Oriental Journal of Computer Science & Technology Vol. **3**(1), 05-11 (2010)

Software quality measurement: A Revisit

MAZHAR KHALIQ[1], RIYAZ A. KHAN[2] and M.H. KHAN[3]

[1]Department of Computer Science, Amiruddauala Islamia Degree College, LU, Lucknow, (India).
[2]Faculty of Applied Science, Integral University, Lucknow, (India).
[3]Department of Computer Science, Institute of Engineering and Technology, UPTU, Lucknow, (India).

(Received: April 12, 2010; Accepted: June 04, 2010)

Abstract

This paper presents a review on software quality measurement. An effort has been made to put forth the demand and dependency of quality software in our modern society. The demand has increased the size and complexity of computer software system during the past decades. The software industry is responding to the demand for high quality product by spending resources to improve the quality of software product. A brief discussion on quality and software quality has been given. Further, the discussion on measurement and software quality measurement are presented. Finally, it throws the light on the role of various software quality models and software metrics for software quality measurement.

Keywords: Quality, Software Quality, Measurement, Software Quality Measurement, Software Quality Models, Software Metrics.

INTRODUCTION

During the past decades, computer plays a significant role in every field of our modern society and the computer software is affecting the human being in every aspect of life. Due to our society's increasing dependence on software, the size and complexity of computer software has grown at a very high peak. Dependency and requirements on computer software increases the difficulties and failures of software and the often-devastating effect that a software error can have in terms of life, financial loss, or time delays[1]. To avoid such circumstances, the demand of quality software continues to increase.

Our increasing reliance on software systems and the ever-increasing domain of software applications puts a high premium on the standard of quality these systems offer us. Considering the varied interpretations of quality itself, as well as the lack of uniform, quantitative quality indicators; assuring 'quality systems' is not as straightforward as it seems. Software Quality has intangible aspects, which cannot be embodied in standards and thus renders it rather difficult for quantitative analysis and interpretations[2,3].

Rest of the paper is organized as follows: In section 2, discussion about quality and software quality. In section 3, measurement and software quality measurement has been discussed. In section 4 & 5, the discussion has been made on software quality models and software metrics respectively.

Software Quality

Quality is a vague and multifaceted concept, which means each type of customer may have their own conception of quality. The assessment of quality is a matter of perception. So, the identification of a set of quality attributes that completely represent quality assessment depends on a lot many things including management objectives, business goals, tools and techniques employed and of course the people who produce the product. Evaluation of the product is probably the only way out in standardizing some kind of quantitative measures for quality, as the process

and people factors are both dynamic and variant and hence cannot be controlled[3].

The relationship between product characteristics and the quality attributes has been recognized as a major issue in software engineering based on the premise that improvement in product attributes will lead to higher quality. An important assumption here is that internal product characteristics (internal quality indicators) influence external product attributes and by evaluating a products internal characteristics, some reasonable conclusions can be drawn upon the products external quality attributes. Since there is a wide range of potential software quality attributes, it is not possible to optimize a system for all these attributes. So a critical part is selection of the quality attributes according to requirements. Here it is vital to understand that different quality attributes are relevant in each phase of the Software Development Life Cycle and a decision as to where to focus is very important in order to produce quality software. Considering the fact that inefficiencies in software design account for the maximum errors thereby contributing to maintenance costs, it is only wise to isolate the errors as early as possible in the design phase to eliminate ripple costs[2,3].

Quality has been defined by various standards. Some of them are as follows:

German Industry Standard

"Quality comprises all characteristics and significant features of a product or an activity which relates to the satisfying of given requirements."

ANSI(American National Standards Institute) Standard

"Quality is the totality of features and characteristics of a product or a service that bears on its ability to satisfy given needs."

IEEE(Institutes of Electrical and Electronics Engineers) Standard

"The totality of features and characteristics of a software product that bears on its ability to satisfy given needs."

"The degree to which a customer or user perceives that software meets his or her composite expectations."

"The composite characteristics of software that determines the degree to which the software in use will meet the expectations of the customer."

Software Quality Measurement

It is well accepted fact that measurement enables designers and managers to obtain quantitative measures of attributes in entities and also serves as a baseline for classification, comparison and analysis of these attributes. Software measurement contributes to software quality from various aspects, such as understandability, complexity, reliability, testability and maintainability, as well as performance and productivity of software projects[2,3].

Software measurement has become essential for good software engineering. DeMarco states the importance of measurement, as "You can't control what you can't measure". There is no standard definition of measurement for software artifacts that is universally accepted[6]. Abreu[7] defines measurement as `Measurement is the experimental process in which, to precisely describe the entities or events in real world, numbers or other symbols are assigned to its attributes by using a given scale. The result of the measurement is called measure.' Another definition given by Fenton[8] 'Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules'. Thus, measurement contains information about attributes of entities. An entity is an object (such as a person, place, event etc.). Entities are described by the characteristics that are important to distinguish one entity from another. An attribute is a feature or property of an entity.

The software industry is responding to the demand for high-quality product by spending resources to improve the quality of their product. Periodic analysis and measurement of software products throughout the life cycle are very important to manage and improve the software quality. The latest industry surveys indicate that more than 50 percent of a software project's budget is spent on activities that are related to enhance quality of the software[9].

Software measurement has become an essential requirement for software development to help to ensure the processes. The estimation of measures may be the principal objective of the experimental software engineering. Measuring software quality is not a new theme, as it has been investigated for years in software engineering discipline. Despite this, asking a software developer to measure the quality of a product, can sound and probably in most of cases will sound like an unknown or even a new problem. Since there is no clear definition in what aspects of software quality should be considered qualitative. It may be difficult to find a suitable way to measure software quality and other related aspects. About half of the software problems are reportedly caused at requirements specification level, about 25 percent or more are caused at the design level and just a few are caused during coding. Measures may also be used as an integral part of the design process to assist software developers in improving their product quality. In this regard, the criteria that make it possible to measure positive and negative elements of system design needs to be identified. Another important issue is the use of such measurement to improve the design process[1].

Software Quality Models

Software Quality being a vague concept cannot be embodied in standards. Typically, the way we measure quality depends on the viewpoint we take. This makes the direct assessment of quality is very difficult. In order to better quantify quality, researchers have developed indirect models that attempt to measure software product quality by using a set of quality attributes, characteristics and metrics[2,3]. Till now there have been some quality models, most of which are hierarchical models, which are based on group of quality criteria, associated with group of metrics. Almost all models can be categorized into three kinds according to the means by which the model is generated[10], the first is the theoretical model based on hypothesis relations among variables. The second is data-driven that are based on statistical analyses. The third is the combined model in which intuition is used to determine the basic type of the model and data analysis is used to determine model's constants. In most cases, the combined model is practically adapted. From another perspective, the metric model is also classified into process metrics models and product metrics models. While process metrics models deal with improving the process of software development, product metrics models focus on measuring the internal attributes in the software product and relating them to product external quality[4]. The following models are in use in software industry:

Factor-Criteria-Metrics (FCM) Model

This has been generally accepted as a basis for software measurement. The basic principle of this model is that each attribute can be decomposed into a set of factors which themselves can be decomposed into a set of criteria. And these criterion values can be assessed from a set of software-related measurements, software metrics.

McCall model[11,12]

This is the most well known early model based on FCM model, which was first introduced in 1976-7 by the US Air Force Electronic System Decision (ESD), the Rome Air Development Center (RADC), and general Electric (GE), with the aim to improve and enhance the quality of product. This model is used in the United States for military purposes, space, and public domains. The basic aim of McCall's quality model is to assess the relationship among external quality factors and product quality criteria.

Boehm model[11,13]

This model was introduced in 1978, added some characteristics to McCall's model and developed an improved quality model with the emphasis on the maintainability of software product. The proposed model was also able to evaluate the software product with respect to the utility of program. Upon analyzing the Boehm model one sees that it begins with the software's general utility. It looks at utility from various dimensions, considering the types of user expected to work with the system once it is delivered. General utility is broken down into Portability, Utility and Maintainability.

FURP Model[11]

The FURP model was proposed in 1987 by Robert Grady and Hewlett-Packard. The characteristics is decomposes in two categories of requirements namely functional requirements and

non-functional requirements. Functional requirements are defined by input and expected outputs, while non-functional includes usability, reliability, performance and supportability. When the FURPS model is used, two steps are considered: setting priorities and defining quality attributes that can be measured. Grady and Caswell note that setting priorities is important given the implicit trade-off, i.e. one quality characteristic can be obtained at the expense of another.

ISO 9126 model[11]

This model is a derivation of McCall's Model, was proposed in recent years. It defines software quality as `the totality of features and characteristic of a software product that bear on its ability to satisfy stated or implied needs'. The model proposes a set of six independent high-level quality characteristics, which are defined as a set of attributes of a software product by which its quality is described and evaluated.

REBOOT (Reuse Based on Object Oriented Technology)[14]

This was proposed as a general quality model and reusability model based on the FCM model. To arrive at a reasonable computation of quality and reusability, a questionnaire was distributed to software engineers in five European countries and then adapted the requirements, the software engineers considered as important components as factors in their FCM model. All the factors are cost-related, productivity-related, or probability related. With the starting point of decomposing an activity, which is counterpart of a factor, into a subset of activities, a set of criteria of this factor are then defined.

Dromey's Quality Model[15]

Geoff Dromey recently proposed a quality framework for building product based quality models, which is a methodology for the development of quality models in a bottom-up fashion. This model addresses some of the problems of the earlier models such as the McCall's and IS09126. The framework prescribes a methodology for the development of quality models in a bottom-up fashion, ensuring that the lower level details are well specified and computable, the lack of which has been a severe drawback of the previous models.

SATC Quality Model[16,17]

SATC(Software Assurance Technology Center) has developed a software quality model for quantitatively measurement of quality of the software. These models have the goals to find the stability in Requirements and Design phase[18]. the SATC model defines a set of goals like Requirements Quality, Product Quality, Implementation Effectively and Testing Effectively. The goals are then related to software product and process attributes that allow indications of the probability of success in meeting these four goals[16].

QMOOD (Quality Model for Object Oriented Design)[19]

This was proposed as a hierarchical model used to assess object-oriented design quality[10]. It is a quality model for assessing high-level external quality attributes such as reusability, functionality, flexibility etc. of object-oriented designs based on the internal properties of design components. Jagdish Bansiya and Carl G Davis proposed this model. In this model, tangible design properties (both structural and functional) of object oriented design components such as classes are used to generate object oriented design metrics, which evaluate the extent of the tangible properties in the design components[1].

MQMOOD (Metric Based Quality Model for Object Oriented design)[20]

The Dromey's generic quality model[15] has been used to develop MQMOOD for the assessment of SATC's identified high-level quality attributes(such as efficiency, complexity, understandability, reusability and testability/ maintainability) in object oriented design[17]. The object oriented design properties to the set of SATC's have been evaluated using a suit of object oriented design metrics.

Software Metrics

Software metrics are tools of measurement. The term `metrics' is also frequently used to mean a set of specific measurements taken on a particular item or process. The IEEE Standard Glossary of Software Engineering Terms defines metrics as, "a quantitative measure of degree to which a system, component, or process possesses a given attribute". Software metrics are an attempt

to quantify some aspects of a product generated during a software project. Software metrics generally categorized into three categories: product metrics, process metrics and project metrics[1]. The subjective measurement of quality may assess by human estimation, but software metrics are the mechanized tools to assess the value of internal attributes of quality in an objective manner.

Quality software depends on many factors including process being used to develop the product, the design and characteristics of implementation. In order to improve quality of software, it is necessary to identify the locations of bad design or programming practices in software system. One way to identify the problem area is to define the metrics. Software metrics provide some confidence in the software quality estimation as it relates directly to reliability, maintainability, and adaptability, all of which are essential requirements for quality. The software metrics may define the success and failure for a product, process or person in quantitative manner. It may allow to identify and quantify the improvement, lack of improvement or degradation in product, process and person. The managerial and technical decision can be made by software metrics[1,4,5]. Some of the approaches have been reviewed and described in brief, as follows:

SATC's Approach[17,18]

The metrics was proposed by SATC which are supported by most literatures and Object oriented tools. These metrics were based on focusing the critical constructs of object oriented design and may be used to evaluate the object oriented design properties like methods, classes, coupling and inheritance. The proposed metrics focus on both the internal and external measures of efficiency of an algorithm, machine resources and psychological complexity factors that affects the ability of programmer. SATC suggests three traditional and six object oriented metrics.

Abreu's Approach[21]

The set MOOD (Metrics for Object Oriented design) was introduced by Abreu. There are six metrics was proposed in order to achieve the basic goals of MOOD are as follows:

(i) to improve the object oriented design process to get better maintainability.

(ii) to improve the object oriented estimation process to achieve better resource allocations.

The IEEE Computer Society has validated the MOOD set on the basis of the framework proposed by Kitchenham[22].

Chidamber and Kemerer's Approch[23]

The Metrics for Object Oriented Software Engineering (MOOSE) set was proposed by Chidamber and Kemerer. The researchers have been validated the set and have shown that the MOOSE are useful quality indicators for predicting fault-prone classes and maintenance effort. The set consists six object oriented metrics.

Jagdish Bansiya and Carl Devis's Approch[24]

This was proposed a set QMOOD++(Quality Metrics for Object Oriented Development) as an automated tool that consists a suite of over 30 object oriented metrics. The essential features of the metrics that object oriented development may be analyzed at both the system and class level. The set QMOOD++ covers all constructs used in the creation of an object oriented system, so the conclusion that all object oriented attributed need to be evaluated by these metrics[1].

CONCLUSION

Every aspects of human life is affected by the software, quality of Software is a necessary factor for software industry as it may cause for human life, time and budget. Periodic analysis and measurement of software products throughout the life cycle are very important to manage and improve the software quality. Measures may also be used as an integral part of the design process to assist software developers in improving their product quality. In this regard, the criteria that make it possible to measure positive and negative elements of system design needs to be identified. In order to define quality, it is required to identify the important aspects of quality. Measurement of software quality after the completion of development work is no longer needed but it is more important to monitor and manage the quality of software when it is under development. Such a task is the purpose of the software quality models and software metrics.

Quality models may be used to provide early signs of warning or of improvement so that timely action may be planned and implemented. These software quality metrics models must be used in the early phases of development. The earlier a metrics model can detect the signs of quality problems or improvements, the more time is available for proactive planning, and will be less expensive.

REFERENCES

1. R A Khan, K Mustafa, S I Ahson, 'Software Quality Concepts and Practices', Narosa Publishing House Pvt. Ltd., 2006.
2. Khan R. A. & Mustafa K: A Survey of Object Oriented Design Metrics, Proceedings, National Conference on Information Technology, Operations research and Computing, April 10-12, 2004, Agra.
3. S. Afzal, Khan R. A. & Mustafa K., Assessment of Quality Contributor Attributes - An object oriented software perspective, Vol 4(11), pp. 35-40.
4. Pressman, R. S., Software Engineering: A Practitioner's Approach, McGraw-Hill Book Company, 2005.
5. Pankaj Jalote, 'An Integrated Approach to Software Engineering', Narosa Publishing House, Thired Edition, 2008.
6. Archer, C.; Stinson, M. "Object-Oriented Software Measures", CMU/SEI-95-TR-002, Carnegie Mellon University, Pittsburgh, PA, EUA, Software Engineering Institute 1995.
7. Abreu. Brito F. and Carpuca, Rogerio, "Candidate Metrics for Object Oriented Software within a Taxonomy Framework.", Proceeding of AQUIS'93, Venice, Italy, October 1993; selected for reprint in the Journal of Systems and Software, **23**(1): 87-96, (1994).
8. Fenton N. E., "Software Measurement: A Necessary Scientific Basis", *IEEE Software Eng.*, **20**(3), 199-206, (1994).
9. Massood Towhidnejad, Thomas B. Hilburn, "Software qualityAcross the Curriculum", Proceedings, 15`h Conference on Software Engineering Education and Training, 2002 IEEE.
10. Conte S. D., Dunsmore H. F., Shen V. Y. "Software Engineering Metrics and Models", Menlo Park: Benjamin/Cummings, 1986.
11. Maryoly Ortega , Maria A. Perez & Teresita Rojas, Construction Of A Systemic Quality Model For Evaluating A Software Product, Software Quality Journal, 11:3, July 2003, pp. 219-242. Kluwer Academic Publishers, 2003.
12. McCall, J. A., Richards, P. G., and Walters, G. F. "Factors in Software Quality", Vols. I, II, and III (NTIS AD(A-049 014/015/055), Springfield: NTIS, 1977.
13. Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., Macleod, G. J., and Merritt, M. J., "Characteristics of Software Quality", Amsterdam: North-Holland, 1978.
14. Even-Andre Karlsson. Chichester, "Software Reuse: A Holistic Approach -Measuring the Effect of Reuse Chapter", New York: Wiley, 1995: 113-180.
15. Dromey, R. G. "A Model for Software Product Quality", IEEE Transaction on Software Engineering 21(2), Feb. 1995, p. 146-162.
16. Lawrence E. Hyatt and Linda H. Rosenberg, "A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality".
17. Rosenberg Linda, "Software Quality Metrics for Object Oriented System Environments", A report of SATC's research on object oriented metrics.
18. R A Khan, K Mustafa : A review on SATC Research on OO Metrics, Proceedings of National Conference on Software Engineering Principles and Practices(SEPP'04), CSED, TIET, Patiala, 5-6 March 2004.
19. Bansiya Jagdish, "A Hierarchical Model for object- oriented Design Quality Assessment", IEEE Transaction on software engineering, **28**(1), January 2002.
20. R A Khan, K Mustafa, "Metric Based Model

for object oriented Design Quality Assessment", Information Technology Journal 4, 2005.

21. Fernando Brito e Abreu, "Design Metrics for Object Oriented Software Systems", ECOOP'95 Quantitative Methods Workshop Aarhus, August 1995.
22. Kitchenham B., Fenton N., & Pfleeger, S. L, "Towards a framework for software measurement validation", IEEE Transaction Software Engineering, vol. 21, no.12, pp. 929-944, 1995.
23. Chidamber, S and Kemerer, C., "A Metrics Suite for Object Oriented Design", IEEE Transaction on Software Engineering, June 1994, pp. 476-492.
24. Bansiya Jagdish & Devis Carl, "Automated Metrics and Object Oriented Development", Dr. Dobb's Journal December 1997.