Identification & Analysis of Parameters for Program Quality Improvement: A Reengineering Perspective

Harmeet Kaur[1]*, Shahanawaj Ahamad[2], Gurvinder N. Verma[3]
1. Ph.D. (Computer Applications) Research Scholar, Punjab Technical University, Jalandhar, Punjab, India.
2. Astt. Professor, Dept. of C. S. & Soft. Engg., College of Comp. Sc. & Engg., University of Ha'il, K.S.A.
3. Professor & HoD-Applied Sciences, Shri Sukhmani Institute of Engg. & Tech. Derabassi, Punjab, India.
* e-mail of corresponding author: hrmt01@yahoo.com

Abstract
The nature of software development is very dynamic and more complex by the perspective of reengineering or further program maintenances so the developed programs must be flexible, reusable and more scalable and which will be possible by the optimum quality parameters satisfaction. Having these issues in concern the software development houses trying to find out some established, usable methods to improve the quality of programs, the work presented in this paper is to identify, describe and analyze various parameters for quality which can affect the productivity and reusability of the software program and its future maintenance in form of reengineering. The work presented also analyzes the literature, previous research with different aspects and issues, and discussed its affects on present quality of the program because it is necessary to consider the potential impact on other requirements when designing a program to meet quality parameters requirements. Quality parameters are the overall factors that affect run-time behavior, system design, and user experience because many of these parameters are major concern to the program design and architecture, and also applied to establish program functionality, reusability, performance, reliability, and security which indicates the success of the design and the overall quality of the program and its application, integration.
Keywords: AOSD, DoD, FURPS,LOC,Parameters.

1. Introduction

The rise in demand for efficient systems has increased greatly in last few years, but there has been also criticism about the quality of the software systems being currently used. As the expectations for quality have raised, it is insufficient to deliver the software product which is technically sound but does not meet the end users expectations i.e. easy to use and it should fit best in the work practices and activities of the end- user. Assessing and controlling software quality is very difficult and one cannot hold it or touch it, yet its behavior has an impact on our day to day's life. Today, everyone is stakeholders in the drive to improve the quality of the software that they work with, yet very few people are able to explicate precisely how they define measures to discriminate between "poor" quality and "high" quality software. The key mindset is to remember that a software product is developed to provide a range of services for a user group, in order to help them achieve certain needs or goals. Thus, it should be made clear that for any software project one should precisely know what those needs or goals are. These are the key drivers behind the identification of not just the functional requirements, but also the quality requirements.

All are aware of the problems encountered during the development of software systems, sometimes the cost of developing a software overruns the estimated cost or the software systems delivery is delayed or the delivered system do not work adequately. Software as such is of greatest concern for the stake holders because of its cost and the critical functions it performs (Kosy,1974). For this reason considerable emphasis is given to the software quality by the research community. Producing high quality software is prerequisite for all the systems which require high reliability and error free operations. For example air force, DoD, Industries etc.

A major problem while dealing with the software is that there are no quantitative measures of the quality of a software product. One problem while dealing with quality aspect of the software is the absence of widely accepted definition of software quality which leads to confusion while specifying the quality aims of software system. Another problem which hinders developing high quality software is the time of delivery of software systems, operation and maintenance phase of the software; one can determine how far the software system meets the expectations of the user and at that stage if modifications or enhancements required in the software system it becomes very expensive.

Since the software testing indicates only expected error frequency, while verification corresponds to the functional requirements of the software system, so, a new process is needed to measure quality of the software

system. This process should indicate which software characteristics relates to which quality attribute such as reliability, functionality, flexibility, usability, efficiency, reusability, integrity etc.
The lifecycle for program development is complex and is of many stages, output of the previous stage acts as an input to the next stage and every intermediate deliverable has certain quality parameters that affect the quality of end program. If each stage of development cycle meets the requirements of the next stage the end program thus developed will meet the specified requirements of the end-user so for better reengineering and developer satisfaction the requirements for these quality parameters are also to be considered along with functionality of the programs in the planning, design, implementation and testing process.

2. Literature Review

In the last five decades lots of quality models for software engineering are proposed as in late 70's, two main models were proposed, in 1977, McCall et al. proposed a model called McCall's Software Quality Model, and it is also called Classical Quality Model this McCall's Quality Model was adapted and revised as the MQ Model by Watts in 1987. In year in 1978, Boehm et al. proposed a model using McCall's quality model, called Boehm's Software Quality Model, while in late 80's, three quality Models (in 1987, Evans & Marciniak's Quality Model and FURPS Quality Model and next year 1988, Deutsch & Will's Quality Model) were proposed.
Roger Pressman (2001), groups at HP have agreed on an acronym, FURPS, abbreviated from their perspective: functionality, usability, reliability, performance and supportability, but in recent years, FURPS has became FLURPS+ by some teams at HP, adding localization and another catchall category (the +). McCall, Richards and Walters suggested 11 different elements, in 2003, Karl Wiegers suggested 12, RADC(1991) had 13 . Experimentally Wiegers has added 2 more to his list, and currently uses 14 as two additional elements are install ability and safety.
Makoid et. al.,(1985), suggested that different definitions of usability may include different parameters such as user's satisfaction or type of errors. Butler (1985), suggested that a system is considered usable if the users can complete a given task within a predetermined amount of time. Reed (1986), defines usability as the ease with which a system can be learned and used.
In same year, Shackel (1986), presented an operational definition of usability that allows a system to be evaluated throughout the development life cycle. He presented one of the most widely used definitions of usability. He suggests that a system is usable to the extent that it is effective, learnable, flexible and subjectively pleasing.
In 1991, ISO 9126 contains 21 attributes, arranged in six areas: functionality, reliability, usability, efficiency, maintainability, and portability, from which usability attracted the attention of most researchers.
In year 1992, IEEE defined usability as it is an easiness for user to learn, to operate, prepares inputs, and interprets outputs of a system or components, Nielsen (1993), defines usability as it is having least aspects of learning, competence, able to memorize, error recovery.
Because till 90's lot of quality models were proposed which lead more confusion among software practitioners which model to follow actually? Therefore, Int. Org. of Standardization/Int. Electro-technical Commission (ISO/IEC) started promote new development to standardize new quality models considering the entire repository of various quality models proposed so far.
In 1991, ISO/IEC proposed ISO/IEC Quality Model latterly renamed as ISO/IEC 9126 Quality Model since ISO 9126 was part of the ISO 9000 standard. Later on in 1995, R.G. Dromey (1995) proposed a quality model adding one characteristic into ISO/IEC 9126 Quality model. The model is called Dromey's Software Quality Model. All the above defined software quality models were derived based on either legacy software or object-oriented software.
Kumar et.al.,(2009), and I. Castillo et al., (2010). have proposed software quality model for AOSD and called it as Aspect-Oriented Software Quality Model (AOSQUAMO). It is a common system, based on UML conceptual model the Requirements, Aspects and Software Quality model. The model integrated some new characteristics/factors and sub-characteristics/sub-factors of AOSD in AOSQUAMO Model as a base ISO/IEC 9126 Quality Model and proposed a new quality model for Aspect-Oriented Programming Paradigm, and is called Aspect-Oriented Software Quality (AOSQ) Model.

3. System & Model for Software Quality Measurement

Before considering the problems of identifying and analysing software quality parameters, we have to discuss what quality is. For the development of particular software it is necessary first to understand, define, and then prioritize the quality requirements of the end-users. During the software development, a group of users represents the customer who commissions the software product. Keeping in view the user requirements the aim is to concentrate on timescale issues of quality parameters such as maintainability, reliability, and usability. Consistent with ISO 9126-1 (1998), quality is characterized as a set of characteristics and qualities of product or service that bear on its capacity to fulfill stated or inferred necessities. In the view of this definition, a product which does not meet functional requirements, or which is delivered late or which has a greater-than-agreed cost, has low quality. The fact that individuals disagree about "What Quality is" is an argument in itself for greater clarity in the definition of quality parameters and measures for a given software project (and user base), and in the longer term for the appearance of some quality standards.

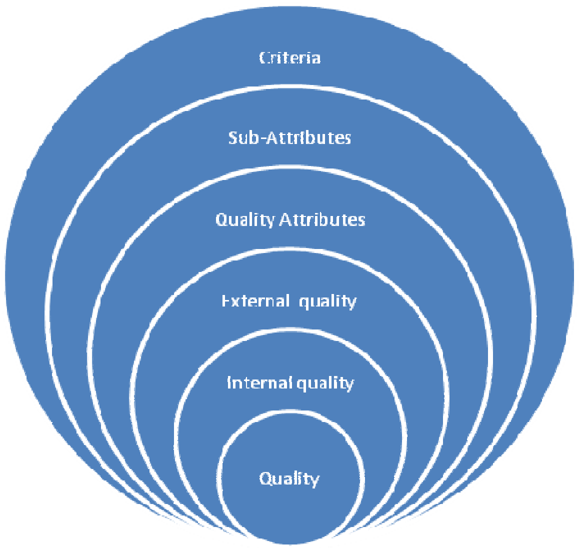

Figure 1: Software quality system

The classic quality models propose that quality can be decomposed into a hierarchy of quality parameters at different levels of abstraction. In Boehm's "Quality Characteristics Tree" (Shackle,1986), the key quality characteristics are maintainability and utility. Maintainability can further be divided into such characteristics as modifiability, understandability, and testability, whereas utility can be further divided into reliability, efficiency, and human engineering. Further decomposition yields even more primitive characteristics, e.g., structuredness, completeness, conciseness, integrity, accuracy-which may be considered more amenable to measurement and analysis. Overall quality of the end product can be computed as a weighted function of the primitive characteristics, where the weightings depend on user requirements. While quality models provide much insight with regard to the parameters of quality and how these may be related, they do not adequately define them. Too much reliance is placed on common understandings of everyday language terms. In practice, much variation in interpretation occurs between different users of such models.

Different Software quality perspectives which can be measured are user view, as the quality of the final product; developer view as the quality of the intermediate products, end-user manager view, as the marketing requirements. The overall quality of the software system can be expressed by combining different views in our context, the user and developer views will be used. The refinement between two levels of quality characteristics: factors and criteria were done by Mc Call in (1977). The former can't be measured straightforwardly, while the later could be subjectively measured. On this groundwork, the McCall's model was further streamlined by ISO 9126-1, into the ISO 9126-1 quality model, now generally acknowledged in the state-of-the specialty of software product quality specification and proposed set of six autonomous high level quality attributes, which are characterized as a set of parameters of a software product. The quality of software product is depicted and assessed by the aforementioned quality characteristics which are utilized as the focuses for validation (external quality) and verification (internal quality) at the different phases of development. They are refined (Fig.2) into sub-characteristics, until the parameters or measurable properties are obtained.

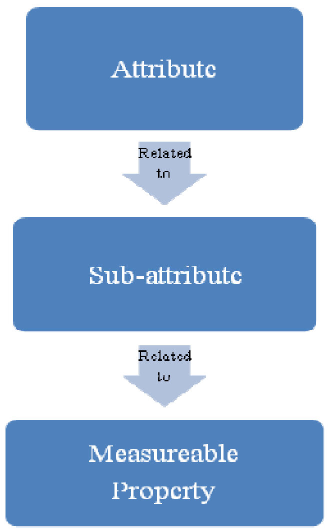

Figure 2: Relations among the quality elements

To control and monitor software quality during the development process, the external quality requirements are translated into intermediate products, obtained from development activities. Depending on the stakeholder personal experience, the conversion and selection of the parameters is a non trivial activity unless an organization provides an infrastructure to collect and to analyze previous experience on completed projects. The various quality parameters used for measuring are discussed in Table 1. The model should be adapted or customized to the specific application or product domain.

4. Quality Parameters

Software properties, like time efficiency and memory consumption, are often called as quality parameters and cover important aspects of software quality. Quality parameters can be measured by software developers to ensure that its software adheres to certain standards or customer requirements. Quality as per the IEEE Glossary of Software Engineering Terminology (1990) is the degree to which software meets user's needs. Schneidewind (1995), states that a quality factor is an attribute of software that gives to its quality. Consistent with the aforementioned two definitions it might be stated that quality factors are parameters that clients need to find in the software and subsequently software quality components might be stated to be client or client oriented. It is contended that distinctive stakeholders have distinctive discernments about quality parameters. According to Gillies(1992), there are diverse perspectives of quality, which might conflict with one another.

Type of View	**Description**
The Transcendent view	The classical definition of quality meaning "elegance".
The Product-based view/ The Economist's view	Higher quality equals higher cost.
The User-based view	It is meeting the users' requirements and fitness for purpose.
The Manufacturing view	Measures quality in terms of conformance to requirements.
The Value-based view	Provide what the customer requires at a price they can afford.

Table 1: Different Views of Quality

Quality plays a vital role to achieving a competitive advantage, based on the notion of continuous improvement throughout the entire organization. Quality is intertwined with process management, human resources management, organizational characteristics, and strategic and technological approaches.

4.1 Identify quality parameters

Quality parameters can be assumptions, constraints or goals of organization. By analyzing the initial set of requirements, the potential quality parameters are identified. For example, if the owner of an ATM card holder presents his card for transaction of money automatically bank transfers are performed then security is an issue that the system needs to address. Another fundamental quality attribute is response time, that is an issue, when, an ATM card holder punches his/her ATM card at the ATM, the machine have to react in time so that the user of ATM can act according to the instructions and deliver required services. Other concerns are identified in a similar fashion: Multiuser System, Compatibility, Legal Issues, Correctness and availability. Quality parameters define global properties of a system usually these are only dealt with in the later stages of a software development process, such as design and implementation.

Sr. No.	Attribute	Description	Variables effect Parameters	Category
1	Conceptual Integrity	Conceptual integrity characterizes the consistency and rationality of the overall design	Design of Modules, components, coding style, variable name.	Design Qualities
2	Maintainability	Maintainability is the capability of the system to experience changes with a degree of ease. The aforementioned progressions could sway parts, utilities, characteristics, and interfaces.	Services, Features and interfaces. when adding or changing the functionality, fixing errors, and meeting new business requirements	Design Qualities
3	Reusability	Reusability outlines the ability for segments and subsystems to be suitable for utilization in different provisions and in different situations. Reusability minimizes the duplication of segments and additionally the execution time.	Components, time	Design Qualities
4	Availability	Availability outlines the extent of time that the system is useful and working.	Percentage of the total system downtime. System errors, infrastructure problems, malicious attacks, system load.	Run-time Qualities
5	Interoperability	Interoperability is the capability of a system or distinctive systems to work successfully by corresponding and trading informative content with other outside systems composed and run by external parties. An interoperable system makes it less demanding to trade and reuse informative content internally and externally.	information, external system, communication	Run-time Qualities
6	Manageability	Manageability demarcates how simple it is for system managers to supervise the provision, generally through sufficient and useful instrumentation uncovered for utilization in screening systems.	Debugging and performance tuning	Run-time Qualities
7	Performance	Performance is a sign of the responsiveness of a system to execute any activity within a given time interim. It might be measured regarding idleness or throughput.	No. of events, time	Run-time Qualities

Sr. No.	Attribute	Description	Variables effect Parameters	Category
8	Reliability	Reliability is the capacity of a system to remain operational as time goes on. Reliability is measured as the likelihood that a system won't fail to perform its planned capacities over a specified time interval.	Time, function	Run-time Qualities
9	Scalability	Scalability is capability of a system to either handle increase in burden without effect on the performance of the system, or the capacity to be promptly developed.	System load	Run-time Qualities
10	Security	Security is the capacity of a system to counteract vindictive or incidental movements outside of the planned use, and to avoid exposure or misfortune of informative data.	Malicious attacks, accidental actions, loss of information	Run-time Qualities
11	Supportability	Supportability is the capacity of the system to furnish qualified information accommodating for distinguishing and determining issues when it cannot work accurately.	Information	System Qualities
12	Testability	Testability is a measure of how straightforward it is to make test criteria for the system and its parts, and to execute the aforementioned tests with a specific end goal to verify if the criteria are met.	Faults ,components	System Qualities
13	Usability	Usability outlines how well the provision meets the necessities of the client and customer	Changing user requirements	User Qualities

Table 2: The different quality parameters and their classification

4.2 Analysis of Quality Parameters

Measuring quality in the beginning of software development is the key to develop high quality software. There must be a way to assess software quality as early as possible in the development cycle. The factors that affect software quality can be categorized in two broad groups:

1. Factors that can be directly measured (e.g. defects recovered during testing) and
2. Factors that can be indirectly measured (e.g. usability or maintainability)

McCall proposed a suitable order of factors that influence software quality as indicated in figure 3.

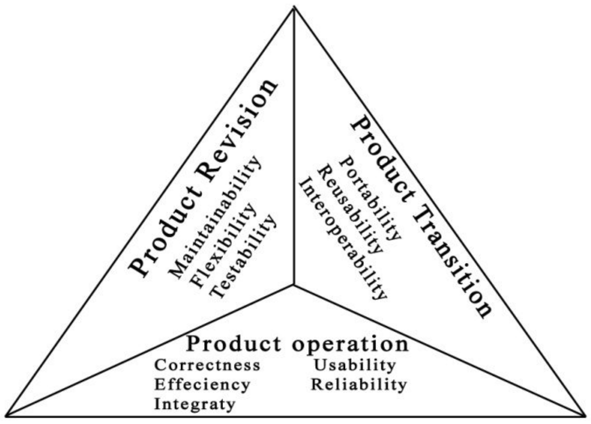

Figure 3: McCall's Quality factors

It is evident from the figure that during various phases of software development various quality parameters and sub parameters affect the quality of the software. During the product operation phase correctness, reliability, usability, integrity and efficiency play important role in the quality of the software product where as in product transition phase quality is affected by interoperability, portability and usability. In product revision phase maintainability, flexibility and testability are important factors of the software product. In this phase the product is tested for errors and it is closely observed whether it is maintainable and flexible. As far as the quality of the software is concerned it is affected by its parameters and sub-parameters such as functionality, usability, reliability, portability, supportability. Usability of the software is related to number of error messages and the length of the user manual that is lesser the numbers of errors greater will be its usability whereas portability is size of the program measured in LOC (Line of code) and number of parameters. Reliability is affected by LOC, cyclomatic complexity and number of error messages portability, and usability are interrelated more is the portability more will be its usability which will enhance the reliability and increases the functionality of the software system and leads to the development of the quality software.

For any software system correctness is one of the most important attribute which is required for proper working of the software. Another factor of concern is maintainability; which can be corrective and adaptive. Corrective maintainability deals with fixation of errors whereas, adaptive maintainability deals with changes in the requirement of the user and perfective changes deals with requirement of the software. Maintainability is directly proportional to the amount of re-work.

5. Conclusion

During the study it is observed that quality parameters and sub- parameters play a very important role in the development of the quality software system. Quality of software is affected by number of parameters for example conceptual integrity is influenced by variable name, coding style whereas availability is affected by malicious attacks, system load etc. The various existing quality parameters are identified, discussed and analyzed and it is seen that reliability, portability and usability are closely related to each other. As far as the maintainability of the software system is concerned it is directly proportional to the amount of rework.

References

[1] B. W. Boehm, et.al.,(1978), "Quantitative Evaluation of Software Quality," IEEE Computer Society Press, Page No.: 592 – 605.

[2] Butler, K. (1985): Connecting Theory and Practice: A Case Study of Achieving Usability Goals, In Proceedings of CHI 85, ACM, New York, pp. 85-88.

[3] Castillo,et.al.,(2010), "REquirements, Aspects and Software Quality: the REASQ model," Journal of Object Technology, Volume 9, Number 4, Page No.: 69 – 91.

[4] Fenton, et.al.,(1995),1995," Software Quality Assurance and Measurement, A Worldwide Perspective" International Thomson Computer press, London.

[5] Francisca Losavio et.al., 2003,"Quality Characteristics for Software Architecture" Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering Vol. 2, No. 2.

[6] Gillies Alan C.,(1992), “Software Quality, Theory and management” London, Chapman & Hall.
[7] IEEE, (1990) “IEEE Glossary of Software Engineering Terminology”, 610.12.
[8] ISO/IEC 9126-1, (2001), “Software Engineering - Product Quality- Part 1: Quality Model”, International Organization for Standardization, Switzerland.
[9] ISO/IEC 9126-2, (2002), “Software Engineering - Product Quality- Part 2: External Metrics”,
[10] International Organization for Standardization, Switzerland.
[11] ISO/IEC 9126-3, (2003), “Software Engineering - Product Quality- Part 3: Internal Metrics”,
[12] International Organization for Standardization, Switzerland.
[13] J. A. McCall, et.al.,(1977), “Factors In Software Quality - Concept and Definitions of Software Quality,” Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, New York, Vol. 1, No. 3.
[14] J. Tian,(2005), “Software Quality Engineering-Testing, Quality Assurance and Quantifiable.
[15] Improvement,” IEEE Computer Society, 2005.
[16] Karl Wiegers,(2003), Software Requirements (2nd Edition), Microsoft Press,ISBN 0735618798.
[17] Makoid, L., Forte, C., and Perry, J., (1985), “ An Empirical Model for Usability Evaluation Based on the Dynamics of the Human-Computer Interface”, Technical Report TR-85-15, North Carolina State University.
[18] Msdn library http://msdn.microsoft.com/en-us/library/ee658094.aspx
[19] Nielsen, J., (1993), Usability Engineering, Academic Press.
[20] R. G. Dromey,(1995), “A Model for Software Product Quality,” IEEE Transactions on Software
[21] Engineering, Volume 21 Number 2, Page No.: 146 - 162.
[22] Roger Pressman,(2001), Software Engineering: A Practitioner’s Approach (5th Edition), McGraw Hill,ISBN 0073655783
[24] The Rome Air Development Center is now the Rome Laboratory, as of 1991.
[25] Reed, P., (1986),” Usability Testing in the Real World”, In Proceedings of CHI 86, ACM, Boston, 212.
[26] Shackel, B. (1986), “Ergonomics in design for usability”, In Harrison, M. D. and Monk, A. F., editors, People and computers, Proc. Second conf. of the BCS HCI specialist group, pp. 45–64, Cambridge, Cambridge University Press.
[27] Schneidewind Norman. F.,(1995), “ Controlling predicting the quality of space shuttle software using metrics”, Software Quality Journal 4, 49-68.

Author's biography

Harmeet Kaur has seven years of experience in academic and research, she is currently pursuing Ph.D. in Computer Applications from Punjab Technical University, Jalandhar, India; after completing two years Master of Technology (I.T.) degree in year 2011; three years Master of Computer Applications (M.C.A.) degree in year 2008; she had the recipient of UNDP scholarship to conduct high quality research, is member of IEEE and has qualified National Eligibility Test (NET) conducted for lectureship. Presently she is working as Assistant Professor of Software Engineering in Green Hills Engineering College, Solan (HP). She has published three research articles; her area of research includes Program Analysis, Software Reengineering and Quality Assurance.

Dr. Shahanawaj Ahamad is an active academician and researcher in the field of Computer Science, Software Reverse Engineering with twelve years of research and academic experience including five years in abroad, working with College of Computer Science & Engineering of University of Ha'il, K.S.A., before joining UoH he has worked with King Saud University, Al-Khraj University of K.S.A. and Shobhit University, Meerut (Delhi-NCR) and Uttar Pradesh Technical University of INDIA as HoD-I.T., Assistant Professor. He is professional member of British Computer Society, U.K., senior member of Computer Society of India, including membership of various national and international academic and research organizations, member of research journal editorial board and reviewer. He is currently working on Service-Oriented Migration, Multi Agent System Reverse Engineering, published more than twenty five research articles in his credit in national and international journals and conference proceedings. He holds M.Tech. followed by Ph.D. in Computer Science with specialization in Software Engineering from Jamia Millia Islamia Central University, New Delhi, India. He has supervised bachelor senior projects, master and Ph.D. dissertations.

Prof. (Dr.) Govinder N. Verma has more than twenty years of wide experience in research, academics and administration, held various positions as Director, Professor, lecturer in universities and engineering colleges after completion of Ph.D. in Applied Mathematics from Himachal Pradesh University, Shimla, with excellent grade; he has completed M.Phil, M.Sc., B.Sc. in Applied Mathematics all in first division from Himachal Pradesh University, Shimla; he has published five research papers in reputed journals and supervising Ph.D. thesis; currently he is designated as Professor and Head of Department of Applied Science of Shri Sukhmani Institute of Engineering and Technology, Derabassi, affiliated to Punjab, Technical University, Jalandhar, India.