ISSN 2278-5612

Maintainability analysis of Consumer Electronics using Software Quality Metrics

Dr.Vijay pal dhaka[1], Swati Agrawal[2]
Head of Computer & Engg.Deptt. Jaipur National University
Vijaypal.dhaka@gmail.com
Research scholar, Computer & Engg.Deptt. Jaipur National University
swatinoty@yahoo.co.in

ABSTRACT

Software quality is a complex mix of factors that will very across different application and the customers who request them [1]. Determining the quality of products software quality is important factor in consumer electronic software. Consumer electronics domain demands high performance, low cost and easy to use continuous need for new product innovation. So, maintainability is considered as a solution to satisfying such demand. Maintainability is an important quality goal for CE product software. In this paper, we have identified some quality characteristics of McCall as critical quality factors, these factor help in our model to improve the quality of product in business and define the product properties. Also we have identified metrics which can be measured by static analysis tool (veracode, codesonar) for critical quality factors and then, we found some problems which affect the software quality. We design a model for CE products, derived quality model can be utilized for quality evaluation and quality improvement in CE domain.

1. INTRODUCTION

A quality model is defined as the set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality [2]. Consumer electronics including personal computing equipment, home entertainment devices, appliances, cell phones, and cameras has increased in recent years, and quality of software is more critical. Today the Consumer Electronics industry is facing unexpected changes. Day by day new products are introduced frequently [3]. These products have a short life that means to maximize revenues per product in a short time frame. Consumer electronics companies always try to achieve new heights of effectiveness with their demand, planning and inventory optimization.

Software quality is evaluated based on each domain characteristic and it is important to identify quality attributes to evaluate quality of CE product software [4].

Earlier software quality models such as McCall's model and Boehm quality model identified characteristics for software quality, and they provided how to evaluate the quality characteristics. But they are not evaluated software quality in CE product, because they are not determining which factor included in the quality definition. In the consumer electronics industry, the life of product is short and a successful launching of product processes a greatest margin. Almost 50% of new products which are introduced in the market without knowledge of market expectations are failed to achieve their success.

Consumer electronics industry can improve their odds unit having better demand planning and multi-echelon inventory optimization [5]. Where demand is based on consumer income, the rate of product innovation, manufacturing efficiency, effective marketing and distribution determines profitability. Success in this highly competitive industry is determined by isolating winning technologies that can touch consumer lives. The Consumer Electronics companies must offers spanning consulting, application development and maintenance, and infrastructure services, high involvement & low involvement product lifecycle, customer retention and brand imaging. The need of client for new developed products in high performance, easy to use and low cost [6].

In this paper, we determine weight for each quality characteristic in McCall, based on maintainability as quality goal for CE domain, and identify quality characteristics as critical quality factors. Maintainability means the ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to changed environment. Maintainability is an important quality goal for CE product software. We utilized two static code analysis tools (Veracode and codesonar) for quality evaluation. Also, we have introduce some useful metrics which can be measured by static analysis tools for critical quality factors. Derived quality model can be utilized for quality evaluation.

2. BACKGROUND RESEARCHES

There are more renowned predecessors of today quality model this is used and appreciated approach for dealing with quality issues in software developing enviourment, such as McCall's model, Boehm's model, and ISO/IEC 9126 to understand and measure quality. We introduce McCall and Boehm quality model.

2.1. McCall's Quality Model

One of the more renowned quality models presented by Jim McCall also known as the general electrics model of 1977. McCall quality model attempts to bridge the gap between users and developers by focusing on a number of software quality factors that reflect both users view and the developer's priorities. The defined software quality as a hierarchy of factors, criteria and metrics. McCall propose a useful categorization of factors that affect software quality. The McCall quality model has three major perspectives for defining and identifying the quality of a software product: product revision (ability to undergo changes), product transition (adaptability to new enviourments) and product operations (operation characteristics). The quality factors describe different type of system [7]. Behavioral characteristics and the quality criterions are attributes to one or more of the quality factors. But the McCall quality model has a week point. The actual quality metrics is achieved by answering yes and no question that then are put in relation to each other and it is based on hierarchy of 11 quality factors. It is difficult and in some cases impossible to develop direct measure of these quality factor.

2.2. Boehm quality model

Boehm addresses the contemporary shortcomings of models that automatically and quantitatively evaluate the quality of software. In essence his models attempts to qualitatively define software quality by a given set of attributes and metrics. Boehm's model is similar to the McCall Quality Model in that it also presents a hierarchical quality model structured around high-level characteristics, intermediate level characteristics, primitive characteristics - each of which contributes to the overall quality level. The high-level characteristics represent basic high-level requirements of actual use to which evaluation of software quality could be put – the general utility of software. The intermediate level characteristic represents Boehm’s 7 quality factors that together represent the qualities expected from a software system:

- Portability (General utility characteristics): Code possesses the characteristic portability to the extent that it can be operated easily and well on computer configurations other than its current one.
- Reliability (As-is utility characteristics): Code possesses the characteristic reliability to the extent that it can be expected to perform its intended functions satisfactorily.
- Efficiency (As-is utility characteristics): Code possesses the characteristic efficiency to the extent that it fulfills its purpose without waste of resources.
- Usability (As-is utility characteristics, Human Engineering): Code possesses the characteristic usability to the extent that it is reliable, efficient and human-engineered.
- Testability (Maintainability characteristics): Code possesses the characteristic testability to the extent that it
- Facilitates the establishment of verification criteria and supports evaluation of its performance.
- Understandability (Maintainability characteristics): Code possesses the characteristic understandability to the extent that its purpose is clear to the inspector.
- Flexibility (Maintainability characteristics, Modifiability): Code possesses the characteristic modifiability to the extent that it facilitates the incorporation of changes, once the nature of the desired change has been determined. The lowest level structure of the characteristics hierarchy in Boehm’s model is the primitive characteristics metrics hierarchy. The primitive characteristics provide the foundation for defining qualities metrics – which was one of the goals when Boehm constructed his quality model. Consequently, the model presents one ore more metrics supposedly measuring a given primitive characteristic [8].

Boehm’s quality mode model is based on a wider range of characteristics with an extended and detailed focus on primarily maintainability. Boehm focuses a lot on the models effort on software maintenance cost effectiveness – which, he states, is the primary payoff of an increased capability with software quality considerations. But these characteristics do not have the some importance or priority for each type of software.

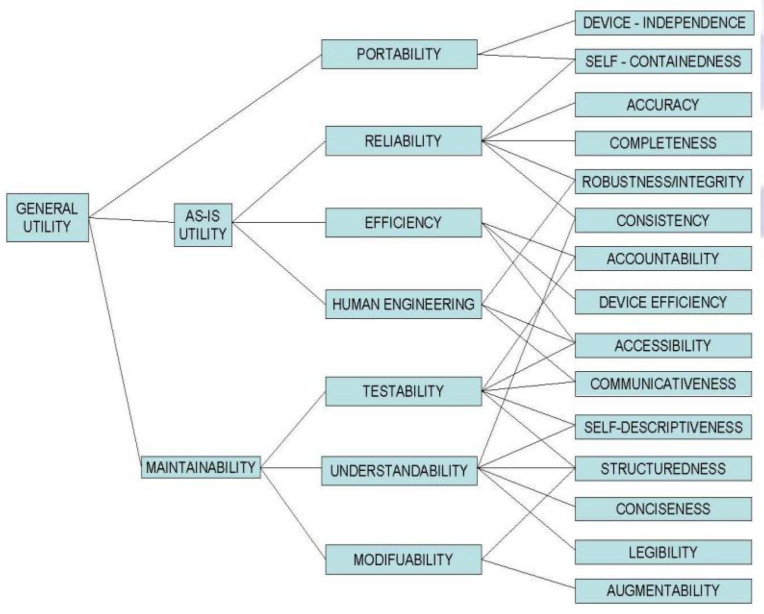

Figure 1. Boehm software quality model [9]

Hence, we need to customize it based on CE domain's characteristics.

3. SOFTWARE QUALITY IN CE PRODUCT

We have accepted McCall being used for quality evaluation as our base model. The 11 quality factors represent the qualities expected from a software system.

Quality can be defined as the totality of feature and characteristics of a product of service that bear on its ability to satisfy implied needs. Software quality prediction helps minimize software cost by allowing the mitigation of risks in early stages of software development process. A quality model is defined as "the set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality" [10]. We can identify the critical quality factors for CE product software. These quality factors help to produce good quality software which results in satisfied customer and healthier return on investment.

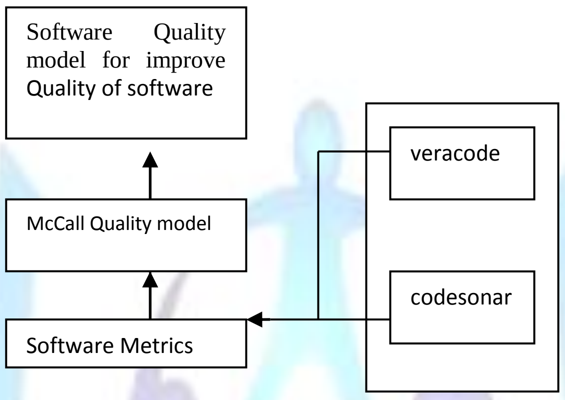

Figure 2. Software Quality model determine the quality of product

The quality characteristics cannot be measured directly. Each quality characteristic is realized with related sub-characteristics, and each sub-characteristic can be measured by related metrics [11].

3.1. Identifying Critical Quality Factors for CE Product Software

Maintainability is an important quality goal for CE product software. Maintainability needs amount of resource, time and effort. One of the main approaches in controlling maintenance cost is to monitor software metrics during the development phase. Software maintainability means the ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to changed environment [12]. It is important to get source code to maintain product with small cost and effort. Maintainability is depend on other characteristics directly and indirectly so it can not be added as a new quality characteristics. An alternative way is determining weight for each existing quality characteristic based on the relationship with maintainability. Our approach is based on it.

3.2. Prioritizing Quality Characteristics

With Quality evaluation system (QUES), we have calculated weight for each characteristic. QUES is one of the most appropriate methods for weighting the criteria. It is a strong managerial tool for multicriteria decision making.

Table 1: Quality characteristic based on QUES

	M	T	F	P	R	I
M	2.00	0.49	1.59	1.28	0.42	0.32
T	1.62	0.33	3.00	0.68	0.85	0.82
F	0.84	1.26	1.00	0.39	1.62	0.78
P	0.49	2.38	0.79	1.02	2.01	1.82
R	3.29	0.81	0.82	2.08	0.65	1.09
I	2.16	0.63	2.10	1.69	0.01	2.01

M: maintainability T: testability
F: flexibility P: portability
R: reusability I: interoperability
CI=0.043 CR= 0.032

Table 2: Weight for quality characteristic

Quality factor	Weight	Normalized weight
M	73.5344	0.1977
T	36.4213	0.0861
F	42.8614	0.0732
P	82.0321	0.2810
T	57.0852	0.1871
I	21.0381	0.0082

Table 1 is a comparison matrix based on the average values. The value of Consistency Index (CI) and Consistency Rate (CR) for comparison matrix is 0.043 and 0.032. CR is less than 0.1, so we can say that the comparison data has consistency. Weights for the quality characteristics, which are derived from QUES are shown in Table 2. Based on the weight, we found that three characteristics – maintainability, reusability and portability are relatively important for CE product software. They cover about 82% of whole quality characteristics. We have identified three quality characteristics with high weight as critical quality factors.

3.3 Metrics for quality factors

Quality factors can not be measure directly because they are abstract concept. The measurement of software quality can be difficult because there is no simple variable to look at, or because the measurement process is costly, or because it requires a complex infrastructure [13]. It is necessary to find measurements, or metrics, which can be used to quantify them as non-functional requirements. They can be measure by metrics, it is important to select good metrics describing Quality factors. We have to identify Metrics, which are good for specifying quality factor and finding out improvement criteria. Our metrics are based on code analysis which can be measure by static analysis tools. The relationship between each metrics and critical quality factor is displayed in table 3:

Table 3: Mapping metrics to quality factors

Metrics	R1	R2	P1	M1	M2
Local variable		*			
Global variable	*			*	
Source lines			*		
Comment lines	*				*
Number of files	*		*	*	
Number of function	*	*		*	

R1: fault tolerance
R2: Recoverability
P1: Replaceability
M1: Testability
M2: Stability

4.1. Quality Evalution

Quality evaluation is useful for Understanding the current status of the Software and identifying improvement areas [14]. We describe an evaluation that we have applied our approach to a software module. We utilized two static code analysis tools to avoid inefficient work.

4.2. Static Analysis Tools

Two static code analysis tools – Understand for Veracode and codesonar has been selected for the quality evaluation. Brief descriptions for them are follows:

- Veracode delivers an automated, on-demand, application security testing solution that is the most accurate and cost-effective approach to conducting a vulnerability scan. Veracode is the industry's most accurate vulnerability scan tool because it combines three different testing methodologies: static analysis, dynamic analysis, and manual penetration testing. Veracode's static analysis provides an innovative and highly accurate testing technique called binary analysis. Veracode delivering a accurate and comprehensive analysis.
- CodeSonar is a source code analysis tool that performs a whole-program, interprocedural analysis on C and C++, and identifies programming bugs and security vulnerabilities at compile time. CodeSurfer is a program-understanding tool [15].

4.3. Result of Quality Evaluation

We have tried to measure metrics from source code of target software with ‘Understand veracode and codesonar’, and then, we found some problems which affect the software quality, and we could improve the quality of software, by solving the issues as define in Table 4.[16]

Table 4. Issues and solution found by Quality evaluation

S.n	Critical quality factor	Issues	Solutions
1	Reusability	Minimum defect found	Defect has been removed
		Unused function found	Unused function will be removed
2	Maintainability	Global variable found	Convert in macro
		Unused variable found	Removed unused variable
		Average complexity	Refractor function
3	Portability	Potential defect	Remove all of them

All issues is eliminate by quality evaluation, because they influence critical quality factor directly. But it not possible to eliminate all of them issue and some issue is too difficult to improve, but developer may try to improve it and discussed about solution. We had evaluated the source code and improvement was successful. It is recommended to eliminate all issues found by quality evaluation, because they influence software quality. So some defect is removed and minimized by source code. Table 5 described the improvement result.

Table 5.Quality improvement result

Metrics	Improvement Rate
Defect	56%
Unused function	87%
Global variable	92%
Read only global variables	89%
Unused files	98%

5. CONCLUSION AND FUTURE WORKS

It is not easy to improve, protect and safe quality of software, because software quality is a complicated concept which consists of attributes in various aspects. We have identified quality goal which should be achieved by CE product software maintainability has been identified as quality goal of CE product software in this paper. We have specified three critical quality factors – portability, maintainability and reusability from quality characteristics in McCall, based on the relationship with maintainability as quality goal. Well-designed metrics with documented objectives can help an organization obtain the information it needs to continue to improve its software products, processes, and services while maintaining a focus on what is important. We have identified metrics which can be measured by static analysis tool for critical quality factors. CE product can be assured by improvement of metrics, Our approach utilized for quality evaluation and quality improvement in CE domain.

Our quality model tries to include more metrics to improve the quality in future.

REFERENCES

[1] Yi Liu,Jeng-Foung Yao, Gita Williams,Gerald Atkins, “Studying Software Metrics Based on Real-World Software Systems”,Consortium for Computing Sciences in Colleges,Mid-South Conference,2007,p.p. 55-61.

[2] Li H., Meissner J., “Improving Quality in Business Process Outsourcing through Technology”, 2008.

[3] Arun Sharma, Rajesh Kumar, and P. S. Grover, “Esti-mation of Quality for Software Components: an Empiri-cal Approach”, ACM SIGSOFT Software Engineering Notes, Vol. 33, Issue.6, pp.1-10, 2008.

[4] Avadhesh Kumar, Rajesh Kumar, P.S. Grover, “An Evaluation of Maintainability of Aspect-Oriented Sys-tems: a Practical Approach”, International Journal of Computer Science and Security, Vol -1, Issue-2, pp. 1-9, Aug 2007.

[5] ISO/IEC 9126-1, International Standard Information Technology, 2001. Software Engineering - Product Quality – Part 1: Quality Model

[6] Ma Q., Pearson J., Tadisina S., “An exploratory Study into Factors of Service Quality for Application Service Providers”, Information and Management 42, pp. 1067–1080, 2005

[7] McCall, J. A., Richards, P.K., and Walters, G. F., Factors in Software Quality, Vol. 1, 2, and 3, Nat’l Tech. Information Service, Springfield, Va., 1977.

[8] Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M., Characteristics of Software Quality, North Holland, 1978.

[9] Boehm, B. W., Brown J. R, and Lipow, Mlity, “Quantitative evaluation of software quality”, International Conference on software Engineering, 1976.

[10] Lehal M.S, “Software Quality: A mosaic of abstractions” in IJMIT Vol. 1, No. 1 Page No. 17-25, 2012.

[11] Tung-King See et al, "Multi attribute Decision Making Using Hypothetical Equivalents", Proceedings of Design Engineering Technical Conferences and Computers and Information in Engineering Conference ASME (2002).

[12] Software Engineering Institute, The Product Line Practice (PLP) Initiative, Carnegie Mellon University, www.sei.cmu.edu/activities/plp/plp_init.html.

[13] Taghi M. Khoshgoftaar, Boca Raton,”The Necessity of Assuring Quality in Software Measurement Data”,10th International Symposium on Software Metrics, 2006.

[14] Tim Menzies, Justin S. Di Stefano, Mike Chapman and Ken McGill,,”Metrics That Matter”, IEEE Software Engineering Workshop, 2003.

[15] Akingbehin, K., “A Framework for Software Engineering Metrics”, ACIS 4th International Annual Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD’03, Lubeck, October 2003.

[16] Kan, Stephen, “Metrics and Models in Software Quality Engineering”, 2nd Edition, Addison-Wesley, 2003.