**Software Quality Model**

Rochak Kunwar

CS 416

A software quality model is the representation of criteria which is followed by software quality management, so they can reach a certain level of quality in the software when it reaches to users. The main objective of the software quality department must be to fulfill customer needs. (tutorialspoint & Jas Preet Kaur)

The quality model presented will be for bigger software project than the project done in 165 and 246, but it will not be bigger than a project that can be done in the final project. This quality model will be useful for the application focused software. This model can be followed to test the quality of software that will have some sort of interaction with humans.

When we talk about the quality, generally we talk about what product can provide, but in software quality process we even go further to check if the software is reusable, cost of making it and many more. I think the best way to define software quality will be to divide it into 4 aspects: Impression quality, functionality, code quality, and process quality.

Process Quality 🡪🡪🡪🡪 Code Quality 🡪🡪🡪🡪 Functionality

🡪🡪🡪🡪Impression quality

Fig: Quality testing related to software lifecycle

A close up of a sign

Description automatically generated

Fig: Model for Software Quality Testing

I think if we want to test the software of average size before making of any prototype, then the best way to test it will be to go in reverse order. We can go from testing the quality of final product to the testing of process quality in a sequence. The first step we will do or test will be impression quality.

**Impression quality: (**Acceptance, Attractiveness, Compatibility)

This quality aspect will define how the user will feel when they get to use the project. It will measure how much impression a software can make if it launches in the market. I have kept four different metrics under this category, and the description is listed below.

Acceptance: A degree by which the product is accepted in the market. Eg: likes and dislike of users. It can also be an indication of how much product can grow.

Attractiveness: A degree by which it is attractive and unique to the users.

Compatibility: The ability by which the software can manage available resources in the most efficient way. The software shouldn’t take systems large processing power and memory space. It can also go in functional quality in a big project, but for a small project like quality testing for the app, it will be more towards impression.

**Functional Quality:** This aspect explains the robustness of software and finds if the software is functional in a way it should. I have kept a few metrics inside this aspect all of them are listed and described below.

Accuracy: It describes the degree of accuracy a certain software can give under average conditions. A software providing inaccurate result will be limited to a good project which will not be able to compete in the market for more than a moth based on circumstance and product type itself.

Reliability: It is a level of trust that we put on some software. Reliability increases when a system can perform according to the expectation in a certain condition.

Robustness: The degree by which the system can execute an operation without any problem under normal condition. It is related to reliability in a way that it helps in increasing trust towards the system.

Efficiency: The degree by which system can function smoothly respect to the amount of resources and time used.

**Code Quality:**  It basically checks the quality of code that we put on to build a system. It is very hard to check this aspect in comparison to other aspects. I have listed four different metrics for this aspect of software quality inspection.

Testability: The degree by which each part of the code can be tested. The degree of testability highly depends on the organization and understandability of code.

Modifiability: The degree by which the code of a system is modifiable. It also covers the metric maintainability and understandability.

Understandability: It is a degree by which different developer can understand the code written by someone else. It has a lot to do with code organization. A higher degree of organization means higher degree of reusability, modifiability, and testability.

Efficiency: This metric is under code quality and functionality because the code has a lot to do with efficiency and it affects the overall function of the system. And this metric is so important that is worth mentioning in both aspects of quality test.

**Process quality:** This aspect covers the overall process of software development. The quality of software development affects the quality of software we get, and time and resources used by developers. Below are the few metrics that I have listed under it. (David Chappell)

Deliverability: It defines the way if the sub-project is completed on time so the main project can be completed in desired time. It is more like a Boolean (Yes/No) then coming up with the way to deliver the project on time.

Budgeting: It is also like Yes/No question. We are trying to check quality rather than making changes in the development process. It answers if the project is deliverable in the allocated budget.

Time management and group meeting: Since this model is for small group project, we need to make sure if the time has been allocated enough for group meeting as well as for development.

**Rationale:**

For most of the metrics, I have mentioned above that why I chose specifically those. But in general, I choose all those metrics closely related to my aspect of software quality testing. Sometimes, like one in the code quality aspect, I chose the most relevant one which can cover other metrics too.

My model is based on almost all model that I have read, because while designing this model I never thought “I liked this from this model, I will keep these characteristics. I don’t like this, so I will not keep these things.” These things were happening in my subconscious mind, and I can say my design is influenced a lot by Boehm Model and ISO 9126 Model. I was trying to group my metrics so every related metrics can be checked together which can save a lot of time. Also, it helps to make your testing model clear and understandable.

My model is the one design and it can compete with other models that are in the market for small-sized software. My model defines all the metrics useful to check the quality of application-based software. It is very clear and easy to follow. It saves a lot of your time in testing because one can assign a group of people or a person to do quality check in one aspect. Getting specialized in one aspect helps not only saving time but also money.

Cited:

[1].https://www.tutorialspoint.com/software\_quality\_management/software\_quality\_management\_discussion.html

[2]. J. Kaur, “Software Quality Model and Standard.”

[3]. J. Miguel et al., "A review of software quality models for the evaluation of software products," International Journal of Software Engineering & Applications, vol. 5, no. 6, pp. 31-54, Nov. 2014, [Online] Available: http://airccse.org/journal/ijsea/papers/5614ijsea03.pdf

[4]. <http://www.arisa.se/compendium/node6.html>

[5]. D. Chappell, “THE THREE ASPECTS OF SOFTWARE QUALITY: FUNCTIONAL, STRUCTURAL, AND PROCESS.”

[6]. D. Chappell, “THE THREE ASPECTS OF SOFTWARE QUALITY: FUNCTIONAL, STRUCTURAL, AND PROCESS.”

[7]. Gilad, “What Software Quality (Really) Is and the Metrics You Can Use to Measure It.” .