Comparative Analysis of Different Software Quality Models

Ranbireshwar S. Jamwal, Deepshikha Jamwal & Devanand Padha
Jamwal.grandee@gmail.com, Jamwal.shivani@gmail.com, dpadha@rediffmail.com
Lecturer, Research Scholar, H.O.D

ABSTRACT
Software Quality is increasingly important in today's marketplace. An organization’s focus on the strategic importance of software quality depends on whether they are producers or users of software. Software users see software as a tool to be used to support them in the way they do business in their specific sector.

Quality is a composition of many characteristics. Because of that, quality is usually captured in a model that depicts the characteristics and their relationships. The models are useful; they show what people think is important when speaking about quality. Different organizations use different quality models based upon the requirements. Different concepts of software quality characteristics are reviewed and discussed in this paper. Also comparative analysis of various software quality models used by various organisations is being discussed in this paper.

KEY WORDS
Software Quality, software Quality Models.

1. INTRODUCTION
Software quality is conformance to explicitly state functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.”

Some organisations try to develop standard definitions for quality. We now present some definitions of international and standard organisations:

• ISO 9126: Software quality characteristic is a set of attributes of a software product by which its quality is described and evaluated”.

• German Industry Standard DIN 55350 Part 11: Quality comprises all Characteristics and significant features of a product or an activity which relate to the satisfying of given requirements”.

• ANSI Standard (ANSI/ASQC A3/1978): Quality is the totality of features and characteristics of a product or a service that bears on its ability to satisfy the given needs.

• IEEE Standards (IEEE Std 729-1983): The totality of features and characteristics of a software product that bear on its ability to satisfy given needs. For example, conformance to specifications. The degree to which software possesses a desired combination of attributes. The degree to which a customer or a user perceives that software meets her composite expectations[1].

1.1 KEY DIMENSIONS TO SOFTWARE QUALITY

• Level of satisfaction. The degree to which customers or users perceive that a software product meets their composite needs and expectations.

• Product value. The degree to which a software product has value for its various stakeholders relative to the competitive environment.

• Key attributes. The degree to which a software product possesses a combination of desired properties, e.g., reliability, portability, maintainability[2].

• Defectiveness. The degree to which a software product works incorrectly in target user environments due to debilitating operational defects.

• Process quality. In relation to the development process by which the product is produced, it means good people doing the right things in an effective way.

1.2 MEASUREMENT OF SOFTWARE QUALITY
1. The aim of software quality engineering is to investigate the relationships among in-process metrics, project characteristics, and end-product quality.
2. In general, software quality metrics are more closely associated with process and product metrics[3].

1.3 QUALITY METRIC
Quality metric provides a numerical value that can be scaled to measure a quality factor. Metrics must be complete and detailed sufficiently to be the firm foundation of a quality model. There is a strange relationship between internal and external quality. External quality is quality as measured by the customer. Internal quality is quality as measured by the programmers [4].

2. SOFTWARE QUALITY MODELS
Quality is a composition of many characteristics. Because of that, quality is usually captured in a model that depicts the characteristics and their relationships. The models are useful; they show what people think is important when speaking about quality. There are a number of software quality models described below [5].
A. THE ISO 9126 STANDARD QUALITY MODEL
Having a single universal model makes it easier to compare one product with another. The ISO 9126 quality model was proposed as an international standard for software quality measurement in 1992. It is a derivation of the McCall model. ISO 9126 defines 21 attributes that a quality software product must exhibit (Figure 1). The 21 attributes are arranged in six areas: functionality, reliability, usability, efficiency, maintainability and portability. Although we have these attributes, measuring quality still is not easy. Software quality measurement techniques allow us to measure some of the attributes (in bold in Figure 5). But still, many of the attributes seem to be very hard to measure in a straightforward way. ISO 9126 is the most common used quality standard model. There are several others though, such as IEEE 1061[6].

The ISO 9126-1 software quality model identifies 6 main quality Characteristics, namely:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

B. McCall Software Quality Model.
One of the more renowned predecessors of today’s quality models is the quality model presented by Jim McCall et al. [9-11] (also known as the General Electrics Model of 1977). The McCall Model aimed at system developers. It is used during development process. It identifies 3 areas of software work: -

- **Product Operation** – refers to the product’s ability to be quickly understood, efficiently operated and capable of providing the results required by the user.
- **Production Revision** - is related to error correction and system adaptation.
- **Product Transition** - distributed processing, rapid change in hardware [7].

C. Boehm Software Quality Model
The second of the basic and founding predecessors of today’s quality models is the quality model presented by Barry W. Boehm. Boehm addresses the contemporary shortcomings of models that automatically and quantitatively evaluate the quality of software. The intermediate level characteristic represents Boehm’s 7 quality factors that together represent the qualities expected from a software system:

- **Portability (General utility characteristics):** Code possesses the characteristic portability to the extent that it can be operated easily and well on computer configurations other than its current one.

- **Reliability (As-is utility characteristics):** Code possesses the characteristic reliability to the extent that it can be expected to perform its intended functions satisfactorily.
- **Efficiency (As-is utility characteristics):** Code possesses the characteristic efficiency to the extent that it fulfills its purpose without waste of resources.
- **Usability (As-is utility characteristics, Human Engineering):** Code possesses the characteristic usability to the extent that it is reliable, efficient and human-engineered.
- **Testability (Maintainability characteristics):** Code possesses the characteristic testability to the extent that it facilitates the establishment of verification criteria and supports evaluation of its performance.
- **Understandability (Maintainability characteristics):** Code possesses the characteristic understandability to the extent that its purpose is clear to the inspector.
- **Flexibility (Maintainability characteristics, Modifiability):** Code possesses the characteristic modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined [8].

D. Dromey’s Quality Model
Dromey has built a quality evaluation framework that analyzes the quality of software components through the measurement of tangible quality properties. Each artefact produced in the software lifecycle can be associated with a quality evaluation model. Dromey gives the following examples of what he means by software components for each of the different models:

- Variables, functions, statements, etc. can be considered components of the Implementation model;
- A requirement can be considered a component of the requirements model;
- A module can be considered a component of the design model;

According to Dromey (1995), these components all possess intrinsic properties that can be classified into four categories:

- **Correctness:** Evaluates if some basic principles are violated.
- **Internal:** Measure how well a component has been deployed according to its intended use.
- **Contextual:** Deals with the external influences by and on the use of a component.
- **Descriptive:** Measure the descriptiveness of a component (for example, does it have a meaningful name).

Dromey proposes a product based quality model that recognizes that quality evaluation differs for each product and that a more dynamic idea for modelling the process is needed to be wide enough to apply for different systems. Dromey is focusing on the relationship between the quality attributes and the sub-attributes, as well as attempting to connect software product properties with software quality attributes.
Implementation, Correctness Internal Contextual Descriptive, Functionality, reliability Maintainability, efficiency, reliability, Maintainability, reusability, portability, reliability, Maintainability, reusability, portability, usability.

Dromey’s Quality Model is further structured around a 5-step process:
1) Chose a set of high-level quality attributes necessary for the evaluation.
2) List components/modules in your system.
3) Identify quality-carrying properties for the components/modules (qualities of the component that have the most impact on the product properties from the list above).
4) Determine how each property affects the quality attributes.
5) Evaluate the model and identify weaknesses [9].

E. Capability Maturity Model:

The Carnegie Mellon Software Engineering Institute (SEI), non-profit group sponsored by the DoD work at getting US software more reliable. Examples of relevant material produces from SEI are the PSP and TSPi. While PSP and TSPi briefly brushes the topic of this technical paper, SEI has also produced a number of more extensive Capability Maturity Models that in a very IEEE and ISO 9000 similar manner addresses the topic of software quality:

CMM / SW-CMM
P-CMM [32]
CMMI [33]
- PDD-CMM
- SE-CMM
- SA-CMM

The limitation of current models:

- Two categories cannot fully reflect the characteristics (human, time, equipment, etc) are limited; some of fault-prone modules should be tested with higher priority.

F. The FURPS Model

A later, and perhaps somewhat less renown, model that is structured in basically the same manner as the previous two quality models (but still worth at least to be mentioned in this context) is the FURPS model originally presented by Robert Grady (and extended by Rational Software - now IBM Rational Software - intoFURPS+3). FURPS stands for:
- Functionality – which may include feature sets, capabilities and security
- Usability - which may include human factors, aesthetics, consistency in the user interface, online and context sensitive help, wizards and agents, user documentation, and training materials.
- Reliability - which may include frequency and severity of failure, recoverability, predictability, accuracy, and mean time between failure (MTBF).
- Performance - imposes conditions on functional requirements such as speed, efficiency, availability, accuracy, throughput, response time, recovery time, and resource usage
- Supportability - which may include testability, extensibility, adaptability, maintainability, compatibility, configurability, serviceability, install ability, localization (internationalization).
The FURPS-categories are of two different types: Functional and Non-functional.
- Functional Requirements (F): Defined by input and expected output.
- Non-functional Requirements (NF): Usability, Reliability, Performance, and Supportability [10].

3. ANALYSIS

In order to achieve the objective that is the comparison of different quality models, the authors have collected data from various organizations using both primary and secondary research methods. In the Primary method the authors have used the questionnaire .In addition to the questionnaire, the students were interviewed personally. Questionnaire to 25 organizations (Govt / Semi-Govt. and software companies) have been circulated and feedback was received. The questionnaire contains various software quality characteristics like correctness, maintainability, testability, Integrity, Reliability, Usability, Understandability, and Interoperability etc. and their suitability is checked for various software quality models. Secondary methods included journals, Research papers, Articles, Periodicals etc. On the basis of the survey analysis is done using different analysis tools and analysis is done, which is shown below in the form of tables.


Table 1: Comparison between criteria/goals of the McCall, Boehm and ISO 9126 quality models.

<table>
<thead>
<tr>
<th>Criteria/Goals</th>
<th>McCall Model</th>
<th>Boehm Model</th>
<th>ISO 9126, 1993</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>X</td>
<td>X</td>
<td>Maintainability</td>
</tr>
<tr>
<td>Integrity</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Maintainability</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Testability</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Interoperability</td>
<td>X</td>
<td>X</td>
<td>Maintainability</td>
</tr>
<tr>
<td>Flexibility</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Reusability</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Clarity</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Modifiability</td>
<td>X</td>
<td>X</td>
<td>Maintainability</td>
</tr>
<tr>
<td>Documentation</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Resilience</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Understandability</td>
<td>X</td>
<td>X</td>
<td>Maintainability</td>
</tr>
<tr>
<td>Validity</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Generality</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Economy</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Comparison between Different Software Quality Models.

<table>
<thead>
<tr>
<th>Quality Characteristic</th>
<th>Boehm</th>
<th>McCall</th>
<th>FURPS</th>
<th>ISO 9126</th>
<th>Dromey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testability</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Correctness</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Understandability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Flexibility</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Human Engineering</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integrity</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Interoperability</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Maturity</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Changeability</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portability</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Reusability</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

4. CONCLUSION

Software quality engineering needs a quality model that is usable throughout the software lifecycle and that it embraces all the perspectives of quality model suitable for such a purpose, through the comparative evaluation of existing quality models and their respective support Quality engineering. From the research it is observed that the:

- Users play a role in the measurement of Software Quality.
- Users have a direct and equal impact on the software quality.

- Identifies the requirements, which needs improvement
- Requires a good amount of time for the evaluation.
- Can be used to improve the quality of later versions of the software.

5. FUTURE WORK

- To measure the effectiveness of our metric by using realistic data.
- In depth analysis of the various software quality models using more exhaustive questionnaire.
- To distribute the questionnaire among more multinational companies and study the various aspects of software quality involved.
- To compare the efficiency of this metric with the existing Software Quality Metrics.

REFERENCES