Comparative Study of Software Quality Models

Saniu Abdul Sattar, Nimbal Pandita, Prateek, Dr. Parul Gandhi

1, 2, 3 Student, FCA, Manav Rachna International University, Faridabad, Haryana, India
4 Associate Professor, FCA, Manav Rachna International University, Faridabad, Haryana, India

Abstract—Quality that is identified independent of any measurable characteristic is defined as the capability to meet the customer expectations, whether the requirements are explicit or not. A recent innovation in the Systems Area is the development of a set of mechanisms and models for evaluating quality. This article compares the design of a Quality Model with its factors and criteria. Different quality models were studied: McCall, Boehm, FURPS, Dromey, and GEQUAMO in an attempt to identify the aspects present in these models that are deemed important in Software Quality. GEQUAMO software quality model is the best among the models compared, since it can be customizable as per requirement in a dynamic and a flexible way.

Keywords—Software Quality, Software Quality Models, GEQUAMO, Generic Quality Model

I. INTRODUCTION

Our goal of software engineering is to create exclusive quality and endurable software with in time and budget. In order to achieve such a milestone quality is a crucial factor. Software quality focuses on three key aspects; meeting the user’s requirements, the software production process, and the software as a product itself [1]. Character that is diagnosed independent of any assessable feature is defined as the potentiality to meet customer needs, whether the requirements are explicit or not. The ISO and IEEE define software quality as the entirety of functionality and feature of a software product that weight on its capability to satisfy quantified or signified needs [2]. Even though we understand quality, appreciate quality, quality itself is perspective oriented. Thus making it very difficult to truly understand what quality is specifically. To understand how and what to deal with software quality, there are four attribute domains [3].

The Quality Attributes include:

A. Reliability
B. Usability
C. Maintainability
D. Adaptability

A. Reliability

This domain can be described as the extent by which software perform its intended functions without failure. The attributes of this domain are:

- Correctness – The term to which the software meets its specification.
- Consistency & Precision – The term to which software is consistent and give results with precision.
- Robustness – The extent to which software tolerates the unexpected problems.
- Simplicity – The term to which software is simple in its operations.
- Traceability – The extent to which an error is traceable in order to fix it.

Accurate estimation depends on two factors: Component Reliability and Glue Code Reliability [16]. Still much more research is expected to estimate reliability in a better way

B. Usability

This domain can be defined as the extent of efforts required to learn, understand and operate the functions of the software. The attributes of this domain are:

- Accuracy – Meeting specifications with precision.
- Clarity & Accuracy of Documentation – The extent to which documents are clearly and accurately written.
- Conformity of Operational Environment – The extent to which software is inconformity of the operational environment.
- Completeness – The extent to which software has specified functions.
- Efficiency – The amount of calculating resources and codes need by software to perform operation.
Testability – The effort needed to test software for ensure that it execute its intended operation.

C. Maintainability
This domain can be defined as the effort needed to locate and fix an error in the middle of maintenance phase.

The attributes of this domain are:

- **Clarity & Accuracy of Documentation** – The extent to which documents are clearly and accurately written.
- **Modularity** – The extent of ease to implement, test, debug and maintain the software.
- **Readability** – The extent to which software is readable in order to understand.
- **Simplicity** – The extent to which software is simple in its operations.

D. Adaptability
This domain can be defined as the extent to which software is adaptable to new platforms & technologies.

- **Modifiability** – The effort required to modify software during maintenance phase.
- **Expandability** – The extent to which software is expandable without undesirable side effects.
- **Portability** – The efforts needed to transfer a program from a platform to another platform.

These domain attributes and its attributes are entrusted by the customer and most commonly understood by the software community. And these become a basis for any software quality model built to satisfy as the factors and criteria for sound software quality.

II. SOFTWARE QUALITY MODEL

A software quality model assesses quality of a software by measuring its attributes. Metrics are used to obtain objective measurements in numbers. Product-view is assumed on the influences of the internal attributes over the external attributes. Various software quality models are used in sequence to assess usual and exclusive software quality product, they were processed based on well-known software quality models, namely McCall, Boehm, FURPS, Dromey, and GEQUAMO [4-5]. The comparisons of these models are operated based on different characters and sightedness, and at the components level, to further understand how the attributes of quality were defined and applied to the software as a product and as well as a process.

1) McCall’s Software Quality Model
McCall’s Software Quality Model evaluates and pinpoints the property of a software product by highlighting 3 working areas in system [6]:

- **Product Operation** - The product’s capability to operate effectively and efficient at providing the outputs expected by the consumer. Product operation criteria covers correctness, reliability, efficiency, integrity and usability quality factors.
- **Product Revision** - The capability to elapse modifications to the system, from tasks like testing and system maintenance. Product revision criteria covers maintainability, flexibility and testability quality factors.
- **Product Transition** - The ability to adapt to new environments, when undergoing platform changes and transfer onto a different system. Product transition criteria covers portability, reusability and interoperability quality factors.

![Fig 2.1 Software Quality Factors (McCall’s Software Quality Model)](image-url)
Since the model is hierarchical and distinguishes between high level quality attributes known as quality factors and second level quality attributes known as quality criteria. And since these quality factors and quality criteria can be measured either directly, subjectively or objectively; there are no factors related to architectural integrity or analysability included explicitly. Therefore, this model is introduced for common applications system.

2) Boehm’s Software Quality Model

The Boehm’s Software Quality Model was introduced in 1978 [7-8]. This model evaluates software quality automatically and quantitatively. Similar to its predecessor the McCall’s Software Quality Model, this model also represents a hierarchical structure of characteristics each which contributes to total quality. This model defines three levels of quality attributes:

- Primary Uses
- Intermediate Constructs (Similar to McCall’s SQM Quality factors)
- Primitive Constructs (Similar to McCall’s SQM Quality criteria)

Boehm’s Software Quality Model emphasis successful software should include the needs and expectations of the users, similar to McCall’s Software Quality Model. Apart from the McCall’s Software Quality Model, it also consist of features of hardware execution that are missing from the McCall’s Software Quality Model [9].

![Diagram of Boehm's Software Quality Model](image)

Fig 2.2 The Boehm’s Software Quality Model

3) FURPS Software Quality Model

The features that are taken noticed in FURPS models have been described in many studies [10-11]. It includes:

- Functionality – Factors cover like; feature sets, capabilities and security.
- **Usability** – Factors cover like; human factors, steadiness in the appropriator interface, online and context-sensitive help, wizards, user documentation, and training materials.
- **Reliability** - Covers frequency and severity of failure, recoverability, predictability, accuracy, and mean time between failures.
- **Performance** - prescribes conditions on functional requirements such as speed, efficiency, availability, accuracy, throughput, response time, recovery time, and resource usage.
- **Supportability** - includes testability, extensibility, adaptability, maintainability, compatibility, Configurability, serviceability and localizability /internationalization.

This software quality model considers only the user’s requirements and disregards the developer’s consideration. Setting priorities are important given the implicit trade-off, i.e. one quality characteristic can be obtained at the expense of another [12]. Hence, the model fails to take into account some of the software product characteristics, such as portability and maintainability.

4) Dromey’s Software Quality Model

Dromey’s Software Quality Model states that the evaluation is different for each product, hence a dynamic idea for process modeling is required [13-14]. Therefore, the main idea of the proposed model was to obtain a model broad enough to work for different systems. The model seeks to increase understanding of the relationship between the attributes (characteristics) and the sub-attributes (sub-characteristics) of quality. This model defined two layers, high-level attributes and subordinate attributes. Therefore, this model suffers from lack of criteria for measurement of software quality [15].

5) GEQUAMO

The software quality model GEQUAMO (Generic, Multilayered and customizable model) was created by E. Georgiadou and consists of the gradual breakdown into sub layers of features and characteristics and is intended to encapsulate the various user requirements in a dynamic and flexible way. In this form the user (end user, developer, and manager) can build their own model reflecting the emphasis (weight) for each attribute and/or requirement.

The GEQUAMO—generic and hence customizable, multilayered software quality model has been presented together with the profiling/visualization techniques Composite Features Diagram(CFD) and Kiviat diagrams for qualitative and quantitative representation. It was demonstrated that developers, sponsors and users have synergistic but often conflicting requirements [5].

### III. COMPARISON

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>McCall</th>
<th>Boehm</th>
<th>FURPS</th>
<th>Dromey</th>
<th>GEQUAMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Adaptability</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Analysability</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Attractiveness</td>
<td></td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Changeability</td>
<td>☑</td>
<td></td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Correctness</td>
<td>☑</td>
<td></td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Efficiency</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Flexibility</td>
<td>☑</td>
<td></td>
<td></td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Functionality</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Human engineering</td>
<td>☑</td>
<td></td>
<td></td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Installability</td>
<td></td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Interoperability</td>
<td>☑</td>
<td></td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Maintainability</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Maturity</td>
<td></td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Modifiability</td>
<td></td>
<td></td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Portability</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
<tr>
<td>Reliability</td>
<td>☑</td>
<td>☑</td>
<td>☑</td>
<td></td>
<td>☑</td>
</tr>
</tbody>
</table>
IV. CONCLUSIONS

Based on the discussion of the five quality models and on the comparison between them, the following points could be elaborated about them.

- The McCall’s software quality model is proposed for general application systems, because the quality is subjectively measured based on the requirements of the user.
- The Boehm’s software quality model is similar to the McCall’s software quality model in that it represents a hierarchical structure of characteristics, each of which contributes to total quality and subjectivity towards the requirements of the user. But the Boehm’s software quality model also adds the hardware yield characteristics not encountered in the McCall software quality model. Hence making it a bit better than its predecessor.
- The major two steps that are considered in FURPS software quality model are: setting priorities and defining quality attributes that can be measured. One drawback of this model is that it fails to take account of the software product’s portability.
- Dromey’s software quality model proposes three models, depending on the products resulting from each stage of the development process: requirements model, design model, and implementation quality model. It is important to point out the weighting Dromey’s software quality model gives to Process Maturity, an aspect not considered in previous models.
- The GEQUAMO software quality model is a unique multilayered model where the model can be customizable as per requirement in a dynamic and a flexible way with the help of CASE tools and Kiviat diagrams for qualitative and quantitative representation. Hence, making this model best fit at situations where there are conflicting requirements between the developer and the user.

ACKNOWLEDGMENT

Time, effort and hard work bring out good results. It was an honour to work on this paper together with my colleagues and I am grateful for Dr. Parul Gandhi and Dr. Prasenjit Banerjee for the assistance and the relentless support provided to us through out the course of the preparation of this paper.

REFERENCES