2009 Second International Conference on Environmental and Computer Science

Requirements of Software Quality Assurance Model

Yas A. Alsultanny
College of Graduation Studies
Arabian Gulf University
Kingdom of Bahrain
alsultanny@yahoo.com

Ahmed M. Wohaishi
College of Graduation Studies
MSC Student
Arabian Gulf University
Kingdom of Bahrain

***Abstract*—the Varity and complexity of software increased from day to day, the software quality assurance must be used to make a balance between quality and productivity. The practice of applying software metrics to a software process and to a software product is a complex task that requires study and discipline and which brings knowledge of the status of the process and / or product of software in regards to the goals to achieve. In this paper we are suggested a software quality model to test the different factors that affect on the quality of software as well as increasing the productivity of the software by taking in consideration the software complexity that can faces the designer and implementer of the software systems. The proposed model showed how to provide safe, reliable and quality engineering software product to develop, by taking all factors affecting software quality to reach the ISO 9126.**

Keywords-component; software quality, software complexity, ISO.

I. INTRODUCTION

Every application or business domain faces a specific set of software quality issues, and software quality must be defined accordingly. It is important for each software development project to define its specific meaning of software quality during the planning phase. Such a definition contributes to the basis for setting objectives and practical measures of quality progress and determination of readiness for release to customers [1].

Quality improvements affect operations performance in various ways, such as increasing revenue, reducing costs and improving productivity. Quality has been regarded as one of the major drivers of competitive strategy in every industry [2]. However, as Reeves and Bednar [3] stated that "no universal, Parsimonious, or all-encompassing definition or model of quality exists". The American National Standards Institute (ANSI) and American Society for Quality (ANQ) define quality as: *"The totality of features and characteristics of a product or service that impact its ability to satisfy given needs."*

Many researchers have developed different quality frameworks. For example, Garvin [4] developed a quality framework considering an eight dimension product quality, and Parasuraman et al. [5] derived a five dimension model of service quality as shown in table (1).

It is difficult, however, to measure service quality due to three unique natures of services:

- *Intangibility:* service cannot be measured, counted, inventoried, tested and verified in advance of sale.
- *Heterogeneity:* the consistency of service from personnel is difficult to measure.
- *Inseparability:* the difficulty in separating consumption from production [6].

Zeithaml et al. [7] states that customers not only judge service quality based on the outcome of the service but also consider the process of service delivery.

TABLE I. DIMENSIONS OF QUALITY

Frame work	Dimension	Definition
Product quality (Garvin (1987))	1. Performance	Primary operating characteristics.
	2. Feature	Supplements to basic functioning characteristics.
	3. Reliability	Does not malfunction during specified period.
	4. Conformance	Meets established standards.
	5. Durability	A measure of product life.
	6. Serviceability	The speed and ease of repair.
	7. Aesthetics	How a product looks, feels, tastes and smells.
	8. Perceived quality	As seen by a customer.
Service Quality (Parasuraman et al. (1991))	1. Tangibility	Physical facilitates equipment and appearance of personnel.
	2. Reliability	Ability to perform the promised service dependably and accurately.
	3. Responsiveness	Willingness to help customers and provide prompt service.
	4. Assurance	Knowledge and courtesy of employees and their ability to inquire trust and confidence.
	5. Empathy	Caring, individualized attention the firm provider gives its customers.

II. INTERNATIONAL QUALITY MANAGEMENT

ISO 9001 is an international quality management system standard applicable to organizations within all type of businesses. ISO 9001 internally addresses an organization's processes and methods and externally at managing (controlling, assuring etc.) the quality of delivered products and services. ISO 9001 is a process oriented approach towards quality management. That is, it proposes designing, documenting, implementing, supporting, monitoring, controlling and improving (more or less) each of the following processes [8-12]:

- Quality Management
- Resource Management
- Regulatory Research

978-0-7695-3937-9/09 $26.00 © 2009 IEEE
DOI 10.1109/ICECS.2009.43

- Market Research
- Product Design
- Purchasing
- Production
- Service Provision
- Product Protection
- Customer Needs Assessment
- Customer Communications
- Internal Communications
- Document Control
- Record Keeping
- Planning
- Training
- Internal Audit
- Management Review
- Monitoring and Measuring
- Nonconformance Management
- Continual Improvement

III. SOFTWARE PRODUCT EVALUATION

Besides the famous ISO 9000, ISO has also release the ISO 9126 (see Figure (2)): Software Product Evaluation: Quality Characteristics and Guidelines for their Use-standard [13-15]. The ISO 9126 series of standards now consists of one international standard and three technical reports [16]:

- ISO 9126-1: Quality model
- ISO TR 9126-2: External metrics
- ISO TR 9126-3: Internal metrics
- ISO TR 9126-4: Quality in-use metrics

The first document of the ISO 9126 series-Quality model-contains two-part quality model for software product quality:

- Internal and external quality model
- Quality in-use model

The first part of the two-part quality model determines six characteristics in which they are subdivided into twenty-seven sub characteristics for internal and external quality, as shown in Figure (3). These sub characteristics are a result of internal software attributes and are noticeable externally when the software is used as a part of a computer system. The second part of the two-part model indicates four quality in-use characteristics. All the quality characteristics and their corresponding sub-characteristics are defined in ISO 9126-1.

The 2nd, 3rd and 4th documents of the ISO 9126 series provide the following information:

- Sets of metrics for each external quality sub characteristic, internal quality sub-characteristic and quality in-use characteristic.
- Explanations of how to apply and use these sets of metrics.
- Examples of how to apply these metrics during the software product lifecycle.

IV. 4. MODEL TO PROVIDE SOFTWARE QUALITY ASSURANCE

Software complexity is added during the development stages that following the requirements phase, primarily during the designing and coding phase. In our case to provide software quality assurance, we can define the software complexity as the mount of resources required from optimal solution of the problem. The software have many problems in their design and implementation, these make software more complex. To make a balance between the software complexity and software quality we are suggested the software model shown in Figure (4).

Complexity is one of the problems that faces the software quality assurance, to have good quality with a high productivity, we must make the software complexity as simple as possible, but this is not simple specially during software implementation, these are comes from the way of solving the problems and the constrains that faces any project, these are representing the resources of the complexity of the software quality assurance. The following subsection will describe the function of each factor affecting the quality of software quality assurance.

A. Problem with Software Quality Assurance

Complexity of the problem which is the inherent complexity, created during the requirements phase. This type of complexity is added during the development stages following the requirements phase, primarily during the designing and coding phases, for this approach, the complexity of a problem can be defined as the amount of resources required for an optimal solution of the problem.

1) Solution

The solution can be regarded as the resources needed to implement a particular problem.

i. Design:

While requirements are meant to specify what a program should do, design is to specify how the program should do it. The usefulness of design is also questioned by some, but those who look to formalize the process of ensuring reliability often offer good software design processes as the most significant means to accomplish it.

ii. Code

To a computer, there is no real concept of "well-written" source code. However, to a human, the way a program is written can have some important consequences for the human maintainers. Many source code programming style guides, which stress readability and some language-specific conventions, are aimed at the maintenance of the software source code, which involves debugging and updating.

2) Development

Software development is the translation of a user need or marketing goal into a software product. Software development is sometimes understood to encompass the processes of software engineering combined with the research and goals of software marketing to develop computer software products. This is in contrast to marketing software, which may or may not involve new product development.

B. Constrains on Software Complexity Model

These resources have at least two aspects: time, i.e. computer time and man-hours, and space, i.e. computer memory. On other hand, this will help us to classify which constrains either budget; time, logistical or manpower need to be analyzed to find which aspects will have an impact to the program complexity. The idea behind this typology is that when we can measure how and when complexity is brought into the project, we will also know which stage in the product life cycle we need to focus on, to be able to control and hopefully reduce the software complexity.

i. Budget:

Every project has a certain budget allocated to it. The money allocated for each project covers the entire scope of work (complexity) which includes design, development until final completion of the project.

ii. Time:

Any project has to be completed within a given time. Only then the project becomes viable and hence profitable. If there is delay in completing the project due to unforeseen circumstances, it leads to various other complications and sometimes even liquidated damages from the client.

iii. Logistic:

For carrying out a project, proper logistics have to be planned well in advance. For example, this could be constraints in transport, infrastructure, UPS power when power failure occurs, and so many such issues external to, but which aid in making the program a success.

iv. Manpower:

If adequately trained manpower are not available or leave half-way in a project, it could create a big problem. Hence having enough backup of trained manpower is absolutely essential for the success of any project.

C. 4.3 Software Qualities

In the case of most software-based system, the achievable software quality goals largely depend on the availability of resources for performing software quality improvement activities. As a general rule, the more resources the project can devote towards testing and software quality assurance, the better the software quality of the end product.

1) Testing

Software testing is the process of checking software, to verify that it satisfies its requirements and to detect errors. Software testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding software bugs. The following issues are the main factors to be taken in consideration during the software testing:

- Size
- Methods
- Usability
- Environment
- Management and Risk

Risk is defined as the possibility of loss, and the degree of probability of such a loss. Barry Boehm 1989, in his book "Software Risk Management", defines risk management as a combination of risk assessment and risk control. Risk assessment includes identification, analysis, and prioritization; risk control includes management, planning, resolution and monitoring. In order to develop a software quality model that is useful in the risk management process, it is necessary to identify a set of risks that is common to most projects and based on the project manager's idea of software quality. Given that definition, the risk areas are:

- Correctness
- Reliability
- Maintainability
- Reusability
- Schedule

The project manager might have a different prioritization of the listed risks for different segments of software that are being developed by the project.

2) Reliability

Reliability is the capability of the software product to maintain a specified level of performance when used under specified conditions. Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. The high complexity of software is the major contributing factor of Software Reliability problems. Hardware reliability is often defined in terms of the Mean-Time-To-Failure, or MTTF, of a given set of equipment.

3) Modify or Change

Software maintainability is defined as the ease of finding and correcting errors in the software. It is analogous to the hardware quality of Mean-Time-To-Repair, or MTTR. While there is as yet no way to directly measure or predict software maintainability, there is a significant body of knowledge about software attributes that make software easier to maintain. These include modularity, self (internal) documentation, code readability, and structured coding techniques.

D. Software Productivity

Software productivity is the capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use. The interest of a productivity measure is to appreciate the efficiency of a production process and to determine how to improve it. The issue is to find when the minimal cost is reached to realize the optimum profit.

1) Maintainability

Maintainability is the capability of the software product to be modified. Modifications may include corrections, improvements, or adaptation of the software to changes in environment, and in requirements and functional specifications. Software complexity may have a direct impact upon maintenance costs as well as the costs incurred through the presences of software errors.

2) Simplicity (Understandability)

The attribute of simplicity relates to the ability of the developers to understand clearly what is met by the requirements document. The programmer is currently looking for good ways to automatically evaluate understandability.

3) Compatibility

Compatibility of software is defined as minimum hardware resources required to perform its functions. There are many other software qualities. Some of them will not be important to a specific software system, thus no activities will be performed to assess or improve them. Maximizing some qualities may cause others to be decreased. For example, increasing the compatibility of a piece of software may require writing parts of it in assembly language. This will decrease the transportability and maintainability of the software.

V. SOFTWARE QUALITY GUIDELINES

In general, the processing of data consists of a set of linked stages where each stage can be considered as a separate system. Each system needs to be considered separately, as does the models and algorithms implemented by the software within each system. Each software system typically has inputs consisting of data (processed by earlier systems) and quality indicators for that data (assigned by earlier systems).

A. Model validation

In modeling, we must distinguish the functional model that describes the functional relationships between quantities (including those that are measured) from the statistical models that describe the probability distributions (and the uncertainties) associated with those quantities. We must also consider the particular issues of "continuous" models where the solutions consist (mathematically) of a continuum of values and solved by finite element methods and similar techniques.

B. Algorithm Validation

An algorithm is a mathematical specification of the computational steps to perform a calculation, e.g. to determine the output quantities given a physical model. Although an algorithm is expressed in mathematics, it is understood that it is operates using finite precision arithmetic on finite precision data and therefore different algorithms whose specifications are mathematically equivalent may behave numerically in different ways.

C. Software Validation

Software development should follow the procedures of the European Community Space Standard ECSS-E-40B [1] or similar. However this standard does not specify particular techniques to assure that software are "fit for purpose" and the necessary evidence to enable a QI to be robustly assigned to its outputs.

D. Assessment and Auditing

1) Evidence

All the validation techniques that are used at all stages of the software system must be documented and their results recorded. This provides evidence of the use of the validation technique and can be used to demonstrate that the software system has been properly developed. Also, the risk analysis and the reasons for the choice of validation techniques must be recorded. A pro-forma (electronic or paper) can easily be produced for collation of such evidence.

2) Assessment

To assess that sufficient and appropriate validation techniques have been applied, the choice of techniques and the evidence from those techniques must be examined. This assessment should be based on a sound technical understanding of modeling, numerical analysis and software engineering (respectively).

3) Auditing

Ideally the functionality and performance of software systems should require third-party assessment of the validation steps carried out. This can easily be performed by auditing of the documentation of the risk analysis and the evidence from the application of the validation techniques; as opposed to the need for any direct observation or repetition of the validation techniques.

VI. CONCLUSIONS

Software quality assurance ensures that the process of cooperating quality in the software is done properly, and that the resulting software product meets the quality requirements. The degrees of conformance to quality requirements usually must be determined by analysis whole functional requirements are demonstrated by testing. Software quality assurance is usually facing the important factor, which is the software complexity. The software complexity is affecting the software productivity rate to realize the optimal profit with minimal profit.

In this paper the proposed model showed the relationship between the problem and constrains that might be affected the quality and productivity of the software. We study the resources that effect the testing such as usability, reliability, management and risk which help us to improve the quality to develop the software complexity; also we study the compatibility, maintainability and understandability that affecting the model which will influence the software productivity. To explore the nature of complexity, we can classify the concept, in order to look at it from different view points. A program that is more complex than another is also more likely to contain more errors and generate a larger system.

REFERENCES

[1] Lazic L., Kolasinac A., Avdic D., "The Software Quality Economics Model for Software Project Optimization", WSEAS Transaction on Computers, Issue 1, Vol. 8, January 2009.

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

[3] Reeves C., Bednar D., "Defining Quality: Alternative and Implications", Academy of Management Review 19 (3), pp. 419-445, 1994.

[4] Garvin A., "Competing on the Eight dimensions of Quality" Havard Business Review Nov-Dec 101–109, 1987.

[5] Parasuraman A., Berry L., Zeithaml V., "Refinement and Reassessment of the SERQUAL scale", Journal of Retailing 67(4), pp. 420–450, 1991.

[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] Zeithaml V., Berry L., Parasuraman A., "The Nature and Determinants of Customer Expectations of Service Quality", Journal of the Academy of Marketing Science 21(1), pp. 1-12, 1993.

[8] ISO, International Organization for Standardization, "ISO 9000:2000, Quality Management Systems-Fundamentals and Vocabulary", 2000.

[9] ISO, International Organization for Standardization, "ISO 9000-2:1997, Quality Management and Quality Assurance Standards — Part 2: Generic guidelines for the application of ISO 9001, ISO 9002 and ISO 9003", 1997.

[10] ISO, International Organization for Standardization, "ISO 9000-3:1998-Quality Management and Quality Assurance Standards – Part 3: Guidelines for the application of ISO 9001_1994 to the development, supply, installation and maintenance of computer software (ISO 9000-3:1997)", 1998.

[11] ISO, International Organization for Standardization, "ISO 9001:2000, Quality Management Systems-Requirements", 2000.

[12] ISO, International Organization for Standardization, "ISO 9004:2000, Quality Management Systems-Guidelines for performance improvements", 2000.

[13] ISO, International Organization for Standardization, "ISO 9126-1:2001, Software Engineering- Product quality, Part 1: Quality model", 2001.

[14] McCall, A., Richards, K., and Walters, F., "Factors in Software Quality", Nat'l Tech.Information Service, Vol. 1, 2 and 3, 1977.

[15] Boehm, Barry W., Brown, J. R, and Lipow, M., "Quantitative Evaluation of Software Quality, International Conference on Software Engineering", Proceedings of the 2nd international conference on Software engineering, 1976.

[16] Al-Qutaish R., "Measuring the Software Product Quality during the Software Development Life-Cycle: An International Organization for Standardization Standards Perspective", Journal of Computer Science 5 (5), pp. 392-397, 2009.

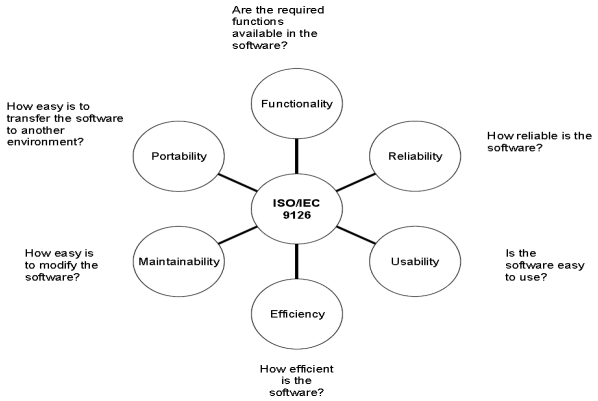

Figure. 1 The ISO 9126 quality model.

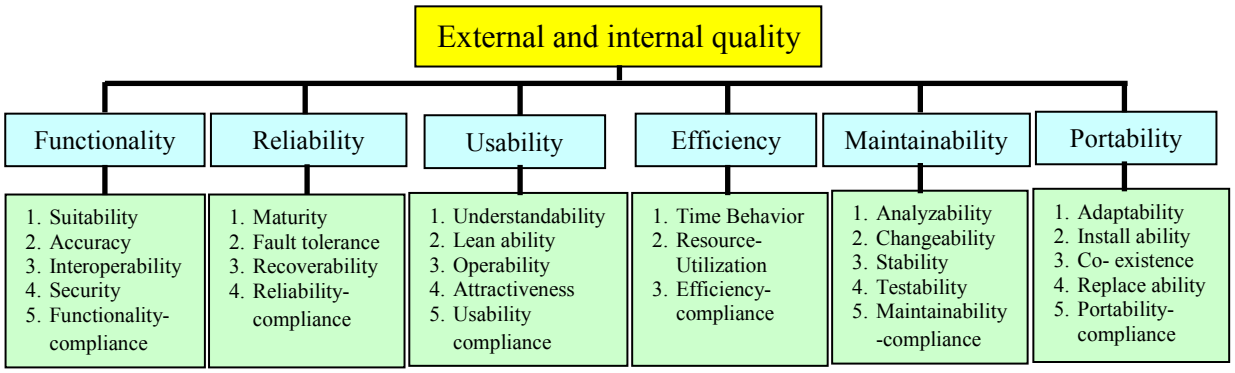

Figure 2. ISO 9126 quality model for external and internal quality

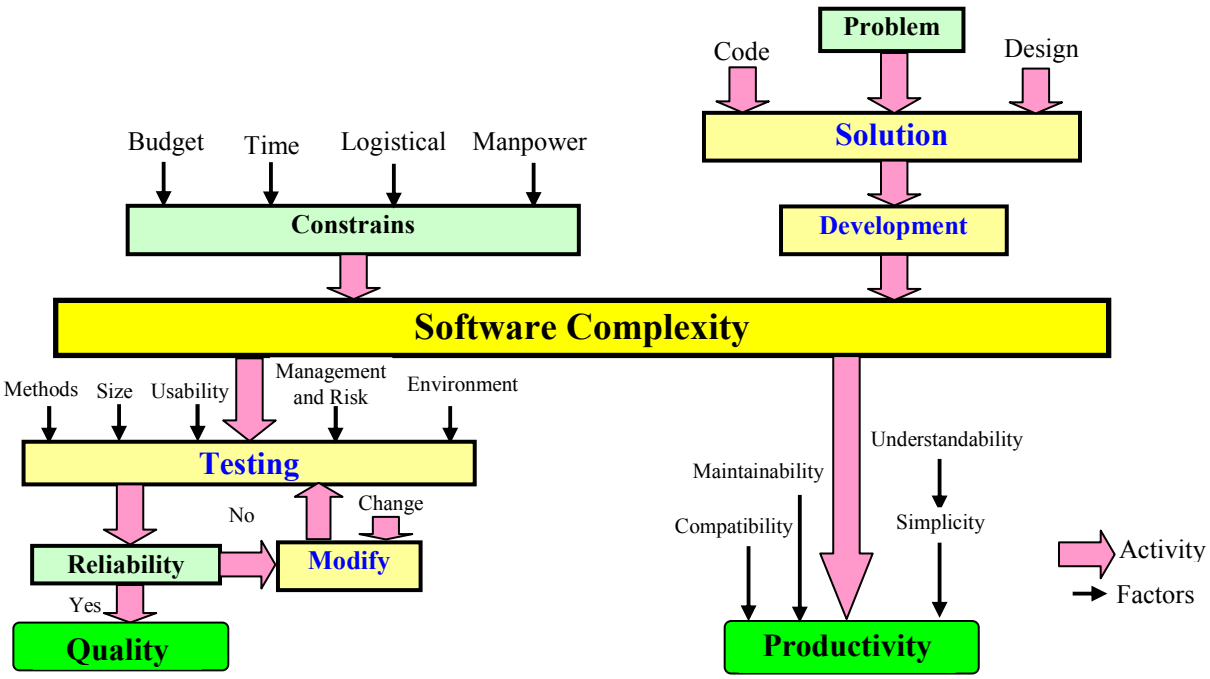

Figure 3. A model of software Quality Assurance