GLOBAL JOURNAL OF PURE AND APPLIED SCIENCES VOL. 17, NO.1 2011: 93-98
COPYRIGHT© BACHODU SCIENSE CO. LTD PRINTED IN NIGERIA ISSN 118-0579
www.globaljournalseries.com, Email: info@globaljournalseries.com

LESSONS AND CHALLENGES FROM SOFTWARE QUALITY ASSESSMENT: THE CASE OF SPACE SYSTEMS SOFTWARE.

MONICA AGU AND FRANCIS BAKPO

(Received 17 February 2010; Revision Accepted 13 July 2010)

ABSTRACT

Software development activities have continued to be plagued by a number of problems even with the availability of so many esoteric software technologies and paradigms such as object oriented development, etc. Several studies can be traced back to the software processes adopted. Other contributing factors include lack of knowledge of available systems standards, tools and techniques employed by system practitioners. This paper presents lessons and challenges gained over the last 10 years of experience as software system administrator as well as lecturers in the computer science department. Over this period of time, we have managed a number of in-house and purchased project software amongst them are banking, airtime billing, human resource, result computation etc. We discussed these lessons and challenges across two measurable characteristics namely quality of design (life cycle stages) and quality of conformance. Finally, we also recommended the lessons and challenges from software quality management for space system software.

KEYWORDS: Software, Software Quality ,Quality Standard, Characteristics, Assessment, Challanges, lessons

1. INTRODUCTION

As software becomes a more critical component in systems, concerns about software quality are increasing. Consequently, a number of organizations have developed quality standards that are specific to software or that can be applied to software. One of these organizations is the International Organization for Standardization (ISO) which has developed standards for quality management and assurance. Software quality assessment is a field which has come into greater focus as the global drive for systemic quality assurance continues to gather momentum. There is a general consensus within the field on the elements needed to measure the quality of a software product.

There is no generally accepted and widely held definition of software quality. Some people seem to have their own definition which supports their particular point of view and which relates to the issues and concerns that they see as important. In Fitzpatrick et al (2004) software quality is an abstract concept and it is perceived and interpreted differently based on one's personal views and interest. Software quality can be defined as the features and characteristics of a software product that bear on its ability to satisfy stated and implied needs (SO/IEC-9126). To resolve this ambiguity, ISO/IEC-9126 (International Organization for Standardization 2001) provided a framework for the evaluation of software quality defining six software quality attributes, and these attributes are often referred to as quality characteristics. This focuses on the characteristics of a product relative to current needs requirement. Georgios et al (2007), ANSI Standard (ANSI/ASQCA3/1978) gave the definition of quality as "The totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs."

Software is a critical core industry that is essential to national interests in science, technology, etc. It is in several places at the same time in today's society and coexists with hardware in our medical systems, financial and communication, etc. The software in a modern financial system consists of code that helps to control the transfer rate of information/data across the globe. In this and other applications, issues concerned with improving the quality and productivity of the software development process are of paramount importance. It is therefore very important to address relevant and pressing problem of software development.

The aim of this work is to discuss the challenges associated with software quality assessment in relation to our practical experience over the years of teaching and programming. The paper also discusses the lessons that can be learnt from these challenges. In doing so the ISO standard for software quality assessment was also discussed. We hope that this paper will be of immense help to software developers/users when assessing the software developed.

In this paper the quality of the software design is discussed putting into consideration the software life cycle. The life cycle is shown in fig.1 and comprises the following stages:

- Understanding the problem,
  Analyzing the problem (input, processing, output required to solve the process.)
- Developing an algorithm
- Coding the algorithm
- Testing the programme
- Debuging the programme
- Implementation
- Documenting -

M. Agu, Department of Computer Science, University of Nigeria, Nsukka, Nigeria
F. Bakpo, Department of Computer Science, University of Nigeria, Nsukka, Nigeria

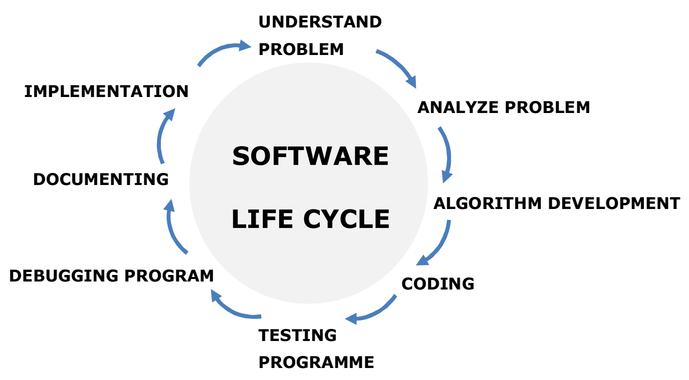

Fig1: Software life cycle

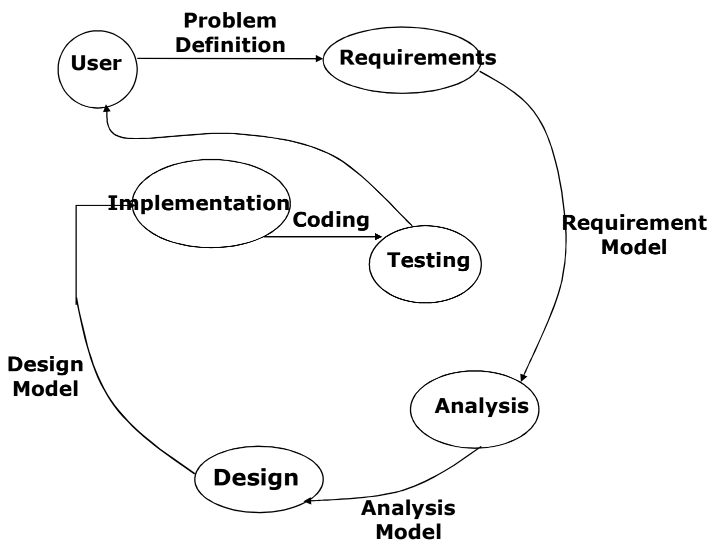

Fig. 2: Phases in software system development

According to (Boehm 1978) the first step of the software life cycle is the generation of system requirements whereby functionality, interactions and performance of the software product are specified in numerous documents. Software development has many phases. These phases include requirement engineering architecture, design, implementation, testing, software deployment and maintenance as shown in fig. 2.

In our today's changing environment and also to survive and grow in this environment, organizations must be adaptable and ready for change. With ever-changing requirement from users and hardware models appearing and disappearing at breakneck speeds as well as software updates from commercial vendors bombarding organizations, applications must be postured for growth and evolution (Robert and Stephen 1998). In addition as software undergoes maintenance and enhancement, it becomes brittle, complex and susceptible to errors.

The quality assessment process produces clear and objective risk rating of the overall quality of the software products evaluated. These ratings comprise risk drivers and risk mitigators inherent in the system artifacts under evaluation. The question arises, how should quality of software be defined? The definition depends on an individual point of view and it relates to issues and concerns that the person sees as important Robert and Stephen (1998). This can be defined in such a way that a quality system minimizes its lifecycle risks. Software systems are expected to be modified overtime and therefore should support ease of modification and evolution. We think of systems that minimize the risk of introducing errors during the development and maintenance phases of the system. With this definition we explore the types of issues that we will have to examine in order to obtain a measure of system quality.

2.0 Characteristics of software quality

Collaborating ISO/IEC-9126 which is an extension of the previous work done by McCall (1977), Beohm (1978), we define a set of characteristics for measuring a software quality to include:
(i) Functionality: This is defined as the essential purpose of any product or service and the more function it has the more complex it becomes to define its functionality. In designing software, functionality can be expressed as a totality of its essential functions. Some essential attributes of functionality include:

- Suitability. This refers to the appropriateness of the functions of the software
- Accurateness. This refers to the correctness of the functions. If the software is suitable to handle the function, it should have this second attribute.
- Interoperability. This concerns the ability of the software component to interact with other components or systems since a software system cannot function in isolation.
- Compliance. The software has to be compliant to the performance of the functions for which it was designed.
- Security. Security refers to the authorized access of the software functions.

(ii). **Reliability**: This is the next characteristics for measuring software quality.It defines the capability of the system to maintain its service provision. The reliability is related to maturity, fault telorance and recoverability. Maturity of the software concerns the frequency of its failure while fault tolerance explains the ability of the software to withstand/recover from component or environmental failure. Recoverability is concerned with being able to bring back a failed system

(iii) **Usability:** This refers to the ease of use for a given function. It has three sub characteristics which explain the usability of the software and they include the understandability of the software and that determines the ease with which the system functions can be understood, learn ability which concerns the learning effort for different users and operability the ease of being operated by a given user in a given environment.

(iv) **Efficiency:** This concerns system resources when providing the required functionality. This is in respect of disk space, memory, etc. and is measured by time behavior and resource behavior. Time behavior characterizes response time for a given throughput and Resource behavior characterizes resources used namely memory usage, CPU, etc.

(v) **Maintainability:** This addresses ability to identify and fix a fault within a software component and it is a measure under the following

- Analyzability. This is the ability to identify the root cause of a failure within the software
- Changeability. This takes care of the amount of effort to change a system.
- Stability. This is the effort needed to verify a system change.
- Adaptability. This is the ability of the system to change to new specifications or operating environments.

(vi) **Portability:** It refers to how well the software can adopt to changes in its environment or with its requirements. The following are sub characteristics of portability

- Installability. The effort required to get it installed
- Conformance. This is similar to compliance in functionability which addresses how capable the software can be compliant.
- Replaceability. This talks of the plug and place aspects of software components and how easy it is to exchange a given software component within a specified environment.

As in Robert and Lawrence (1996) the main areas for quality of the software were identified as follows:
Maintainability. This ranges from architectural design issues to implementation and documentation. Here we look at the ease of locating and fixing software failures and making minor modifications.

Evaluability. This is concerned with the ease of changing software to accommodate changes in requirements. This deals with the questions of how difficult it is to change the capabilities of the system. A lot determines the systems evaluability and these include, how simple the design is, good control of errors, accurate informative documentation.

Portability: This concerns when the system may need to migrate or upgraded
to another hardware platform or when operating system changes.
Descriptiveness: This refers to both the external printed material about the system and the source code resident documentation. What matters is that the documentation is adequate to support the maintenance, porting and enhancement activities that will occur through the systems life.
Having defined the areas, the quality of these areas are measured based on the factors. Each of the factors is considered to measure the quality of those areas while using some set of attributes. To measure the quality factor these attributes are distinct measurable questions that address the various ways the concept of the factor may be implemented in the code.
As in Robert and Lawrence (1996), Robert and Mary (1996) the analysis of a structure for software quality assessment was done originally by B. Boehm and associates and incorporated by McCall and others in 1978. This structure involved quality attributes related to

quality factors which was decomposed into quality criteria which lead to quality measurement. In his work he designed a quality profile model in which the functions that determine quality factors and merit indices can either be formulated using an algorithm or statistic. What composes these functions can be derived from theoretical understanding of relationship between measurement parameters and the derived properties of the system. One good example of these quality factors in software is the degree of structuredness of some code. Therefore a quality attributed for a comparable program property would be a Boolean value which indicates whether or not all structures in a program are consistent with some standards and this standard represents the ISO for 2001 which defined six software quality attributes. The quality factors are concerned with defining the quality of given software. This quality must surely have to do with both structure of the system and perceived performance in achieving its objectives. Based on this the subject i.e. and the objective measure are used to capture all the important issues which surrounds the quality of a system, where the subjective quality factor is one that has little or no theoretical support while the objective parameters express some essential qualitative aspect. Having looked at the different quality factors and different quality areas, fig 3 shows a mapping of the quality areas with respect to the quality factors.

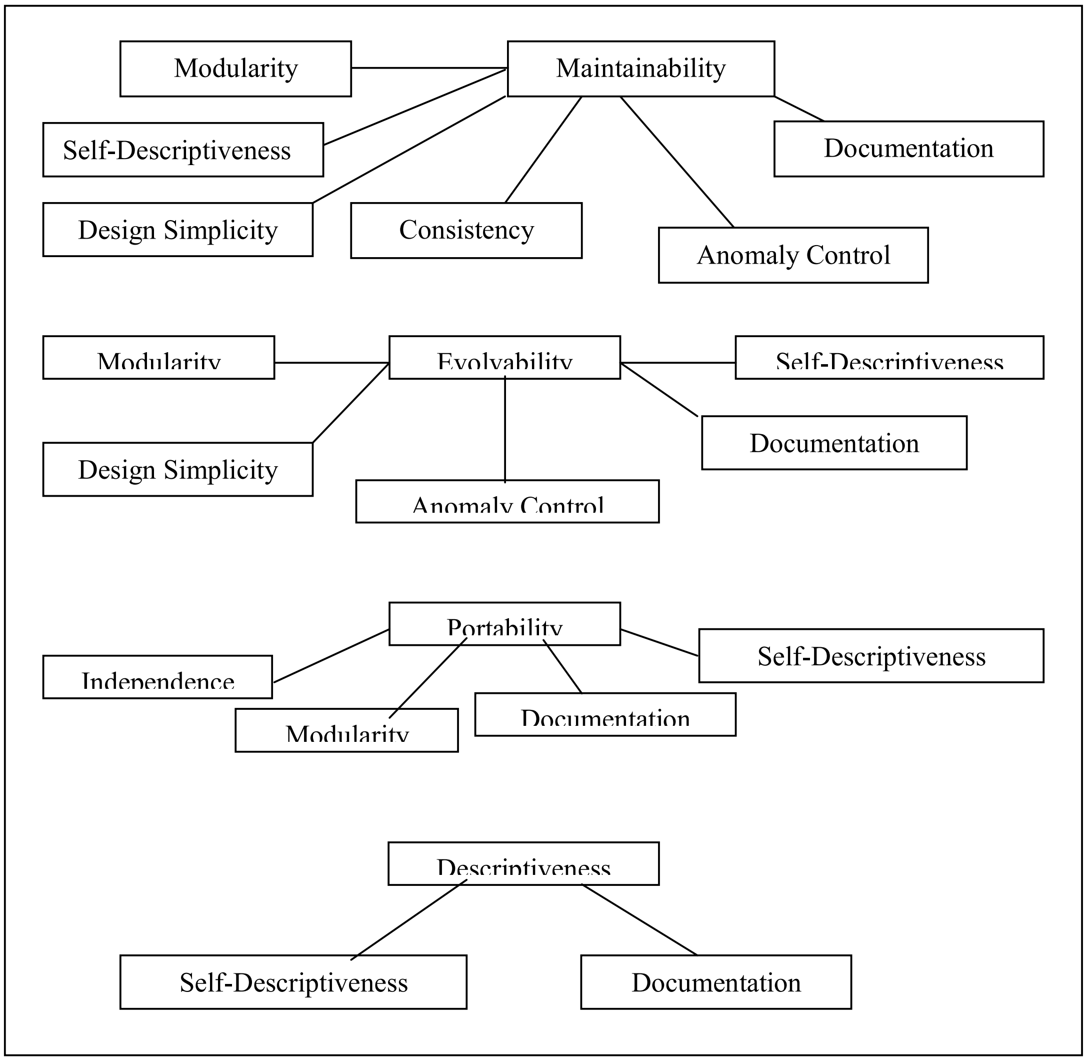

Fig 3: Quality Areas to Quality Factors Map.
Source: Robert A. Martin and Lawrence H. Shafer (1996)

From fig 3. It can be seen that self descriptiveness and documentation contributes to all the four quality areas. Having described the factors that are used to assess the quality of the software, we now discuss the challenges we are faced with.

3.0 Challenges of software quality assessment.

Challenges of software quality assessment bother on the sub characteristics of the non-functional components. Evaluation of quality in use is partly affected by the user's knowledge and experience Georgios gousious et al (2007). These challenges of software quality assessment embrace the following:

Security: this is a nonfunctional requirement and it needs to be addressed in every software project. A badly written software may be functional but subject to buffer overflow attacks. Here the software designed is not considered because what matters is the functionality of it.

Designing: This is a major factor for software quality when proven design principles are adhered to. This allows a software system to evolve as it makes modifications to be cheaper. Therefore by evaluating the quality of the design of a system one can estimate its overall quality.

Reliability: Nobody thinks of the failure of the software. However when a software is not able to withstand or recover from failures it becomes unreliable.
Usability: This concerns the ease/difficulty of learning how to use the software by different users. In some cases the software users are not able to operate the software. Here, it does not have a good user interface design. In our situation as long as the experts know how to use it then the problem is solved.

Maintainability: A number of questions are raised. Systems are designed but can they be easily maintained? If it has a problem, can the root cause of the problem be identified and if not is it advisable to have software that cannot be maintained? Again what effort is required for the system to be changed? The software bought or developed does it adapt to this situation? How stable is the system in respect to changes to the system? What efforts do we need to verify the change to new specifications? In our situations, software bought/designed by our students have not had the facilities of testing this characteristics which are necessary to certify the quality of the software

Efficiency: When we think of this nobody bothers about the through put/ turnaround time of the system which is one of the attributes that measures efficiency. The interest is always on the functional characteristics of the system.

Portability: In this the efforts needed to install the system is a problem. Another problem is the ease/difficulty of changing a given component of the software within a specified environment. We are confronted with all these challenges when dealing with softwares bought and those developed by our students. One of the greatest challenges is not having the facilities to assess software bought or developed. Unfortunately when people buy a piece of software, they only look at the functionality aspect of it. Once it is able to solve their problem, the other non functional aspects which they are not aware of are not taken into consideration. Consequently, the software is discarded as soon as problem arises. A badly-written, software may be functional but subject to buffer overflow attacks.

4. Lessons

The lessons gathered include the following:

1. Mature process help ensure consistent quality of products
2. Assessment of the quality of the products that the process produces provides more accurate analysis of the organization's capability.
3. Ignoring the quality of the products developed by a process leads to an incomplete understanding of the risk that a development effort presents.
4. Ignoring product quality can also result in the misallocation of resources in process improvement and planning.

A method for evaluating the quality of software products is based on the ISO standard for product evaluation. This product assessment focused on software quality assessment standard of four quality areas and these include the following, maintainability, portability, evaluability and descriptiveness. As in Robert and Stephen (1998) their experience shows that processed product assessments produce a more accurate understanding of software development organizations capabilities instead of using the standard SCE that uses process Matura.

Assessment of product quality can identify deficiencies that will increase the risk of using the product as the basis for further development. Problems with product quality can also provide an indication of lack of commitment to disciplined development practices and rigorous process enforcement.

CONCLUSION

Using the ISO/IEC-9126 we have been able to identify the major characteristics used to assess the quality of software. The same standard should also be applied to assess that of the space system software. Since the space system software is a complex and complicated one, all characteristics must be evaluated and must be correct. Consequently for space system software a more serious effort must be made to enforce this standards. This is because once those standards are not enforced it can cause a disaster. From the paper a lot of challenges based on the non functionality characteristics of the standard are highlighted. The benefits of using the standards were also discussed. It was observed that a lot of challenges were based on our ignorance of those characteristics. Also we were handicapped because the facilities needed to assess our developed software were not available. Our interest was based on only the functionality of the system which did not actually tell us how good the software developed was and when those standards are to be put in place.

REFERENCES

ANSI Standard (ANSI/ASQCA3/1978)

Boëhm, B., 1978. "Characteristics of software quality", 1, of TRW series on software technology, North-Holland, Amsterdam, Netherlands.

Fitzpatrick, Ronan; Smith, Peter; and O'Shea, Brendan, 2004. "Software quality challenges.". Proceedings of the Second Workshop on Software Quality at the 26th. International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland. Published by IEEE. http://arrow.dit.ie/scschcomcon/5/

Georgios Gousios, Vassilios Karakoidas, Konstantinos Stroggylos, Panagiotis Louridas, Vasileios Vlachos and Diomidis Spinellis, 2007. Software quality assessment of open source softtware.

http://www.dmst.aueb.gr/dds/pubs/conf /2007-PCI-SQOOSS/html/GKSL07htm

ISO/IEC 9126-1, 2001. International Standard Software engineering – Product quality – Part 1: Quality model, International Organisation for Standardisation, Genève, Switzerland

ISO/IEC-9126 Software Quality characteristics. http://www.sqa.net/iso9126.html

McCall, J., Richards, P. and Walters, G., 1977. “Factors in software quality”, Vol I-III, Rome Aid Defence Centre, Italy.

Robert A. Martin and Andrey E. Taub, 1998. Improving software quality norms within military systems. http://www.mitre.org/work/tech_papers/tech_papers_98/martin_software_quality/

Robert A. Martin and Lawrence H. Shafer, 1996. Providing a framework for effective software quality assessment: Making a science of risk assessment. Paper presented at the 6th annual internal symposium of INCOSE “Systems Engineering”: Practices and Tools. http://www.mitre.org/work/tech_transfer/pdf/riskassessment.pdf

Robert A. Martin and Mary T. Drozd, 1996. Using Product Quality Assessment to Broaden the Evaluation of Software Engineering Capability. Paper presented at the 1996 Software Engineering Process Group Conference. “Broadening the Perspective for the Next Century”. http://www.mitre.org/work/tech_papers/pdf/secapability.pdf

Robert A. Martin and Stephen A. Morrison, 1998. Managing software quality throughout the lifecycle. http://www.mitre.org/work/tech_papers/tech_papers_98/martin_software_quality/ martin_software_quality_paper.pdf