ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 2, Issue 1, January 2013

A Comparative Study of Software Quality Models

Sheikh Fahad Ahmad, Mohd. Rizwan Beg, Mohd. Haleem

Abstract: Computers play a vital role in industry and government. Since the hardware of modern systems relies heavily on the supporting software, which can critically affect lives. Many applications (e.g., life support systems, mission-critical systems, nuclear power plants etc) require correct and reliable software. The increased importance of software also places more requirements on it. Thus, it is necessary to have precise, predictable, and repeatable control over the quality of software development process and product. Metrics evolved out of this need are used to measure software quality. This paper aims at an empirical survey of various software quality metrics and comparing their performance in software development.

Index Terms— in-process metrics, product metrics, quality metrics, Software quality.

I. INTRODUCTION

Software metric is a field of software engineering that is associated with diverse measurements of computer software and its developments. Software metrics [1] [2] [3] is one of the important tools for analyzing the software product in an effective way. In other words software metrics are measures that enable software developers and software analyst to gain insight into the efficiency of the software process and projects that are conducted using the process as framework. Software metrics measures different aspects of software complexity and therefore play an important role in analyzing and improving software quality [3]. With the help of software metric we are able to understand the software product in an effective way. Software metric is a field of software engineering that is associated with diverse measurements of computer software and its development. According to Tom DeMarco that "You cannot control what you cannot measure". With the help of software metric we are able to measure some property of software or its component.

***Sheikh Fahad Ahmad**, Deptt of Computer Science & Engg, Integral University, Lucknow, India, 09919491430*

***Mohd Rizwan Beg**, Deptt of Computer Science & Engg, Integral University, Lucknow, India, 09839384611.*

***Mohd Haleem**, Deptt of Computer Applications, Integral University, Lucknow, India, 08604178177.*

Software metric [4] [5] are helpful in improving the quality of software, planning the budget, its cost estimation etc. with the help of software metric we are able to understand the software product in an effective way.

II. DEFINING: METRICS

Famous quote of a management consultant Peter Drucker: "If you can't measure it, you can't manage it", if a project manager and team members are not able to precisely measure what they are going to do then it would not be possible for them to effectively manage and improve the performance of a project. The success of a software project is the primary goal. Metrics that help to measure the success or failure of a project are very diverse and these metric hardly have a good deal in commonalities [6]. Metrics are also helpful to determine current status of a project and evaluate its performance. Software metrics can be classified into three categories: product metrics, process metrics, and project metrics. Product metrics describe the characteristics of the product such as size, complexity, design features, performance, and quality level. Process metrics can be used to improve software development and maintenance. Project metrics describe the project characteristics and execution.

III. QUALITY METRICS

Software quality metrics are the subset of software metrics that focuses on the quality aspect of software. Software quality metrics are associated with process and product metrics than with project metrics. Software quality metrics can be divided further into end-product quality metrics and in-process quality metrics. The need of software quality engineering is to determine the relationships among in-process metrics, project characteristics, and end-product quality in order to improve quality in both process and product. Moreover, we should view quality from the entire software life-cycle perspective and, we should include metrics that measure the quality level of the maintenance process as another category of software quality metrics.

IV. SOFTWARE QUALITY FACTORS(SQF)

Software Quality Factors (SQFs) are those attributes that users consider important for software to possess

[6]. When these factors are merged with software metrics, the result is an SQM. There are many Quality Factors, and the meaning of each depends on a given definition and measure. For example, a practitioner may measure complexity by counting the decisions in a program and another may measure it by counting the source lines of a program.

A quality factor can be used at any time in the life-cycle of the product. The procedure for selecting a quality factor consists of four steps [6]:

First Step- Identifying functions

Second Step- Assigning quality factors and goals

Third Step- Considering interrelationships

Fourth Step- Considering costs

The quality factors are chosen based on the function of the software and the objectives of the application. For example, a mission-dependent function will need to be more rigorously tested for reliability than one that is not critical to the mission.

An application designed efficiently in handling real-time transactions may be extremely difficult to maintain. In cases where the factors conflict and the practitioner still wishes to analyze opposing attributes, the metric scores can be weighted, giving the more important factors more significance in the results.

The following quality factors are presented alphabetically. These factors represent important attributes of software products.

1. Accuracy: Size Accuracy is the extent to which a program's outputs are sufficiently precise to satisfy their intended use [7] [8] [9].

2. Clarity: Clarity measures how clearly a person can understand a program [6] [7] [8].

3. Completeness: Completeness is the extent to which software fulfils the overall mission requirements [6] [7] [8].

4. Complexity: If complexity is viewed as the degree of decision-making found in a program [6], this quality factor could be measured using software metrics.

5. Conciseness: Conciseness is the ability of a program to satisfy functional requirements using a minimum amount of software [6] [7] [8].

6. Consistency: Consistency can be divided into two types: internal and external. Internal consistency is the degree to which software satisfies specifications [7] [8]. External consistency is the extent to which a software product contains uniform notation, symbols, and terminology [6].

7. Correctness: Correctness measures the extent to which the software design and implementation conform to specifications and standards [6].

8. Efficiency: Efficiency is considered to be a system quality factor [6]. It is the extent to which resources are utilized.

9. Expandability: Expandability is the amount of effort required to increase the software's capabilities or performance [6].

10. Flexibility: Flexibility is defined as the amount of effort required to change the software's mission, functions, or data to satisfy other requirements [6].

11. Integrity: Integrity is the measure of the ability of a program to perform correctly on different sets of input [10].

12. Maintainability: It is the measure of the effort and time required to fix bugs in the program [7].

Modifiability: Modifiability measures the cost of changing or extending a program [11].

13. Modularity: In the computer industry, applications are becoming increasingly complex and diverse. The high-powered technology that continues to spawn faster and more efficient computers has led to larger, more complex software applications. This complexity must be reduced to manageable chunks so it can be understood.

14. Performance: Performance is perhaps the broadest quality factor. It encompasses several of the other quality factors, and is concerned with how well a software product functions. Performance asks the following questions [6].

15. Portability: Portability measures how easily a software product will run on a computer configuration other than the current one [7].

16. Reliability: Reliability is simply a measure of the number of errors encountered in a program [11].

*17. Reusability***:** Reusability is the extent to which a system can be applied to other environments [8].

18. *Simplicity:* Simplicity is the ease with which functions can be implemented and comprehended [11].

19. Testability: Testability is the extent to which software facilitates the establishment of acceptance criteria and supports evaluation of its performance [7].

20. Understandability: Understandability is the ease with which a program can be understood [11].

21. Usability: Usability measures the effort required to train a person to use the software [6].

V. SOFTWARE QUALITY METRICS(SQM)

SQM are composed of software metrics and SQFs.

In 1977, McCall et al. [15] proposed a software quality model called McCall's Software Quality Model. In 1978, Boehm et al. [16] proposed another software quality model using McCall's quality model, called Boehm's Software Quality Model. Later in late 80's, three quality Models (in 1987, Evans & Marciniak's Quality Model and FURPS Quality Model and next year 1988, Deutsch & Will's Quality Model) came into existence. Among these models, FURPS Quality Model [17] [18] is more popular because it was the first industrial approach based quality model developed by Hewlett Packard (HP).

Till 90's, many software quality models were proposed. This led to confusion among practitioners, that which model to be actually followed. Therefore, International Organization for Standardization/International Electro-technical Commission (ISO/IEC) developed and standardized a new quality model considering the entire repository of various quality models proposed till date. In 1991, ISO/IEC proposed a quality model,

called ISO/IEC Quality Model. Later , the name was changed to ISO/IEC 9126 Quality Model [19] [20] [21] since ISO 9126 was part of the ISO 9000 standard. In 1995, R.G. Dromey [14] proposed a quality model adding one characteristic into ISO/IEC 9126 Quality model. This model is known as Dromey's Software Quality Model.

The three major families of software metrics (ISO 9126, McCall, Boehm), based on which most of the striking SQM are based, are presented here.

A. THE ISO 9126 STANDARD QUALITY MODEL

The ISO 9126 quality model was proposed as an international standard for software quality measurement in 1992. It was derived using the McCall model. ISO 9126 defines 22 attributes that a quality software product must exhibit. The 22 attributes are arranged in six areas: functionality, reliability, usability, efficiency, maintainability and portability. Software quality measurement techniques allows to measure some of the quality attributes. ISO 9126 is the most commonly used quality standard model. There are several others such as IEEE 1061[12]. The characteristics and sub-characteristics are shown in Table I. The ISO 9126 software quality model identifies 6 main quality characteristics, namely:

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

TABLE I
ISO 9126 QUALITY MODEL

Quality Type	Product Perspective	Characteristics	Sub-Characteristics
Quality Software Product		Functionality	Suitability
			Accuracy
			Interoperability
			Security
		Reliability	Maturity
			Fault Tolerance
			Recoverability
		Usability	Understandability
			Learn-ability
			Operability
			Attractiveness
		Efficiency	Time Behavior
			Resource
		Maintainability	Analyzability
			Changeability
			Stability
			Testability
		Portability	Adaptability
			Replace-ability
			Install-ability
			Co-Existence

B. MC CALL SOFTWARE QUALITY MODEL

One of the most famous predecessors of today's quality models is the model proposed by Jim McCall et al.[13] (also known as General Electric's Model of 1977). Quality criteria associated with a set of quality metrics are used and defined to provide a scale and method for measurement [22]. The factors and criteria are shown in table 2. It identifies 3 areas of software work: -

• Product Operation – refers to the ability of the product to be quickly understood, efficiently operated and also capable of providing the results required by the user.

• Production Revision – refers to error correction and system adaptation.

• Product Transition – distributed processing and rapid change in hardware [14].

TABLE II
MCCALL'S QUALITY MODEL

Quality Type	Product Perspective	Factors	Criteria
Quality Software Product	Product Operation	Correctness	Traceability
			Completeness
			Consistency
		Reliability	Accuracy
			Error-Tolerance
			Consistency
		Efficiency	Execution
			Storage
		Integrity	Access Control
			Access Audit
		Usability	Operability
			Training
			Communicativene
	Product Revision	Maintainability	Simplicity
			Conciseness
			Self-
			Modularity
		Testability	Instrumentation
			Self-
			Simplicity
			Modularity
		Flexibility	Simplicity
			Expandability
			Generality
			Modularity
	Product Transaction	Portability	Simplicity
			Software System
			Machine
		Reusability	Simplicity
			Generality
			Modularity
			Software System
			Machine
		Interoperability	Communications
			Data

C. BOEHM SOFTWARE QUALITY MODEL

The second of today's software quality model was developed by Boehm (1978), which emphasises on the maintainability for software product into McCall's Quality Model is known as Boehm's Quality Model [16] and is shown in Table III. Boehm proposed the contemporary shortcomings of models that automatically and quantitatively evaluate the software quality. The intermediate level characteristic addresses Boehm's quality factors that together represent the qualities which are expected from a software system. These qualities are:

1. Portability: Code possesses this characteristic to the extent that it can be easily operated and well on computer configurations except its current one.

*2. Correctness: U*se case scenarios are used to explore numerous functional requirements. However, the right interpretation of user vocabulary leads to correct set of valid requirements. Vague requirements like 'shall', 'multi-user', 'user-friendly', are thoroughly analysed as an input to design phase.

QC= Nv/(Nnv*N)

Where

Nv is number of valid requirements,

Nnv is number of still not valid requirements and

N is Total number of requirements

3. Completeness: It reflects the depth/ breadth of requirements in a scenario. Each requirement is unique and need to serve the user as a complete package. Though many requirement metrics are discussed and practiced, it is hard to quantify the quality of requirements metric.

TABLE III
BOEHM QUALITY MODEL

Quality Type	**Product Perspectiv**	**Factors**	**Criteria**
General Utility		Portability	Device-Independent
			Completeness
	As is utility	Reliability	Accuracy
			Completeness
			Consistency
		Efficiency	Device Efficiency
			Accessibility
		Human Engineering	Communicativeness
			Accessibility
	Maintainability	Testability	Communicativeness
			Accessibility
			Structuredness
			Self-
		Understandability	Consistency
			Structuredness
			Self-
			Conciseness
			Legibility
		Modifiability	Structuredness
			Augment-ability

V. ANALYSIS

For the comparison of the most renowned quality models, we have collected data from a number of organizations using questionnaires and interviewing many students personally. Questionnaires were circulated in many software companies and Govt/Semi-Govt institutions and feedback was received. We have also collected data from various journals, research papers, articles, periodicals etc. On the basis of the data collected, survey analysis is done which is shown below in Table IV.

TABLE IV
COMPARISON OF QUALITY MODELS

Quality Factors	**McCall**	**Boehm**	**ISO 9126**
Correctness	X	X	
Maintainability	X	X	
Reliability	X	X	Maintainability
Integrity	X	X	X
Usability	X	X	
Efficiency	X	X	X
Testability	X		X
Interoperability	X	X	Maintainability
Flexibility	X	X	
Reusability	X	X	
Portability	X	X	
Clarity		X	X
Modifiability		X	Maintainability
Documentation		X	
Resilience		X	
Understandability		X	
Validity			Maintainability
Functionality			X
Generality		X	
Economy		X	

VI. CONCLUSION AND FUTURE WORK

Software quality engineering desperately needs a quality model which can be used throughout the software lifecycle and which embraces all the perspectives of quality model, using the comparative analysis of existing quality models and their supportive Quality engineering. This analysis can be used to define "how to measure the functionality of the software and how we can their characteristics be improved". The increasing significance of software measurement has to lead to an increase amount of research on developing the new software measures. In this paper, we have presented the three basic quality models. They provide a basis for measuring many of the characteristics like size, complexity, performance, quality etc. This paper provides some help for researchers and practitioners for better understanding and selection of software metrics for their purposes. Future work can include comparison of other software quality models with realistic data which can be used to propose a new and rather efficient software quality model.

REFERENCES

[1] H F Li, W K Cheung "An Empirical Study of Software Metrics" Software Engineering IEEE Transactions on (1987) Volume: SE- 13, Issue: 6, Pages: 697-708
[2] N E Fenton "Software Metrics" Conference Proceedings of on the future of Software engineering ICSE 00(2000) Volume: 8, Issue: 2, Publisher: ACM Press
[3] Manik Sharma, Gurdev Singh, "Analysis of Static and Dynamic Metrics for Productivity and Time Complexity", IJCA, *Volume 30– No.1, September 2011*
[4] Manik Sharma, gurdev singh, "A Comparative Study of Static Object Oriented Metrics", IJoAT
[5] S.R Chidamber and C.F. Kemerer, "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering, Vol.20 No.6
[6] Bowen, Thomas P., Gary B. Wigle, and Jay T. Tsai, "Specification of Software Quality Attributes", RADC-TR-85-37, RADC, Griffiss Air Force Base, NY, Volumes I, II, and III, February 1985.
[7] Boehm, B. W. et al., Characteristics of Software Quality, North-Holland Publishing Company, New York, NY, 1978.
[8]. Gilb, T., Software Metrics, Winthrop, Inc., Cambridge, MA, 1977.
[9]. Walters, G. F., "Applications of Metrics to a Software Quality Management (QM) Program", Software Quality Management, Petrocelli, New York, NY, 1979.
[10] Kreitzberg, Charles B. and Ben Shneiderman, FORTRAN Programming: A Spiral Approach, Harcourt Brace Jovanovich, Inc., 1982.
[11] McCall, Jim A., Paul K. Richards, and Gene F. Walters, "Factors in Software Quality", RADC-TR-77-369, Volumes I, II, and III, RADC, Griffiss Air Force Base, NY, November 1977.
[12] Robert S. Oshana, Richard C. Linger, "Capability Maturity Model Software Development Using Cleanroom Software Engineering Principles - Results of an Industry Project," hicss, pp.7042, Thirty-second Annual Hawaii International Conference on System Sciences-Volume 7,1999.
[13] McCall, J. A., Richards, P. K., and Walters, G. F.,"Factors in Software Quality", Nat'l Tech.InformationService, no. Vol. 1, 2 and 3, 1977.
[14] C. V. Ramamoorthy, "*Evolution and Evaluation of Software Quality Models*," ictai, pp.543, 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02), 2002.
[15] J. A. McCall, P. K. Richards and G. F. Walters, "Factors In Software Quality - Concept and Definitions of Software Quality," *Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, New York*, Volume 1, Number 3, November, 1977.
[16] B. W. Boehm, J. R. Brown and M. Lipow, "Quantitative Evaluation of Software Quality," *IEEE Computer Society Press*, Page No.: 592 - 605, 1978.
[16]. R. Grady, D. L. Caswell, "Software Metrics: Establishing a Company-wide Program," *Prentice Hall*, 1987.
[17]. R. Grady, "Practical software metrics for project management and process improvement," *Prentice Hall*, 1992.
[18]. ISO/IEC 9126-1: Software Engineering - Product Quality- Part 1: Quality Model, *International Organization for Standardization*, Switzerland, 2001.
[19]. ISO/IEC 9126-2: Software Engineering - Product Quality- Part 2: External Metrics *International Organization for Standardization*, Switzerland, 2002.
[20]. ISO/IEC 9126-3: Software Engineering - Product Quality- Part 3: Internal Metrics, *International Organization for Standardization*, Switzerland, 2003.
[21]. R. G. Dromey, "A Model for Software Product Quality," *IEEE Transactions on Software Engineering*, Volume 21 Number 2, Page No.: 146 - 162, February 1995, *International Organization for Standardization*, Switzerland, 2002.
[22]. B. Al-Badareen, M. H. Selamat, M. A. Jabar, H. Din and S. Turarv, "Software Quality Model: A Comparative Study," *Springer ICSECS'11*, Page No.: 46 - 55, 2011.

Author Profile

Sheikh Fahad Ahmad
received B.Tech degree in Computer Science & Engg from Integral University, Lucknow in 2008 and currently pursuing M.Tech degree in Computer Science & Engg from Integral University, Lucknow. From 2010 onwards he is working as a Lecturer in the department of Computer Science & Engg, Integral University, Lucknow, India.

Prof. (Dr.) Mohd Rizwan Beg
received B.Tech degree in Computer Science & Engg in 1995 and completed his M.Tech and PhD degree in Software Engineering. Currently he is working as the Dean of Computer Application Deptt and Head of Computer Science & Engg Deptt, Integral University, Lucknow. He is also the member of Editorial Boards, Program and Technical Committees of many International Journals as well as he is in the advisory committees of many International Journals. Currently he is supervising eight candidates in PhD Program and he has around 78 International Publications.

Mohd. Haleem
received his BSc degree from Aligarh Muslim University in 2007 and MCA degree from HBTI, Kanpur in 2009 and currently pursuing M.Tech degree in Computer Science & Engg from Integral University, Lucknow. From 2011 onwards he is working as a Lecturer in the department of Computer Application, Integral University, Lucknow, India.