World Applied Sciences Journal 6 (Supplement 1): 01-06, 2009
ISSN 1818-4952

The Analytical Comparison of Qualitative Models of Software Systems

R. Khayami, A. Towhidi and K. Ziarati

Department of Computer Science and Engineering,
Shiraz University, Shiraz, Iran

Abstract: The main purpose of most software producers is to present a qualitative software system. Software quality is a multi-dimensional content which is easily distinguishable and measurable. To determine this content more exact, the qualitative models have been presented in which different aspects of this matter are investigated. But the existences of different models and using different expressions have made the comprehension of this content a little hard. This paper attempts to introduce mentioned about models and their analytical comparison, determine software qualification and its qualitative characteristics more clearly. This article can be used as a reference to investigate software qualification and its models. Also, it can help stakeholders of software systems to explain the qualitative requirements more exactly.

Key words: Software quality models · Qualitative requirements · Software qualitative factors

INTRODUCTION

All software developers have a main purpose that is, producing qualitative software. But there is no exact and specified definition of software qualification that can be confirmed by all specialists. Software qualification is not one-dimensional content but is defined by a number of characteristics. Considerable qualitative characteristics in software system are presented in a semi-tree construction called qualitative software model. Presented qualitative models although have many common points, are different in some contents and fields. This research by investigating and comparing the presented models, tries to make a better comprehension of software qualification. This analysis can make or provide an opportunity in order to present new qualitative models especially for software systems which have specific subjects. This paper can provide a situation for stakeholders of software systems to understand content qualification better and to express their qualitative requirements more completely and more exactly. This essay after the introduction, investigate the content qualification in software system.

The rest of the paper is organized as follows. Next section introduces the presented qualitative models for software systems. Third section analyzes the involved elements of qualitative models. Finally, the last section compares the mentioned models analytically.

SOFTWARE QUALIFICATION

Qualification content is ambiguous and all the people express it as a purpose in producing the productions and in giving the services. But it is difficult for all persons to give an exact content and to determine the qualification. The qualification is usually determined by expressing the characteristics which belong to production. One known expressed definition for software qualification is as "Conformance to explicitly stated functional and performance requirements, explicitly documented development standards and implicit characteristics that are expected of all professionally developed software" [1].

So it is cleared that we should search for those characteristics in producing software which have conformity with above definition. In this way we can divide these system requirements in to two parts: functional requirements and non-functional requirements. Functional requirements are those which a system is designed to do them and determine the functional and executive aims of a system. The non-functional requirements are expressional called qualitative requirements of a system. In fact in these requirements in addition to system output, the function of a system is considered. Matters like efficiency, extensibility and maintainability, security and even the expense of production can be mentioned here. So at first we should attempt to define the qualitative characteristics and then

Corresponding Author: Mr. R. Khayam, Ph.D. Student of Computer Science and Engineering, Shiraz University, Shiraz, Iran

to determine clear senses and proofs for them in order to evaluate the degree of achieving those characteristics by using them. Expressing these characteristics depend on interested experiences and knowledge. If the level of their awareness and experiences is not suitable and sufficient, it is possible that in expressing the requirement some qualitative characteristics which should be in software, is going to be ignored or given with less importance. Another reason for investigating the qualitative models is to determine all qualitative characteristics based specialists' view. So in qualitative models in addition to organizing the software qualitative characteristics, it is tried to determine exactly these characteristics. In fact, qualitative characteristics are expressed by the aim of defining"good and suitable" software.

SOFTWARE QUALIFICATION MODELS

To explain and determine the qualitative characteristics usually the qualitative models are used. These models commonly have been expressed as a tree - construction of qualitative characteristics and their relationships. Investigating the main and known models, determine more better the suitable qualitative characteristics in a software system. Interested by studding these models can express their needs more exacter and know that must define something in explaining their requirements in this field.

Mc-call Model: In this model a useful category or classification for factors which affect the software qualification has been expressed. This way gives and suggests the software qualification based on three aspects: Product Operation, Product Revision and Product Transition. The classification of factors is as follow [2]:

Product Operation: Correctness, reliability, efficiency, integrity, usability.

Product Revision: Maintainability, flexibility, testability.

Product Transition: Portability, reusability, interoperability.

Each factor in this model has been designed based on a question expressed by one of the aspect (Figure 1). The power point of this model is the relationship of external quality factors with qualitative criteria of the production. External quality is a quality that is measured by active characteristics of a code which is under performing and the qualification which is measured by means of stable characteristic of code by programmer is called internal quality [3].

Regarding that measurement of these factor in most occasions and cases is very hard, in this model is suggested that based on a series of metrics, they are going to be evaluated. The relationships between the factors and the standards of this model have been showed (Figure 2) and their relationships are in this formula:

$$F_q = c_1 m_1 + c_2 m_2 + \ldots + c_n m_n \quad (1)$$

In this relation F_q is software quality factor, m_n is nth criteria and c_n is regression coefficient which affect on relative factor. Of course many of determined and defined criteria in this model are measured only mentally and subjectively.

The amount of any criteria of low scale to high scale has been considered. The content given to these criteria's depending on design consideration and production is determined [3].

Boehm Model: With the previous model another model was presented synchronically. In this model, characteristics have been classified from the point of end user, users in different view and users in different times. This model has divided the characteristics in to three levels that main characteristics have common points from point of effective factors. This model in comparison to previous model has designed and expressed some new characteristics. Also it has hierarchy as well as the previous one, but has mentioned the characteristics in three levels. In following figure the relationship of qualitative characteristics of this model has been showed [4]. Despite of this fact that many more characteristics have been given in this model, but there is not any way to measure and evaluated them (Figure 3) [9].

ISO/IEC Model: This model expresses these following characteristics as main characteristics of software quality: functionality, reliability, usability, efficiency, maintainability, portability.

In this model, factors are divided into two levels: characteristics and sub-characteristics; and like the previous models has hierarchy with this difference that the sub-characteristics in this construction have been considered in one characteristic [6]:

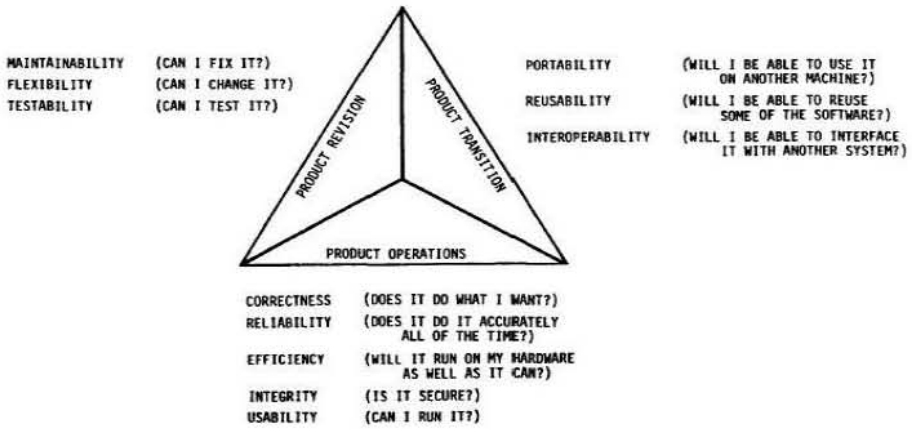

Fig. 1: McCall's software quality factors

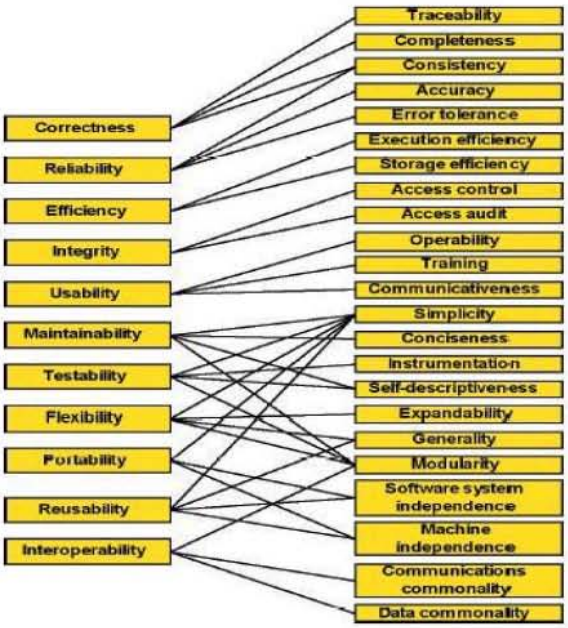

Fig. 2: McCall quality framework

Functionality: Suitability, accurateness, interoperability and security.

Reliability: Maturity, fault tolerance, recoverability.

Usability: Understandability, learnability, operability.

Efficiency: Time behaviour, resource behaviour.

Maintainability: Analyzability, changeability, stability and testability.

Portability: Adaptability, instalability, conformance, replaceability.

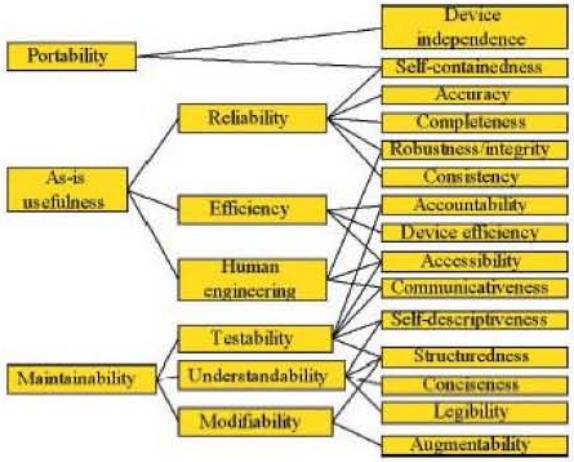

Fig. 3: Boehm´s Quality Model

Dromey Model: This model has introduced 8 characteristics as high level qualitative characteristics: function ability, reliability, usability, efficiency and maintain ability, portability, reusability and maturity process. In fact it is like ISO/IEC model that reusability and maturity process have been added to it.

FURPS Model: Suggestive model by Robert Grady and Hewlett-Packard called "FURPS" divides the qualitative factors in to two groups: functional requirements and non-functional requirements.

Functional (F) Requirements: With software operation or by investigating the outputs, the expectancy of inputs are defined.

Non-functional Requirements: By means of usability (U), reliability (R), performance (P) and supportability(S), the software is defined [7].

One weak point of this model is the lack of attention to software portability; this model also has not defined any way to evaluate exactly the factors. With regard to quality factors and defined characteristics, this model can be a base for designing the qualitative standards for different stages of software cycle. Relative characteristics to any of these factors are as follow [5]:

Function: By means of aspects and program abilities, the general properties and general security of system is defined.

Usability: Regarding human factors, general beauty and documentation is achieved.

Reliability: Measuring repetition speed and failure intensity, the accuracy of output results, average time between two successive failures and recoverability from failure and program predictability is evaluated.

Efficiency: It is expressed by means of processing speed, reply time, amount of resource use, amount of out put and amount of efficiency.

Supportability: Extensibility, adaptability and serviceability (these 3 characteristics and usually called maintainability) with testability, configurability, ease of installation and ease of adaptation to local situation are characteristics that determine the supportability.

Kazman Model: It is one of the groups which have been suggested by Kazman and his competitors, in Software Engineering Institute. This grouping divides the qualitative characteristics in to two observable groups during the time performance and those which are not defined during the performance and show during the software existence cycle [8]. The characteristics of these two groups are as follow:

- Efficiency, security, availability, function
- Modifiability, portability, reusability, inheritability and testability

This group, in fact, has not presented specific qualitative model, but had given ATAM evaluation way to investigate the quality of software architecture. In that way system interested should define their qualitative model regarding their needs. Qualitative model construction has been mode of a base called "utility" and after that, there are 3 levels. The last level of this tree is defined by a series scenario in order to test the qualitative characteristics [9].

IEEE Model: IEEE Institute, in fact, has given a scale and standard to provide a qualitative model and has not given a clear qualitative model. It has presented a tree construction to show qualitative model and emphasizes on how to design the measurement ways of qualitative factors. This suggested construction is semi-tree and has three levels. Last level is software metrics [10]. In this model it is allowed to define metrics for any factors if there is a direct measureability after the first level. For the first suggestion, a tree with factors and sub-factors are given as follow:

Efficiency: Temporal economy and resource economy.

Reliability: Nondeficiency, error tolerance and availability

Function: Completeness, accuracy, security, compatibility and interoperability.

Supportability: Testability, extensibility and correctability.

Portability: Independency from hardware, independency from software, instalability and reusability.

Usability: Comprehensibility, ease of learning, usability, communicativeness.

Comparing the Software Quality Models: In all of the models, software quality or infect the production profitless is described by means of some factors. Those factors are defined by sub-factors and get more corrected aspects. Most of the factors and sub-factors are qualitative and are not directly measurable. Sub-factors in order to be more concrete like factors, must be defined more exactly.

This refinement is continued to the point which at it, a series of system character is tics have a direct relation with considerable factors. These characteristics are sometimes implementation and it is hard to define a specific size for them. Some of them are qualitative and their size and measure depend on a person who evaluates them. So it is tried to describe them in a way that measuring becomes independent from the persons

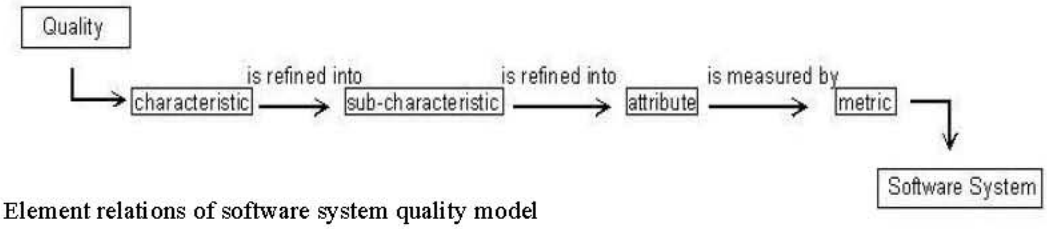

Fig. 4: Element relations of software system quality model

Table 1: Constructional comparison of quality models

Characteristic Structure	McCall	Boehm	ISO/IEC	Dromey	FURPS	Kazman	IEEE
First Level	Factor	H-Level Characteristic	Characteristic	H-Level Attribute	Factor	Quality Attribute	Factor
Second Level	Criteria	Primitive Characteristic	Sub - Characteristic	Subordinate Attribute	Attribute	Quality Attribute Refinement	Subfactor
Relation	n : m	n : m	1 : n	1 : n	1 : n	1 : n	1 : n

Table 2: Comparison of main quality characteristics

Quality Characteristic	McCall	Boehm	ISO/IEC	Dromey	FURPS	Kazman	IEEE
Testability	*	*				*	
Correctness	*						
Efficiency / Performance	*	*	*	*	*	*	*
Understandability		*			*		
Availability / Reliability	*	*	*	*	*	*	*
Flexibility	*					*	
Functionality			*	*	*	*	*
Human Engineering		*					
Security / Integrity	*					*	
Interoperability	*						
Process Maturity				*			
Maintainability	*	*	*	*	*	*	*
Modifiability		*					
Portability	*	*	*	*			*
Reusability	*			*			
Usability	*		*	*		*	*

(Figure 4). On the other hand measurable metrics in producing software usually describe a qualitative characteristic in a production. Sometimes we can give several metrics for a characteristic that these metrics have a relation with this characteristic. Also it is possible that a metric emphasizes on several characteristics.

By investigating the suggested models it becomes clear that, model producers as presenting their own model define an observable definition of software quality. Although their aim had been the same but in practice, suggested models are different. So they are not exactly the same in construction and concepts and qualitative characteristics. Even their naming of different levels of qualitative models (semi-tree construction) is not the same and sometimes cause not to understanding the main concepts for reader.

McCall model has not considered the operation characteristic, Boehm model has presented a model regardless to any suggestion for measuring the characteristics, FURPS model has ignored the portability, ISO/ IEC model has showed the qualitative characteristic measurement Kazman, McCall model are those which have presented security at the first level.

Although in different models, factors and first level qualitative characteristics are different from each other, but it is important to note and important to keep in mind that, these factors are not independent from each other semantically and sometimes they cover each other and what has been presented in one model and at the first level may have been given in first or second levels factors. For example some second level factors (ISO/IEC) have been presented in other models at the first level (Table 1).

For example some factors of second level of ISO/IEC model have been given in other models at the first level like: testability, comprehensibility, security, interoperability, correct ability which, infect, are in factors of the first level of this model. These matters have been presented in Table 2 which has compared first level factors of qualitative models with each other.

It is clear that some factors like efficiency, reliability and availability, maintainability, supportability portability, usability and operability are in most of models. On the other hand, based on Table 2 we could be found which models cover less characteristics and which ones cover more characteristics. It seems that if a new model is going to be presented, can use these concepts and be assure it covers efficiency, reliability and availability, maintainability, supportability portability, usability and operability factors in itself. To find a model with most coverage level, model designers must pay attention to this point that second level factors cover which concepts. In fact, sub-factors of each models, show the main coverage level. From the point of first level, ISO/IEC, IEEE, Dromey and McCall models cover more common factors, but if we consider the second level sub-factors, ISO/IEC has more complete level and covers more aspects of software qualification.

CONCLUSION

Regarding the different definitions which exist for software quality, the main aim of this paper is to introduce and compare the given models for this concept. In this way the ambiguity is vanished and a suitable view of main concept of software quality is provided. This paper introduces not only the qualitative factors, but also gives a framework for analytical comparing software quality models. This comparison consists of common points and differences of qualitative models and despite of different expressions, makes a good comprehension of software quality. This essay can be considered as a reference for qualitative models of software systems. This paper tries to help stakeholders to have a correct comprehension about software quality. Based on this matter, they can express their qualitative requirements well and follow them in software production cycle correctly.

As future works, this research can be as an introduction to new qualitative models design, particularly in special purpose software systems. Also, it can make a good basis for designing the qualitative metrics of software systems.

REFERENCES

1. Pressman, R.S., 2000. Software Engineering: A Practitioner' approach. McGraw-hill.
2. Cavano, J.P. and J.A. McCall, 1978. A Framework for the Measurement of Software Quality. Procs. ACM Software Quality Assurance Workshop, pp: 133-139.
3. Bevan, N., 1999. Quality in use: Meeting user needs for quality. Journal of System and Software, Elsevier., 49(1): 89-96.
4. Boehm, B.W., J.R. Brown, H. Lipow, G.J. Macleod and M.J. Merrit, 1978. Characteristics of Software Quality. Elsevier North-Holland.
5. Astudillo, H., 2005. Five Ontological Levels to Describe and Evaluate Software Architecture. Rev. Fac. Ing. Univ. Tarapacá, 13(1): 69-76.
6. ISO/IEC 9126, 1991. Information Technology-software Product Evaluation: Quality Characteristics and Guideline for Their Use.
7. Khosravi, K. and Y. Gueheneuc, 2004. A Quality Model for Design Patterns, M. S. Thesis, Laboratory of Open Distributed Systems and Software Engineering, Dept. of Informatics and Operations Research, University of Montreal.
8. Kazman, R., L. Bass and P. Clements, 2003. Software Architecture in Practice 2Ed. Addison Wesley.
9. Klein, M., P. Clements and R. Kazman, 2002. Evaluating Software Architectures: Methods and Case Studies. Addison Wesley.
10. IEEE, 1993. IEEE Standard for a Software Quality Metrics Methodology, IEEE Std 1061-1992, IEEE Computer Society.