TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

A Quality Model for Software Quality

Klaus Lochmann, Stefan Wagner

TUM-I1328

Abstract

A large number of terms describing aspects of software quality, called quality attributes, are defined by taxonomies in standards like ISO 9126 or ISO 25010. These definitions have come under critique for being ambiguous, overlapping and incomplete. Based on our experience in quality modeling, we developed a quality model, defining a hierarchy of quality attributes. It relies on an activity-based paradigm and thus makes use of a clear decomposition criteria. We believe this clearer decomposition solves some of the problems of other taxonomies.

1 The Quality Model

In this article, we present a quality model defining quality attributes for software systems. In different research projects at the Chair for Software and Systems Engineering at Technische Universität München, we gained experience in quality modeling regarding all different tasks of quality assurance. Together with industry partners, we developed an activity-based quality model for maintainability, used for the generation of guidelines and checklists [2]. In the research project Quamoco, together with several research and industry partners, we extended the scope of the quality model beyond maintainability and extensively worked on automatic measurement and aggregation of measurement results [4, 8]. In other studies, we explored the usefulness of activity-based quality models for the specification of quality requirements [6, 7, 5] and applied an activity-based quality model to both security [9] and usability [10].

Our quality model in this article relies on the paradigm of activity-based quality models [2, 1]. Such quality models define its quality attributes by referring to activities that are conducted with or on the system. For instance, the classical quality attribute *maintainability*, is described as the *efficiency and effectiveness* of conducting the *maintenance* activity. The benefit of reasoning about activities is that they provide a clear decomposition criteria: activities may be decomposed into sub-activities. For instance, maintaining a system means conducting the following sub-activities: analyzing the change request and the existing system, modifying the system, and releasing the modified system. This leads to the introduction of *analyzability*, *modifiability* and *releasability* as sub-quality attributes of *maintainability*.

As a starting point for the development of our quality model, we used the quality attributes of ISO 25010 [3]. We redefined them according to the activity-based paradigm, removed and added new terms where necessary.

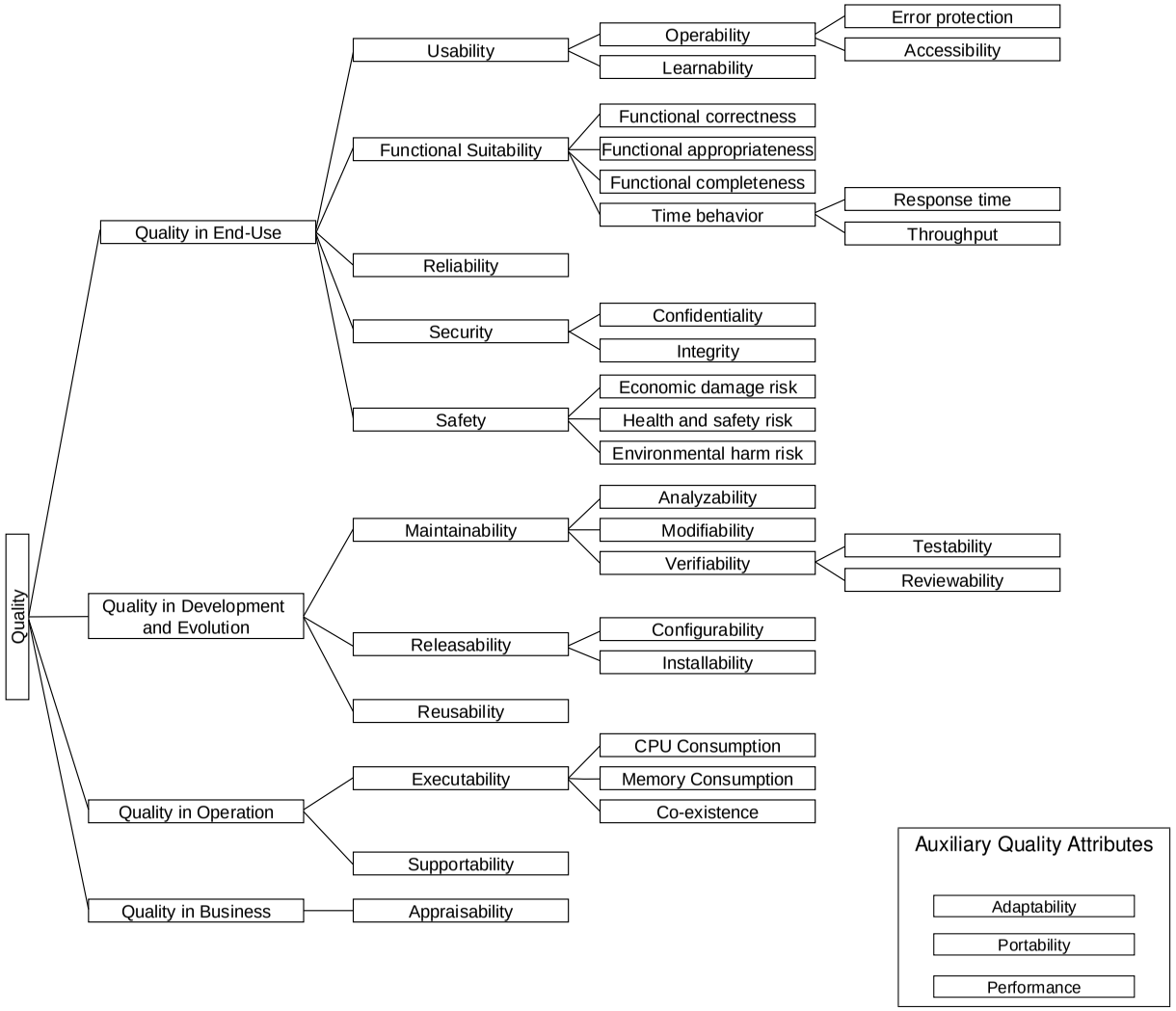
1.1 Structure

The main part of the developed quality model consists of a tree of quality attributes. This tree contains all quality attributes constituting the quality of a system. They were defined in a way to be as non-overlapping as possible. Figure 1.2 shows the hierarchy of these quality attributes.

Besides the main tree of quality attributes, there is a list of auxiliary quality attributes, which are overlapping with the main quality attributes. Each of these quality attributes includes multiple of the main quality attributes or of parts of them. For instance, the auxiliary quality attribute portability is a combination of adaptability and releaseability, whereby adaptability is a special type of maintainability.

A third kind of quality attributes are orthogonal to other kinds of quality attributes. They are provided as a separate list.

1.2 Main Quality Attributes



Quality. the degree to which the system satisfies the requirements of all stakeholders.

Quality in End-Use. the degree to which the system satisfies the requirements of the end-user.

Usability. the degree to which the system enables effective use with the end-user. Using the system includes operating (i.e. interacting with it) and reading/understanding the documentation.

Operability. the degree to which the system enables effective operation with the end-user. Operation by the end-user includes providing input to the system and perceiving and understanding the output of the system.

Error protection. the degree to which the system prevents the end-user from wrong and/or accidential input to the system.

Accessibility. the degree to which the system enables end-users with disabilities operating the system efficiently and effectively.

Learnability. the degree to which the documentation of the system is suited to efficiently and effectively instruct the end-user in operating the system.

Functional suitability. the degree to which the system provides functionality that supports the tasks of the end-user.

Functional correctness. the degree to which the system provides the correct results with the required degree of precision.

Functional appropriateness. the degree to which the functionality of the system supports the tasks of the end-user.

Functional completeness. the degree to which the tasks of the end-user are covered by the functionality of the system.

Time behavior. the degree to which the system satisfies required response times and throughput rates.

Response time. the degree to which the system satisfies required response times.

Throughput. the degree to which the system satisfies required throughput rates.

Reliability. the probability of the system to be functionally correct (see *functional correctness*) at any time.

Security. the degree to which the system prevents unauthorized actors from (1) reading or modifying data of the system (2) hampering authorized actors from using the system.

Confidentiality. the degree to which information and data are protected from unauthorized disclosure.

Integrity. the degree to which the system prevents unauthorized reading or modifying of data.

Safety. "the degree to which a product or system does not, under specified conditions, lead to a state in which human life, health, property, or the environment is endangered" [3].

Economic damage risk. "the degree of expected impact of harm to commercial property, operations or reputation in the intended contexts of use" [3].

Health and safety risk. "the degree of expected impact of harm to people in the intended contexts of use" [3].

Environmental harm risk. "the degree of expected impact of harm to property or the environment in the intended contexts of use" [3].

Quality in Development and Evolution. the degree to which the system satisfies the requirements of the stakeholders concerned with tasks regarding the development and evolution of the system. The evolution of a system includes maintaining and releasing it.

Maintainability. the degree to which the system can be maintained efficiently and effectively. Maintaining the system means modification of the system to correct faults, to improve it to prevent future faults, or to adapt the product to satisfy changed requirements.

Analyzability. the degree to which the systems enables (1) the study of the feasibility and scope of a requested modification

and (2) the devision a preliminary plan for design, implementation, test, and delivery.

Modifiability. the degree to which the systems can be mofified by using the results of the design phase, the current source code, and project and system documentation.

Verifiability. the degree to which the system enables the test for satisfaction of the changed requirements.

Testability. the degree to which the system enables to conduct software tests to assess the satisfaction of requirements.

Reviewability. the degree to which the system enables to conduct reviews of it.

Releaseability. the degree to which the system can be efficiently and effectively released to customers. Releasing means building, naming, packaging, releasing of a particular version of the system, installing, and making operational at the customers'.

Configurability. the degree to which the system can be efficiently and effectively adapted to certain circumstances by means of using build-in functionality of the system; i.e. without changing the system itself.

Installability. the degree to which the system can be efficiently and effectively installed and/or uninstalled in a specified environment.

Reusability. the degree to which the system can be efficiently and effectively used as part of another software.

Quality in Operation. the degree to which the systen satisfies the requirements of the stakeholders concerned with operating the system. Operating includes operation of the hardware, and providing support to end-users.

Executability. the efficiency with which the system can be executed on the target hardware.

CPU consumption. the efficiency with which the system uses the computing resources of the CPU.

Memory consumption. the efficiency with which the system uses the memory of the hardware.

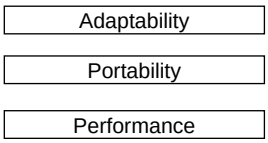
Co-existence. the degree to which the system can co-exist with other independent systems in a common environment sharing common resources without any detrimental impacts.

Supportability. the degree to which the system enables providing technical assistance, consulting with the user, recording user support requests, and triggering maintenance activities.

Quality in Business. the degree to which the system satisfies the requirements of the stakeholders concerned with acquiring software.

Appraisability. the degree to which acquisitioners can efficiently and effectively assess whether the systems satisfies their requirements.

1.3 Auxiliary Quality Attributes

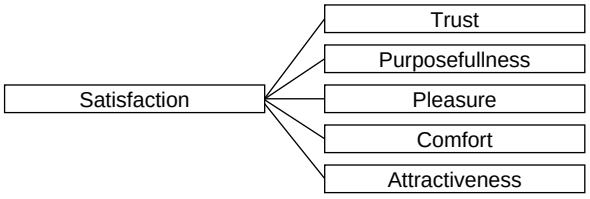


Adaptability. a special type of maintenance, with the goal of adapting the system to satisfy changed requirements. Two other special types of maintenance are corrective and preventive maintenance.

Portability. the degree to which the system can be efficiently and effectively transferred from one hardware, software or other operational or usage environment to another. Transferring the system means adapting it, and releasing it. Thus, *portability* is a combination of *adaptability* and *releasability*.

Performance. subsumes the *time behaviour*, *CPU consumption*, and *memory consumption*.

1.4 Orthogonal Quality Attributes



Satisfaction. the degree to which the system makes the end-user feel satisfied by using it.

Purposefullness. the degree to which the end-user "is satisfied with their perceived achievement of pragmatic goals, including acceptable perceived results of use and consequences of use" [3].

Trust. the degree to which the end-user "is satisfied that the product will behave as intended" [3].

Pleasure. the degree to which the "end-user obtains pleasure from fulfilling their personal needs" [3].

Attractiveness. the degree to which the end-user considers the product to be attractive.

Bibliography

[1] F. Deissenboeck. Continuous Quality Control of Long-Lived Software Systems, PhD thesis, Technische Universität München. 2009.

[2] F. Deissenboeck, Stefan Wagner, M. Pizka, S. Teuchert, and J.-F. Girard. An Activity-Based Quality Model for Maintainability. In Proc. of the *International Conference on Software Maintenance (ICSM '07)*. IEEE Computer Society, October 2007.

[3] ISO. 25010, Systems and software engineering – System and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models.

[4] K. Lochmann and L. Heinemann. Integrating Quality Models and Static Analysis for Comprehensive Quality Assessment. In Proc. of the *International Workshop on Emerging Trends in Software Metrics (WETSoM '11)*. ACM, May 2011.

[5] K. Lochmann, D. Mendez Fernandez, and S. Wagner. A case study on specifying quality requirements using a quality model. In Proc. of the *Asia-Pacific Software Engineering Conference (APSEC 2012)*. IEEE Computer Society, December 2012.

[6] S. Wagner, F. Deissenboeck, and S. Winter. Erfassung, Strukturierung und überprüfung von Qualitätsanforderungen durch aktivitätenbasierte Qualitätsmodelle. In Proc. of the *Software Engineering Konferenz (SE '08)*. GI, February 2008.

[7] S. Wagner, F. Deissenboeck, and S. Winter. Managing quality requirements using activity-based quality models. In Proc. of the *International Workshop on Software Quality (WoSQ '08)*. ACM, May 2008.

[8] S. Wagner, K. Lochmann, L. Heinemann, M. Kläs, A. Trendowicz, R. Plösch, A. Seidl, A. Goeb, and J. Streit. The Quamoco Product Quality Modelling and Assessment Approach. In Proc. of the *International Conference on Software Engineering (ICSE '12)*. ACM, June 2012.

[9] S. Wagner, D. Mendez Fernandez, S. Islam, and K. Lochmann. A Security Requirements Approach for Web Systems. In Proc. of the *Workshop Quality Assessment in Web (QAW '09)*. Springer, June 2009.

[10] S. Winter, S. Wagner, and F. Deissenboeck. A Comprehensive Model of Usability. *Engineering Interactive Systems*, (4940):106–122, 2008.