Quality Models To Measure Software Quality:A Descriptional Review

Deepu Sajeev
Mechanical Department
Trinity College of Engineering Trivandrum
Trivandrum, India
deepusajeev@outlook.com

S Sivakumar
Mechanical Department
College of Engineering Trivandrum
Trivandrum, India
profssk@yahoo.com

Abstract - The software industry is largely affected by cost-overruns, delays, poor customer satisfaction and quality issues that are costing clients and customers world-wide lots of money each year. The phenomenon is known as "The Software Crisis". Software Quality Engineering is an emerging discipline that is concerned with improving the approach to software quality. It is important that this discipline be firmly rooted in a quality model satisfying its needs. Software Quality Engineering needs a quality model that is usable throughout the software lifecycle and that it embraces all the perspectives of quality. Software quality models are a well-accepted means to support quality management of software systems. Over the last 30 years, lots of quality models have been proposed and applied with varying degrees of success. Despite successes and standardisation efforts, quality models are still being criticized, as their application in practice exhibits various problems. The goal of this paper is to find out a quality model suitable for such a purpose in Indian Software development Sector, through the comparative review study of existing quality models used outside India and their respective support for Software Quality Engineering.

Keywords— Software Project Management, Software Quality, Software Testing, Quality Models.

I. INTRODUCTION

Software technology has become an integrated and ubiquitous element in all kinds of human activity. The Internet grew in less than two decades to achieve the status of the largest information repository in human history. Computers, interconnected by complex and interdependent networks, are running software applications that control air traffic, satellite positioning, banking transactions and hospital emergency etc. With this increased dependence on information systems, technology failures might have disastrous effects. Such failures may result from both the hardware and software elements of the system, but while hardware design and manufacture has accumulated an admirable track record of reliability and dependability, software reliability has attracted much less attention.

Different approaches have been proposed to address the software quality issue. Proposed solutions include testing tools and methodologies, software development techniques, project management disciplines and training and development schemes. The field of software testing in particular grew substantially in the last decade. Researchers and practitioners within this field are developing innovative methods for ensuring the reliability, dependability and trustworthiness of software.

II. SOFTWARE QUALITY

A. Introduction

Research on software quality is as old as software research itself. As in other engineering and science disciplines, one approach to understand and control an issue is the use of models. Therefore, quality models have become a well-accepted means to describe and manage software quality.

B. Definition of Software Quality

What exactly constitutes quality? There are different perspectives which define quality. For some it is "[the] degree to which a set of inherent characteristics fulfills requirements" (ISO/IEC 1999b) while for others it can be synonymous with "customer value" (Highsmith, 2002), or even "defect levels" (Highsmith, 2002). It is also said as "something toward which we strive as an ideal, but may never implement completely." (Kitchenham & Pfleeger, 1996). The manufacturing perspective represents quality as conformance to requirements. This aspect of quality is stressed by standards such as ISO 9001, which defines quality as "the degree to which a set of inherent characteristics fulfills requirements" (ISO/IEC 1999b).

III. SOFTWARE QUALITY MODELS

The last three decades in quality modelling generated a multitude of very diverse models commonly termed "quality models". Examples on the spectrum of diverse models include taxonomic models like the ISO 9126 metric-based models like the maintainability index (MI) and stochastic models like reliability growth models (RGMs). The ISO 9126 is mainly used to define quality, metric-based approaches are used to assess the quality of a given system and reliability growth models are used to predict quality. Consequently, we term the ISO 9126 as definition model, metric-based approaches as assessment models and RGMs as prediction models.

A. McCall's Quality Model

D McCall (McCall, Richards & Walters, 1977) introduced his quality model in 1977. According to Pfleeger (2001), it

was one of the first published quality models. Figure 1 presents this quality model. Each quality factor on the left hand side of the figure represents an aspect of quality that is not directly measurable. On the right hand side are the measurable properties that can be evaluated in order to quantify the quality in terms of the factors. McCall proposes a subjective grading scheme ranging from 0 (low) to 10 (high).

Regarding this model, "unfortunately, many of the metrics defined by McCall et al. can be measured only subjectively"[7]. It is therefore difficult to use this framework to set precise and specific quality requirements. Furthermore, some of the factors and measurable properties, like traceability and self-documentation among others, are not really definable or even meaning at an early definable or even meaningful at an early stage for non-technical stakeholders.

This model is not applicable with respect to the criteria outlined in the IEEE Standard for a Software Quality Metrics Methodology for a top to bottom approach to quality engineering. Furthermore, it emphasizes the product perspective of quality. It is therefore not suited as a foundation for Software Quality Engineering.

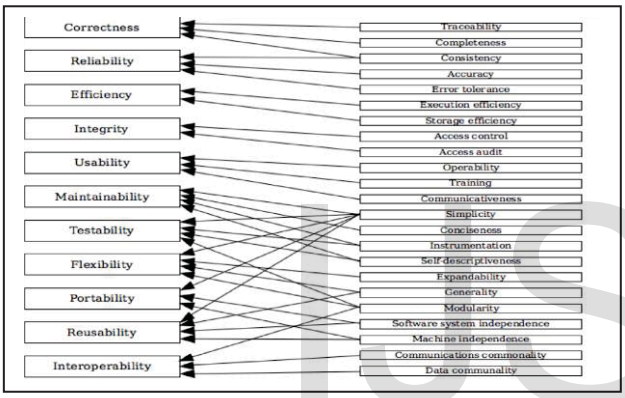

Fig 1: McCall's Quality Model Adapted from Pfleeger (2003) and McCall et al. (1977)

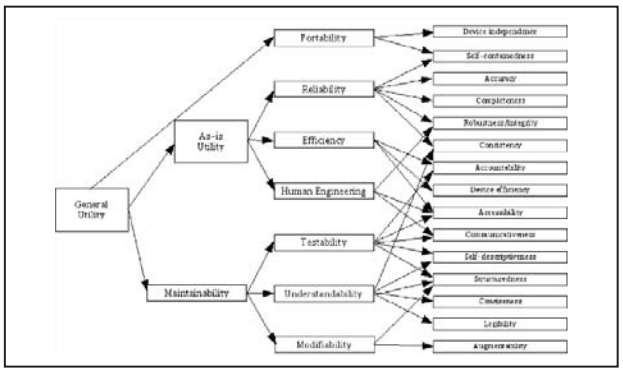

Fig.2 Boehm's Quality Model, adapted from Pfleeger (2003), Boehm et al. (1976; 1978)

B. *Boehm's QualityModel(1978)*

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. 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.

As Figure 2 shows, this quality model loosely retains the factor-measurable property arrangement. However, for Boehm and his colleagues, the prime characteristic of quality is what they define as "general utility". According to [6], this is an assertion that first and foremost, a software system must be useful to be considered a quality system. For Boehm, general utility is composed of as-is utility, maintainability and portability [3]:

- How well (easily, reliably, efficiently) can I use it [software system] as-is?
- How easy it to maintain is (understand, modify, and retest)?
- Can I still use it if I change my environment?

It is interesting to note that in opposition to McCall's model, Boehm's model is decomposed in a hierarchy that at the top addresses the concerns of end-users while the bottom is of interest to technically inclined personnel. Like the McCall model, this model is mostly useful for a bottom to top approach to software quality (i.e. it can effectively be used to define measures of software quality, but is more difficult to use to specify quality requirements).

C. *FURPS/FURPS+*

A later, and perhaps somewhat less renown, model that is structured in basically the same manner as the previous two quality models is the FURPS model originally presented by Robert Grady and extended by Rational Software now IBM Rational Software - into FURPS+3). FURPS stands for:

- Functionality
- Usability
- Reliability
- Performance
- Supportability

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*

An even more recent model similar to the McCall's, Boehm's and the FURPS (+) quality model, is the quality model presented by R. Geoff Dromey . Dromey proposes a product based quality model that recognizes that quality evaluation differs for each product. 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. Dromey has built a quality evaluation framework that analyzes the quality of software components through the measurement of tangible quality properties (Figure 3). Each artifact produced in the software lifecycle can be associated with a quality evaluation model.

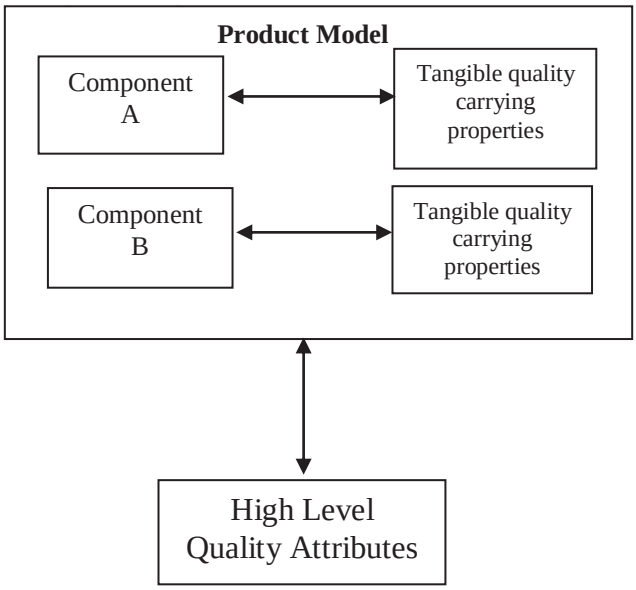

Fig 3: Dromey's Quality Model

E. System Dynamics Modeling

System dynamics is a strong simulation technique for analyzing and managing complex feedback system. It has its origin from industrial dynamics introduced by Forrester (1961). Industrial dynamics is the study of the information feedback characteristics of industrial system that aims for the design of improved organizational form and guiding policy. In industrial management, system dynamics has been applied to a wide range of problems such as human resource and knowledge managements, inventory management, product development, transportation, engineering service, supply chain management etc. The system dynamics models have been used to capture the dynamic behavior of supply chain using information feedback structures with the models represented by differential equations [4].

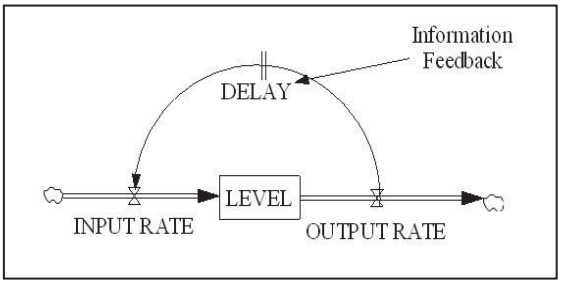

Fig 4: System dynamics structure

System dynamics approach consists of two basic structures: physical structure and information structure. Physical structure shows the resource converted between states in the system. An information feedback system exists whenever the environment leads to a decision that results in actions that effect the environment and thereby influences future decisions. [4]

Developing a System Dynamics model of software testing based on a "stocks and flows" view, and supported by one of the available software simulation packages, would enable the behaviour of the system to be simulated and, crucially, to conduct true "what if?" experiments by altering the values of constituent variables or "policies" and demonstrating how this affects other values within the model. [1]

The System Dynamic approach is a method that focuses on portraying complex systems, and simulate the relationship between variables across time and space. This is achieved through the concept of internal feed-back loops and time-delays that will influence behaviour in the system as a whole. In System Dynamics dynamic misperceptions can be identified and corrected if we have correctly calculated and represented key factors and behaviors inside the system itself. Hence, the system dynamics approach allows to build and test policies and assumptions in order to improve understanding of system behaviour or to change the observed behaviour. [5]

IV. CONCLUSION

Throughout this paper the aim has been to briefly survey some different models of quality – without going deep into a particular model and to identify which model is more appropriate. It was found that the models proposed by McCall, Boehm and Dromey focus on the product perspective of quality. Furthermore, they are primarily useful in a bottom up approach to quality that is not suitable for Software Quality Engineering.

Criteria/goals	***McCall, 1977***	***Boehm, 1978***
Correctness	*	*
Reliability	*	*
Integrity	*	*
Usability	*	*
Effiency	*	*
Maintainability	*	*
Testability	*	
Interoperability	*	
Flexibility	*	*
Reusability	*	*
Portability	*	*
Clarity		*

Modifiability		*
Documentation		*
Resilience		*
Understandability		*
Validity		*
Functionality		
Generality		*
Economy		*

Table 1: Comparison between criteria/goals of the McCall and Boehm quality models

This paper would suggest that System Dynamics would be the most appropriate methodology to achieve dynamic software quality issues. The reason is that most importantly it can simulate the relationship between variables across time and space and allows to build and test policies and assumptions. The most valued aspect is that it can shifts the focus from one aspect of a system to the behavior of the system as a whole. Further research is needed to see if the factors and measures associated with System Dynamics make this model usable for Software Quality Engineering in practice.

Acknowledgment

I would like to express my deep sense of gratitude and sincere thanks to all who helped me to complete this paper successfully. I am deeply indebted to my guide **Prof. S. Sivakumar,** Dept. of Mechanical Engineering for his excellent guidance, positive criticism, valuable comments, suggestions and constructive feedback. I avail my opportunity to express my profound thanks to my colleagues in Trinity College of Engineering, Trivandrum.

References

[1] Abdel-Hamid, T.K., 1991: Software Project Dynamics: An Integrated Approach, PrenticeHall, Upper Saddle River, USA.

[2] Boehm, B., 1981: Software Engineering Economics, Prentice-Hall, Englewood Cliffs, USA.

[3] B.W. Boehm, J.R. Brown, J.R. Kaspar, M. Lipow, G. MacCleod (1978) , "Characteristics of Software Quality", North Holland, Amsterdam.

[4] Forrester, J. W. & Senge, P. M., (1980): Tests for Building Confidence in System Dynamics Models, in Legasto, jr., A.A, Forrester, J.W. & Lyneis, J.M. (Eds.), System Dynamics: TIMS Studies in the Management Science, 14, North-Holland, New York, USA, 209-228.

[5] Sterman, J.D., 2003: Business Dynamics: Systems Thinking and Modeling for a Complex World, Irwin/McGraw-Hill, Chicago, USA.

[6] Rudiger Lincke, Tobias Gutzmann and Welf Lowe, Software Quality Prediction Models Compared, *10th International Conference on Quality Software*, 2010

[7] M. Ruiz et al., A Simplified model of software project dynamics, *The Journal of Systems and Software* 59 (2001) pp. 299-309