Software Quality

Sukhvir Kaur
Assistant Professor,
CTIEMT, Jalandhar

ABSTRACT

In the given paper I present the information regarding quality metrics. Different quality factors can be imposing with relation to ***cost, schedule and rework****. It's very important to make quality assurance plans. For that you have to cover many milestones which can be represented in* ***time sheets****. The main characteristics of the quality models have covered in this paper.*

Keywords

Quality, Cost, Quality assurance plans, Fault, Reliability, Milestone

1. SOFTWARE QUALITY

The software quality is a planned and systematic set of activities to ensure that quality is built into the software. It consists of software quality assurance, software quality control, assessment and other aspects. According to the IEEE 610.12(IEEE, 1990) standard, software quality is a set of attributes of a software system and is defined as:

- The degree to which software, or process meets customer or user needs or expectations.
- The degree to which software, or process meets specified requirements.
- Quality comprises all characteristics and significant features of a product or an activity which relate to the satisfaction of given requirements.

These standards are in existence for a long time and their relevance might be a little too broad. IEEE standard expresses quality in terms of customer expectation. If a customer's expectation is nil, it doesn't mean that a product with nil characteristics is a quality product.

A high quality product is one which has associated with it a number of quality factors. These could be described in the requirements specification, or they could be quality factors which the developer regards as important but are not considered by the customer and hence not included in the requirements specification.

Quality is the totality of features and characteristics of a product or a service that bears on its ability to satisfy the given needs, for example, conformance to specifications. It is the degree to which a customer or user perceives that software meets his or her composite expectations. The evaluation of quality for a software system depends upon the following:

- Quality Model
- Quality characteristics which may further be classified into several subs - characteristics.
- Metrics to measure the attributes of characteristics and sub - characteristics

In the ISO standard 8402 (ISO, 1994), a software quality model is defined as:

"The set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality"

Software quality models have been proposed to provide many benefits like these can be used as a base to define a commonly agreeable quality framework, which consolidates the different views on quality, they can be tailored to specific contexts, and they provided a measurable base to the evaluation of software quality.

2. QUALITY ATTRIBUTES FOR SOFTWARES

2.1 Functionality

Functionality is a set of attributes that bear on the existence of a set of functions and their specified properties (ISO, 1991). It means that the software should provide the functions and services as per the requirement when used under the specified Condition. Pre-existing software with low cost, faster delivery of end product .The sub-characteristics under functionality are:

- ***Suitability:*** Suitability expresses how well the software fits into the developer's requirements. As means requirement can only be known to system Developer, it cannot be measured by software developer during its development.
- ***Accuracy:*** It evaluates accuracy of the software with correct precision level required by the system developer.
- ***Interoperability:*** This indicates whether the format of the data, handled by the target software is compliant with any international standard.
- ***Security:*** It refers how the software is able to control the unauthorized access to the services provided to it.
- ***Compliance:*** This characteristic indicates if software is conforming to any international standard or certification etc.

2.2 Reliability

In general, reliability is the probability that a system or software will produce failure within a given period of time. In other words, reliability expresses the ability of the software to maintain a specified level of fault tolerance, when used under specified conditions. Reusability aspect of the same software with multiple applications will increase the reliability of that software as it may be observed here that this software would have been thoroughly tested before using it in previous applications. Reliability is broken into the following sub -characteristics:

- ***Maturity:*** In software context, it deals with the number of commercial versions of the software and the time interval between each version.

- ***Recoverability****:* It measures the ability for software to recover from an unexpected failure and also to recover the lost data along with the original performance.

- ***Fault Tolerance:*** This sub-characteristic indicates whether the software ca n maintain a specified level of performance in case of faults.

2.3Usability

It is the ability of software to be understood, learned, used, configured, and executed, when used under specified conditions. Obviously, here the user of the software is application/system developer rather than end user. Therefore the usability of the software should be less complex and more reusable and developer friendly so that it can be assembled properly in the system. Sub characteristics of Usability are defined as under.

- ***Understandability:*** It is the capability of the software to enable the user (system developer) to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.

- ***Learn ability:*** This sub-characteristic refers the ability of the software to enable the system developer to learn the software. For example, the user documentation and the help system should be complete; the help should be context sensitive and explain how to achieve common tasks, etc.

- ***Attractiveness:*** It indicates the capability of the software to be attractive to the user. Here as stated earlier, the user is system developer, and he/she may be more interested in the programmatic interfaces, API's defining the services provided by the software so that they can be composed and integrated with the target system rather than it's a attractiveness or GUI interfaces. Therefore, we may omit this sub-characteristic from the proposed model.

- ***Compliance****:* This characteristic indicates if software is conforming to any international standard or certification etc. Relating to usability. Currently, no standards have been set up for this sub -characteristic in the software context; therefore it may be omitted from the quality model for the time being.

2.4 Efficiency

This characteristic expresses the ability of a software to provide appropriate performance, relative to the amount of resources used. Efficiency is affected by technology, mainly though rough resource usage by the runtime system but also by interaction mechanism. Module can be internally optimized to improve performance without affecting their specifications. Modules should be tested on various platforms to check the performance. The two sub-characteristics of efficiency are defined as follows:

- **Time behavior**: This characteristic indicates the ability to perform a specific task at the correct time, under specified conditions.

- **Resource behavior**: This characteristic indicates the amount of the resources used, under specified conditions.

2.5 Maintainability

It describes the ability of software to be modified. As the developers do not have the source code of the software, they can only adapt it, reconfigure it and finally test it before integrating it in the final product. The sub -characteristics under maintainability are:

- ***Customizability:*** A system developer can only adapt, reconfigure, test and finally embed it into the system, as he is not having the Source code of the software. Customizability refers to the ability to modify the software through its limited available information, like interfaces and Parameters.

- ***Testability:*** It refers whether the software provides some sort of tests or test suites that can be performed to the software to check its functionality inside the final system in which the software will be integrated.

- ***Stability****:* It refers to the software ability to handle the unexpected changes during the maintenance. In other words, it is the degree to which software is composed of discrete software's such that a change to one software has zero impact on the other software's or the system.

- ***Analyzability:*** It refers to the auto-analysis of software. It identifies the parts of the software's, which are to be modified. However, software rarely consists of methods for its auto analysis. Therefore analyzability may not be required and is removed from the model.

2.6 Portability

This characteristic is defined as the ability of software to be transferred from one environment to another with little modification, if required. The software should be easily and quickly portable to new environments if and when necessary, with minimized costs and schedules. The specification of software should be platform independent. Various sub-characteristics defined under portability are:

- ***Replace ability:*** This sub-characteristic indicates whether the software is backward compatible with its previous

versions. This means that the new software can substitute the previous ones without any major efforts.

• ***Adaptability****:* It refers whether the software can be adapted to different specified platforms.

• ***Install ability:*** It is the capability for software to be installed easily on different platforms.

3. Quality Models for Systems

Quality model is a set of characteristics and sub-characteristics, as well as the relationships between them that provide the basis for specifying quality requirements and for evaluating quality of the component or the system. Quality model establishes a framework to perform some kind of measurement of the specific features that are needed in the final system and described by the developer. Measurement of quality attributes is concerned with deriving the numerical values by using the appropriate metrics for that attribute.

There is several quality models proposed so far. The most well known model is McCall Model purposed by McCall and Joseph, in 1978. They presented a software quality framework 17 and classified the quality attributes into three groups namely:

- product operation
- product revision
- Product transition.

McCall and Joseph proposed 11 characteristics under these categories.

Second model Boehm was proposed which in addition to most of the McCall factors; also include hardware characteristics, documentation, and others.

Third model FURPS model decomposed quality into two different categories of requirements,

- Functional requirement
- Non Functional requirement

Functional requirements defined by input and expected output and non-functional requirements like usability, reliability and others.

Third Model Dromey added two more characteristics i.e. reusability and process maturity to propose a new model. It categorized the characteristics into four categories:

- Correctness
- Internal
- contextual
- Descriptive.

ISO proposed a quality standard ISO 9 126 (ISO, 2001) providing a generic definition of software quality, in terms of six main characteristics.

- Functionality
- Reliability
- Performance
- Usability
- Portability
- Maintainability.

3.1 McCall Model

The first quality model was proposed by McCall and Joseph in 1976. They presented a Software Quality Factor Framework and classified the quality attributes into three groups

• ***Product Operation:*** The system's ability to be quickly understood, efficiently operated and capable of providing the results required by the user i.e. involving attributes such as correctness, reliability, efficiency, integrity and usability.

• ***Product Revision:*** It relates to error correction and system adaptation. This aspect is generally considered the costliest part of the software and involves attributes such as maintainability, flexibility and testability.

• ***Product transition:*** It refers to distributed processing, together with rapidly changing hardware and involves attribute such as portability, reusability and interoperability.

Main Factors are:

• Correctness

• Reliability

• Maintainability

• Testability

• Portability

The major advantage of this model is the relationship created between its quality characteristics; however, the main drawback is that it does not include the functionality aspect of the software product. Also, some of the factors and measurable properties like Traceability and self-documentation among others are not really definable or even meaningful at an early stage for non-technical stakeholders. It is, therefore, difficult to use this framework to set precise and specific quality requirements. This model is not applicable with respect to the criteria outlined in the IEEE Standard for a software quality metrics methodology for a top-down approach to quality engineering.

3.2 Boehm's Model

The Boehm model is similar to the McCall model in that it represents a hierarchical structure of characteristics, each of which contributes to total quality and it includes users' needs like McCall's does. Boehm's model looks at utility from various dimensions, considering the types of user expected to work with the system once it is delivered. General utility is broken down into portability, utility and maintainability. Utility is further broken down into reliability, efficiency and human engineering. Maintainability is, in turn, broken down into testability, understandability and modifiability. However, Boehm's

model does not elaborate the methodology to measure these characteristics.

3.3 FURPS Model

FURPS takes into account the five characteristics that make up its name: Functionality, Usability, Reliability, Performance, and Supportability. The model proposed by Robert Grady from Hewlett -Packard Co. decomposes characteristics into two different categories of requirements:

- ***Functional requirements:*** Defined by input and expected output.
- ***Non-functional requirements:*** Usability, Reliability, Performance, and Supportability.

One disadvantage of the FURPS model is that it does not consider the portability aspect, which may be an important criterion for application development, especially for software-based systems.

3.4 Dromey's Model

Dromey proposed a quality evaluation framework that analyzes the quality of software component through the measurement of tangible quality properties. All these software's possess properties can be classified into four categories:

- ***Correctness:*** Evaluates software whether some basic principles are violated.
- ***Internal:*** Measure how well software has been deployed according to its intended use.
- ***Contextual:*** Deals with the external influences by and on the use of software.
- ***Descriptive:*** Measures the descriptiveness of software.

These properties are used to evaluate the quality of the software's. It is difficult to see how it could be used at the beginning of the lifecycle to determine user quality needs.

3.5 ISO 9126 Model

International Organization for Standardization (ISO) proposed a standard, known as ISO 9126 (ISO, 2001), which provides a generic definition of software quality in terms of six main characteristics for software evaluation. These characteristics include: Functionality, Reliability, Usability, Efficiency, Maintainability and Portability. Which are further subdivided into many parts like:

Suitability, Accuracy, Interoperability, Recoverability, Fault-Tolerance, Understandability, Operability, Testability

One of the advantages of this model is that it identifies the internal and external quality characteristics of a software product. The quality models described above (McCall and Joseph, Boehm, Grady, Dromey, and ISO, 2001) contain several factors in common, like maintainability, efficiency, reliability. However, some of the factors like correctness, maturity, and understandability are not so common and are found in one or two models.

4 Quality Model Attributes of Boehm, McCall, FURPS, Dromey and ISO 9126

The main attribute or characteristics of quality models is shown in table 1. Only 'reliability' is common to all quality models.

Table 1 . Quality Attribute of Quality Models

Quality Attribute	McCall	Boehm	Dromey	FURPS	l ISO 9126
Maintainability	×		×		×
Flexibility	×				
Testability	×	×			
Correctness	×				
Efficiency	×	×	×		×
Reliability	×	×	×	×	×
Integrity	×				
Usability	×		×	×	×
Portability	×	×	×		×
Reusability	×		×		
Interoperability	×				
Understandability	×	×			
Functionality	×		×	×	×
Performance	×			×	
Supportability	×			×	

4. Reliability

As reliability is common with all the models the main point to consider in case of reliability of software is faults. There are major groups of approaches to deal with faults:

- Fault avoidance
- Fault Removal
- Fault tolerance

Fault avoidance: To avoid or prevent the introduction of faults by engaging various design methodologies, techniques and technologies, including structured programming, object-oriented programming, software reuse, de-sign patterns and formal methods.

Fault removal. To detect and eliminate software faults by techniques such as reviews, inspection, testing, verification and validation.

Fault tolerance. To provide a service complying with the specification in spite of

5 Conclusions:

In the following paper I had proposed different models which give bench mark to quality. Different models include different factors in relation to functional and non functional requirements but reliability is a important factor in measuring quality of Software. It's very important to have very good software specification requirements to have best results. It should be noticed that phases like planning, requirement, coding implementation should be done in proper manner to have quality.

6 References:

[1] Deepak Gupta, Vinay Kr.Goyal and Harish Mittal "Comparative Study of Soft Computing Techniques for Software Quality Model" *International Journal of Software Engineering Research & Practices Vol.1, Issue 1, Jan, 2011*

[2] Jie Xu, Danny Ho and Luiz Fernando Capretz, An Empirical Study On The Procedure To Derive Software Quality Estimation Models, *International Journal of computer science & information Technology (IJCSIT) Vol.2, No.4, August 2010*

[3] Rafa E. Al-Qutaish," Quality Models in Software Engineering Literature: An Analytical and Comparative Study", 2010

[4] Ranbireshwar S. Jamwal & Deepshikha Jamwal," Issues & Factors For Evaluation of Software Quality Models ", *Proceedings of the 3rd National Conference, INDIACom-2009*

[5] Jean-Louis Letouzey, Thierry Coq, "An analysis model compliant with the representation condition for assessing the Quality of Software Source Code" 2010

[6] http://www.jot.fm/issues/issue_2010_07/article4.pdf

[7] http://www.4shared.com/document/F0iqxJIV/Requirements_of_Software_Quali.html

[8] http://people.uncw.edu/simmondsd/courses/Fall2011 /csc 592 /papers/qualityModels.pdf

[9] http://www.sqa.net/iso9126.html

[10] http//www.jot.fm/issues/issue_2002_09/Article4