Engineering Quality Requirements using Quality Models

Klaus Lochmann
Chair for Software & Systems Engineering
Technische Universität München
Boltzmannstr. 3, 85748 Garching b. München, Germany
lochmann@in.tum.de

***Abstract*—In this paper ongoing research for a PhD-thesis is presented. The objective of this thesis is to develop an approach for engineering quality requirements. Quality requirements are an important part of requirements on software and their thorough specification is a prerequisite for the successful development of high quality systems. The presented approach relies on a quality model, that defines software quality and that serves as a structured knowledge-base. The quality model is integrated with an use-case based approach for eliciting and analyzing quality requirements. This way an effective communication with stakeholders as well as the quantification of quality requirements is assured.**

I. Introduction

It is generally accepted that the development of high-quality software is an important challenge to the industry. Nonetheless the term of quality itself remains unclear and ambiguous since quality itself is a "complex and multifaceted concept" [1]. Despite the variety of definitions of quality, most of them have in common that conformance to requirements is seen as a substantial part of quality.

Although the importance of an accurate specification of requirements for a successful development is generally accepted [2], requirements on the quality of a system – called quality requirements – are often neglected [3].

A. Problem Statement

The reasons for the difficulty in specifying quality requirements become evident in typical characteristics of them. One characteristic is the *cross-cutting nature*, that means that quality requirements usually are intermingled with other requirements as well as with development artifacts. Therefore they cannot be specified separately and their realization is not confined to a single part of the system. Another important characteristic is the *interacting nature* of quality requirements. Different quality requirements are influencing each other in the realization. This means that improving one quality aspect may worsen another one.

These characteristics lead to typical problems observed in practice. Mostly, quality requirements are not specified in *sufficient completeness*, because the stakeholders have problems in expressing their quality requirements or they express them in a very vague manner. Furthermore, the expressed requirements are often not *correct*, i.e. they are either impossible or unrealistic to realize. Other problems attributed to correctness are subjective estimations, missing context information, and oversimplifications. Because quality requirements are often specified in a vague manner, they do not fulfill the criterion of *verifyability*: Requirements are defined as verifiable, if there exists some cost-effective method that can check that the software product meets the requirement [4]. Also the *operationalization* of quality requirements poses problems. They are often not sufficiently concretized, i.e. they give no indication how they can be realized.

B. Objective

The main research question can be formulated as follows:

> How can *quality requirements* be unambiguously specified, to be *supportive during development*, and to be *verifiable* at the end of development?

The developed approach has the objective to support the engineering of quality requirements in a manner that the resulting requirements fulfill the characteristics of good requirements according to IEEE-830-1998 [4]; the requirements should be: Correct, Unambiguous, Complete, Consistent, Ranked for importance and/or stability, Verifiable, Modifiable, and Traceable.

II. Envisioned Solution

The underlying concepts of the developed approach are a value-based view on quality [1] and the cost-of-quality concept [5]. The basic idea of these concepts can be summarized as follows: high quality of a product means, that it incurs low costs and provides high benefits.

In our approach the costs are expressed in terms of effort needed to perform activities on and with the system. In this terms high quality means that the activities can be performed with low effort. It has to be noted, that the activities are not limited to end-user interaction with the system, but they comprise all activities from all stakeholders, e.g. maintenance-activities from developers, attack-activities of intruders, etc.

To define quality by the means of activities and their costs, activity-based quality models (ABQM) [6], [7] are used. The idea of an ABQM is to break down quality into detailed facts and their influence on activities. For ABQMs an explicit meta-model was defined, that consists of the following elements (see Fig. 1): An *entity* can be any thing, that can have an

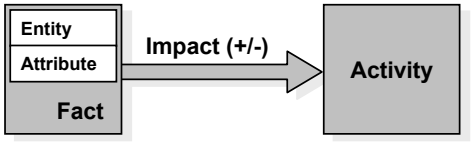

Fig. 1. Activity-based Quality Model

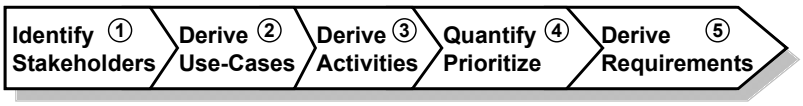

Fig. 2. Requirements Engineering Process

influence on the software's quality, e.g. source code. These entities are characterized by *attributes* such as STRUCTUREDNESS or CONFORMITY. The combination of an entity and an attribute is called a *fact*, e.g. [source code | STRUCTUREDNESS]. An *impact* specifies whether a fact has a positive or negative influence on an *activity*, e.g. [source code | STRUCTUREDNESS] $\xrightarrow{+}$ [Maintenance].

Since realistic quality models can contain a very large number of entities and activities, hierarchies were introduced as a means for structuring. For example the activity hierarchy contains the top-level activity Maintenance, which is refined to sub-activities like Code comprehension and Modification.

The supposed requirements engineering process (see Fig. 2) incorporates the concept of use-case modeling and relies on the ABQM as a knowledge-base to support the elicitation and analysis of requirements. A similar approach has been proposed by Wagner et al. [8].

In step ① Identify Stakeholders all stakeholders relevant to the system-to-be are identified. This step makes use of classical RE methods, such as goal modeling to find all stakeholders and their interests.

In step ② Derive Use-Cases use-cases for each stakeholder are derived, whereby beside end-users also use-cases for stakeholders interacting in other ways with the software have to be regarded, e.g. a maintainer could perform the use-case "process a change-request". In this step the ABQM is used as a knowledge-base by providing activities that can be used as suggestions for use-cases.

In step ③ Derive Activities to the single steps of the use-cases, corresponding activities in the ABQM are identified. If no corresponding activities can be found, new ones are added to the ABQM in order to extend the knowledge-base.

In step ④ Quantify / Prioritize a quantification and prioritization of the activities is taking place. The activities are quantified by the stakeholders according to the costs they incur; and estimation of these costs is done by information like frequency of execution, duration, etc.

In step ⑤ Derive Requirements based on the given activities, the ABQM is used to derive detailed requirements. That means, starting from the activities, all impacts in the ABQM are traced to facts. Each fact expresses a (un-)desired characteristic of the system, which can directly be turned into a requirement.

III. Example

A case-study adopting a similar approach has been conducted in the context of security [9]. In this case-study the stakeholder "attacker" and its intention to attack the system – in our case Tomcat 6.0 – was considered. The attack-scenarios were modeled as misuse-cases. The ABQM was build of available security-guidelines and was then used to derive detailed requirements. To validate the approach, the derived requirements were compared to published vulnerabilities of the web server. The result was that 16 of 19 (84%) of the requirements were referring to actual vulnerabilities and their implementation in Tomcat would have prevented the vulnerabilities.

IV. Conclusion

In this paper a concept for an approach for engineering quality requirements has been presented. By using use-cases in a structured way the communication with the stakeholders is improved and the completeness and correctness of requirements is increased. Furthermore, the quality model enables the refinement and operationalization of the quality requirements, to eventually get verifiable requirements. Through the link of detailed requirements to activities, use-cases, and stakeholders a traceability between requirements is established.

The next steps in this research will be the conduction of industrial case studies, to get experiences with this RE method and to further elaborate the method. Another step for further research is to incorporate this RE approach into existing artifact-based RE approaches.

Acknowledgments

This work has partially been supported by the German Federal Ministry of Education and Research (BMBF) in the project QuaMoCo (01 IS 08023B).

References

[1] D. A. Garvin, "What does product quality really mean," *Sloan Management Review*, vol. 26, no. 1, pp. 25–43, 1984.

[2] B. W. Boehm, "Software Engineering Economics," *Software Engineering, IEEE Transactions on*, vol. SE-10, no. 1, pp. 4–21, 1984.

[3] M. Glinz, "On Non-Functional Requirements," in *Requirements Engineering Conference, 2007. RE '07. 15th IEEE International, New Delhi, India, October 15-19*, 2007, pp. 21–26.

[4] "IEEE recommended practice for software requirements specifications," *IEEE Std 830-1998*, pp. –, 1998.

[5] A. V. Feigenbaum, "Total Quality Control," in *Harvard Business Review*, 1956, vol. 34 (6), pp. 93–101.

[6] F. Deissenboeck, Stefan Wagner, M. Pizka, S. Teuchert, and J.-F. Girard, "An Activity-Based Quality Model for Maintainability," in *Proceedings of the 23rd International Conference on Software Maintenance (ICSM 2007)*. IEEE CS Press, 2007.

[7] M. Broy, F. Deissenboeck, and M. Pizka, "Demystifying Maintainability," in *Proceedings of the 4th Workshop on Software Quality*. ACM Press, 2006.

[8] S. Wagner, F. Deissenboeck, and S. Winter, "Managing quality requirements using activity-based quality models," in *WoSQ '08: Proceedings of the 6th international workshop on Software quality*. ACM, 2008, pp. 29–34.

[9] S. Wagner, D. Mendez Fernandez, S. Islam, and K. Lochmann, "A Security Requirements Approach for Web Systems," in *Workshop Quality Assessment in Web (QAW 2009)*, 2009.