Comprehensive Research Journal of Management and Business Studies (CRJMBS) Vol. 1(2) pp. 015 – 024 October, 2016
Available online https://comprehensiveresearchjournals.org/crjmbs

Review

An analytical comparative analysis of the software quality models for software quality engineering

Machogu Moronge Abiud[1], Philip Mbugua[2]

[1]Adventist University of Central Africa, P.O Box 2461, Kigali, Rwanda
[2]Mount Kenya University, P.O.Box 5826 Kigali, Rwanda

Accepted 12 October, 2016

As software becomes more and more pervasive, there has been a growing concern about software quality. This concern arises from the fact that the main objective of software engineering is to produce good quality maintainable software in time and within budget. Quality is the combination of many characteristics, like conformance to requirements, fitness for the purpose, level of satisfaction. Software Quality Engineering needs a quality model that is usable throughout the software lifecycle and that embraces all the perspectives of quality. The goal of this paper is to do a comparative evaluation of existing quality models and their respective support for Software Quality Engineering, so as to analyze if the users of various software's have a direct and equal impact on the Software Quality, and if they indeed play a pivotal role in the measurement of Software Quality.

Keywords: Software Quality, McCall's Quality model, Boehm's, Quality Model, FURPS Quality Model, ISO 9126 Model, Quality engineering.

INTRODUCTION

Over the last decade, the general focus of the software industry has shifted from providing ever more functionality to improving what has been coined as the user experience. The user experience refers to characteristics such as ease-of-use, security, stability and reliability.

Suryn and Georgiadou, (1986) indicates that ttraditionally, software requirements have been classified either as functional or non-functional with eventual notions of quality hidden in the latter. As the industry focus is shifting from functionality to improving quality, a new category of requirements focused on quality is emerging. In order to define these new quality requirements, quality itself must be defined. A quality model provides the framework towards a definition of quality. In technical terms quality is Conformance to specification, Meeting customer needs, Fitness for use (Ibrahim 2001).

*Corresponding Author's E-mail: mabiud@yahoo.com

As in other engineering and science disciplines, one approach to understand and control an issue is the use of models. A quality model provides the framework towards a definition of quality (Suryn and Georgiadou, 1986). The, quality models have become a well-accepted means to describe and manage software quality. Software quality plays an important role in success of the overall software system. So it is considered as a very important aspect for the developers, users and project managers. Software quality is the extent to which an industry-defined set of desirable features are incorporated into a product so as to enhance its lifetime performance (Dubey, Ghosh, and Rana, 2012). To formulate the requirements for a software product quality model, Dromey and Brisbane, (1998) indicates that three issues must be addressed: the different interest groups need to be identified; the intended applications of the model need to be spelled out; and it is necessary to establish the quality needs or perspectives/views of the different interest groups. They further spell out that in constructing a model for software product quality it is appropriate to apply both bottom-up and top-down strategies.

Suryn and Georgiadou, (1986) argues that eengineers have long recognized that in order for something to find its way in a product, it should be properly defined and specified. Unfortunately, the push towards software quality that can be observed in the industry today is lacking a solid foundation in the form of an agreed upon quality model that can be used not only to evaluate software quality, but also to specify it. The objective of this paper is to identify and analyze and compare the various software quality model that have been used as a foundation to Software Quality Engineering.

THEORETICAL AND RELATED LITERATURE REVIEW

Definition of Software Quality

Quality must be defined and measured if improvement is to be achieved. Yet, a major problem in quality engineering and management is that the term quality is ambiguous, such that it is commonly misunderstood. The confusion may be attributed to several reasons. First, quality is not a single idea, but rather a multidimensional concept. The dimensions of quality include the entity of interest, the viewpoint on that entity, and the quality attributes of that entity. Second, for any concept there are levels of abstraction; when people talk about quality, one party could be referring to it in its broadest sense, whereas another might be referring to its specific meaning. Third, the term quality is a part of our daily language and the popular and professional uses of it may be very different.

Kan, (2002) defines quality as "conformance to requirements" and "fitness for use", while Malhotra and Pruthi, (2012) defines it as the degree to which a system, component, or process meets customer or user needs or expectations. "Conformance to requirements" implies that requirements must be clearly stated such that they cannot be misunderstood. Then, in the development and production process, measurements are taken regularly to determine conformance to those requirements. The non conformances are regarded as defects - the absence of quality. The "fitness for use" definition takes customers' requirements and expectations into account, which involve whether the products or services fit their uses. Since different customers may use the products in different ways, it means that products must possess multiple elements of fitness for use. According to Kan (2002), each of these elements is a quality characteristic and all of them can be classified into categories known as parameters for fitness for use.

In Juran (1998) Quality Control Handbook" he provides two meanings to quality: First, he says quality consists of those product features which meet the need of customers and thereby provide product satisfaction. Secondly quality consists of freedom from deficiencies. Nevertheless, in a handbook such as this it is most convenient to standardize on a short definition of the word quality as "fitness for use" which indicates references to requirements and products characteristics. Juran's definition could be interpreted as a "conformance to specification" definition more than a "meeting customer needs" definition. Juran (1988) proposes three fundamental managerial processes for the task of managing quality. The three elements of the Juran Trilogy include:

- Quality planning: A process that identifies the customers, their requirements, the product and service features that customers expect, and the processes that will deliver those products and services with the correct attributes and then facilitates the transfer of this knowledge to the producing arm of the organization.
- Quality control: A process in which the product is examined and evaluated against the original requirements expressed by the customer. Problems detected are then corrected.
- Quality improvement: A process in which the sustaining mechanisms are put in place so that quality can be achieved on a continuous basis. This includes allocating resources, assigning people to pursue quality projects, training those involved in pursuing projects, and in general establishing a permanent structure to pursue quality and maintain the gains secured.

Ishikawa, (1985) writes the following in his book "What is quality control? The Japanese Way", we engage in quality control in order to manufacture products with the quality which can satisfy the requirements of consumers. The mere fact of meeting national standards or specifications is not the answer, it is simply insufficient. International standards established by the International Organization for Standardization (ISO) or the International Electro technical Commission (IEC) are not perfect. They contain many shortcomings. Consumers may not be satisfied with a product which meets these standards. We must also keep in mind that consumer requirement change from year to year and even frequently updated standards cannot keep the pace with consumer requirements. How one interprets the term "quality" is important. Narrowly interpreted, quality means quality of products. Broadly interpreted, quality means quality of product, service, information, processes, people, systems etc. etc.

Ishikawa's perspective on quality is a "meeting customer needs" definition as he strongly couples the level of quality to every changing customer expectations. He further means that quality is a dynamic concept as the needs, the requirements and the expectations of a customer continuously change. As follows, quality must be defined comprehensively and dynamically. Ishikawa also includes that price as an attribute on quality – that is, an overprized product can neither gain customer satisfaction and as follows not high quality.

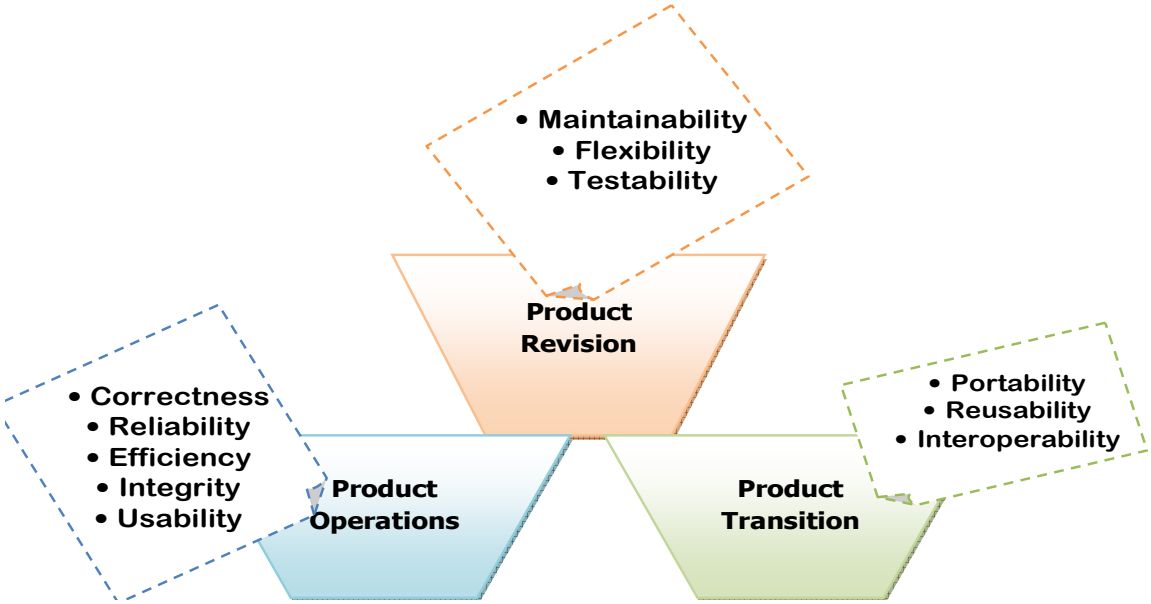

Figure 1. The McCall quality model organized around three types of quality characteristics

EVALUATION OF SOFTWARE QUALITY MODELS

McCall's Quality Model (1977)

- One of the more renowned predecessors of today's quality models is the quality model presented by Mc Call J. A. McCall Quality Model defines and identifies the quality of a software product by addressing three perspectives as indicated in figure 1 -Product revision (ability to change), Product transition (adaptability to new environments), Product operations (basic operational characteristics), which are further divide into the 11 external quality factors (from point of view of user):

1. Product operation: is the product's ability to be quickly understood, operated and capable of providing the results required by the user. It covers correctness, reliability, efficiency, integrity and usability factors.
2. Product revision: is the ability to undergo changes, including error correction and system adaptation. It covers maintainability, flexibility and testability factors.
3. Product transition: is the adaptability to new environments, distributed processing together with rapidly changing hardware. It covers portability, reusability and interoperability factors.

The idea behind McCall's Quality Model is that the quality factors synthesized should provide a complete software quality picture. The model details the three types of quality characteristics (major perspectives) in a hierarchy of factors, and criteria (Kitchenham, and Pfleeger, 1998).

- 11 Factors describes the external view of the software, as viewed by the users.
- 23 quality criteria describe the internal view of the software, as seen by the developer.

Barry W. Boehm's Quality Model (1978)

Boehm's quality model improves upon the work of McCall and his colleagues. At the highest level of his model, Boehm defined three primary uses (or basic software requirements), these three primary uses are:-

a) The extent to which the as-is software can be used (i.e. ease of use, reliability and efficiency).

b) Maintainability, ease of identifying what needs to be changed as well as ease of modification and retesting. How easy is it to maintain (understand, modify, and retest)?

c) Portability, ease of changing software to accommodate a new environment. Can I still use it if I change my environment?

These three primary uses had quality factors associated with them, which further represent the next level of Boehm's hierarchical model. If the semantics of McCall's model are used as a reference, the quality factors could be defined as: Portability, Reliability, Efficiency, Human Engineering, Testability, Understandability and Modifiability.

- Portability, the extent to which the software will work under different computer configurations (i.e. operating systems, databases etc.).
- Reliability, the extent to which the software performs as required, i.e. the absence of defects.
- Efficiency, optimum use of system resources during correct execution.
- Usability, ease of use.
- Testability, ease of validation, that the software meets the requirements.
- Understandability, the extent to which the software is easily comprehended with regard to purpose and

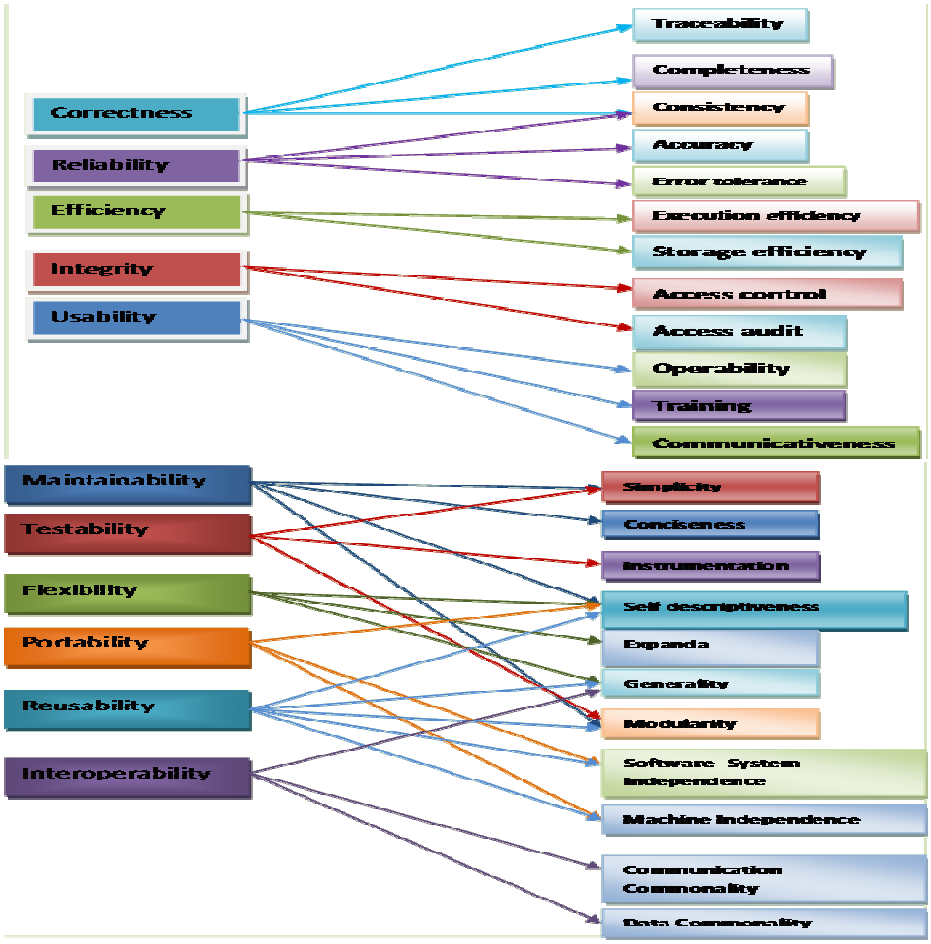

Figure 2. McCall's Quality Model (cont.) illustrated through a hierarchy of 11 quality factors (on the left hand side of the figure) related to 23 quality criteria (on the right hand side of the figure).

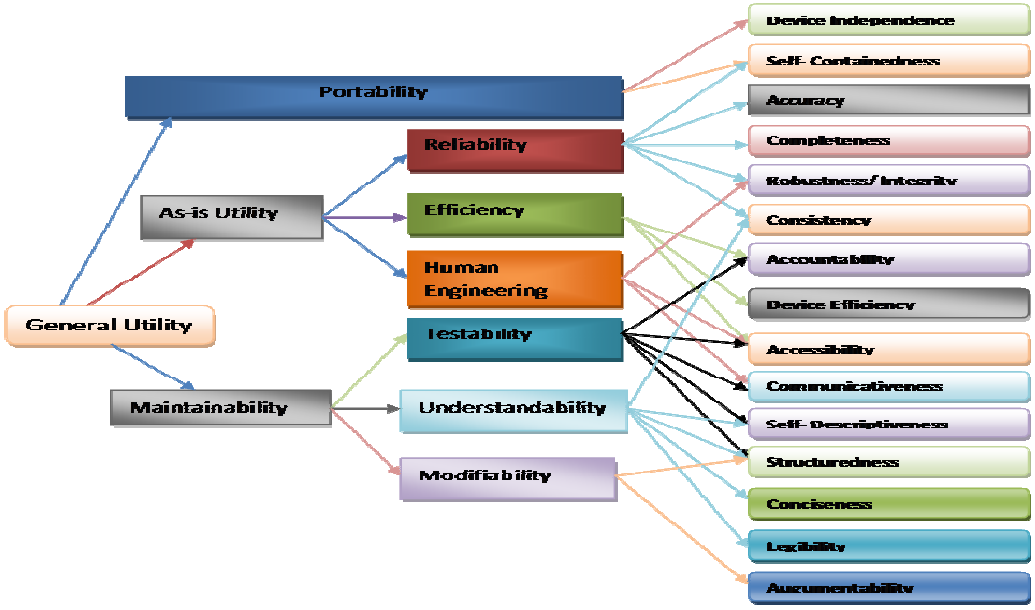

Figure 3. Boehm's quality model. Source: Adapted from Pfleeger (2003), Boehm et al. (1976; 1978)

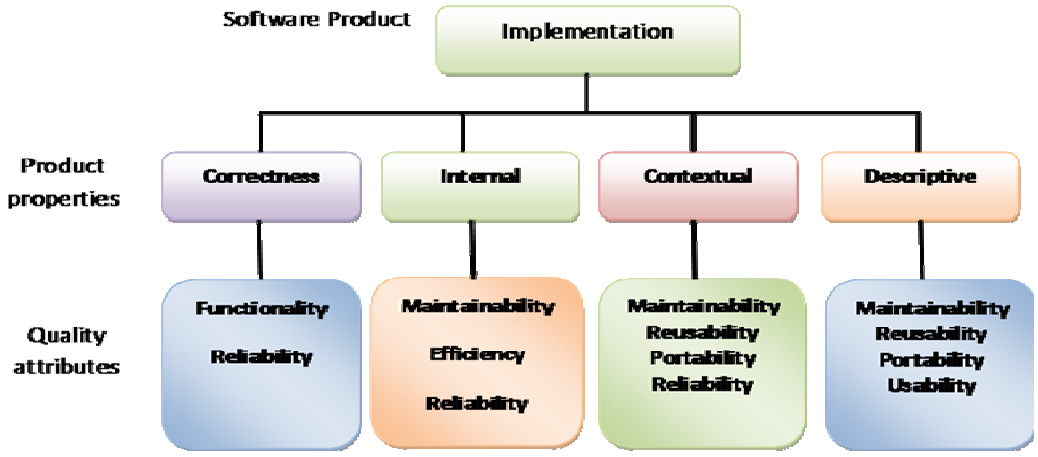

Figure 4. Principles of Dromey's Quality Model

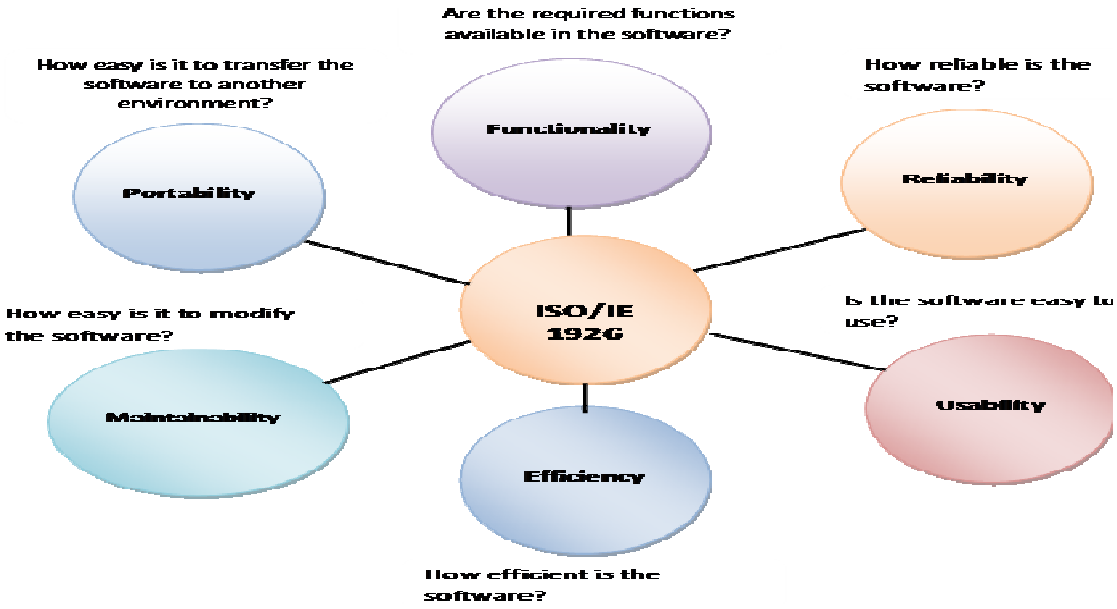

Figure 5: The ISO 9126 quality

structure.

- Flexibility, the ease of changing the software to meet revised requirements.

FURPS (Functionality, Usability, Reliability, Performance and Supportability).

A later, and perhaps somewhat less renown, model that is structured in basically the same manner as the McCall, Boehm, quality models is the FURPS model originally presented by Robert Grady and extended by Rational Software - now IBM Rational Software – into FURPS (Gary 1992).

FURPS stands for:

- Functionality: includes feature sets, capabilities and security.
- Usability: includes human factors, aesthetics, consistency in the user interface, online and context sensitive help, wizards and agents, user documentation, and training materials.
- Reliability: includes frequency and severity of failure, recoverability, predictability, accuracy, and mean time between failures (MTBF).
- Performance: imposes 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.

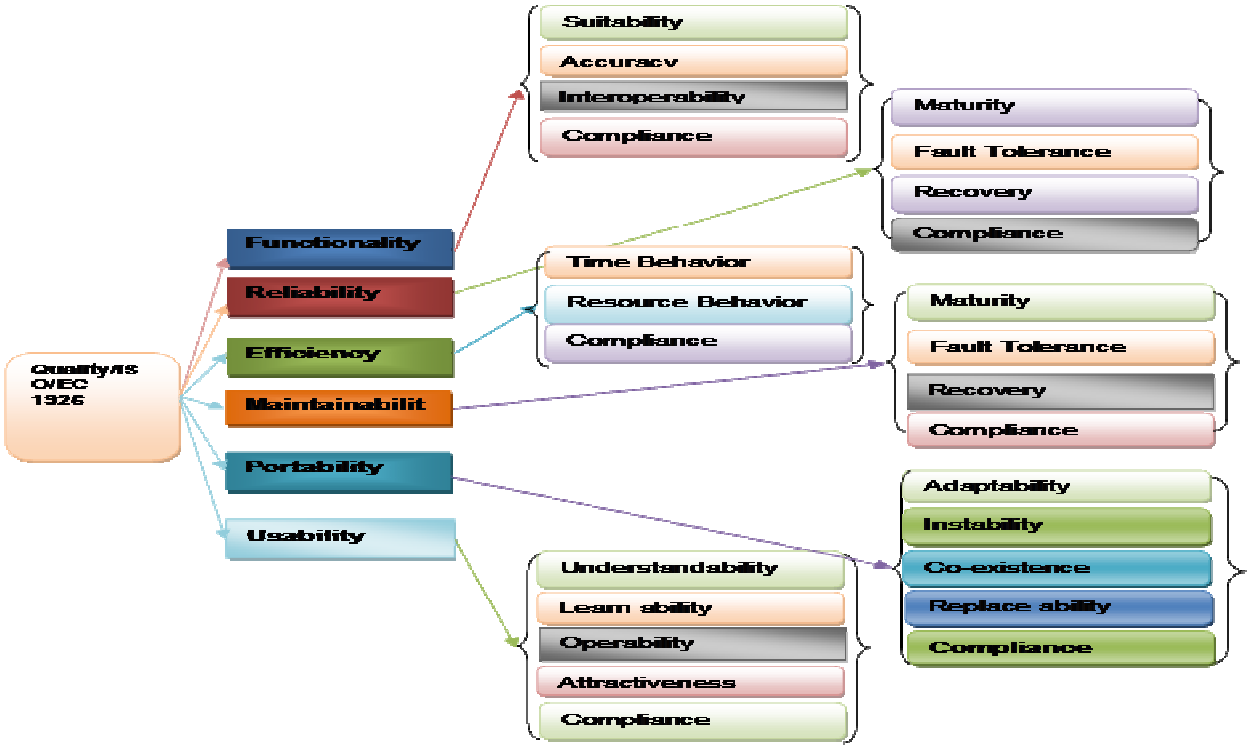

Table 1: Comparative analysis between Dromey's, Boehm's, ISO 9126 and McCall's software quality models factors describing the external view of the software as seen by the users

Criteria/goals	Dromey's Quality Model	ISO 9126	McCall, 1977	Boehm, 1978
Correctness	*		*	*
Reliability	*		*	*
Integrity			*	*
Usability	*	*	*	*
Efficiency	*	*	*	*
Maintainability	*	*	*	*
Testability			*	
Interoperability			*	
Flexibility			*	*
Reusability	*		*	*
Portability	*	*	*	*
Clarity				*
Modifiability				*
Documentation				*
Resilience				*
Understandability				*
Validity				*
Functionality	*	*		
Generality				*
Economy				*

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's main focus is in the relationship between the quality attributes and the sub-attributes, as well as attempting to connect software product properties with software quality attributes (Dromney 1996).

Criteria/goals	Dromey's Quality Model	ISO 9126	McCall, 1977	Boehm, 1978
Traceability			*	
Completeness			*	*
Consistency			*	*
Accuracy		*	*	*
Error tolerance		*	*	
Execution efficiency			*	
Storage efficiency			*	
Access Control			*	
Access audit			*	
Operability		*	*	
Training			*	
Communicativeness			*	*
Simplicity			*	
Conciseness			*	*
Instrumentation			*	
Self descriptiveness			*	*
Expandability			*	
Generality			*	
Modularity			*	
Software-System Independence			*	
Common commonality			*	
Data commonality			*	
Device Independence				
Self-Contentedness				
Robustness/ Integrity				
Accountability				
Device Efficiency				
Accessibility				
Structuredness				
Legibility				
Augumentability				
Suitability		*		
Interoperability		*		
Compliance		*		
Time behavior		*		
Resource Behaviour		*		
Learn Ability		*		
Attractiveness		*		
Maturity		*		
Adaptability		*		
Instability		*		
Co-existence		*		
Replaceability		*		

Table 2: Comparative analysis between Dromey's ,Boehm's, ISO 9126 and Mccall's software quality models factors describing the internal view of the software as seen by the developer.

ISO 9126

In 1991, the International Organization for Standardization introduced a standard named ISO/IEC 9126 (1991). The ISO 9126 standard was based on the McCall and Boehm models. Besides being structured in basically the same manner as these models, ISO 9126 also includes functionality as a parameter, as well as identifying both internal and external quality characteristics of software (Malhotra and Pruthi, 2012). This standard aimed to define a quality model for

software and a set of guidelines for measuring the characteristics associated with it (Naragani and Uniyal, 2013). The standard ISO 9126 consists of four parts:

- ISO 9126-1: Quality model
- ISO 9126-2: External metrics
- ISO 9126-3: Internal metrics
- ISO 9126-4: Quality in use metrics

Each quality factors and its corresponding sub-factors are defined as follows:

A. Functionality: A set of attributes that relate to the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

a. Suitability: Attribute of software that relates to the presence and appropriateness of a set of functions for specified tasks.

b. Accuracy: Attributes of software that bare on the provision of right or agreed results or effects.

c. Security: Attributes of software that relate to its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.

d. Interoperability: Attributes of software that relate to its ability to interact with specified systems.

e. Compliance: Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.

B. Reliability: A set of attributes that relate to the capability of software to maintain its level of performance under stated conditions for a stated period of time.

a. Maturity: Attributes of software that relate to the frequency of failure by faults in the software.

b. Fault tolerance: Attributes of software that relate to its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.

c. Recoverability: Attributes of software that relate to the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.

d. Compliance

e. Usability: A set of attributes that relate to the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

f. Understandability: Attributes of software that relate to the users' effort for recognizing the logical concept and its applicability.

g. Learn ability: Attributes of software that relate to the users' effort for learning its application (for example, operation control, input, output).

h. Operability: Attributes of software that relate to the users' effort for operation and operation control.

i. Attractiveness:

j. Compliance: Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.

C. Efficiency: A set of attributes that relate to the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

a. Time behavior: Attributes of software that relate to response and processing times and on throughput rates in performing its function.

b. Resource behavior: Attributes of software that relate to the amount of resources used and the duration of such use in performing its function.

c. Compliance

D. Maintainability: A set of attributes that relate to the effort needed to make specified modifications.

a. Analyzability: Attributes of software that relate to the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.

b. Changeability: Attributes of software that relate to the effort needed for modification, fault removal or for environmental change.

c. Stability: Attributes of software that relate to the risk of unexpected effect of modifications.

d. Testability: Attributes of software that relate to the effort needed for validating the modified software.

e. Compliance:

E. Portability: A set of attributes that relate to the ability of software to be transferred from one environment to another.

a. Adaptability: Attributes of software that relate to on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered.

b. Install ability: Attributes of software that relate to the effort needed to install the software in a specified environment.

c. Conformance: Attributes of software that make the software adhere to standards or conventions relating to portability.

d. Replace ability: Attributes of software that relate to the opportunity and effort of using it in the place of specified other software in the environment of that software.

IEEE Model

IEEE has also release several standards, as illustrated below:

- IEEE Std. 1220-1998: IEEE Standard for Application and management of the Systems Engineering Process
- IEEE Std 730-1998: IEEE Standard for Software Quality Assurance Plans
- IEEE Std 828-1998: IEEE Standard for Software Configuration Management Plans – Description
- IEEE Std 829-1998: IEEE Standard For Software

Test Documentation
- IEEE Std 830-1998: IEEE recommended practice for software requirements specifications
- IEEE Std 1012-1998: IEEE standard for software verification and validation plans
- IEEE Std 1016-1998: IEEE recommended practice for software design descriptions
- IEEE Std 1028-1997: IEEE Standard for Software Reviews
- IEEE Std 1058-1998: IEEE standard for software project management plans
- IEEE Std 1061-1998: IEEE standard for a software quality metrics methodology
- IEEE Std 1063-2001: IEEE standard for software user documentation
- IEEE Std 1074-1997: IEEE standard for developing software life cycle processes
- IEEE/EIA 12207.0-1996: Standard Industry Implementation of International Standard ISO/IEC 12207: 1995 (ISO/IEC 12207) Standard for Information Technology Software Life Cycle Processes

COMPARATIVE ANALYSIS

Boehm's and McCall's models

Though Boehm's and McCall's models might appear very similar, as indicated in figure 3 and 4, 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 mode model is based on a wider range of characteristics with an extended and detailed focus on primarily maintainability. Figure 6: compares the two quality models, quality factor by quality factor.

CONCLUSION

An impressive development of quality models has taken place over the last decades. These efforts have resulted in many achievements in research and practice. As an example, take a look at the field of software reliability engineering that performed a wide as well as deep investigation of reliability growth models. In some contexts these models are applied successfully in practice. The developments in quality definition models even led to the standardization in ISO 9126 that is well known and serves as the basis for many quality management approaches. However, the whole field of software quality models is diverse and fuzzy. There are large differences between many models that are called "quality models".

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. Based on the discussion of the four quality models and on the comparison between them, it has been observed that users have a direct and equal impact on the Software Quality, and thus play a pivotal role in the measurement of Software Quality. The quality attributes that can then be conclusively seen as a measure of software quality because these attributes are seen to cut across in most of the software quality models are : correctness, reliability, usability, efficiency, maintainability, reusability and portability.

REFERENCES

Boehm BW, Brown JR, Kaspar H, Lipow M, McLeod G, Merritt M(1978). Characteristics of Software Quality, North Holland.

Boehm, Barry W, Brown JR, Lipow M (1976). Quantitative evaluation of software quality, International

Conference on Software Engineering, Proceedings of the 2nd international conference on Software engineering.

Crosby PB (1979). Quality is free: the art of making quality certain, New York, McGraw-Hill.

Deming WE (1988). Out of the crisis: quality, productivity and competitive position, Cambridge Univ. Press.

Dromey G, Brisbane N (1998, March 2009). SOFTWARE PRODUCT QUALITY: Theory, Model, and Practice. AUSTRALIA.

Dromey RG (1996). "Concerning the Chimera [software quality]", IEEE Software. 1: 33-43.

Dromey RG (1996). "Concerning the Chimera [software quality]", IEEE Software, no. 1, pp. 33-43.

Dubey SK, Ghosh S, Rana A (2012). Comparison of Software Quality Models: An analytical Apprioach. Int'l. J. Emerging Tech. Adv. Engr., 2 (2).

Feigenbaum AV (1983). Total quality control, McGraw-Hill.

Grady RB (1992). Practical software metrics for project management and process improvement, Prentice Hall.

Grady RB (1992). Practical software metrics for project management and process improvement, Prentice Hall.

Grady R, Caswell D (1987). Software metrics: Establishing a companywide program, Prentice-Hall, ISBN 0138218447.

Hoyer RW, Hoyer BBY (2001). "What is quality?" quality progress. 7: 52-62.

Hyatt LE, Rosenberg LH (1996). Product Assurance Symposium and Software Product Assurance Workshop, Proceedings of meetings, European Space Agency. 209

Hyatt LE, Rosenberg LH (1996). A Software Quality Model and Metrics for Identifying Project

Ibrahim K.El-Far, James AW (2001). "Model-based Software Testing", Encyclopedia on Software Engineering, Wiley.

IEEE 1993. Standard for Software Maintenance, Software Engineering Standards Subcommittee of the IEEE Computer Society.

International Standard, ISO 9126-1. 2001. Institute of Electrical and Electronics Engineers, Pt. 1, 2, 3: Quality model.

Ishikawa K (1985). What is total quality control?: the Japanese way, Prentice-Hall.

Ishikawa K (1985). What is total quality control?: the Japanese way, Prentice-Hall.

ISO /IEC. 1986. International Standard 8402: Quality –Vocabulary

ISO 9126. 1991. Software Product Evaluation: Quality characteristics and guidelines for their use, ISO/IEC Standardization ISO 9126.

ISO 9126. 2000E. Standard ISO/IEC, Information technology- Software product quality – Part1: Quality Model, ISO/IEC FDIS 9126-1: 2000(E)

ISO/ IEC 25030. 2006. Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE), Quality Requirements.

ISO/ IEC CD 25010. 2007. Software Engineering: Software Product

Quality Requirements and Evaluation (SQuaRE) Quality Model and guide.
ISO/IEC TR 9126-3. 2002. Software Engineering Product Quality.
Jacobson I, Booch G, Rumbaugh J (1999). The Unified Software Development Process, Addison Wesley Longman Inc.
Juran JM (1988). Juran's Quality Control Handbook, McGraw-Hill.
Juran JM (1988). Juran's Quality Control Handbook, McGraw-Hill.
Kazman R, Bass L, Clements P (2003). Software Architecture in Practice 2Ed. Addison Wesley.
Khayami R, Towhidi A, Ziarati K (2009). The Analytical Comparison of Qualitative Models of Software Systems, World Applied Sci. J. 6, IDOSI Pub.
Khomh F, Haderer N, Antoniol G (2009). SQUAD: Software Quality Understanding through the Analysis of Design, Reverse Engineering, WCRE'09, 16th working conference.
Khosravi K, Gueheneuc Y (2005). On issues with software quality models, 9th ecoop workshop on quantitative approaches in object-oriented software engineering.
Kitchenham B, Pfleeger SL (1996). "Software quality: the elusive target [special issues section]", IEEE Software. 1: 12-21.
Kitchenham B, Pfleeger SL (1996). "Software quality: the elusive target [special issues section]", IEEE Software. 1: 12-21.
Klein M, Clements P, Kazman R (2002). Evaluating software architectures: methods and case studies, Addison Wesley.
Kruchten P (2000). The Rational Unified Process An Introduction - Second Edition, Addison Wesley Longman, Inc.
Kumar A, Kumar R, Grover PS (2006). A change Impact Assessment in Aspect-Oriented Software Systems, In the proceedings of International Software Engineering Conference Russia, (SECR-2006), Dec. 83-87.
Marciniak J (2002). Encyclopedia of software engineering, 2(2ed). Chichester: Wiley.
McCall JA, Richards PK, Walters GF (1977). Factors in Software Quality. 1, 2 and 3, AD/A 049-014/015/055. Nat'l. Tech. Info. Serv. Springfield.
McCall JA, Richards PK, Walters GF (1977). "Factors in Software Quality", Nat'l Tech.Information Service. 1, 2, 3.
Naragani DP, Uniyal P (2013). Comparative Analysis of Software Quality Models. Int'l. J. Comp. Sci. Manag. Res. 2 (3).
Pressman RS (1992). Software Engineering a practitioner's Approach. McGraw-Hill, Inc.
Rational Software Inc., RUP - Rational Unified Process, www.rational.com, 2003.
Risks and Assessing Software Quality, European Space Agency Software Assurance Symposium and the 8th Annual Software Technology Conference, 1996.
Robson C (2002). Real world research: a resource for social scientists and practitioner-researchers, Blackwell Publisher Ltd.
Sehra SK, Brar YS, Kaur N (2011). Soft Computing Techniques for Software Project Effort Estimation, Int'l J. Adv. Comp. Math. Sci. ISSN 2230-9624, 2(3): 160-167.
Sharma A, Kumar R, Grover PS (2008). Estimation of Quality for software components: an empirical approach, ACM SIGSOFT Software Engineering Notes, 33(6): 1-10
Shewhart WA (1931). Economic control of quality of manufactured product, Van Nostrand.
Suryn W, Georgiadou E (1986). Software Quality Model Requirements for Software Quality Engineering.