Software Quality Models: A Comparative Study

Anas Bassam AL-Badareen, Mohd Hasan Selamat, Marzanah A. Jabar, Jamilah Din, and Sherzod Turaev

Faculty of Computer Science and Information Technology
University Putra Malaysia
Anas_badareen@hotmail.com,
{hasan,marzanah,jamilah,sherzod}@fsktm.upm.edu.my

Abstract. In last decade, researchers have often tried to improve the usability, portability, integrity and other aspects of software in order for it to be more users friendly and gain user trust. Several approaches and techniques have been proposed to reduce the negative effects of software size and complexity. Moreover, several software quality models were proposed to evaluate general and specific type of software products. These models were proposed to evaluate general or specific scopes of software products. The proposed models were developed based on comparisons between the well-known models, in order to customize the closed model to the intended scope. These comparisons are leak of criteria that is conducted based on different perspectives and understanding. Therefore, a formal method of comparison between software quality models is proposed. The proposed method is applied on a comprehensive comparison between well-known software quality models. The result of the proposed method shows the strength and weaknesses of those models.

Keywords: Quality Model, Model Comparison, Model Development.

1 Introduction

US Air force Electronic System Division (ESD), the Rome Air Development Centre (RADC) and General Electric [1] intends to improve the quality of the software products and to make it measurable. therefore, McCall [2] model was developed in 1976-7, which is one of the oldest software quality models. This model started with a volume of 55 quality characteristics which have an important influence on quality, and called them "factors". The quality factors were compressed into eleven main factors in order to simplify the model. The quality of software products was defined according to three major perspectives, product revision (ability to undergo changes), product transition (adaptability to new environments) and product operations (its operation characteristics).

Since McCall model was proposed, new factors have been added to the original and some of them are redefined [3]. Second model was defined is Boehm model[4], the model was based on McCall model, he defined the second set of quality factors. SPARDAT is a commercial quality model was developed in the banking environment. The model classified three significant factors: applicability, maintainability, and adaptability.

J.M. Zain et al. (Eds.): ICSECS 2011, Part I, CCIS 179, pp. 46–55, 2011.

Nowadays, several software quality models were proposed in order to evaluate general and specific software quality products, they were developed based on well-known models, such as McCall, Boehm, FURPS, Dromey, and ISO. The method of develop a software quality models is started based on comparisons between selected well-known models in order to customize the closed model to the intended scope, such as [5-9]. The comparisons were conducted based on different perspectives and understanding, and at the factors level. Therefore, a contradiction of the software quality factors definition is occurred.

Table 1. Sample of Models Comparisons

Quality Factors	Hamada, 2008 [3]			Rawashdeh, 2006 [4]			Haiguang, 2008 [7]		
	McCall	Boehm	ISO	McCall	Boehm	ISO	McCall	Boehm	ISO
Integrity	×	×	×	×		×	×	×	
Efficiency	×	×	×	×	×	×		×	Maintenance
Reusability	×	×		×			×	×	
Changeability		×	Maintenance		×			×	Maintenance
Testability	×		Maintenance	×	×	×	×		

Table 1 shows sample of comparisons were conducted between same models and same factors, but with a different results. It shows that the researchers were conducted the comparisons based on different points of views. For example, Hamada [5] shows that the integrity is included in McCall, Boehm, and ISO models. Rawashdeh [6] shows that the integrity in included under McCall and ISO. Haiguang [9] shows that the integrity is included in McCall and Boehm. Hamada [5] and Rawashdeh [6] show that the efficiency is included in all of the selected models, whereas Haiguang [9] shows that it is included in Boehm as a main factor and in ISO a sub factor under the maintainability.

However, the inconsistency in the definitions of software quality factors results contradictions in the developed models. Therefore, different software quality models are developed intends to achieve same goals of software quality evaluation. Thus, in this study we propose a method of comparison based on the analysis and discussion of four well-known software quality models. Moreover, we analyzed and compared several comparisons were conducted between these models and we identified the main differences between them.

This study intends to develop a formal method that can be used to compare and differentiate between software quality models mathematically. That will help to avoid any contradictions that may occur during development. Moreover, it helps to define a standard basic for developing a software quality model.

In section two, we present the well-known software quality models. In section three, we discuss and describe the proposed method. In section four, we describe the case study and how the weight is assigned. In section five, we present and discuss the result. Section seven present the conclusion and future work.

2 Quality Models Background

Whereas, several software quality models are proposed, in order to evaluates different types of software products. This section presents the most popular software quality models, were considered in different studies.

2.1 McCall Model

McCall's model [2] was developed by the US air-force electronic system decision (ESD), the Rome air development center (RADC), and general electric [1], to improve the quality of software products. This model was developed to assess the relationships between external factors and product quality criteria. Therefore, the quality characteristics were classified in three major types, 11 factors which describe the external view of the software (user view), 23 quality criteria which describe the internal view of the software (developer view), and metrics which defined and used to provide a scale and method for measurement. The number of the factors was reduced to eleven in order to simplify it. These factors are Correctness, Reliability, Efficiency, Integrity, Usability, Maintainability, Testability, Flexibility, Portability, Reusability, and Interoperability. The major contribution of this model is the relationship between the quality characteristics and metrics. However, the model not consider directly on the functionality of software products.

2.2 Boehm Model

Boehm [4] added new factors to McCall's model and emphasis on the maintainability of software product. The aim of this model is to address the contemporary shortcomings of models that automatically and quantitatively evaluate the quality of software. Therefore, Boehm model represents the characteristics of the software product hierarchically in order to get contribute in the total quality. Furthermore, the software product evaluation considered with respect to the utility of the program. However, this model contains only a diagram without any suggestion about measuring the quality characteristics.

2.3 FURPS Model

FURPS model was proposed by Robert Grady and Hewlett-Packard Co. The characteristics were classified into two categories according to the user's requirements, functional and non-functional requirements. Functional requirements (F): Defined by input and expected output. Non-functional requirements (URPS): Usability, reliability, performance, supportability. And then, the model was extended

by IBM Rational Software – into FURPS+. Therefore, this model considered only the user's requirements and disregards the developer consideration. However, the model fails to take into account the software some of the product characteristics, such as portability and maintainability.

2.4 Dromey Model

Dromey (1995) [10] states that the evaluation is different for each product, hence a dynamic idea for process modeling is required. Therefore, the main idea of the proposed model was to obtain a model broad enough to work for different systems. The model seeks to increase understanding of the relationship between the attributes (characteristics) and the sub-attributes (sub-characteristics) of quality. This model defined two layers, high-level attributes and subordinate attributes. Therefore, this model suffers from lack of criteria for measurement of software quality.

2.5 ISO IEC 9126 Model

Since, the number of the software quality models were proposed, the confusion happened and new standard model was required. Therefore, ISO/IEC JTC1 began to develop the required consensus and encourage standardization world-wide. The ISO 9126 is part of the ISO 9000 standard, which is the most important standard for quality assurance. First considerations originated in 1978, and in 1985 the development of ISO/IEC 9126 was started.

In this model, the totality of software product quality attributes were classified in a hierarchical tree structure of characteristics and sub characteristics. The highest level of this structure consists of the quality characteristics and the lowest level consists of the software quality criteria. The model specified six characteristics including Functionality, Reliability, Usability, Efficiency, Maintainability and Portability; which are further divided into 21 sub characteristics. These sub characteristics are manifested externally when the software is used as part of a computer system, and are the result of internal software attributes. The defined characteristics are applicable to every kind of software, including computer programs and data contained in firmware and provide consistent terminology for software product quality. They also provide a framework for making trade-offs between software product capabilities.

3 The Comparison Method

In order to show the clear differences between software quality models, mathematical comparison method is proposed. The method aims to show the clear and accurate differences between quality models, which consider the sub factors in addition to the factors. It consists of four main tasks: model selection, assigning values, factors comparison, and models comparison.

Models Selection: The process of models selection is depends on the scope intended to be evaluated, usually the well-known software quality models are considered in developing a new model.

Factors Selection: The factors of the selected models are collected and combined in one structural tree (Fa, Fb…Fn) (*See figure 1*).

Different sub-factors are considered in each factor from different model, the sub factors are combined under their factors (S1, S2, Sn). According to the aim and definitions of each factor and sub factor, the repeated are excluded.

Factors Weighting: After analyze the scope that needed to be evaluated, the experts in this field are required to assign the weight of these factors (W1, W2…….Wn) and sub factor (Wa, Wb……Wm) are assigned.

Factors Values: the value of each factor in the original models is calculated, based on the weight that assigned in the previous step.

- First, the value of the same factor within the selected models is calculated (*Formula 1*).
- Second, the total value of each model is calculated (*Formula 2*), based on the calculated values of their factors.

The Comparison: the total value for each factor is compared between the selected models. That shows the comprehensiveness differences between these factors in different models.

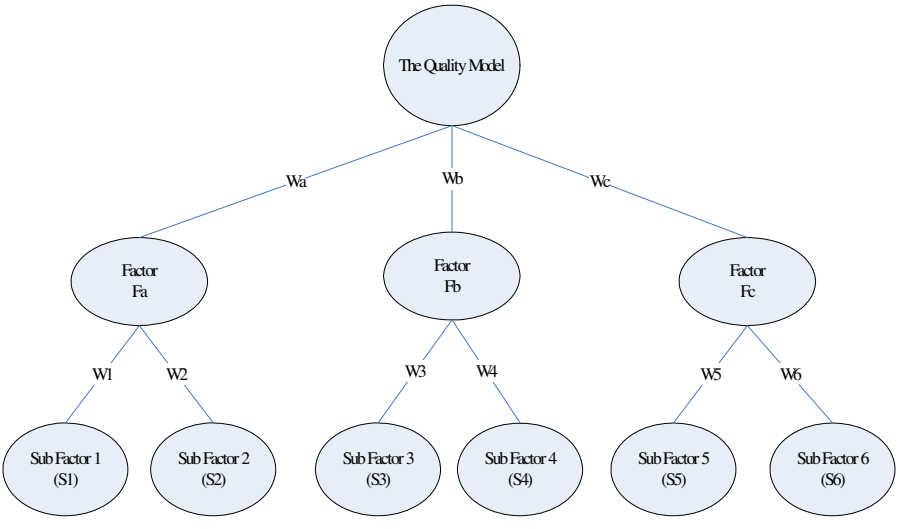

Fig. 1. The Model Weighting

$$F = \sum_{1}^{n} Wn \tag{1}$$

$$Qm = \sum_{1}^{m} Fm \times Wm \tag{2}$$

$$Qm = \sum^{m} Wm \times \sum^{n} Sn \tag{3}$$

4 Case Study

The case study considered a general comparison between most popular software quality models. The comparison shows the main differences between these models. The following steps are followed in order to perform the task:

Step 1: combine the factors of the selected models and remove the repeated
Step 2: combine the sub-factors for each factor
Step 3: assign the weight for each factor
Step 4: assign the weight for each sub factor
Step 5: calculate the weight for each factor in every model independently
Step 6: compare the values of same factors in all of the selected models

Whereas, the comparison is a general term comparison, the weight of the software characteristics is considered equivalently. While, in order to compare the models for specific scope, the expertise weighting is very important to show the most model enclose to the scope and which model emphasize on the characteristics of the intended scope.

A simple formula is used to assign the values for these attributes. In order to present whether the characteristic is considered as a factor, 50% was given to the factor and 25% to the sub factor. For the rest of the percentage it was equivalently divided between the numbers of the sub factors included in the comparison.

5 Result and Discussion

The first step of this study was to collect the factors that included by selected models and remove the repeated according to the definition of each of them. The second step is combining the sub-factors from all of the models for specific factor. The repeated sub characteristics were removed according to the definition of each of them.

The values were seated equivalently which gave 50% of the value to present whether the factor is included in the model, whereas 25% was given if the characteristic is included as a sub factor. Because of the generality of this comparison which not considered any type of software or any specific software domain, the value of the factors are same. Therefore, the second 50% was divided between the sub factors that included by the selected factor equivalently, where the 50 is divided into the number of the sub factors were collected for the factor from those models.

Finally, in order to calculate the value for each model, the values for each factor within the same model are calculated according to the same formula that was used to calculate the values of the factors. Table 2 presents the total value for each model, whereas figure 2 shows the graphical presentation of these values. The total value was

decomposed between those factors equivalently, where each factor was given 7.14% from the total quality of the model. Figure 3 shows the difference between those models per factor.

McCall model is a hierarchical model with two levels, the models has many to many relationships. This model considered the most of the software product characteristics, except the functionality of the software and the human engineering, whereas software understandability is covered implicitly through the sub-characteristics that required for

Table 2. The total value of the software quality models

Factor/Model	*ISO*	*McCall*	*Boehm*	*FURPS*	*Dromey*
Efficiency	80%	70%	55%	25%	50%
Integrity	25%	100%	25%	25%	0%
Reliability	70%	65%	50%	65%	50%
Usability	63%	60%	60%	73%	0%
Correctness	0%	100%	0%	0%	50%
Maintainability	73%	68%	64%	0%	50%
Testability	25%	78%	53%	0%	0%
Changeability	25%	83%	42%	0%	0%
Interoperability	25%	100%	25%	0%	0%
Reusability	0%	100%	0%	0%	50%
Portability	78%	67%	61%	0%	50%
Functionality	86%	0%	0%	71%	50%
Understandability	0%	0%	25%	0%	0%
Human Engineering	0%	0%	75%	25%	0%
Total	**39.27%**	**63.67%**	**38.19%**	**20.34%**	**25.00%**

other factors such as maintainability, flexibility, and reusability. Furthermore, there is no any mention about the human characteristics, which is important to identify the usability characteristic for the software product. Moreover, the model is the highest model which cover the most of the software product characteristics, whereas lake to the relationships between the factors, which cause overlapping in its relations.

Boehm model is similar to McCall model, which is hierarchical structure. This model focused on the structure of the quality characteristics, whereas several characteristics are dropped from this model such as software correctness, reusability, in addition to the functionality. Furthermore, this model considered new quality characteristics there were not included in McCall model such as Human Engineering and developer understandability. The model is very success in software maintainability relations, which reduce the overlapping and represent perfectly the meaning of the software maintainability. At the same time, the does not consider the software from the end user perspective and just be content with the developer perspective.

FURPS model is a two level hierarchical model with one to many relationships. This model evaluates the software products only from the user's viewpoint, whereas no any mention from the developer viewpoint such as maintainability and reusability. However, this model succeed to cover the user concerned in the software product, where is missed the software portability. Furthermore, the evaluation on this model considered also the supportability facilities that required for operating the system properly, which are not considered any model else.

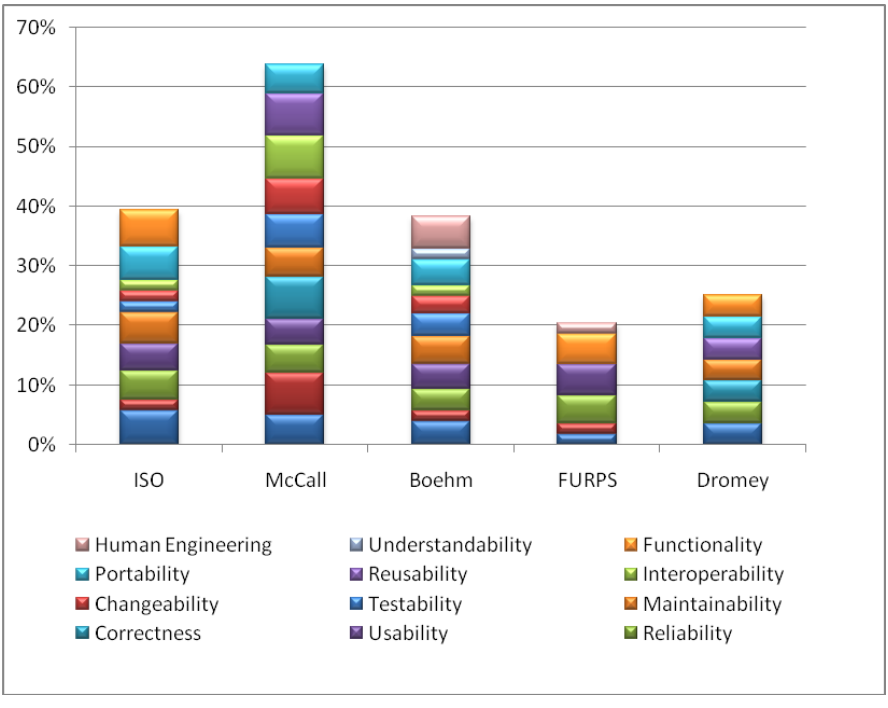

Fig. 2. Total Values Comparison

Dromey model is a dynamic software quality model which aims to evaluate different type of software products. The model considered four levels of software products, Correctness, Internal, Contextual, and Descriptive. Moreover, with each level the software quality attributes are considered. Whereas, the model suffer from the lack of criteria for measure the software quality. Furthermore, the model does not consider the usability of the software.

ISO model which proposed to overcome the confusing of the software quality models, considered six main quality factors related to twenty one criteria. However, this model in addition to the generality the model lakes to several factors were considered in previous models such as software correctness, reusability, and human engineering.

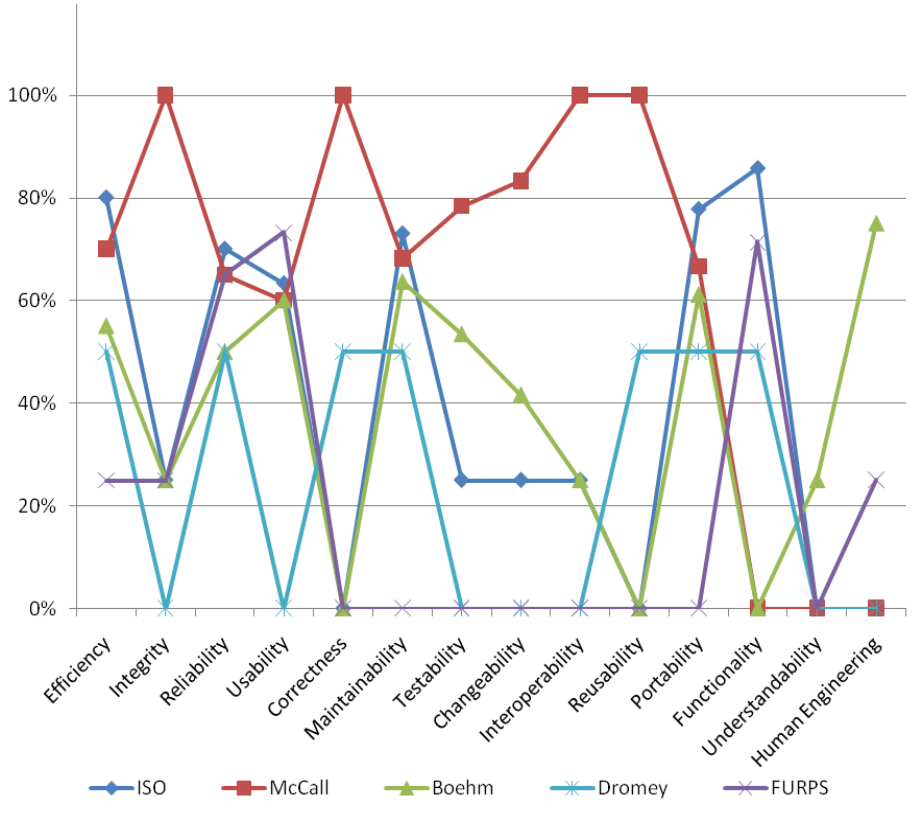

Fig. 3. Models Comparison

6 Conclusion

In this paper, well-known software quality models were presented. Hence, each model was discussed in details, the advantages and disadvantages were expressed. Finally, comprehensive comparison between the selected models was presented. The comparison goes behind the definitions of the software quality factors into sub factors and criteria.

Furthermore, new comparison method was proposed, in order to get clear and accurate differences between software quality models. The comparison was basic on mathematical formula, in order to show graphically the differences between those models. This method requires assign values for the sub factors moreover the main factors. Which is gave a clear picture of the differences between the models.

The values in this study were given equivalently between the factors and between the sub factors that is because this comparison was generally. In specific domain, the values for each factor and sub factor have to be defined according to the selected domain. For future work, the proposed method has to be verified by apply it in specific domain.

References

1. Ravichandran, T., Rothenberger, M.A.: Software reuse strategies and component markets. Commun. ACM 46, 109–114 (2003)
2. McCall, J.A., Richards, P.K., Walters, G.F.: Factors in Software Quality. Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command (1977)
3. AL-Badareen, A.B., Selamat, M.H., Jabar, M.A., Din, J., Turaev, S., Malaysia, S.: Users' Perspective of Software Quality. In: The 10th WSEAS International Conference on Software Engineering, Parallel And Distributed Systems (SEPADS 2011), pp. 84–89. World Scientific and Engineering Academy and Society (WSEAS), Cambridge (2011)
4. Boehm, B.: Characteristics of software quality (1978)
5. Hamada, A.A., Moustafa, M.N., Shaheen, H.I.: Software Quality model Analysis Program. In: International Conference on Computer Engineering & Systems, pp. 296–300 (2008)
6. Rawashdeh, A., Matalkah, B.: A new software quality model for evaluating cots components. Journal of Computer Science 2, 373–381 (2006)
7. Behkamal, B., Kahani, M., Akbari, M.K.: Customizing ISO 9126 quality model for evaluation of B2B applications. Information and Software Technology 51, 599–609 (2009)
8. Kumar, A., Grover, P.S., Kumar, R.: A quantitative evaluation of aspect-oriented software quality model (AOSQUAMO). SIGSOFT Softw. Eng. Notes 34, 1–9 (2009)
9. Haiguang, F.: Modeling and Analysis for Educational Software Quality Hierarchy Triangle. In: Seventh International Conference on Web-based Learning, pp. 14–18 (2008)
10. Dromey, G.: A Model for Software Product Quality. IEEE Transactions on Software Engineering 146, 21 (1995)