated the factors allowing quality to be measured and managed.

McCall's quality model was followed by Boehm's quality model [14]. Like McCall's model, Boehm's model presents product quality in a hierarchy with three high level characteristics linked to seven intermediate factors, which are in turn linked to 15 primitive characteristics. Boehm's model has a wider scope than that of McCall's, with more emphasis on the cost-effectiveness of maintenance [16].

More recently work has been done to create an international standard for software product quality measurement – ISO 9126 [13]. This standard is again organised in a hierarchy with six characteristics at the top level and 20 sub-characteristics with indicators used to measure the sub-characteristics. In addition to aspects of internal and external quality, covered by McCall and Boehm's models, ISO 9126 includes quality characteristics of functionality [16]. Internal, external and functional qualities are also mixed at all levels of the hierarchy. However, ISO 9126 does not clearly state how quality should be measured [14].

None of these three models present a rationale for the selection of characteristics to be included in the quality model and it is not possible to tell if a model presents a complete or consistent definition of quality [14]. Further the placement of items appears arbitrary in ISO 9126, with no justification as to why Interoperability is not related to Portability.

Dromey presents a different type of model that attempts to address some of the issues presented and support developers build product quality [9]. Dromey believes

that it is impossible to build high-level quality attributes like reliability or maintainability into a product, but developers must instead build properties that manifest in achieving these goals. The distinction this model makes is important, as using it will verify that it allows the quality required to be achieved [14]. Before Dromey's model can be successfully applied, the various groups involved in the development of a software product must agree on what quality attributes should be achieved and to what level. This process can be supported using other models.

2.3 Merging Perspectives on Software Quality

Software product quality can easily become an area of problems and conflict, as each stakeholder group has its own perspective on what is important. A number of methods can be applied to help reconcile this situation and select the best way forward. These methods include expert judgement, the NFR Framework, Quality Functional Deployment and Theory-W.

Expert judgement involves one or more experienced professionals using their experiences and knowledge to make a decision on an issue. The decisions are not necessarily supported by modelling or numerical assessment.

The NFR Framework uses diagrams to relate non-functional requirement goals with different decisions that can be made in the design and operation of a system that affect it positively or negatively, allowing trade-offs to be identified and made [8]. While this method makes the results of a choice to be made more explicit, it requires a set of common priorities to be identified to allow effective decisions to be made.

Quality function deployment (QFD) considers the priority of customer and technical requirements in achieving the goals of the system to help prioritize the requirements [11]. However, the other perspectives involved in the development of the software product are not considered.

Value-based software engineering (VBSE) recognises the problems created by conflicting perspectives in the software development process [6]. Central to resolving conflict in VBSE is Theory-W, which requires [7]:

1. Success-critical stakeholder groups to be identified;
2. The requirements of these groups to be elicited;
3. Negotiation between the groups to create a win-win situation; and
4. A control process to support success-critical stakeholder win-win realisation and adaption to a changing environment.

The key advantage of Theory-W is that it explicitly brings all of the parties on whom success lies together to understand each other's needs, compromise and agree. But in order to be successful Theory-W must be managed to ensure the plans are achieved and any deviations from the plans are corrected [7]. Management requires an understanding of why the goals are being pursued, what is the required result, who is responsible for the result, how the result will be achieved and at what cost the result can be achieved. The answer to these questions will be peculiar to the context in which they are answered.

2.4. Research Objectives

The objective of the research presented in this paper is to create and validate a method capable of determining the level of alignment between the internal success-critical stakeholder groups. The method should be able to identify the degree to which the groups are aligned in how they perceive operations today with respect to quality.

This method should be validated in an industrial case, answering the following research questions:

RQ1: Is the method proposed in this paper capable of identifying the degree to which the internal success-critical stakeholder groups are aligned in how they perceive the priorities on software product quality today?

RQ2: Is the method proposed in this paper capable of identifying what the different internal success-critical stakeholder groups perceive as the ideal set of priorities on software product qualities in the situation today? And to what degree are the groups aligned?

3. Methodology

To address the research questions, a method, using Theory-W as a starting point, is developed. By exploiting the early phases of Theory-W it is possible to determine the level of alignment between the internal success-critical stakeholder groups. This involves identifying the internal success-critical stakeholder groups and eliciting their value propositions with respect to quality.

The results should support the continued application of Theory-W, to negotiate between the success-critical stakeholders to achieve a better situation and realize this goal through clearer management.

3.1. Quality Model

The literature on software product quality recognizes that quality depends both on the perspective of the observer and the actual software product in question. As such, using any model as it appears in the literature risks not adequately defining quality in the context being studied. To use one of the quality models briefly introduced in Section 2 is a good starting, but company specific needs have to be taken into account, as illustrated in the case study in Section 4.

3.2. Questionnaire

This method proposes the cumulative voting (CV) [15] technique to elicit (a) how important each quality is today, and then repeated the exercise to show (b) how important they perceived each quality should be today – answering RQ1. CV asks participants to spend 1000 points across all of the qualities previously identified, to represent their relative influence. For example, if a participant thought testability does

not at all matter today and security was twice as important as scalability they might award these qualities zero, 200 and 100 respectively.

3.3. Analysis

CV allows participants' responses to be grouped logically for analysis – for this method into the success-critical stakeholder groups. The results of each participant in the group can be averaged for each quality, ultimately producing a list that shows each quality and the averaged notion of its importance.

From here it is possible to rank the qualities from most to least influential for each success-critical stakeholder group. The degree to which the groups are aligned can then be calculated pairwise using a Spearman rank correlation matrix.

4. Case Study

The case study was conducted during Autumn 2007 for two products at Ericsson. Ericsson is a world leading company in telecommunication, providing a wide range of products and solutions. Products are developed and sold as generic solutions offered to an open market, although customized versions of the products are also developed for key customers.

4.1 Success-Critical Stakeholder Groups

High-level R&D management supported the authors to identify internal success-critical stakeholder groups for this case study. Participants in the case study represent:

- *Strategic Product Management (SPM):* This group has the strategic product responsibility and decides the overall product development direction.
- *Project Management (PM):* This group is responsible for planning and executing projects aligned with the priorities of the strategic product management.
- *Tactical Product Management (TPM):* This group supports the strategic product management with expert knowledge of the systems and their architecture. It is also responsible for providing analysis of pre-project requirements in the form of feasibility, impact and technical dependencies.
- *Development and Testing (R&D):* These groups are responsible for the implementation, verification and validation of requirements.

The high-level management further recommended that the results of SPM and PM be combined when determining the priorities given to the software product qualities, the first research question. These groups work closely together; with SPM prioritising the development activities that PM is responsible for planning.

A description of this case study was sent out to the managers of the identified success-critical stakeholder groups requesting volunteers from their teams to take part in the case study.

In total 44 potential participants were identified to take part in this case study, with 31 usable results being obtained. A breakdown of the participants can be seen in Table 1. Two of the participants identified felt they were not appropriate and identified other people in their team to replace themselves, two people declined to participate, one questionnaire result was lost in an Excel crash and nine people could not find time to complete the questionnaire.

Table 1 Study response rate

Group	Candidates	Replacements	Complete Responses
Strategic Product Management	15	1	6
Project Management	6		4
Tactical Product Management	9	0	9
Development and Testing	14	1	12
Total	*44*	*2*	*31*

The low response rate for Strategic Product Managers was anticipated, so extra participants for this role were selected to ensure a sufficient number of responses.

The questionnaire was conducted as a one-on-one structured interview, which each participant taking between 30 and 75 minutes. The interviews were conducted over two-month period.

4.2. Software Product Qualities

The process of defining a model of software product qualities was also a collaborative exercise involving the academic and industrial perspectives. The list of qualities was defined specifically for the products studied at Ericsson, maximizing the relevance for this industrial partner and possibilities for using the results to support improvements within the company.

As Ericsson uses an agile development methodology, the quality model is founded on an axiom from Extreme Programming which states there are four variables to control in software development: *time*, *cost*, *quality* and *scope* [3].

The axiom from Extreme Programming was complemented with ISO 9126, the international standard for software product quality, providing more detail on the components of quality and scope. The authors then wrote preliminary definitions for these terms.

A workshop was held within Ericsson to review and refine the terms defining software product quality. The aim was to ensure the final list of terms and definitions would be complete, meaningful and useful to Ericsson. The model was split into three categories – the ISO 9126 qualities relating to *functionality*, the ISO 9126 qualities relating to *system properties* and *project management* to cover *time* and *cost*. Moreover, *security* was moved from *functionality* to *system properties*. Two new qualities were identified and added to *system properties*; these are *scalability* and *performance management/statistics*. Finally, five of the quality terms were complemented with alternative names used in Ericsson.

The terms and definitions used in the case student presented in this paper are available online [1].

4.3 Pilot Study

A questionnaire was developed using the methodology described in Section 3 and piloted. The participants in the pilot had trouble making comparisons between qualities related to *features*, *system properties* and aspects of *project management*. An example of such a comparison could include accuracy of feature, resource behaviour of the system and development cost. In order to address this issue the authors modified the questionnaire to use a hierarchical cumulative voting (HCV) method as described by Berander and Jönsson [4]. This effectively split the questionnaire up into four independent CV exercises; one parent list that included the three category terms – *features*, *system properties* and *project management* – and one list for each of the categories, each containing the relevant qualities. A pilot of the new questionnaire found the participants' capacity to respond much improved.

In order to conduct the analysis each participant's response needs to be changed from HCV to CV, converting the four cumulating voting lists into one that covers all of the qualities.

Remember that one of the four lists includes the categories *features*, *system properties* and *project management*, while the remaining lists each detail the qualities that make up one of the categories. This allows the number of points awarded to each quality to be multiplied with the category from which it came [4]. It is also necessary to multiple each of these results by the number of qualities from the same category as the resultant value. Finally the set of number for each quality can be scaled so that the sum is 1000.

For example, if *200* points are awarded to *project management* and *600* points are awarded to *time*, then *category* × *quality* × *number of qualities* = *200*×*600*×*2* = *240,000*. The scaling of this result then depends on the other values, but if the other values were to sum to *4,800,000* then the number would be scaled to *result* ÷ *total sum* × *1000* = *240,000* ÷ *4,800,000* × *1000* = *50*.

It is necessary to multiply each value by the number of qualities in the same category to ensure that qualities with many categories are not under represented and that categories with few qualities are not over represented.

The individual responses can now be grouped and averaged, allowing ranks to be determined and the Spearman rank correlation can then be calculated.

The final version of the questionnaire is available online [1].

4.4. Software product quality priorities

The first objective of this case study is to determine the degree to which the key stakeholders are aligned regarding how they see software product quality today. The results show the groups are very aligned, with Spearman's rank correlation values between 0.80 and 0.90 indicating each group ranked the qualities in a very similar order. The full results are presented in Table 2.

Similarly the key stakeholder groups are aligned how they ranked the software qualities should be today. The results in Table 3 show correlation values between 0.65 and 0.74 between the groups.

Table 2 Correlation matrix showing the degree to which the groups are aligned in how they perceive the priorities today

	SPM & PM	**TPM**	**R&D**
SPM & PM	1.00	0.80	0.90
TPM		1.00	0.86
R&D			1.00

Table 3 Correlation matrix showing the degree to which the groups are aligned in how they perceive the priorities should be today (ideal)

	SPM & PM	**TPM**	**R&D**
SPM & PM	1.00	0.74	0.71
TPM		1.00	0.65
R&D			1.00

The similarities between the perceived situation today and the perceived ideal situation were striking. Looking at all responses the correlation between the two situations is 0.82. However, each group individually saw the need for more changes. The results in Table 4 show, for example, a correlation of 0.62 comparing what R&D perceived as the priorities today against what they thought the priorities should ideally be today.

Table 4 Correlation between perceived situation today and perceived ideal situation today

Group	**Correlation**
All groups	0.82
SPM & PM	0.72
TPM	0.77
R&D	0.62

While there was variation between participants of the same perspective, there was no individual that stood out as being consistently different in their results to the other members of their group.

Looking at the underlying data it is possible to further understand the differences and similarities between the groups. The remainder of this section highlights key aspects of similarity and difference.

The qualities studied have been grouped into three categories – *features*, *project management* and *system properties*. All of the groups today ranked these groupings in the same order, with *features* as the most important category, followed by *project management* and finally *system properties*. Interestingly all groups would like to see *system properties* overtake *project management* in their perception of the ideal situation. This helps explain the high correlation values attained.

However, looking at the individual qualities it is possible to explain why the collective results from all participants shows less need for change than the result of any of the groups individually. While confidentiality does not allow the ranked qualities to be published in this paper, some groups are more affected by some of the qualities than other groups; so they perceive these qualities as more important, while

the other groups perceive the same quality as less important. For example, *Time Behaviour* is ranked seventh today, with all groups placing it in the seventh or eighth position today. In the perceived ideal situation the overall rank is only increased one place to sixth, but the same criterion is ranked second most important by R&D, sixth most important by TPM and tenth most important by SPM & PM. This situation acts to reduce the correlation coefficients for the individual groups, but still keeps it high when examining all results together.

There was a high level of agreement in the ranks given to qualities relating to features and project management both today and in the ideal situation. The area of greatest contention between the groups concerns qualities relating to *system properties*. SPM differs the most when examining the results of the three groups. The results highlight which qualities the groups agree on as important – such as *Scaleability*, *Time Behaviour*, *Robustness/Stability*, *Configurability/Product Customisability/Adaptability* and *Resource Behaviour* – and which qualities for which there are differing priorities – *Recoverability*, *Operability*, *Performance Management/Statistics*, *Upgradeability/Replaceability*, *Analysability*, *Testability*, *Containment/ ISP/Fault Tolerance* and *Security*.

5. Discussion

The methodology proposed in Section 3 has been applied to identify the level of alignment between internal success-critical stakeholders with the modification made after the pilot study described in Section 4.3. While the results from the case presented in this paper are not generalisable, it highlights what situations can be detected by the method and acts as a reference point for future applications of the method.

While in the case study some changes to the priorities given to the software product qualities would be perceived beneficial by each of the internal success-critical stakeholder groups, the extent of the changes required is reduced when considering all perspectives together. This can be seen most clearly with a number of qualities where the groups agree on their importance today, but some of the groups think some should be more important while other groups think they should be less important and it ends up in almost the same place. This result shows Theory-W in action, with the organisation having to balance conflicting stakeholder perspectives in order to achieve the optimal balance.

While the current processes seem to have done a reasonable job to balance the various concerns of software product quality, this is not explicitly visible to all of the stakeholders who felt that their needs were not being adequately addressed. One of the ongoing aims within Ericsson is to use these results to foster a greater understanding and dialogue between the internal success-critical stakeholders in terms of each other's needs.

6. Conclusion

This paper presents a methodology and results of a case study for examining the alignment between the internal success-critical stakeholder groups in software product quality. The results obtained by the method were interesting, valuable and very positive from the perspective of the industrial partner, Ericsson, with:

- The groups found to be aligned in perceived the priorities placed on different software product quality today; and
- Overall the participants in the case study perceive few changes necessary to improve the current situation.

The case study results highlighted that different stakeholder groups have different priorities, and companies must be able to balance these differing opinions in order to achieve an optimal outcome. Key to achieving this outcome appears to be open and transparent dialogue and cross group communication and understanding.

The case study presented in this paper may not be representative of the software development industry, only involving two products from one company. Still, it provides some insights into how qualities are handled in an industrial context. Furthermore, the method can be applied in other situations to support the alignment of success-critical stakeholders in issues of software product quality. In turn, these additional results can help determine which of the results, if any, can be generalised.

This research will be used in three ways:

- This work is the first in a series of studies examining, with the intention to help improve, the alignment of company strategy, product strategy, product management and development efforts.
- This work is also the first in another series that is looking at different investment options and trade-offs in software development – like features, quality and staff training. Going forward the aim of this work is to support organisations improve the investment choices they make.
- The authors are also hoping to replicate parts of this study at different organisations to help achieve greater alignment in issues of software product quality and draw more general conclusions in this topic area.

Acknowledgments

We would like to thank Ericsson for their active involvement in and support of this research.

This work was partly funded by The Knowledge Foundation in Sweden under a research grant for the project Blekinge Engineering Software Qualities (BESQ) (http://www.bth. se/besq).

References

1. S. Barney, C. Wohlin 2008, ‘Software Product Quality Questionnaire’, [Online], Available: http://www.bth.se/tek/aps/sba
2. V. R. Basili 1997, ‘Evolving and packaging reading technologies’, *Journal of Systems and Software*, vol. 38, no. 1, pp. 3-12
3. K. Beck 2000, *Extreme Programming Explained: Embrace Change*, Addison Wesley Longman, pp. 15-19
4. P. Berander, P. Jönsson 2006, ‘Hierarchical Cumulative Voting (HCV) – Prioritization of Requirements in Hierarchies’, *International Journal of Software Engineering and Knowledge Engineering* (IJSEKE – Special Issue on Requirements Engineering Decision Support), vol. 16, no. 6, World Scientific Publishing Company, Singapore, pp. 819-849
5. B. Boehm, V. R. Basili 2001, ‘Software Defect Reduction Top 10 List’, *IEEE Computer*, vol. 34, no. 1, pp. 135-137
6. B. Boehm, A. Jain 2005, ‘An Initial Theory of Value-Based Software Engineering’, in *Value-Based Software Engineering*, S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, P. Grünbacher, Eds., Springer
7. B. W. Boehm, R. Ross 1989, 'Theory-W software project management principles and examples', *IEEE Transactions on Software Engineering*, vol. 15, no. 7, pp. 902-916
8. L. Chung, B. A. Nixon, E. Yu, J. Mylopoulos 2000, *Non-functional requiremnts in software engineering*, Boston, Kluwer Academic
9. G. Dromey 1996, ‘Concerning the Chimera’, *IEEE Software*, vol. 13, no. 1, pp. 33-43
10. D. Garvin 1984, ‘What does “Product Quality” really mean?’, *Sloan Management Review*, Fall 1984, pp. 25-45
11. G. Herzwurm, S. Scockert, W. Pietsche 2003, ‘QFD for customer-focused requirements engineering’, in Proceedings of *the 11th International Requirements Engineering Conference*, pp. 330-338
12. R. W. Hoyer, B. Y. Brooke 2001, “What is quality?”, *Quality Progress*, vol. 34, no. 7, pp. 53-62
13. ISO9126 Information Technology – Software Product Evaluation - Quality Characteristics and Guidelines for Their Use, *International Organisation for Standards*, Geneva, 2002
14. B. Kitchenham, S Lawrence Pfleeger 1996, ‘Software Quality: The Elusive Target’, *IEEE Software*, vol. 13, no. 1, pp. 12-21
15. D. Leffingwell, D. Widrig 2000, *Managing Software Requirements – A Unified Approach*, Addison Wesley, Upper Saddle River, NJ
16. D. Milicic 2005, ‘Software Quality Models and Philosophies’, in *Software Quality Attributes and Trade-Offs*, L. Lundberg, M. Mattsson, C. Wohlin Eds., Blekinge Institute of Technology, pp. 3-19
17. E. Yourdon 1995, ‘When good enough software is best’, *IEEE Software*, vol. 12, no. 3, pp. 79-81