The Definition of ‚Software Quality': A Practical Approach

Dr. Roland Petrasch
petrasch@computer.org

Motivation

Probably everyone has an idea about the meaning of *quality*. However, when it comes to quality in the real world, i.e. in conjunction with a software development project, disagreements between the persons involved often lead to further problems. Especially in the case of customer complaints about faults in a software product, it seems to be unclear not only what the requirements are, but also if the software has the „right" characteristics with regard to these requirements. This article aims to reduce the confusion arisen about *quality*, *requirement* and *characteristic.*

Introduction

Only when the requirements are defined in conjunction with the characteristics that are relevant for quality, it is possible to measure software quality. We have learned from experience that it is neither possible to give a standard recipe to overcome "software's chronic crisis" [2], nor the deadlock can be broken easily that evolves in many projects because of short-term and hectic activities of error correction to solve acute quality problems to the debit of long-term and well-panned methods to "produce" quality. Nevertheless, one main concept for quality can be presented here.

To explain *quality* the definition of the *International Standardization Organization* (ISO) is a basis but over and above that, software engineering techniques are used later to provide an object oriented model for the context in which *quality*, *requirements* and *characteristics* occur. It can be shown that even in scientific literature and standards the understanding of *quality* is neither uniformed nor consistent, so that the given definition can help especially when standards are interpreted and applied.

Definition of *quality*

Standards shall and can help to define terms like *quality* [9]. Nevertheless, the means of expression used in standards are often not appropriate for the practice. This is also true for the definition of the ISO 8204 for quality: „Totality of *characteristics* of an entity that bears on its ability to satisfy stated and implied *needs*." [3]. That means: We require a *quality software product* to have certain *characteristics* that are related to *requirements* (of the user) and satisfies them.

It is clear that the pair *requirement* and *characteristic* plays a central role in the definition of *quality*. Therefore, an object oriented model[1] contributes to a better understanding for these notions. Figure 1 shows a software product, which is to fulfill *requirements* in having appropriate *characteristics* [10]. The existence of relationships between *requirements* and *characteristics* makes statements about the quality of a product possible.

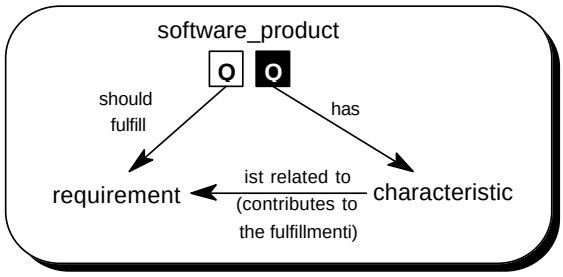

Fig. 1. Relationship between requirements and characteristics in conjunction with quality [10]

Problem of the term *Quality*

Despite the standardized ISO definition of quality, it can be shown that scientific literature lacks consistency and unity regarding the usage of the terms *requirement* and *characteristic.* Examples of this heterogeneity are terms like *feature, attribute* and *characteristic* [11] [12]. Unfortunately, standards like ISO 9001 [4], ISO 15504 [7] and the German V-Modell 97 [1] (similar to DoD MIL-STD-498) give reason for further criticism.

Let's take the ISO 9001 as an example: Because of the reference to the ISO 8204, it can be assumed that the terms *requirement* und *characteristic* are often used. This is the case when it comes to *requirements*, but the standard does not give any information about *characteristics*, e.g. chapter 4.10 (Inspection and testing), which is quite astonishing in the face of the definition of *quality*.

The following example will show that it is possible to describe the management of *requirements* and *characteristics* in the software development process to determine the *quality* of a software product.

Example: Quality Inspection

This example shows an important process that is a part of quality management: The static analytical quality measure "inspection of the user interface". It takes up the central concept that certain *characteristics* of the user interface (e.g. list box, push button) are to be checked against the *requirements* that can be ergonomic requirements of the ISO 9241-10 [6]. According to the *quality* definition, the precondition for any inspection is the assignment: The *characteristics* have to be assigned

[1] It is left open, if the model uses classes in means of the object-oriented paradigm. Important is here that a technical term is brought into a context. This terminus oriented notation is used in [6].

to the *requirements* they are able to fulfill.

At first, a presentation of some *requirements* and the assigned *characteristics* prepare for the inspection. Figure 2 demonstrates the existence of the

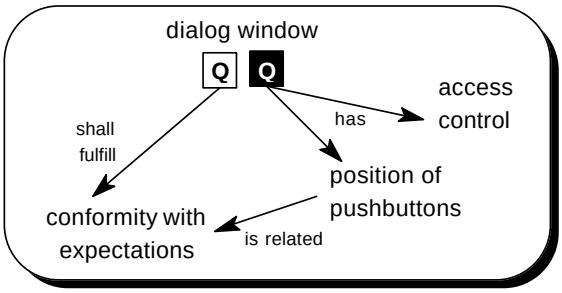

Fig. 2: Example for a requirement and characteristics

requirement "Conformity with user expectations" [6]. The consistency of the dialog layout is necessary. The *characteristic* "position of push buttons for dialog navigation" can help to fulfill this requirement and therefore it is assigned.

The inspection is to check the *characteristic* "position of push buttons" of different windows against the *requirement* "Conformity with user expectations", i.e. in all windows we expect push buttons with the same or analogous functionality of interaction at the same position. The "OK" button (i.e. "save & exit") is such an interaction element and occurs in many dialog windows.

The windows belong to the product Microsoft® Word for Windows 2000®[2] (see Fig. 3 and 4). Push button positions are the values of the *characteristic*. They indicate a difference: Window „Language" places the „OK" button on the left side of the „Cancel" button, while window „Columns" contains the button on top of the other, i.e. the inspection has found an error, because the assessed *characteristic* is not able to fulfill the *requirement* "Conformity with user expectations" which results in the statement: The quality of the dialogs is deficient with regard to the requirement.

This example has shown how to use *requirement* and *characteristic* in conjunction with an inspection and how a quality statement is based on the results.

Conclusion

Software quality is the existence of *characteristics* of a product which can be assigned to *requirements*. In addition to this, we have to look at the characteristics that are not related to requirements: Characteristics, which reduce the software quality (contra-productive) and „neutral" characteristics, which are not relevant for quality. It is clear that not only the presence of characteristics is important, but also the absence of these contra-productive characteristics.

Hopefully in the literature and standards a more unified and consistent usage of the term *quality* can be found in the future. Experiences with the *quality* concept presented here are positive, because the personnel involved in a development project relies on a common terminological fundament.

References

[1] Entwicklungsstandard für IT-Systeme des Bundes: *Vorgehensmodell, Teil 1: Regelungsteil*. Juni 1997

[2] Gibbs, W.: *Software's chronic crisis*. Scientific American. 1994

[3] DIN EN ISO 8402: 1995: *Qualitätsmanagement Begriffe.* 1995

[4] DIN EN ISO 9001: 1994: *Qualitätsmanagementsysteme.* 1994

[5] Horn, E.; Schubert, W.: *Objektorientierte Software-Konstruktion*. 1993

[6] ISO 9241-10:1996: *Ergonomic Requirements for Office Work with Visual Display Terminals - Part 10*. ISO, 1996

[7] ISO/IEC TR 15504-2:1998(E) – *Information Technology - Software Process Assessment: Part 2: A reference model for processes and process capability*. Template Vers. 3.3, ISO/IEC 1998

[8] Myers, G.J.: *Methodisches Testen von Programmen*. 1995

[9] Petrasch, R.: *Einführung in das Software-Qualitätsmanagement.* 1998

[10] Petrasch, R.: *Entwicklung von Modelltypen für das Qualitätsmanagement in der Software-Entwicklung am Beispiel von ausgewählten Qualitätssicherungsmaßnahmen. Dissertation.* Universität Potsdam, 1999

[11] Thaller, G. E.: *Software-Qualität: Entwicklung, Test, Sicherung.* 1990

[12] Wallmüller, E.: *Software-Qualitätssicherung in der Praxis.* 1990

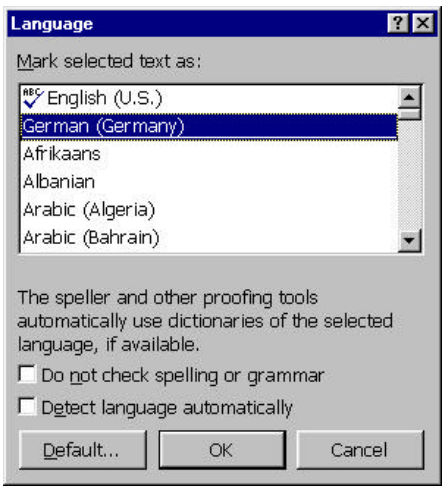

Figure 3. Window „Language"

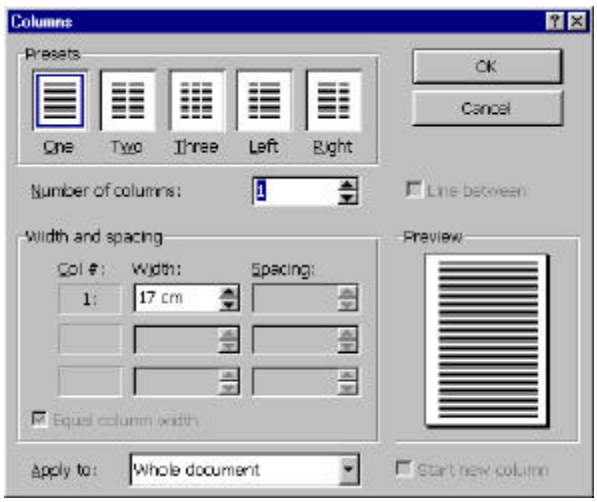

Figure 4. Window „Columns"

[2] Microsoft and Word for Windows 2000 are reg.Trademarks of Microsoft Corp.