Software Quality Models: A Comparative Study

J. Ghayathri, E. Mohana Priya

Abstract- Software testing requires the use of a model for test selection and test verification. Model-based testing is very effective for software design and development. Model-based testing has a lot of benefits, test suite growth with the product, manage the model instead of cases, automation is cheaper and more effective, one implementation per model, then all cases are free. Model-based testing is a general term that signifies an approach that bases common testing tasks such as test case generation and test result evaluation on a model of application under test. The fundamental tasks for Model-based testing including gathering the necessary information to build a model, the steps in building a model, and generating and evaluating tests from a model. There are a number of models, but few models are good for testing. By using some models such as McCall, Boehm, FURPS, Dromey and ISO 9126 the qualities are tested. Each model has its own quality characteristics along with the factors and subfactors. One of the great advantages of model-based development is the opportunity to start simulation and testing activities. This allows the developer to focus on specific errors at each development stage. The benefit of quality models is that they are simpler to use and the benefit of quality management is that they probably more to the point capture the idea of quality. Software quality model is the representation of the characteristics of the software that describe the quality. This paper discusses the features of the models McCall, Boehm, FURPS, Dromey and ISO 9126. By using these models the quality characteristics are compared.

Index Terms- Quality Models; Quality Assurance; Quality Characteristics; Quality Criteria; Quality factors; Metrics and Quality Management

I. INTRODUCTION

The main goal of quality model is to achieve quality (more quality factors) with cost and performance as prime consideration. In technical terms quality is Conformance to specification. Meeting customer needs, Fitness for use [11]. There are numerous such models, and each describes different aspects of software behavior [1]. Then generation by model checking and generation by using an event-flow model are the techniques used in Model-based testing [2].

Conformance to specification: Quality that is defined as a matter of products and services whose measurable characteristic satisfy a fixed specification – that is, conformance to an in beforehand defined specification.

Meeting customer needs: Quality that is identified independent of any measurable characteristics. That is, quality is defined as the products or services capability to meet customer expectations – explicit or not. The definitions of five different approaches (views) are gained for the product qualities are as follows:

- The transcendent view that defines quality as something that can be recognized but not defined.
- The product view that defines quality by specifying the product characteristics that contributes to quality.
- The user view, that regards quality as subjective attribute that is “in the eye of the beholder:.
- The manufacturing view, that quality as conformance to specification.
- The value-based view that regards quality as inherently connected to the costs to achieve it.

There are many models in Quality Assurance (QA) such as McCall’s, Boehm, FURPS, Dromey’s and ISO quality model. The quality of the implementation model largely determines the quality that is correctness and efficiency [3]. Each model gave number of factors that can help to give us.
high quality of software. An important factor in Software Quality Assurance (SQA) and other are explaining relationships between some factors of SQA such as reusability, maintainability, usability and etc. And then also there is relationship between SQA factors and attributes of object oriented [5]. Quality assurance make sure the product will be completed based on the approved specifications, standards and functionality [9]. The testing of software is important for assessing the software to determine its quality then it consumes more effort for systems that require higher levels of reliability [10].

The software quality assurance issue brings four key subjects:

- How much of the software development cost to invest in software testing.
- Communication to the user of the software.
- Defects detected in the package and released into the “production” environment.
- Consequences of defects beyond the intended use of software application [6].

Quality assurance tasks, such as testing, profiling, performance evaluation, have been done in developers and generated field [7].

II. QUALITY MODELS

A. McCall’s Quality Model (1977)

One of the more renowned predecessors of today’s quality models is the quality model presented by Jim McCall et al. In quality model McCall attempts to bridge the gap between users and developers by focusing on a number of software quality factors that reflect both the users’ views and the developers’ priorities. This model is constructed using tree-like fashion. In this model quality factors are not directly measured and set of metrics is needed to develop relationship [4,8]. Then three major perspectives for defining and identifying the quality of a software product: product revision (ability to undergo changes), product transition (adaptability to new environments) and product operations (its operation characteristics). Product revision includes maintainability to locate and fix a fault in the program within its operating environment, flexibility is the making changes required by changes in the operating environment and testability is to ensure that it is error-free and meets its specification. Product transition is all about portability to transfer a program from one environment to another, reusability is reusing the software in a different context and interoperability to couple the system to another system. Quality of product operations depends on correctness which a program fulfils its specification, reliability is the system ability not to fail, efficiency categorized into execution efficiency and storage efficiency and generally meaning the use of resources, integrity is the protection of the program from unauthorized access and usability is the ease of the software. The difficulty in applying this model is no consensus about what high level quality factors are important [15].
Fig. 2: Three types of quality characteristics in a hierarchy of factors, criteria and metric

Fig. 3: McCall’s Quality Model

The quality factors describe different types of system behavioral characteristics, and the quality criterions are attributes to one or more of the quality factors. The problem in this model is no standards, no methods and no tools to measure these quality factors.

B. Boehm’s Quality Model (1978)

The Boehm’s model attempts to qualitatively define software quality by a given set of attributes and metrics. Boehm's model is similar to the McCall Quality Model in that it also presents a hierarchical quality model structured around high-level characteristics, intermediate level characteristics, primitive characteristics - each of which contributes to the overall quality level. The high-level characteristics address three main questions that a buyer of software has:
• As-is utility: How well (easily, reliably, efficiently) can I use it as-is?
• Maintainability: How easy is it to understand, modify and retest?
• Portability: Can I still use it if I change my environment?

The intermediate level characteristic represents Boehm’s 7 quality factors that together represent the qualities expected from a software system:

**TABLE I**

<table>
<thead>
<tr>
<th>Quality Factors</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portability</td>
<td>General Utility Characteristics</td>
</tr>
<tr>
<td>Reliability</td>
<td>As-is Utility Characteristics</td>
</tr>
<tr>
<td>Efficiency</td>
<td>As-is Utility Characteristics</td>
</tr>
<tr>
<td>Usability</td>
<td>As-is Utility Characteristics</td>
</tr>
<tr>
<td>Testability</td>
<td>Maintainability Characteristics</td>
</tr>
<tr>
<td>Understandability</td>
<td>Maintainability Characteristics</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Maintainability Characteristics</td>
</tr>
</tbody>
</table>

The primitive characteristics provide the foundation for defining qualities metrics – which was one of the goals when Boehm constructed his quality model. Consequently, the model presents one or more metrics supposedly measuring a given primitive characteristic.

Though Boehm’s and McCall’s models might appear very similar, the difference is that McCall’s model primarily focuses on the precise measurement of the high-level characteristics “As-is utility”, whereas Boehm’s quality model is based on a wider range of characteristics with an extended and detailed focus on primarily maintainability. The Boehm’s quality model is very difficult to apply in practice, but easy to understand and learn [16].

**Fig.4: Boehm’s Software Quality Characteristics Tree**
C. FURPS/FURPS+

The FURPS model that is structured in basically the same manner as the previous two quality models. The FURPS model originally presented by Robert Grady.

FURPS stands for:
- Functionality – which may include feature sets, capabilities and security
- Usability - which may include human factors, aesthetics, consistency in the user interface, 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.
- 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, installability, localizability (internationalization)

The FURPS-categories are of two different types: Functional (F) and Non-functional (URPS). These categories can be used as both product requirements as well as in the assessment of product quality.

D. Dromey’s Quality Model

The Dromey Quality model similar to the McCall’s, Boehm’s and the FURPS quality model, is the quality model presented by R. Geoff Dromey. 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.

As Figure 5 illustrates, there are three principal elements to Dromey’s generic quality model
- Product properties that influence quality
- High level quality attributes
- Means of linking the product properties with the quality attributes.

E. ISO

1. ISO 9000

The renowned ISO acronym stands for International Organization for Standardization.

ISO 9001 is a process oriented approach towards quality management. That is, it proposes designing, documenting, implementing, supporting, monitoring, controlling and improving (more or less) each of the following processes:
2. ISO 9126

ISO has also released the ISO 9126: Software Product Evaluation: Quality Characteristics and Guidelines.

Are the required functions available in the software?

How reliable is the software?

How easy is to transfer the software to another environment?

How easy is to modify the software?

How efficient is the software?

Fig. 7: The ISO 9126 quality model

This standard was based on the McCall and Boehm models. ISO 9126 also includes functionality as a parameter, as well as identifying both internal and external quality characteristics of software products. ISO 9126 proposes a standard which species six areas of importance, i.e. quality factors, for software evaluation.

The problem with this model is there is no consensus regarding what is a top-level quality-factor and what is more concrete quality criterion [17]. Each quality factors and its corresponding sub-factors are defined as follows:

- **Functionality**: A set of attributes that relate to the existence of a set of functions and their specified properties.

- **Reliability**: Capability of software to maintain its level of performance under stated conditions for a stated period of time.
  - Maturity, Fault tolerance, Recoverability.

- **Usability**: The effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.
Subfactors

- Understandability, Learnability, Operability.
- **Efficiency**: The relationship between the level of performance of the software and the amount of resources used, under stated conditions.
  - Time behavior, Resource behavior.
- **Maintainability**: The effort needed to make specified modifications.
  - Analyzability, Changeability, Stability, Testability.
- **Portability**: Ability of software to be transferred from one environment to another.
  - Adaptability, Installability, Conformance, Replaceability.

III. COMPARATIVE ANALYSIS

From the comparison table each model has its own characteristics. The McCall’s model supports testability, correctness, efficiency, reliability, flexibility, integrity, interoperability, maintainability, usability and portability and it lacks the characteristics such as understandability, functionality, human engineering, process maturity, changeability, clarity, modifiability, documentation, resilience, validity, generality and economy. Boehm’s model provides the characteristics are testability, efficiency, understandability, reliability, human engineering, maintainability, changeability, portability, usability, clarity,
TABLE II
COMPARISON BETWEEN CRITERIA/GOALS OF THE BOEHM, MCCALL, FURPS, ISO 9126 AND DROMEY QUALITY MODELS

<table>
<thead>
<tr>
<th>Quality Characteristic</th>
<th>McCull</th>
<th>Boehm</th>
<th>FURPS</th>
<th>Drome</th>
<th>ISO 9126</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testability</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>Understandability</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>
<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>Integrity</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>Process Maturity</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>Changeability</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>Reusability</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Clarity</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>Documentation</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Resilience</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Validity</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Generality</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Economy</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

modifiability, documentation, resilience, validity, generality and economy and it unable to provide correctness, flexibility, functionality, integrity, interoperability, reusability and process maturity. The FURPS model has the qualities are efficiency, understandability, reliability, flexibility, functionality, maintainability, usability and documentation then it fails to support testability, correctness, human engineering, integrity, interoperability, process maturity, changeability, portability, clarity, documentation, resilience, validity, generality and economy. Dromey’s quality model has the abilities such as efficiency, understandability, reliability, functionality, process maturity, maintainability and portability and it unable to support the qualities are testability, correctness, flexibility, human engineering, integrity, interoperability, usability, clarity, modifiability, documentation, resilience, validity, generality and economy and changeability. ISO 9126 model has the possibilities such as testability, efficiency, understandability, reliability, functionality, integrity, interoperability, maintainability, usability, modifiability and portability and it lacks the qualities such as correctness, flexibility, human engineering, process maturity, changeability, clarity, documentation, resilience, validity, generality and economy.

IV. FURTHER STUDY

The Management of software quality has also evolved, it is directed not only toward the development of software that executes correctly, but also toward satisfying the objectives for which that software is designed. The management of software quality requires a set of quality criteria that the software product must meet, including related software measures to evaluate this quality. The three models of software product related to the three views of quality are an internal quality model, external quality model and quality—in-use model [12]. An internal metrics measure the software itself, external metrics measure the behaviour of the computer-based system that includes the software, and quality in use metrics measure the effects of using the software in a specific context of use. Appropriate internal attributes of the software are prerequisites for achieving the required external
V. CONCLUSION

Most of the quality models presented within this paper probably could be fitted within the user view, manufacturing view or product view. Some advantages are the quality models actually reduce the notion of quality to a few relatively simple and static attributes. The benefit of quality model is that they are simpler to use. Then the software quality models are the process of testing the quality with several factors and sub factors along with metrics. The development of efficient software quality model that will assist in the creation of high-quality software will become one of the most important research areas in the near future.

REFERENCES

[7] Arvind S. Krishna, Douglas C. Schmidt, Electrical Engineering and Computer Science, Vanderbilt University, TN USA, Atif Memon, Adam Porter, Computer Science Department, University of Maryland College Park, MD USA. Diego Sevilla, Department of Computer Engineering, University of Murcia, Spain. “Validating Quality of Service for Reusable Software via Model-integrated Distributed Continuous Quality Assurance”. National Science Foundation under Grant Nos. NSF ITR CCR-0312859, CCR-0205265, CCR 0098158, 2004


J. Ghayathiri is currently working as Associate Professor in Computer Science (PG), Kongu Arts and Science College, Erode, TamilNadu, India. PH: 9842379818, E-mail:

E. Mohana Priya is currently pursuing Master of Philosophy in Computer Science, Kongu Arts and Science College, Erode, TamilNadu, India. PH: 9865628687.