
Fraunhofer Institut
Experimentelles
Software Engineering

# Reusing Knowledge on Software Quality for Developing Measurement Programs

**Authors:**
Olga Jaufman
Bernd Freimut
Ioana Rus

Submitted for publication at
SEKE 2004

IESE-Report No. 009.04/E
Version 1.0
April 15, 2004

---

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach (Executive Director)
Prof. Dr. Peter Liggesmeyer (Director)
Sauerwiesen 6
67661 Kaiserslautern

# Abstract

In order develop high quality software, companies must understand what constitutes quality for their products and stakeholders and then manage and engineer quality accordingly. To support this, the measurement of software quality is essential. Various measurement approaches and quality models have been proposed for this, such as the ISO9126 standard, the models proposed by Boehm and McCall, or the GQM approach. The problem however is, that the application of these approaches implicitly assumes that the measurement personnel has experience in defining and measuring software quality, an assumption that is not true for many software companies. Therefore we propose to address this problem by supporting quality measurement with reuse of previously acquired knowledge. We gathered and organized quality measurement knowledge in a knowledge repository and designed a quality definition process that allows elicitation of quality requirements from multiple stakeholders.

**Keywords:** experience base; measurement program; quality model; quality metrics; software quality

# Table of Contents

# 1 Introduction

Customer satisfaction progressively becomes a more important goal for software development companies who want to be successful and survive competition. Therefore these companies must improve their ability to understand what the customer means by "quality" and to develop software according to these requirements.

The challenge in doing so is that quality might not have the same meaning in the context of different application domains, different systems, or even for different stakeholders of the same system. Thus, there are multiple perspectives on quality that must be integrated for a system. For example, the customers or the users of a system have an "external" perspective, oriented more on the functionality of the system, and other execution properties such as performance, usability, and reliability. The developers on the other hand have an "internal" perspective, focused more on defects in software artefacts. Thus, there is a need for a common understanding and agreement between stakeholders of a specific system on quality definition.

For being achieved, quality must be more than just measured in the final product; it must be properly defined, managed and engineered throughout development. For this purpose, quality indicators that can predict the final product quality must be defined and measured during development. This calls for the definition and selection of appropriate quality measures that require both application domain knowledge and software development knowledge. This can be a challenging task especially for people with less experience and knowledge in this area. The consequences can be firstly an incomplete characterization and measurement of the product quality and secondly a large amount of effort required to set and perform the measurement plan.

We propose to address this problem by supporting quality measurement with reuse of previously acquired knowledge. Some of the software quality knowledge is applicable across multiple systems, and is therefore transferable from one project to another.

We gathered and organized this knowledge in a knowledge repository called software quality characterization and measurement (SQCM) experience base, following the concept of experience factory (EF) [1]. The EF paradigm argues for reuse of experience collected from individual projects by analyzing, synthesizing and packaging this experience for future projects. The SQCM Experience Base captures generic knowledge and experience such as quality properties and cor-

responding measures, as well as when and how these measures should be taken and how to be used.

In this paper we describe the SQCM experience base, and the process of using it in support of measurement programs. The proposed concept was applied in a case study, the findings of which will be reported as well.

The target audience of this paper is envisioned to be project managers, software engineers, and process engineers, who are in the need for practical approaches for defining, documenting, and measuring software quality. Researchers can also benefit, by getting ideas about new and more efficient approaches for quality modeling and measurement.

The remainder of the paper is organized as follows: Section 2 presents related work, and discusses similarities and differences between other approaches and ours. Section 3 describes in more detail the objectives of our research. Section 4 presents the experience base and the method for using it. The case study is discussed in Section 5. Section 6 concludes the paper with a summary and directions for future efforts.

# 2 Related Work

According to [4] there are principally two different ways to define software quality for a given product. Using a fixed-model approach it is assumed that all important quality properties are a subset of those in a defined model. To control and measure each quality property, relationships between measures, internal and external quality characteristics are used.

Examples of such an approach are the models proposed by McCall [7], Boehm [2], and ISO9126 [5]. In these models, key characteristics of quality are identified from a user perspective. These key characteristics are normally high-level external attributes (e.g., reliability, and maintainability). These high-level attributes are decomposed into lower-level attributes, which represent a refined view or internal view on quality. Measures are also proposed for these lower-level attributes.

The disadvantage of this approach is that the model's attributes are too general as they are supposed to be fitted for all contexts.

Therefore a different approach emerged, namely a define-your-own-model approach, as it was apparent that different contexts have different notions of quality. In these approaches, instead of using a pre-defined set of quality characteristics, a consensus on relevant quality characteristics is defined for a given product in co-operation with relevant stakeholders. These characteristics are then decomposed until measurable quality attributes are obtained.

Examples of a define-our-own-model approach are measurement approaches such as the GQM-approach [3] and the SQUID-approach [6]. The GQM-approach provides a framework for measurement programs, where relevant measurement goals for significant stakeholders are identified and then refined via questions into measures, through interviews with stakeholders.

The SQUID quality modelling approach [6] has the objective to define quality requirements for a particular product in a quantifiable manner. One basic idea of this approach is that a quality model has two components: (1) a structure model that defines model elements and their interactions, and (2) a content model that identifies a set of entities (e.g. attributes) linked in accordance with that structure. The structure model identifies, for example, that quality properties are decomposed into sub-properties. Regarding the content model, [6] suggests to adopt one (or more) of the existing models and to adapt it to the organization.

The advantage of these approaches is that they can take into account the context in which quality is to be defined and can incorporate the views of the stakeholders. On the other hand, these models provide little guidance on how to define and elicit quality requirements.

Therefore, we aim at supporting these tasks by re-using experience about quality measurement.

# 3 Objectives

The purpose of our SQCM Experience Factory is to store and re-use experience related to the definition and measurement of quality, in order to support the measurement team in the process of setting up the quality definition for a product or project.

The challenge is, however, that the term "quality" is dependent on the domain, product, or even the stakeholders of a product (e.g., customer, or developer). Consequently, our experience base has to follow a define-your-own-model approach in order to be tailorable to the context in which it is to be used. The approach should allow abstract properties to be decomposed in more concrete sub-properties, specifically for each product and according to its stakeholders' views.

Yet, the implicit assumption with existing define-your-own approaches such as goal-oriented measurement is that the approach will be performed by people with measurement experience and knowledge. They should know what properties constitute product quality, how the relevant quality properties can be measured, when and by whom these properties can be measured, and finally how relevant it is to measure these properties. Unfortunately, not all people who are setting measurement programs have such measurement experience and knowledge.

We address this problem by providing an experience-based support that provides information about the properties that may be relevant to measure, how these properties can be measured, and for whom and when it may be relevant to measure them. In doing so, we are tying to build on top of previous work as much as possible.

The objective of the development of the SQCM Experience Factory is then to provide the support for initiating measurement programs more effectively and more efficiently. "More effectively" means that a measurement program covers all quality aspects that are relevant for product quality from project stakeholders' point of view. "More efficiently" means that less effort is required to set up a measurement program. The approach will also allow different stakeholders to communicate about quality needs using a common vocabulary.

# 4 The SQC-Experience Factory

## 4.1 The Experience Base

From the available define-your-own approaches, the SQUID-approach was appealing to us as its structure model already provides a sound scheme for an experience base. Therefore, we based the scheme of our experience-base on the SQUID structure model. The content of the experience base follows quality properties and their measurement models as proposed in the pre-defined quality models presented in the literature. In this section we discuss both the structure and the content of our experience base.

### 4.1.1 Structure

The structure of our EB scheme, adapted from [6], is shown in Figure 1

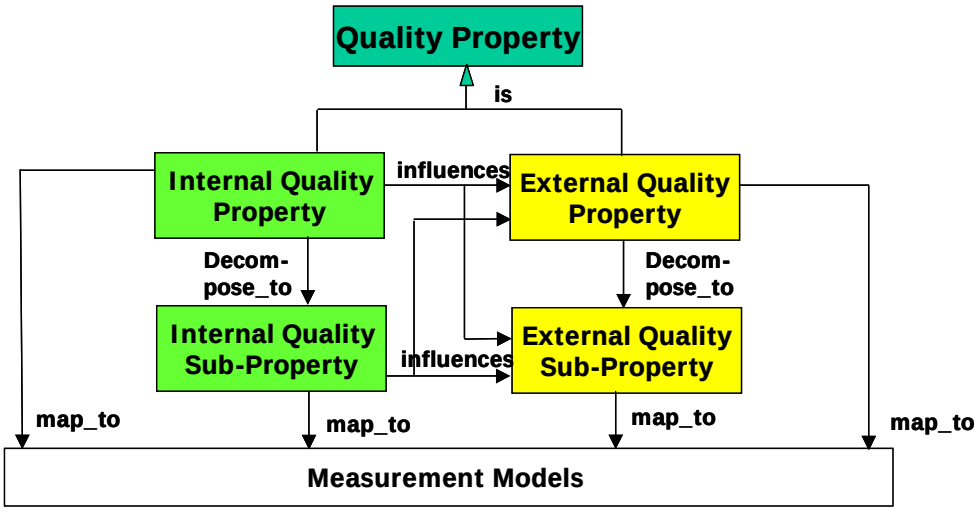


Figure 1 EB Scheme

Since we want to support both the customer-view of software as well as the developer view, we distinguish between internal and external quality properties [4]. A quality property can either be mapped directly to a measurement model that contains a descriptive model [3] according to which the quality property is to be operationally measured, or it is decomposed into more detailed sub-properties. This decomposition allows to organize quality properties hierarchically (e.g., like [7][2][5]) or flat such as within GQM.

In our pilot implementation we used a 4-level hierarchy, as we organized a large number of quality properties. However, in a different context where only a small number of quality attributes needs to be stored, a more flat implementation is also possible.
An important aspect of our objective is to measure quality during development and make inferences about final quality, since we want to evaluate the impact of new technologies on final quality as early as possible. The challenge is here to establish such relationships between internal and external quality by means of empirical (case) studies. Since these relationships tend to be dependent on a particular domain or even system, it is an important corporate asset to be captured in our experience base. Consequently, we aim at maintaining an influence relationship between internal and external quality.

In order to support and ease the selection of relevant quality properties, for each property we store the set of stakeholders and development phases, for which a property might be relevant. Figure 2 shows an entry of our experience base.

| Name of 3rd level property | Definition | Sub-property of | Applicable for object | Relevant for stakeholder | Relevant in phase |
|---|---|---|---|---|---|
| **Reliability** | The probability that the software will not cause a system failure for a specified time, under specified conditions. | Product Operability | Operational Software | User, Customer, Project Manager, Analyst, Designer, Programmer, Tester, Maintainer | Analysis, Design, Code, Test, Operation |

Figure 2 Excerpt of EB

Each quality property is characterized by its name and a comprehensive definition as it was observed [10] that definitions within existing quality models are often terse and too short.

### 4.1.2 Content

Having defined the structure of our experience base, the next task is to populate it with an initial set of quality attributes. This can be achieved in several ways. First, it is possible for an organization to survey past measurement programmes and compile the set of quality properties under study as well as the measures used.

The second approach, which we followed, is to perform a survey of quality models published in the literature. Although these properties are typically not explicitly tailored to a specific domain or company, they can serve as a reasonable start.

Due to the abundance of quality models in the literature we faced several problems: Often quality properties are defined in too short and concise a manner, so that their exact meaning is fuzzy and unclear [10]. Moreover, when comparing models from different sources we observed inconsistent terminology about quality properties in the literature: Properties with the same meaning are denoted with different names, properties having the same names can differ in their meaning. Additionally, the published quality properties are on different layers of abstraction, for example, ranging from Reliability to Mean Time Between Failures.

Therefore, we carefully compared exiting definitions for quality properties and synthesized them in a hierarchy with 4-levels, where each quality property is explicitly defined. Additionally, we classified for each property the development phase and stakeholder, for which the property might be relevant, and identified applicable descriptive measurement models from the literature. To establish a relationship between internal and external quality properties, we adopted the indicators of Reliability identified in [7].

## 4.2 Using the Experience Base

The contents of the experience base are quality properties that have been or can be defined for products in the considered environment. In order to keep the contents up to date we defined processes that evaluate and update the contents as necessary. The focus of this paper is, however, the Quality Definition Process.

The objective of this process is to select and identify appropriate quality definitions and measurements for a given system by re-using such knowledge from the experience base. One key aspect here is the ability to consider and integrate different viewpoints on quality.

The basic idea of the process is to elicit which of the quality properties captured in the experience base are relevant for each stakeholder and to identify those quality properties that are important to several stakeholders or those that are only relevant for a small number of stakeholders. Figure 3 shows an overview of the process.

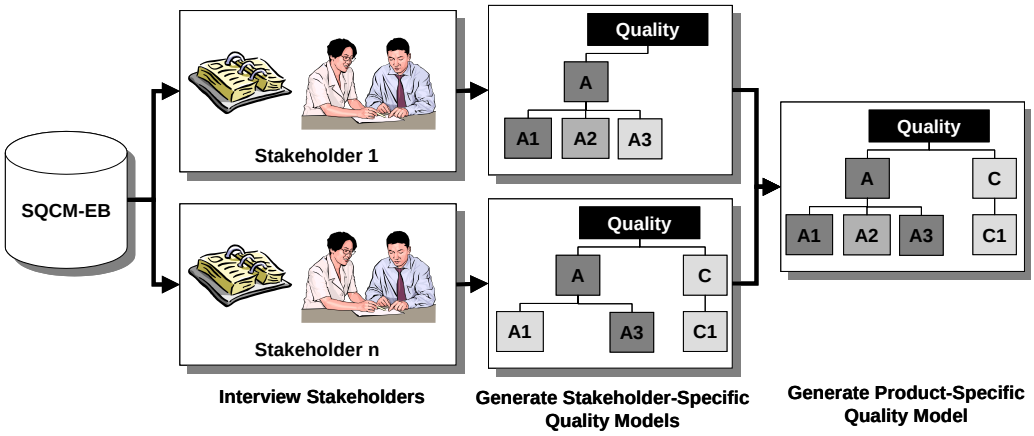


Figure 3 Quality Definition Process

The Quality Definition Process has the following steps: 1) Identify Stakeholders, 2) Interview Stakeholders, 3) Define a stakeholder-specific model for each stakeholder, and 4) Define a product-specific quality model by aggregating the stakeholders' views. In the following we will describe these steps in more detail:

The first step of this process is to identify those stakeholders that are to be considered in the quality definition of the system. Potential stakeholders are users, customers, project managers, contractors/developers, etc.

The second step is to interview individual representatives from the selected stakeholder groups. The objective of these interviews is to identify those quality properties that are important from the viewpoint of the stakeholder group. Therefore, structured interviews are conducted, in which each stakeholder rates the relevance of the quality properties in the experience base. To support these interviews, the interviewer derives a questionnaire from the experience base that includes those properties that might potentially be relevant for the stakeholder. For each quality property its name and definition is given, as well a 4-point-relevance scale. An excerpt from such a questionnaire is shown in Figure 4.

| **Question 18** (Level 2) | **Portability** |
|---|---|
| | The property "Portability" describes how easily it is to transport the software for use in another environments |
| | This quality property is for me .. ❑ highly relevant ❑ relevant ❑ somewhat relevant ❑ not relevant |

Figure 4 Excerpt from questionnaire

In order to minimize the time required for these expert interviews, we exploited the tree structure of our experience base: If a quality property was rated as not relevant, quality properties in the hierarchy below that property were considered as not relevant as well. Here, the questionnaire contained skip patterns that prevented these questions from being asked.

In order to detect whether the set derived from the experience base is complete, the interviewees are asked about missing properties that are to be considered.

In the third step a stakeholder-specific quality model is generated. When multiple persons per stakeholder group are to be interviewed, it is necessary to aggregate the individual viewpoints. Here it is possible [9] to perform a group interview and achieve consensus within the group or to aggregate individual answers mathematically.

The aggregated answers are visualized in a coloured quality tree (cf. Figure 5): the colours indicate whether quality properties have been rated as very relevant, relevant, or somewhat relevant. Quality Properties that were rated as not relevant are not included. Alternatively it is also possible to include only very relevant and relevant properties in the stakeholder-specific tree.

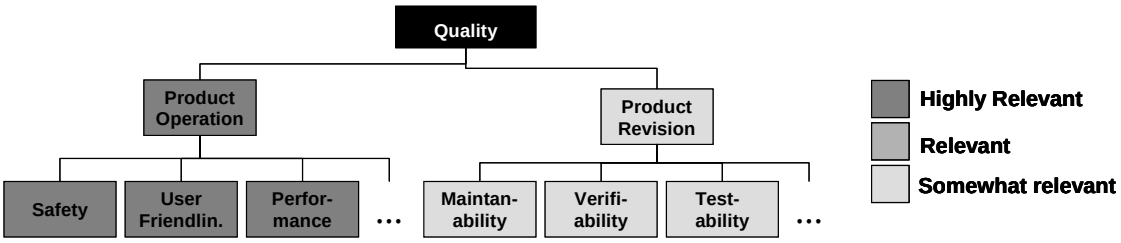


Figure 5 Role-Specific Quality Model (Example)

In the fourth step, a product-specific quality model is generated from the set of stakeholder-specific models. Applying mathematical aggregation it is possible to determine the relevance of a given quality property as the maximum relevance rating of all stakeholders, which implies equal importance of all stakeholders. In a more sophisticated computation, the importance of a group of stakeholders can be assigned a weight that will be considered in the final calculation. An alternative would be that ratings are assigned to the confidence in stakeholder's reply, according to his/her experience. Similar to the previous step, the resulting tree is visualized, as exemplified in Figure 6.

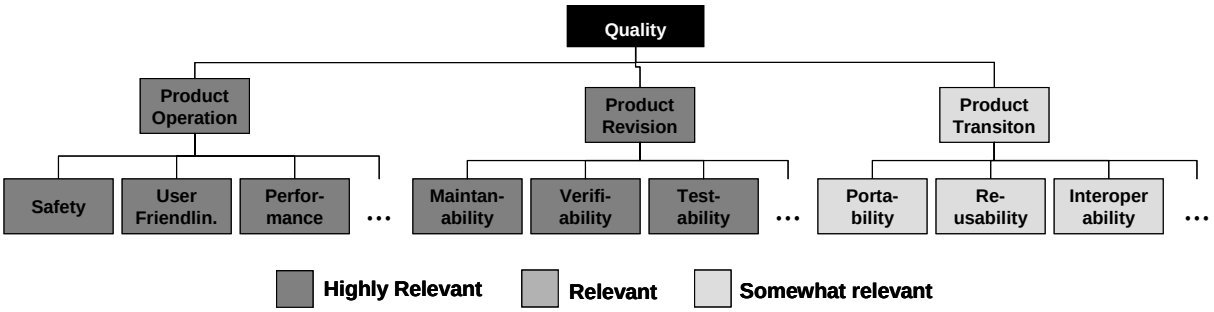


Figure 6 Product-Specific Quality Model (Example)

The advantage of these visualized models is that they clearly show which properties are important to many stakeholders. These are the properties that will be important to measure. In addition, an analysis of the role-specific models allows to make explicit the differences in quality perception from different stakeholders and to prioritize based on stakeholder importance.

Based on this model it is now possible to perform focused GQM interviews or to directly define metrics. For both purposes, the experience base contains measurement models that will support these activities.

# 5 Case Study and Experiences

In order to evaluate our SQCM approach we performed a case study, in which we derived a product quality model for a concrete software product named eWorkshop (from "electronic workshop").

The eWorkshop is a web-based product developed at the Fraunhofer Center for Experimental Software Engineering Maryland. The purpose of this software is to support virtual, on-line workshops with worldwide participants, and to capture the knowledge exchanged during each session for future analysis and dissemination. This software product has been used and has been under evolution for a couple of years.

## 5.1 Application of the Quality Definition Process

Since our initial experience base contained only quality properties from the literature we suspected that several product-specific quality properties were not included. Thus, we performed a first completeness check based on our understanding of the eWorkshop. Thereby we noticed several quality properties that one might expect to be important for such a system but that were not part of our experience base. Consequently we updated our experience base accordingly. This illustrates that it is an important aspect of the interviews to take the quality properties in the experience base as a sound start, but to be open for additional quality perceptions from each stakeholder. Following the Quality Definition Process described in Section 4.2, we then identified in the first step the set of stakeholders to be considered. For this product we identified the User and the Developer as stakeholders.

In order to prepare the interviews of the second step, we produced for each role a questionnaire containing those quality properties corresponding to the stakeholder (cf. Figure 4). Since we did not want to focus on a particular development phase, we considered all phases.

The interviews were held with one representative of the Developer role and two representatives of the User role. In the interview, the interviewees evaluated the relevance of the properties following the questionnaire. The duration of the interviews ranged between 15 and 25 minutes. We consider this time as rather short. Consequently, we regard such interviews as an inexpensive way to define an (initial) quality model.

Next, we generated the stakeholder-specific model for the Developer and the User based on the interview results according to Step 3 of the process. These

models are shown in Figure 7 and 8. (Due to space constraints the figures show only model excerpts.)

These models show that for the Developer quality properties related to product revision (e.g., maintainability, testability) play, naturally, a larger role than for the User. Also it can be observed that the Developer would prefer different measurement models than the User (Fault Density).

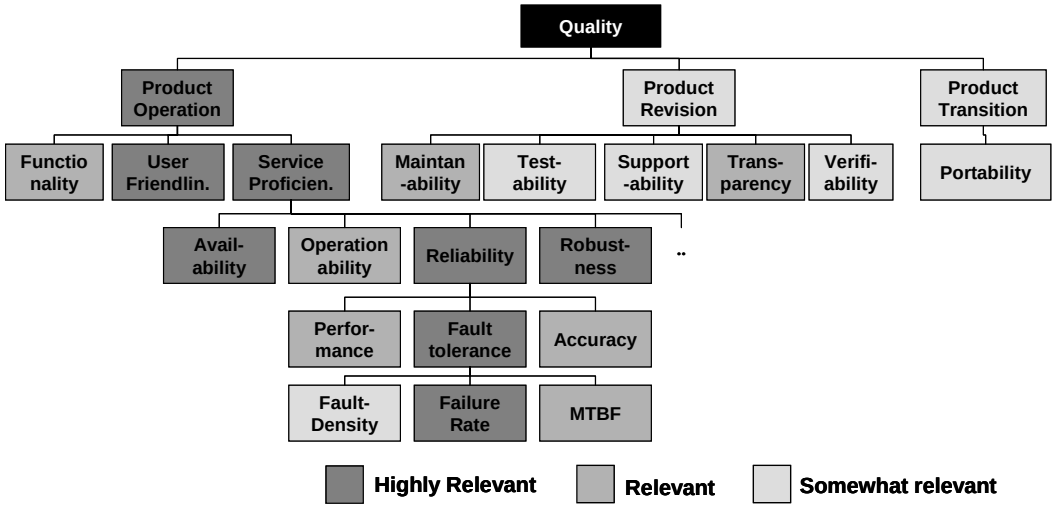


Figure 7 Developer View eWorkshop

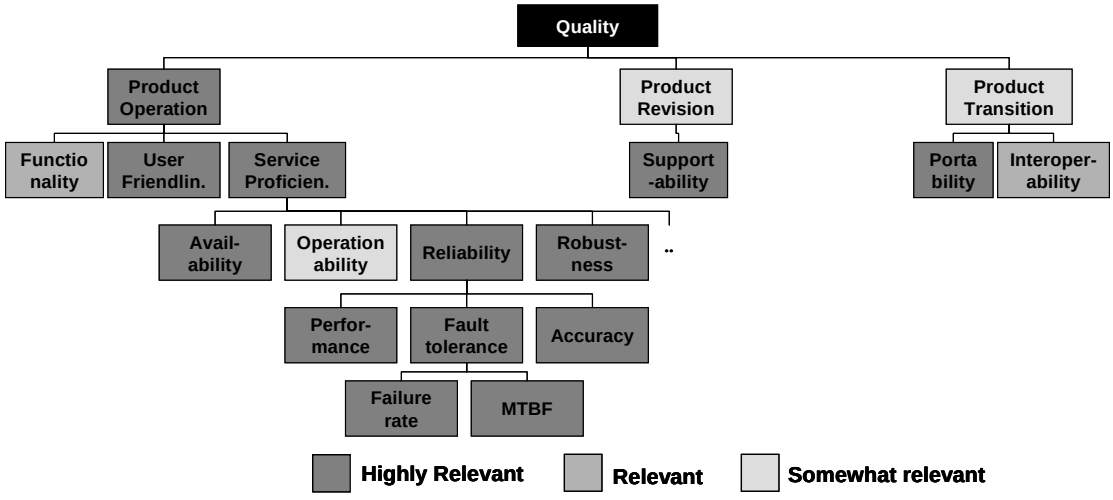


Figure 8 User View eWorkshop

Finally, these two models were aggregated using the maximum-relevance rating from each stakeholder-specific model as shown in Figure 9.

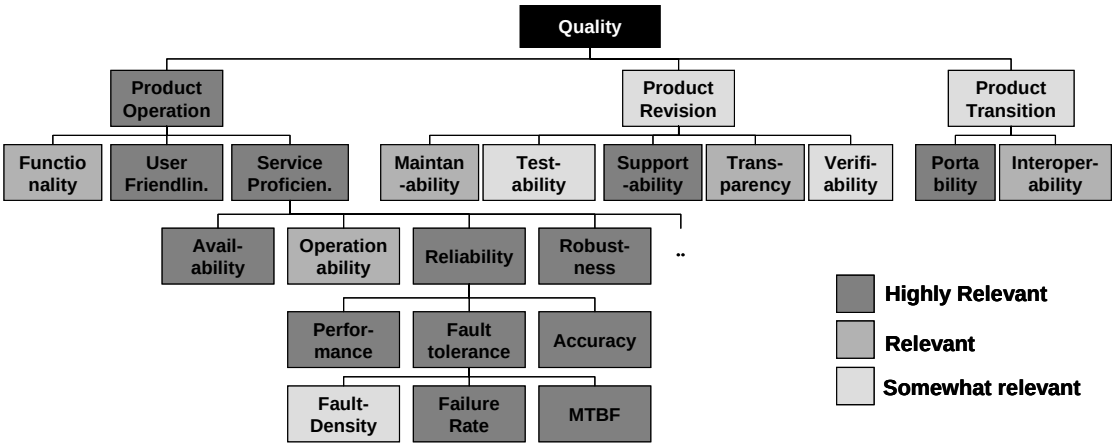


Figure 9 Unified eWorkshop Product Quality Model

These visualizations help to make different views explicit and allow a focused discussion on what to finally measure. The proposal for initial measurement models also helps the measurement team to prepare later steps.

## 5.2 Experiences on the Approach's Usefulness

In order to evaluate the usefulness of our approach, we performed structured de-briefing interviews with each of the interviewees, to assess the approach's pro's and con's. A questionnaire was prepared for these interviews and contained questions that tackled issues like the benefit of the approach, the perception of the time required to participate, the feasibility of the approach, and its improvement opportunities.

Overall, the interviewees considered the approach as useful. Particularly, for the Developer it was interesting to see explicitly, which quality properties were highly relevant for the project manager. Thus, our approach clearly supported the communication about quality between different stakeholders. As overall strengths of the approach were regarded the usefulness for those Software Engineers who have little experience and knowledge in setting up software quality measurement programs and the usefulness for the characterization of a system for which it is not known what quality aspects may be relevant for the system's stakeholders. Moreover, the interviews can trigger the stakeholders to think about quality and to decide which quality attributes matter most to them.

As drawback was the length of the questionnaire considered. Here it would be useful to reduce the questionnaire using a more refined screening about potentially relevant quality properties, for example according to project environment characteristics. Additionally we think that a computer-based support of the questionnaire and experience base might ease the application as well.

Also a User representative said that the approach only gives "a pointer" to important quality attributes, which will have to be more refined and expressed in terms familiar to the project members.

Overall, we conclude from these results that our approach is useful and addresses the problem we wanted to tackle, namely the support of typically inexperienced personnel with an experience base as a preparation for more refined (GQM)-interviews.

Generally, the concept of our experience base can be easily transferred to contexts were quality elements can be structured as shown in Figure 1 or a subset thereof. The key idea of the Quality Definition Process, the relevance rating of proposed properties in interviews, is also easily transferable. However, the concrete structure of the questionnaire and the resulting stakeholder-specific and product-specific models might depend on the number of properties stored and depth of the hierarchy, which were both very high in our case.

# 6 Summary and Future Work

In this paper we proposed a practical approach to support quality measurement with reuse of previously acquired knowledge about the measurement of software quality. For this purpose we developed the SQCM experience base that contains knowledge in the form of quality properties and appropriate measurement models. We defined a quality definition process that allows to select and identify relevant quality properties for multiple stakeholders and thus facilitate communication about different quality needs.

We reported our experience with this approach in a case study. Yet, the work we presented in this paper can be easily transferred into other organizations and enable them to re-use their own knowledge on measuring software quality.

The next steps with this approach are first to develop a tool that supports the tasks of deriving a stakeholder-specific questionnaire and that also supports the stakeholders in easily entering the information and automatically generating the model visualizations.

Also, although our approach helps to make explicit different views on quality, there is still the need to support the resolution of conflicts that result from those different views and to select those properties and measurements that should finally be measured.

# 7 Acknowledgements

This work was supported by the Otto A. Wipprecht Foundation, and in part by the NASA High Dependability Computing Program under cooperative agreement NCC-2-1298. The authors thank their colleagues at the Fraunhofer Center Maryland, USA for their help in performing the case study.

# 8 References


[1] V. Basili, G. Caldiera, and D: Rombach, "The Experience Factory," Encyclopedia of Software Engineering - 2 Volume Set, pp. 469-476, 1994

[2] B. Boehm, J. Brown, H. Kaspar, M. Lipow, G. MacLeod, and M. Merritt, Characteristics of Software Quality. North Holland Publishing Company, 1978.

[3] L. C. Briand, C Differding, and D. Rombach, Practical Guidelines for Measurement-Based Process Improvement, Software Process Improvement and Practice Journal, vol. 2, no. 3, 1997.

[4] N. Fenton and S. Pfleeger, Software Metrics - A Practical and Rigorous Approach. Int. Thomson Computer Press, 1996.

[5] ISO/IEC 9126 International Standard, Software engineering – Product quality, Part 1: Quality model, 2001.

[6] B. Kitchenham, S. Linkman, A. Pasquini, and V. Nanni, The SQUID-Approach to Defining a Quality Model, Software Quality Journal, vol. 6, no. 3, pp. 211-233, 1997.

[7] M. Li, C. Smidts, Ranking Software Engineering Measures Related to Reliability Using Expert Opinion, Proceedings of ISSRE, p. 246 – 258, 2000.

[8] J.A.McCall, P.K.Richards, G.F.Walters, "Factors in Software Quality", US Rome Air Development Center Reports RADC TR-77-369, 1977.

[9] M. Meyer and J. Booker, Eliciting and Analyzing Expert Judgement: A Practical Guide., Academic Press, 1991.

[10] D. Miller, Choice and Application of a Software Quality Model, in Proceedings of the 10th International Conference on Software Quality, 2000.

# Document Information

| | |
|---|---|
| Title: | Reusing Knowledge on Software Quality for Developing Measurement Programs |
| Date: | April 15, 2004 |
| Report: | IESE-009.04/E |
| Status: | Final |
| Distribution: | Public |