Review Article

iMedPub Journals
http://www.imedpub.com

American Journal of Computer Science and Information Technology

2021
Vol. 9 No. 7: 98

Software Quality and Software Quality Models Practical Recommendations and Research Negotiation Points

Seyfali Mahini*

Department of Computer Science, Islamic Azad University, Khoy, Iran

Corresponding author:
Seyfali Mahini, Department of Computer Science, Islamic Azad University, Khoy, Iran

my1341post@yahoo.com

Citation: Mahini S (2021) Software Quality and Software Quality Models Practical Recommendations and Research Negotiation Points. Am J Compt Sci Inform Technol Vol.9 No.7: 98.

Abstract

Quality of software has great and important potential for a newly developed country like Iran. But it is also an enormous challenge at the same time. Quality models contribute to the consolidation and specification of the complex quality issues. As a result of the discussions of quality experts from research and practice, this paper shows which questions in the future must be dealt with most urgently in the field of software quality models to decisively improve the state of research and practice.

Keywords: Software; Computer engineering; ROI expectations; Quality models

Received: June 11, 2021; **Accepted:** June 25, 2021; **Published:** July 02, 2021

Introduction

Software quality models are the basis for all quality-related activities. The use of such models harbors untapped potential, but also problems that practice and research have to face. Not just in Iran, but worldwide, software has a reputation for inevitable quality problems. It is said that the success of the Iranian software industry depends on its quality management skills. Software errors cause high costs through product recalls and a lack of maintainability causes enormous economic damage. In order to survive in global competition, differentiation through quality is becoming more and more important so there is an urgent need for action.

The first question that arises is: what is quality? Product quality can be viewed from different perspectives. Product quality models are a remedy to define more precisely what software quality means in a certain context.

Various standards for quality models, such as the ISO standard 9126, company-specific standards, but also more recent academic approaches, already exist, but leave questions unanswered. For example, it is still not possible to aggregate individual quality measurements to form more significant statements clearly clarified. Another topic is the appropriate modeling of the relationships between quality, costs and benefits [1].

The aim of this paper is to bring together the most pressing issues in the field of software quality models. From these questions we derive a research agenda for software quality models which can give the community from research and practice a common direction. In this context it can discussions and several major challenges are identified. First is the measurement and evaluation of quality, even with the help of quality models, is often too unsystematic or only shows deficits, but no possibilities or levers to remedy them. Second, the relationship between quality and costs/benefits is not dealt with adequately. Thirdly, the adaptation of quality models is a largely unprocessed but practically relevant topic, since software is used in a wide variety of domains and contexts. Fourth, it is important to have a clear strategy the introduction of quality models in companies, as the success of the model depends on it. Despite these points of criticism, quality models provide valuable information.

In the following sections we present the main topics in detail and give practical recommendations and relevant research questions.

Quality Measurements with Quality Models

Quality models represent a systematic abstraction of quality features for software. They generalize individual quality defects and provide a basis for measurements. For many, the definition and refinement of “ilities”, such as reliability or functionality in ISO 9126, are sufficient for a quality model. Other approaches go much further and describe causal relationships at a very detailed level or the relationship with automated quality reviews [2].

When using quality models in software development, there are various usage scenarios that build on one another. For example, a model can first be used to understand quality. For this, the complex and multi-layered aspects of software quality should be described in a quality model in such a way that understanding is improved. Only a structured approach, as offered by a quality model, allows a meaningful definition of quality at all. This definition can in turn be used as a starting point for the quality analysis. The model can define indicators and metrics for this and support the interpretation of the measurement results.

Furthermore, the model should also show ways to improve quality by helping to identify the relevant influencing factors and, ideally, to predict quality at an early stage.

Basically, it can be stated that quality models must not be an end in themselves, just as the measurement of quality cannot be an end in itself. Rather, they always have to support specific purposes. For example, they can help to make a make-or-buy decision or determine whether a software ready for delivery.

Experiences and challenges: Quality planning and assurance in software and systems engineering requires the quantification of target and actual values. This requires an operationalization of the quality aspects in the form of concrete metrics. The selection of suitable metrics depends on a number of factors (such as the development context), without which a quality model loses a large part of its usefulness. Since currently existing standards such as only include a high level of abstraction due to the desired breadth of application, they cannot offer this operationalization and their practical use remains extremely questionable. Therefore, a uniform meta-model would be desirable that allows the description of both necessary and optional components of a quality model. The Factors-Criteria-Metrics (FCM) and methods operate on such a meta-level Goal/Question/Metric (GQM). FCM describes a simple meta-model for quality models, GQM a procedure for developing customized measurement systems [3].

Furthermore, there are only imprecise, implicitly defined terms for many quality aspects (e.g. maintainability, performance, accuracy). Even terms that appear comparatively clear, such as size, cost or effort, can have very different definitions in different environments. Overall, the goal-oriented derivation of suitable metrics and the minimization of the set of metrics to be recorded turn out to be difficult, especially with "soft" influencing factors such as experience, team coherence or the quality of code comments [4].

The lack of appropriate standards and guidance makes it difficult for companies to identify the indicators that are important to them. Often times it is not clear how existing measurement programs can be used to answer new measurement targets. In addition, baselines are usually missing in practice, so that a comparative assessment of software quality is not possible. Another experience with regard to software measurements is that many companies prefer to orient themselves towards standards in order to act in accordance with the state of the art and thus protect themselves against risks (especially legal). As a rule, however, these standards do not sufficiently take into account the context of use. The preference for using rigid metric sets is also likely to stem from the fact that the awareness that there is a significantly higher number of variation parameters in the software area than in production and manufacturing processes and that metrics and quality models are therefore subject to much greater variations is often not available. Examples of challenges with regard to the technical implementation of software measurements are the cost-efficient collection of data (possibly through automated processes), the integration of measurement tools in the development process and the harmonization of measurements in the case of distributed development across organizational boundaries [5].

Recommendations: The challenges result in three core recommendations for quality measurements in practice:

- Metrics have to be derived from goals and fit for the context.
- Existing measurement programs should be used as far as possible to answer measurement goals. It is often helpful to have software measurements taken in the context of others introduce process improvement programs in order to use synergy effects.
- Software measurements should not start with goals that are too ambitious (e.g. forecast). The development of measurement competence in companies usually takes several years. As a start the determination of baselines is recommended.

Research questions: These challenges and recommendations give rise to research questions with regard to the measurement of software quality. In the practical implementation, the question arises for companies how to arrive at the metrics that are important for their goals and boundary conditions. The derivation and definition of such metrics is still a question for research. The task of developing empirically robust industrial standards for quality models that are sufficiently abstract on the one hand and enable systematic instructions for adapting to company contexts on the other hand can be central challenge in quality models. Furthermore, the development of easy-to-use measurement tools that can be easily adapted to these contexts should be mentioned.

Costs and benefits of quality

The relationship between costs and quality, which has been an issue in the manufacturing industry for many years, has also found its way into software engineering. In a profit-oriented company, quality can never be seen in isolation from other factors, but has to be a cost/benefit is subjected to consideration [6].

Experiences and challenges: Measurements and activities relating to software quality are still not a matter of course in many companies and organizations. The aspect of software quality is often ignored or is only taken into account when significant problems have already arisen. Quality aspects are often considered very late in the development process, usually in connection with the first test processes. Most quality defects can no longer be remedied at this point, or only with great effort.

One reason for the lack of awareness of software quality in development organizations is that the economic significance of software quality is often unclear to decision-makers. The relation of measures to improve quality to higher goals of an organization, in particular to business goals, is often not explicitly shown. The costs of measures and measurements are clearly visible, while the benefits in the form of future savings and avoided follow-up costs are not obvious and difficult to quantify [7]. This is related to the fact that software is seen in many companies as an implementing factor and not as a value-adding factor in relation to corporate goals, although it is now the most important driver

of innovation in many industries. In addition comes the contrast between the supposedly long-term amortization of quality measures and short-term planning horizons at management level. Experience from industrial practice shows that there are often high ROI expectations with regard to measures to improve software quality.

Software quality is often not relevant when placing orders. In the case of tenders, the economically more favorable offer usually counts. Long-term costs (in particular costs caused by quality defects) are not or not sufficient when placing the order considered. One of the reasons for this is that the meaning of non-functional properties is difficult to convey and non-functional requirements are more difficult to translate into contractual ones regulations are to be integrated. In the public sector, there are now first approaches to take economic feasibility studies into account for IT investments (in particular the Wi Be procedure).

Recommendations: The following recommendations can be made for the successful introduction of cost/benefit considerations of quality in practice:

- It is important to explicitly explain the costs and benefits of software quality and its measurement. This includes the presentation of the importance of software quality in the context and in relation to IT and business goals of an organization. In some cases, the consequences of poor quality can be presented in the form of additional costs, risks and liability in the legal sense.
- Quality is sometimes in conflict with (short-term) cost and deadline targets or is at least perceived that way. A corresponding trade-off must be included here so that an "appropriate" quality level is achieved within a tolerance threshold to be defined. The medium and long term the positive influence of quality on costs and deadlines must be explicitly shown.
- Contractors should insist on their clients that even non-functional ones requirements are formulated early and precisely.

Research question: A systematic review of the cost-benefit relationships in the area of software quality would make the use of quality models much easier. By providing cost-benefit models that make quantitative statements regarding the long-term benefit of quality activities can make, many statements that are more of an anecdotal nature today could be put on a solid foundation and thus contribute to the factual discussion. Metrics at the engineering level of an organization must match higher; business-relevant metrics (such as a balanced scorecard defined) can be related.

Adaptation of quality models

Software is used in a wide variety of areas and domains, which places very different demands on the systems. This also applies to quality requirements. The standardized models, such as ISO 9126, nowadays do not offer any specific guidelines on how to adapt them to these differences [8]. This is a strong obstacle to their operationalized use in software projects. We have deliberately chosen the term "adaptation" for this topic, as the alternative "tailoring" often only stands for the omission of parts, but not for the expansion.

Experiences and challenges: A number of quality models from a wide variety of sources already exist. Models can be found in the scientific literature and in various general and domain-specific standards such as. In addition, define companies that go further with employing software development, typically in house guidelines that are adapted to the specific requirements in the respective projects. Finally, the use of implicitly specify a corresponding quality model for measuring tools. With quality models for software development, one basically moves between the complete self-definition of the model and the unchanged use of an existing model. The former allows precise tailoring to specific needs, but also requires enormous effort and corresponding expertise. The latter, on the other hand, minimizes the effort, but can cause considerable problems if the model used does not match the actual circumstances in the company or project. Furthermore, experience has been made in various projects that quality models can become very extensive in specific applications, as they have to cover detailed information on a wide variety of quality aspects [9].

For these reasons, we consider a quality model to be useful, which defines a standardized basic model, but also includes a process for adapting to your own requirements and circumstances. In order to develop such a model, however, some fundamental questions still need to be clarified. Only the SQUID approach currently includes a procedure for the specific definition of quality models, but offers fewer details on their operationalization.

Recommendations: An important point in the adjustment of quality models is the identification of the most important adjustment dimensions or the influencing factors. The goals of the company or project are decisive for this. It must be examined how these affect the quality model and the model must be adapted accordingly. A validation must then take place to check whether the adjustments make sense. This can be done, for example, through feedback with developers and quality assurance personnel. Integration into the process model used is also necessary for longer-term use. For example, it must be specified at which decision points (quality gates) the model will be used or when measurements will take place. It is unlikely that the quality requirements and the quality model will be fully available at the beginning. This is why a bootstrapping procedure seems to make the most sense: Requirements and quality model are iteratively and incrementally compared over the course of the project. Priorities and weightings of the individual parts of the quality model are also determined, which in turn could determine the budget for quality, as in activity-based costing. With a fixed budget, this can of course determine the weightings.

Research questions: From these challenges and recommendations we derive a number of research questions, the answers to which we see as necessary for a practical adaptation of quality models. First of all, it has to be examined which spectrum of quality models currently exists and how great the variability is in quality models across domains, projects and technologies. The experiences presented indicate a significant variability, which,

however, has not yet been scientifically investigated. However, this is important in order to derive a usable, sufficiently flexible adjustment mechanism. Furthermore, it must be analyzed which components a quality model necessarily and optionally includes both with regard to the structure of a quality model and with regard to the content. With the cost aspects, the question arises what relationship between existing cost accounting and Quality models as this could be an important factor in customization. It is also still unclear how a comprehensive and extensive quality model (after its adaptation) can be validated.

Introduction of quality models

The basic requirement for successful quality management is the creation of the necessary acceptance in the corresponding companies. Abstract considered the introduction of quality models the goal of taking quality characteristics the voluntability. Similar to functional Requirements should be achieved that quality requirements are of course implemented and not omitted or changed at will. This introduction is a complex process that needs to be carefully planned and to be analyzed in advance on the expected benefits. Which factors are decisive for this, must be examined more closely.

Experiences and challenges: The use of quality models is usually carried out in the context of long-term quality initiatives and rarely to resolve acute problems. Due to this long-term nature, it is often difficult to motivate the persons involved as they cannot easily recognize the immediate benefits of initiatives. On the contrary, quality initiatives require most involved, at least at the beginning, a certain extra work. Unfavorable boundary conditions, such as delivery pressure, lack of resources and lack of management make the introduction complicate.

Another problem with the introduction of quality models can result from organizational separations and the resulting perspectives of the employees of different process phases. So it is difficult to convince developers who are never entrusted with maintenance tasks from the advantages of readable code. Frequently, not all project participants have the appropriate education that would allow them to allow the long-term benefit of quality initiatives or the consequences of non-compliance with quality standards detect.

In addition to these motivational-personnel problems, the already discussed inflexibility of today's quality models is a difficulty as they adapt to the project-specific quality needs complicated and thus contributes neither to the effectiveness nor to the efficiency of quality management. Especially automatic checking methods for quality criteria suffer often under a high number of false positives that can quickly lead to a demotivation of project participants [10].

Recommendations: It is advisable to make the introduction of quality models as carefully as possible in order not to overwhelm the stakeholders by a wealth of innovations. It has been very helpful proved to identify and present an immediate benefit for the individual stakeholders. This helps to increase the acceptance for long-term innovations. Beyond this immediate benefit, of course, it must be ensured that all stakeholders have a consistent understanding of the objectives to be achieved with the help of quality models and are particularly aware of the long-term consequences of non-consideration of quality properties. Here, quality models, which take into account cost/benefit aspects, provide valuable services. Important for the acceptance of quality measurements in software and system development is that it is seen as a means for purpose and not as an end in itself. Measurement results should be communicated in regular feedback to the parties, and insights from analysis should result in actions. The inflexibility of quality models and the results problems with the introduction can be addressed with the adaptation mechanisms discussed in the last section. The following dimension of the adjustment is important here: organizational customization. An adaptation of the quality model to the organization is important in order to demonstrate the project participants that the use of a quality model is not a bureaucratic fulfillment of standards, but a means of increasing the effectiveness and efficiency in development, employee specific perspectives. In addition to the actual adjustment, the quality model for the project participants should provide tailored perspectives. As a result, every project participant is provided only to the proportion relevant to him, and a stress with unnecessary information is avoided, configuration of the checking methods. It is necessary to adapt review tools and methods to the respective context that the number of false positives on one bearable measure (<5%) is limited.

Research questions: From these findings and recommendations, there are a number of research questions regarding the introduction of quality models. In addition to the more technical questions about improved adaptability and better tool support of quality models, especially socio-psychological aspects (Change management) play a role. In particular, a structured work-up of the findings from other disciplines, e.g. process launch, a promising starting point to better understand what problems encounter the introduction of quality models on a socio-psychological level. Beyond these concrete issues, it makes sense to consider the topic of quality even more in the training of future skilled workers and thus achieve a situation in the long term, in which quality models do not have to be introduced, as they are an integral part of each software project.

Discussion and Conclusion

In all these discussions, a point has remained undisputed: a model for the product quality of software is centrally located for dealing with quality in software development. The topic is too diffuse to get along without clear definitions and definitions. For this reason, a central task for research in the next few years is to develop a quality model based on existing standards and recent results, which meets the current challenges. Software quality is a crucial factor for the competitiveness of companies and business locations. Quality models are an accepted means of dealing with software quality. However, there are very different types of use and also very different characteristics in the different domains and companies. For all these characteristics, however, a number

of research questions that need to be answered to make software quality in practice. This paper provides practical recommendations and a research agenda with the central points based on a detailed discussion with academic and industrial experts. It reflects an extensive spectrum of experience. The proposals offer an opportunity to the Iranians to make engineering services popular in the world for software as well.

References

1 Preiss O, Wegmann A (2003) A systems perspective on the quality description of software components. J Syst Cybern Inf 1: 18-24.

2 Broy M, Jarke M, Nagl M, Rombach D (2006) Manifest: Strategische bedeutung des software engineering in deutschland. Infor Spec 29: 210-221.

3 Punter T, Kusters R, Trienekens J, Bemelmans T, Brombacher A (2004) The W-process for software product evaluation: a method for goal-oriented implementation of the ISO 14598 standard. Softw Qual J 12: 137-158.

4 Basili VR (1992) Software modeling and measurement: The goal/question. Metric.

5 Wagner S, Broy M, Deibenböck F, Klas M, Liggesmeyer P et al (2010) Praxisempfehlungen und Forschungsagenda. Infor Spec 1: 25-29.

6 Deissenboeck F, Wagner S, Pizka M, Teuchert S, Girard JF (2007) An activity-based quality model for maintainability. In 2007 IEEE International conference on software maintenance 184-193.

7 Garvin D (1984) What does 'Product Quality'Really Mean. MIT Sloan Manag Rev 26: 1-4.

8 Dromey RG (1995) A model for software product quality. IEEE Trans Softw Eng 21: 146-162.

9 Heidrich J, Münch J, Trendowicz A (2009) Messbasierte ausrichtung von softwarestrategien an geschäftszielen. Int J Inf Manage 2: 82-89.

10 Heitlager I, Kuipers T, Visser J (2007) A practical model for measuring maintainability. 6th Int Conf Inf Commun Technol 20: 30-39.