Review Paper on Software Quality Factors

[1]Ahmed Saleem Abbas
ahmed_saleam@uobabylon.edu.iq
Software department, College of Information technology, University of Babylon

[2]Ashwak Alabaichi
Faculty of Engineering, University of Karbala
ashwaq.alabaichi@gmail.com

[3]Elaf Ali Abbood
Computer Department, Science College for Women, University of Babylon
wsci.elaf.ali@uobabylon.edu.iq

Abstract: Software quality factors are an important and ideal measure of the quality of software, in addition to being a good indicator of the need for software development and maintenance. Software quality factors are gaining importance and acceptance in the corporate sectors where organizations grow in nature and strive to improve the quality of the organization, while quality measures are quantitative measures of the degree to which a program addresses a particular feature that affects their quality. In this paper, we will make a survey on Software Quality factors and categories these Factors into three important classes: Product operation, revision, and transition factors where these factors were declared by McCall’s and named as “McCall’s Factor Model”. There are another two Factor models we will study, they are “Deutsch and Willis Factor Model” and “Evans and Marciniak Factor Model”, by using these factors we can evaluate the software Product quality and make a decision is good or bad.

Keywords: Software engineering, software quality factors, software requirements, McCall’s Model, software evaluation.

1. Introduction

Software quality is an important criterion in evaluating software [1]. Software quality assurance is a major problem for both software engineers and users. Due to the fact that there are many factors that affect the quality of the software, such as incompatibility with the type of device, memory, system used, etc. This greatly affects the success of the program or vice versa. It is therefore necessary to find ways to increase the quality of programs and reduce errors. Software developers and designers in organizations continue to conduct many kinds of research and processes to evaluate and document the quality of their products before marketing them to users, as well as to periodically inspect their software products during each stage of the software life cycle, looking for methods and standards that help to increase the degree of quality. It is well known that the quality of programs depends on various factors on which to evaluate the quality of the programs produced, such as factors that are directly measured, such as logical errors, and factors that are measured indirectly [2]. In this paper, we will discuss three models to measure the quality of programs: McCall’s Factor, Evans and Marciniak Factor Model, and Deutsch and Willis Factor Model.

2. Related works

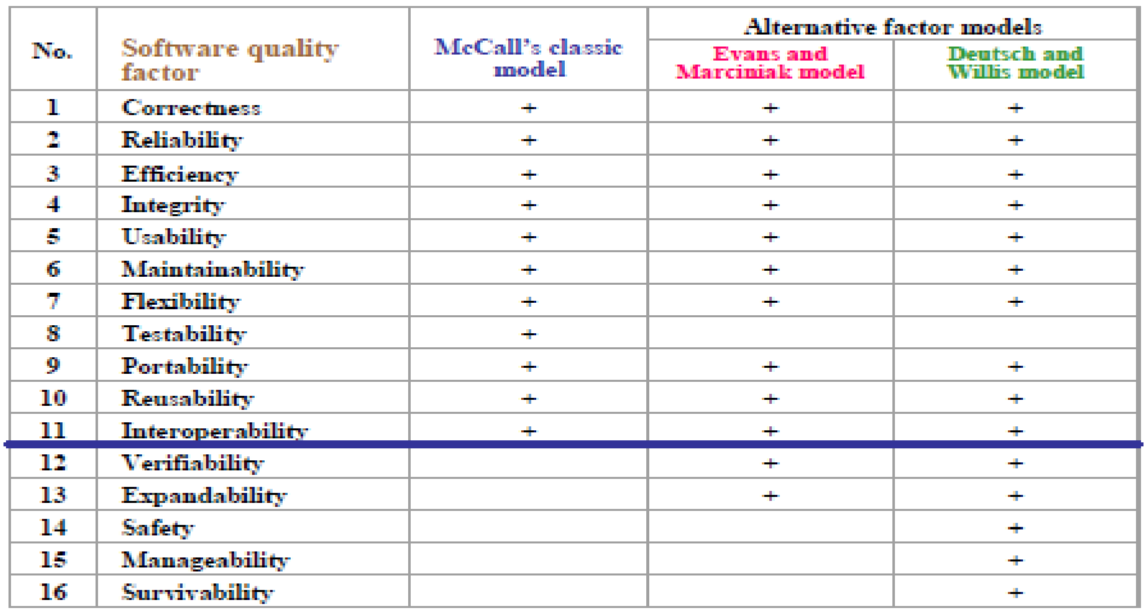
Many studies and a review paper had been conducted to define the different types of software quality measures and their factors models.
Boukouchi et al 2013[19] discussed many types of software quality models and comparing them with each other. The models consisted of many models like Mc Call’s, BOEHM’s, Evans & Marciniak, Deutsch & Willis, ISO 9126’s, and DROMY’s. The comparison between them introduced characteristics that are common between these quality models and other characteristics are spatialized in some classes of these models and explained that in structured tables.

José P. et al 2014[20] introduced a description for the quality models depending on two main categories: basic quality models like Mc Call, Boehm, and Dromey and tailored quality models like Bertoa, Alvaro, Quamoco, and Midas. They tested these quality measures on different applications and found that the communication aspect is a significant element in quality software.
Saba Awan and et al. 2015[21] compared McCall's, FURPS, ISO 9126, Evans and Marciniak, IEEE's and Deutsch & Wills Quality Model and they concluded a special characteristic. They introduced a framework and comparison Methodology that considered the sub-factors in addition to the elements to prove satisfactory and particular compares between quality models. This comparison of these software quality models disclosed the distinctiveness among these models. The evaluation of methodology was performed by joining between quality characteristics from different models and sub-factors of a specific factor. As well as, they removed the redundant according to their description. Then they introduced a quantitative analysis of existing systems against their approach.
Inda and Rosini 2018[22] tested the software quality of the e-SAP application using the Mc Call model in 5 dimensions that included correctness, usability, efficiency, reliability, and integrity. The test is done on users randomly. The research concluded that these dimensions of measures get a good result for this application with a good product operation value.

Nebi and Ayca 2020[23] introduced a semantic review for meta models that consist of many types of software quality models and their evaluation. They search in the most-known seven digital libraries, and many studies were filtered out of 114 studies initially retrieved between 1997-2020 in this area. They concluded that most meta-models are for general purposes, take ISO 9126 as a reference, and suggested different categories of software. Most of them evaluate quality objectively using metric data and provide quantitative results. The majority of them are structured to enable the extension with new quality models.

Hamed Fawareh 2020[24] introduced a study about software maintenance and its failure and success and developed the software quality models to reduce the failure software for specified maintenance purposes. This did by comparing the quality model factors from maintenance aspects that form the software quality factors, sub-factors, and criteria that affect the software failure and success. Also, he used function points as a technique for computing the size and productivity of software systems. It is also used to calculate the size and complexity of applications based on outputs, inputs, queries, internal files, and interfaces. Also, he introduced a set of rules that considered evaluating the software system quality regarding the maintenance factors.

Table 1: models for measure quality of software product



No.	Software quality factor	McCall's classic model	Alternative factor models	
			Evans and Marciniak model	Deutsch and Willis model
1	Correctness	+	+	+
2	Reliability	+	+	+
3	Efficiency	+	+	+
4	Integrity	+	+	+
5	Usability	+	+	+
6	Maintainability	+	+	+
7	Flexibility	+	+	+
8	Testability	+		
9	Portability	+	+	+
10	Reusability	+	+	+
11	Interoperability	+	+	+
12	Verifiability		+	+
13	Expandability		+	+
14	Safety			+
15	Manageability			+
16	Survivability			+

3.1 McCall's Factor Model

As stated by McCall, eleven factors represent the main classes of software requirements. These factors are grouped into three categories which are next.

Table 2: categories of McCall's Model

	Product Factors Category		
	Operation	**Revision**	**Transition**
work with	The requirements, which affect the daily operation of the software directly.	The product revision.	The adaptation of software to interaction with other software and other environments.
have	Reliability, correctness, efficiency, usability and integrity.	Testability, flexibility and maintainability	Reusability, interoperability and portability.

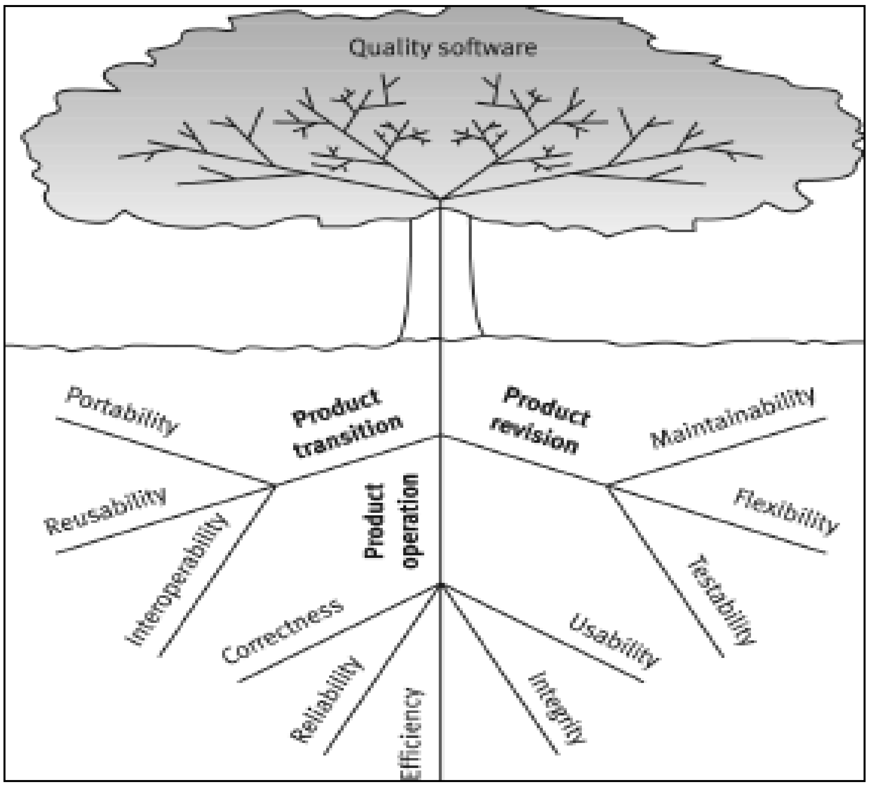


Fig. 1. Categories of McCall's Model

3.1.1 Product Operation Factors

a. Correctness

"That mean a program have and satisfy all its specification." [3]. Correctness is the capability of software products to perform the exact operations that were designed to perform it [4]. The software is called "correct" if it makes the correct results for each input that may be entered. Thus, the testing is very important but at the same, it is consuming a lot of time in programming. The problem is that the set of all possible inputs to a program is extremely large. Therefore, whenever possible, we would prefer to make sure that the software is correct. A proof of correctness needs two parts [5]:-

1. Give a guaranty that the correct output has resulted if the program is completed successfully.
2. Give a guaranty that the program will successfully complete usually.

<u>Example</u>: -

On the fourth of June 1996, one flight ended in failure (Ariane 5 rocket); about 40 seconds after launch it exploded. The reason for this was due to a program reuse error of a procedure that responsible for integer conversion [6].

b. Reliability
"That means a program capable to be maintained so that it can perform its exact function."[3]. Reliability requirements deal with software product failure. They specify the max failure rate permitted of the software product over a period of time and can refer to one module or to the entire system [7].
Reliability Metrics:

- **Mean time between failures**: the average time between sequential failures on a given interval within the system life [7].
- **Mean time to repair**: the average time to maintain or repair equipment [7].
- **Mean time to recover**: the average time to operate a system correctly after a failure [7].
- **Probability of failure**: apply methods to estimate the probability of the system behavior in an expected manner under certain circumstances. The probability of failure is most proper to continuous running systems and safety-critical systems [7].

Example: -
In Independence Bank, exactly its main branch, where this main branch operates one hundred and twenty branches. A requirement of its new software is that it will not fail, on average, more than ten minutes per month through the bank's office hours [8].

c. Efficiency
"The number of computer resources (e.g. external storage or time) or code required for a program to complete its function" [3]. It works with the needed h/w resources to do the various tasks of the software system. It has storage amount, capabilities of processing, and data communication ability. There are many techniques to develop a software product with good efficiency [7]: -

- **Design:** Strategies to enhance cohesion and reduce coupling, normalization methods to minimize data redundancy, and algorithms that enhance process time should be used [7].
- **Operating systems:** latest operating systems have the capability to apply multi-tasking thereby enhancing system performance through facilitating its background operations [7].
- **Programming languages:** Selecting the most suitable programming language for the problem has a major effect on the software product efficiency [7].
- **Access strategies:** Algorithms, which enhance seek time, data transfer time, and rotational delay should always search out and implemented to increase efficiency.
- **Programming techniques:** there are many good programming techniques and practices that are stated as typical techniques, like keep local variables inside procedures, top-down design for complicated problems, and perfect use of parameter passing [7].

Example: -
"A chain of stores is considering two various bids for a computer based system. Each bid encompasses identical computers within the chain's headquarters and its branches. The bids have some differences on storage volume: twenty GB per branch pc and one hundred GB within the head workplace pc (Bid A); ten GB per branch pc and thirty GB within the head workplace pc (Bid B). There's conjointly a distinction within the range of communication lines required: Bid A consists of three lines of communication has twenty-eight-point eight KBPS between every branch and also the head workplace, whereas Bid B is predicated on two lines of communication of identical capability between every branch and also the head workplace. During this case, it's clear that Bid B is a lot of efficient than Bid A as a result of fewer hardware resources are needed" [8].

d. Usability
This means the requirements for staff to execute the software-based system and a new staff member training on this system.
Example: -
In the Home Appliance Services Company, the document of system usability requirements for the junior helper's office should contain these specifications: -
1. A staff member must be capable to handle at least sixty service calls in a day.
2. Two days only to Train a new employee, immediately at the end of these couple days the trainee will have the ability to handle forty-five service calls in a day.

e. Integrity

Dealing with system protection requirements. Means the requirements of the authorized person. To distinguish between those who are allowed to read information and those who are allowed to modify this information.

Example: -

Geographic Information System allowed people access to its information by the Internet; it is just to copy and view data but not to make changes.

3.1.2 Product revision factors: -

a. Maintainability

Including identifying the causes of failure of the system and the efforts required by the maintenance staff and users, correcting the fail, and verifying the success of the correction. This factor's requirements refer to the modular structure of the program, the internal software documentation, and the manual of the programmer.

Example ideal requirements of maintainability: -

The software module size must not exceed thirty statements.

The programming will abide by the company guidelines and coding standards.

b. Testability

Testability requirements need examining computer-based systems and their operation in order to ensure if the system components are properly working and to get a report about discovered errors [3].

When the software system testability is high, then finding faults will be easier by a testing means. In a formal way, some systems can be tested, and some cannot. There are many untestable software systems, or not directly testable.

Example: -

One of the Google's s software system named as "ReCAPTCHA", have no metadata about the images, so it is not a testable system. However, it can be directly tested if there is a tag stored elsewhere for each shown image [9].

c. Flexibility

Flexibility is an attribute that makes the system able to add new features easily, therefore Flexibility helped to reduce the cost of an operational system adjustment. We can say this system had high flexibility whenever we can be added a new feature more easily [3].

Example: -

Object-oriented programming always enables us to write Flexible programs.

3.1.3 Product transition factors:

a. Portability

Portability means software program adaptation with other environments such as different operating systems & different hardware ... etc. in another way we must continue in using the same software in different circumstances [3]. Since the age of good software may equal to or more than fifteen years while for H/W is changed every four years so the Portability became essential [10].

b. Reusability

It points to the capability of a program to be used again in simple several contexts the property condition or quality to be simple machine and program independence. The independence of the S/W or H/W to be utilized freely in every system [11].

Example: -

The code can be reused instead of writing it again through passing different new arguments C language, VB, and OOP development techniques which support the ability of re-use [12]. "Hong Zhu also supports that reusability depends on the generality of the components in a given application domain and the extent to which the components are parameterized and configurable".

c. Inter-operability

The endeavors required to combine one system with another system. Modularity: it is the degree to system's components can be combined and isolated. It makes a difference in the implementation Pareto Guideline that states that eighty percent of impacts came from twenty percentage of areas. Communality ownership of popular highlights or features [13],[14]. Any component interacting with another component successfully. Like word processors communicate successfully with a spreadsheet [15].

Example: -

Take an example of reserving an airway flight. Consider you need to travel from Bagdad to New York. There is no direct flight. You have to travel from Bagdad to Cairo and then take another flight from Cairo to New York. Because there are time constraints, you reserve your flight from Bagdad to Cairo in “Iraqi Airways” and from Cairo to New York in “Virgin Atlantic”. Therefore, that means all your passenger information details were transferred from “Iraqi Airways” to “Virgin Atlantic”. So here, both airways company’s system has independent applications altogether and while reserving your flight, your details of booking were exchanged from Iraqi Airways to Virgin Atlantic in a meaning full way, without previously alert [16].

3.2 Evans and Marciniak Factor Model

a. Verifiability: Identify design and features of programming that enable design and programming verification efficiently of the (simplicity, modularity, commitment to documentation, and program guidelines) [17].

b. Expandability:
Indicate any efforts in the future that will be required to serve a greater population, services improving, or adding new applications for usability improvement. In another mean, “expandability” refers to the ability to scaled and extended to provide more usability. This is similar to McCall’s flexibility [17]. The paper “Software Quality and Productivity Model for Small and Medium Enterprises” is an example of using expandability and verifiability factors [18].

3.3 Deutsch and Willis Factor Model

a. Safety:
Refer to the elimination of hazardous conditions to the instrument because of any error in the process of controlling the SW program [17].
Example: -
In a chemical extract plant, a computer-based system controls the acids flowing based on temperature and pressure changes that occur during production.

b. Manageability:
Identify the tools that the admin uses to support software product modification during the periods of development & maintenance.

c. Survivability:
Clarify the service continuity that specifies the least time permitted between the system fail, and the largest time allowed to recover services.

4. Conclusion

From this work, we have concluded that software quality factors are a critical problem to enhance the overall quality of any software product. These factors can be categories into three main categories, each responsible for some aspects of the validity and success of the software. From all these factors, we can evaluate the product quality of the software and decide whether it is beneficial to the customer or not. In addition to the possibility of diagnosing programs that need maintenance and development, for the purpose of increasing the efficiency of programs, with a focus on everything that contributes to raise the quality ratio.

Acknowledgment

Thanks to my professors and colleagues at the University of Babylon for their hard work in helping me prepare this article, I also thank the researchers involved in the preparation of this article, and finally thanks to Google for making the observations accessible to researchers and to facilitate searches and conduct this study.

الملخص: تعتبر عوامل جودة البرمجيات مقياسًا مهمًا ومثاليًا لجودة البرمجيات ، بالإضافة إلى كونها مؤشرًا جيدًا على الحاجة إلى تطوير البرمجيات وصيانتها. تكتسب عوامل جودة البرامج أهمية وقبولًا في قطاعات الشركات حيث تنمو المنظمات بطبيعتها وتسعى جاهدة لتحسين جودة المنظمة ، في حين أن مقاييس الجودة هي مقاييس كمية للدرجة التي يعالج بها البرنامج ميزة معينة تؤثر على جودتها. في

هذه الورقة ، سنقوم بإجراء دراسة مراجعة حول عوامل جودة البرامج وتصنيف هذه العوامل إلى ثلاث فئات مهمة: تشغيل المنتج ، والمراجعة ، وعوامل الانتقال حيث تم الإعلان عن هذه العوامل بواسطة McCall's وتسمى "نموذج عامل McCall". هناك نوعان آخران من نماذج العوامل التي سنقوم بدراستها ، وهما "نموذج عامل ألمانيا وويليس" و "نموذج عامل إيفانز ومارسينياك" ، باستخدام هذه العوامل يمكننا تقييم جودة منتج البرنامج واتخاذ قرار ما إذا كان جيدًا أم سيئًا.

References

[1] M.-C. Lee, “Software quality factors and software quality metrics to enhance software quality assurance,” *Curr. J. Appl. Sci. Technol.*, pp. 3069–3095, 2014.
[2] K. Khosravi and Y.-G. Guéhéneuc, “A quality model for design patterns,” *Ger. Ind. Stand.*, 2004.
[3] J. A. McCall, P. K. Richards, and G. F. Walters, “Factors in software quality. volume i. concepts and definitions of software quality,” GENERAL ELECTRIC CO SUNNYVALE CA, 1977.
[4] R. S. Pressman, *Software engineering: a practitioner’s approach*. Palgrave Macmillan, 2005.
[5] J. A. Whittaker, “What is software testing? And why is it so hard?,” *IEEE Softw.*, vol. 17, no. 1, pp. 70–79, 2000.
[6] M. Frentiu, “Correctness: a very important quality factor in programming,” *Stud. Univ. Babes-Bolyai, Ser. Inform. L*, pp. 11–20, 2005.
[7] R. Fitzpatrick, “Software quality: definitions and strategic issues,” 1996.
[8] D. Galin, *Software quality assurance: from theory to implementation*. Pearson Education India, 2004.
[9] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, “recaptcha: Human-based character recognition via web security measures,” *Science (80-.).*, vol. 321, no. 5895, pp. 1465–1468, 2008.
[10] J. A. McCall, “Quality factors,” *Encycl. Softw. Eng.*, 2002.
[11] G. Caldiera and V. R. Basili, “Identifying and qualifying reusable software components,” *Computer (Long. Beach. Calif).*, vol. 24, no. 2, pp. 61–70, 1991.
[12] S. Mittal and P. K. Bhatia, “Software component quality models from ISO 9126 perspective: A review,” *IJMRS’s Int. J. Eng. Sci.*, vol. 2, no. 2, 2013.
[13] A. I. Salih, A. Alabaichi, and A. S. Abbas, “A novel approach for enhancing security of advance encryption standard using private XOR table and 3D chaotic regarding to software quality factor,” *ICIC Express Lett. Part B Appl. An Int. J. Res. Surv.*, vol. 10, no. 9, pp. 823–832, 2019.
[14] B. Habib, “Compilation of Software Quality Factors and Criteria along with their Description for a Quality Product,” *Int. J. Comput. Appl.*, vol. 84, no. 3, 2013.
[15] J. Pande, R. K. Bisht, D. Pant, and V. K. Pathak, “On some quality issues of component selection in CBSD,” *J. Softw. Eng. Appl.*, vol. 3, no. 06, p. 556, 2010.
[16] A. P. Sheth, “Changing focus on interoperability in information systems: from system, syntax, structure to semantics,” in *Interoperating geographic information systems*, Springer, pp. 5–291999.
[17] F. Rahmawati, M. A. Musyafa, M. D. Fauzi, and A. Mulyanto, “Quality Testing of Order Management Information System Based on Mccall’s Quality Factors,” *IJID (International J. Informatics Dev.*, vol. 5, no. 2, pp. 12–20, 2019.
[18] J. H. Yahaya, A. Deraman, A. Tareen, and A. R. Hamdan, “Software Quality and Productivity Model for Small and Medium Enterprises,” Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 5, pp. 316–320, 2017.
[19] Boukouchi Youness, "Comparative Study of Software Quality Models", IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 6, No 1, November 2013
[20] José P. Miguel et al, "A Review of Software Quality Models for the Evaluation of Software Products, "International Journal of Software Engineering & Applications (IJSEA)", Vol.5, No.6, November 2014.
[21] Saba Awan et al, " An Efficient and Objective Generalized Comparison technique for Software Quality Models," I.J. Modern Education and Computer Science(MECS), 12, 57-64, 2015.
[22] Inda D Lestantri and Rosini , “Evaluation of Software Quality to Improve Application Performance Using Mc Call Model”, Journal of Information Systems Engineering and Business Intelligence (JISEBI), Vol. 4, No. 1, April 2018.
[23] Nebi Yılmaz, Ayça Kolukısa - Tarhan, “ Meta-models for Software Quality and Its Evaluation: A Systematic Literature Review”, The 30th International Workshop on Software Measurement (IWSM) and the 15th International Conference on Software Process and Product Measurement (MENSURA), 2020.
[24] Hamed Fawareh, “Software Quality Model for Maintenance Software Purposes”, International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 13, pp. 158-162, No. 1, 2020.