A Literal Review of Software Quality Assurance

C. SenthilMurugan
PhD Scholar
Dept of CSA, SCSVMV University
Kanchipuram

S. Prakasam Ph.D.,
Asst .Professor
Dept of CSA, SCSVMV University
Kanchipuram

ABSTRACT

Software development and maintenance is used to make the error-free Software and also concentrate on time-consuming and complex activity. To evaluate the quality of a software product and to keep its level high is much more difficult than to do them for the other industrial products. For maintaining the quality, performance, speed, efficiency and cost of the software the Software quality Assurance activities, principles and its methods are implemented in the early stages of software engineering development phases. In this paper I include the important concepts of Software Quality Assurance that are used to make the quality software in error free and time consuming manners.

Keywords: - Software quality assurance, Software engineering

1. INTRODUCTION

The computers have been used for commercial purpose. With every aspect of computer development, software engineers have been tasked to solve large and complex programs and in a cost efficient manner. The Software engineers are facing many problems, without having a better knowledge in the field, such as late delivery of software, development teams exceeding the budget, poor quality, user requirements are not completely supported by the software, difficult maintenance and unreliable software and lack of systematic approach. Thus number of large size project failed to call software runaways resulted in software crisis.

2. SOFTWARE QUALITY ASSURANCE

Software quality assurance (SQA) is defined as a planned and systematic approach to the evaluation of the quality, software product standards, processes and procedures. SQA includes the process of assuring that standards and procedures are established and are followed throughout the software life cycle.

The modern view of quality assurance takes a more sophisticated view than that of fitness for purpose. A high quality product is one which has associated with it a number of quality factors

The quality factor is categorized into three types

- Portability
- Usability
- Reusability [2].

3. THE QUALITY SYSTEM OF SQA

The quality system should be flexible leads naturally on to a description of how quality system work. The Heart of the quality assurance is called as quality system or as it is increasingly known as a quality management system. This consists of the management structure, responsibilities, activities, capabilities and resources that ensure that software products produces by projects will have the desired quality factors that both the customer and the developer decide will be built into them. This means that a quality system encompasses activities such as

- The auditing of projects to ensure that quality controls are being adhered to.
- The review of the quality system in order to improve it.
- Staff development of personnel employed within the quality assurance area.
- The negotiation of resources which enables staff who carry our quality assurance activities to function properly.
- Providing input into development-oriented improvement activities; for example the adoption of a new notation for requirements specification.
- The development of standards, procedure and guidelines.

4. TECHNICAL ACTIVITIES OF SQA

The technical activities of SQA system are

Requirement analysis: This is the process of discovering the requirements that the customer has a system. These requirements will be functional in that they will describe what the system is to do or non functional in that they constrain some aspect of a system such as its response time.

Requirement specification: This is the process of detailing the properties of a system which have been discovered from requirements analysis. These are embodied in a document which is often called the system specification or requirement specification.

System design: This is the process of specifying the gross architecture of a system in terms of modules.

Detailed design: This is the process of specifying the individual modules in a system.

Acceptance testing: This is the final test of a system which carried out by the customer in the environment where the system will eventually be embedded. The acceptance test checks that the requirements specification has been correctly implemented. This is normally preceded by System Testing

Modular testing: This is the process of checking out the individual modules of a system, the Modular testing sometime called as Unit testing. [5]

5. LIFE-CYCLE PHASES OF SQA SYSTEM DEVELOPMENT:

Software Development Lifecycle is a subset of the SQA System Development Lifecycle. Here, the classical waterfall model is chosen as Software Development Lifecycle model

5.1. Requirements Definition/Analysis: In this phase, software engineers gather customer requirements by defining them with the help of customer and domain experts.

Most of the time, a developing software must have interfaces with existing hardware and software. Therefore the information about them helps to define the interface requirements. In this phase, software engineers must understand the nature of the program to be built. Therefore, understanding the information domain, system's required functions, behaviors, performance and interface are musts.

5.2. Design: In this phase, software designer identifies the system inputs, outputs and processes. Processing algorithms, data structures, databases and software structures are also defined.

5.3. Code Generation: In this phase, software engineers and programmers transform the design into a code by using a selected programming language. Code review, unit tests, unit integration tests is part of this phase.

5.5. Installation and Conversion: After customer approval, the software is installed to serve the customer. If the new software will be used to replace the existing software, a suitable conversion process must be performed to prevent the interruption in the organization's services.

5.6. Operation and Maintenance: Operation phase begins after the installation and conversion is completed. Maintenance activities are performed during the normal operation period which generally continues a few years.

SQA System Development life cycle includes the above software development life cycle phases, but it has a few phases prior to software development life cycle phases:

Pre-Project Phase*:* In this phase, organizational level activities will be performed. None of the activities of this phase are related to a specific project. During this phase, the SQA system of the organization is initiated, organization quality policy and QA methodology are defined, QA staff is assigned an initial SQA component are developed.

Proposal/Contract Phase: Activities of this phase are performed by the proposal team and legal department for a contract between the customer and the organization. During this phase, firstly proposal team develops a proposal draft from the customer requirements document. After reviewing a proposal draft with a customer, a contract draft is developed from a final approved proposal document. After reviewing the contract draft with the customer, a mutually agreed contract which defines source, timetable and cost estimation for the project is achieved.

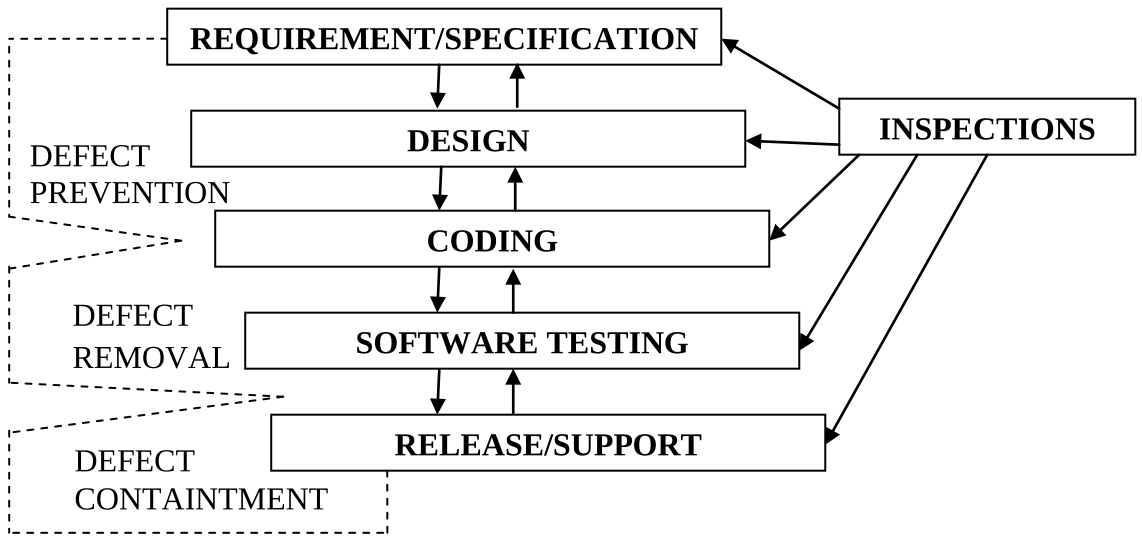

Fig 1. Waterfall Process In the Quality Assurance Activities[13]

5.4. System Testing: In this phase, the system is tested as a whole and system integration is realized. Activities are performed by the testing team.

Project Preparation Phase*:* In this phase, project products, project interfaces, development methodology, development tools, standards, procedures, project schedule, resource and cost estimation, project milestones, staff organization, quality goals, QA activities are identified. [3]

6. TIMING ACTIVITIES OF SQA PROCESSES

The start and end time of the SQA processes are analyzed.

A. Conduct Management's Role in SQA System process is the first step to build an SQA system in an organization. Therefore this process starts at the very beginning of Pre-Project Phase and it never ends.

B. Construct Infrastructure of the SQA System is the base process to establish an SQA system in an organization. It starts at the beginning of Pre-Project Phase and it ends at design phase.

C. Do Standardization and Certification is important to show that the quality of the software products produced by the organization has the level of chosen standards. It starts in the middle of the Pre-Project phase after constructing the infrastructure of the SQA system and never ends.

D. Perform Contract Review starts at the Proposal/Contract Phase after having an offer from the customer and ends at the end of this phase.

E. Develop Project Management Plan, SDP and SQA Plan start at the beginning of Project Preparation Phase just after signing a contract with the customer and ends at the second half of the design phase.

F. Perform Activities to Force and Coordinate the SQA System is necessary to support the SQA system at the organizational level. Therefore it starts at the Project Preparation Phase before entering Software Development Lifecycle phases and it never ends.

G. Develop CM System is necessary to control any changes at the project products (e.g., Source code, documents, data, etc.) and to get the correct version or releases of the products. Therefore, it starts at the end of Project Preparation Phase just before the software development begins. It ends when the project is delivered to the customer completely after completing maintenance phase.

H. Conduct Quality Assurance Audits is an organizational level process to control the departmental SQA activities. Therefore, it starts in the middle of Requirement Analysis Phase just before the project level SQA activity starts. It ends after the project is delivered to the customer.

I. Applying Software Quality Metrics Program is an auxiliary process to collect some data about the project and to use them to control the software development process. Therefore, it starts in the middle of Requirement Analysis Phase just before the project level SQA activity starts. It ends after the project is delivered to the customer.

J. Perform Activities to Support the SQA System is necessary to support the SQA system at department level. Therefore, it starts in the middle of Requirement Analysis Phase just before the project level SQA activity starts. It ends after the project is delivered to the customer.

K. The Apply Risk Management Program is important to take precautions against any possible problems which can be occurred during the development Lifecycle, and can cause undesirable deviations from project schedule and budget. Therefore, it starts in the middle of Requirement Analysis Phase just before the project level SQA activity starts. It ends at the end of installation and conversion phase. Because for maintenance phase, a new risk program must be applied.

L. Perform Documentation Control Activities is necessary to retrieve the project documents later. Therefore, it starts just before the Software Requirements Specification (SRS) document is released and never ends because some documents must be retrieved later.

M. Coordinate Review Meetings is a process which has different review processes. They are starting when a software product is produced and ends when all work listed in the action - item list are completed:

a. The Formal Design Review is performed by the customer after system design is finished by the firm.

b. SRS review is performed after the SRS document is produced at the end of Requirement Analysis Phase.

c. Software Design Description (SDD) review is performed after SDD document is produced at the end of Design Phase.

d. Software Test Plan (STP) review is performed after test plans are completed at the end of Design Phase.

e. Software Test Description (STD) review is performed after test case scenarios are generated at the end of Coding Phase.

f. Code Review is performed after the code is finished and ready to be tested at the end of Coding Phase.

g. The Test Readiness Review is performed to control whether the test environment and source code are ready to test, just before unit integration tests at Coding Phase.

h. Software Test Report (STR) review is performed after tests are completed and a test report is produced at the end of Coding Phase. The similar review process is performed after system testing is completed at the end of System Testing Phase.

i. Software Version Description (SVD) review is performed after all necessary testing and review processes are completed, and the product is ready to release at the end of System Testing Phase.

L. The Apply Test Program is a process to perform all necessary unit levels, system level and customer acceptance level tests. This process starts at the beginning of Design Phase and ends at the end of System Testing Phase.

M. The Apply Process Improvement Program is an auxiliary process to support feedback to the Software Development Process for improvements. Therefore, this periodical process starts when the SQA system frame is established and never ends.

N. Apply SQA Controls to External Participants is an optional process and only performed when at least one part of the project is developed by an external participant. Its purpose is to ensure the quality of the product developed by external participant. Therefore, it starts at the end of Requirement Analysis Phase and ends after the project is delivered to the customer. [3]

7. STANDARDS OF SQA

Standards are the established criteria to which the software products are compared. It established the prescribed methods for developing software. The SQA role is to ensure their existence and adequacy.

Types of standards

- Documentation standards
- Design standards
- Code standards

Documentation standards

The Documentation standards specify a form and content for planning, control and product documentation and provide consistency throughout a project

Design standards

The Design standards specify the form and content of the design product. They provide rules and methods for translating the software requirements into the software design and for representing it in the design documentation.

Code standards

The code standards specify the language in which the code is to be written and define any restriction on the use of language features. They define legal language structures, style convention, rules for data structure and interface and internal code documentation [2]

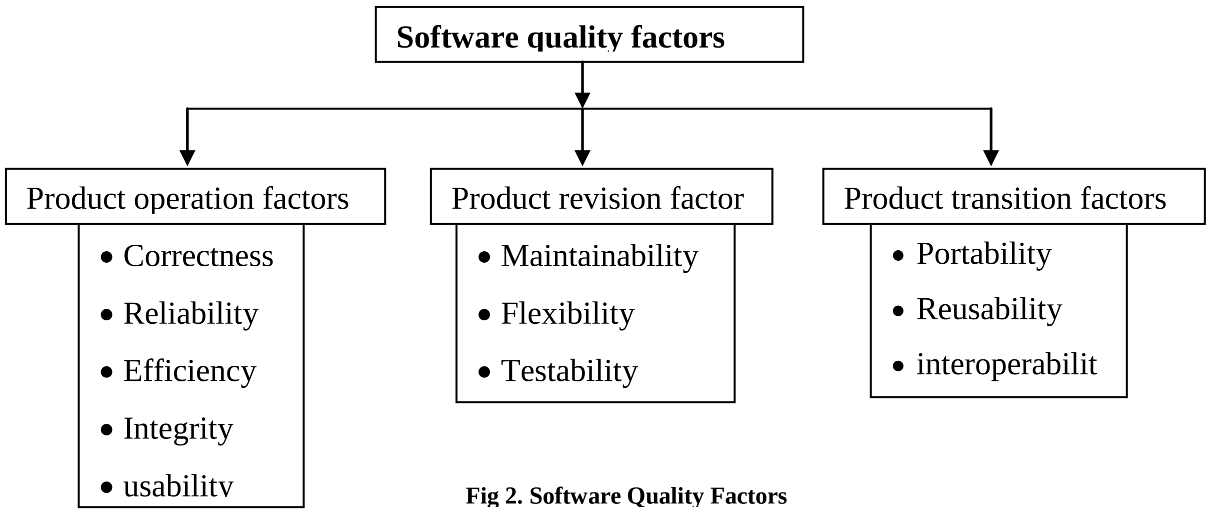

Fig 2. Software Quality Factors

8. SOFTWARE QUALITY FACTORS

Correctness:
- Accuracy, completeness of required output
- Up to datedness, availability of the information

Reliability
- Maximum failure rate

Efficiency
- Resources needed to perform a software function

Integrity
- Software system security, access rights

Usability
- Ability to learn, perform required task

Maintainability
- The effort to identify and fix software failures

Flexibility
- Degree of adaptability (to new customers, tasks, etc)

Testability
- Support for testing (e.g. Log files, automatic diagnostics, etc) [1]

9. SEVEN PRINCIPLES OF QUALITY DEVELOPMENT PROCESS

The First Principles: The Reason It All Exists

A software system exists for one reason: to provide value to its users. Before specifying a system requirement, before noting a piece of system functionality, before determining the hardware platform or development process, ask yourself questions such as: “Does this add real VALUE to the system?” If the answer is “no”, don’t do it. All other principles support this one.

The Second Principle: KIS (Keep It Simple)

There are many factors to consider in any design effort. All design should be as simple as possible, but no simpler. This facilitates having a more easily understood, and easily maintained system.

Simple also does not mean “quick and dirty.” In fact, it often takes a lot of thoughts and works over multiple iterations to simplify. The payoff is software that is more maintainable and less error-prone.

The Third Principle: maintain the vision

A clear vision is essential to the success of a software project. Without one, a project almost unfailingly ends up being “ of two or more minds” about it. Without conceptual integrity, a system threatens to become a patchwork of incompatible designs, held together with the wrong kind use screws.

The Fourth Principle: What You Produce, Others Will Consume
Always specify, design and implement knowing someone else to have to understand what you are doing. The audience for any product of software development is potentially large. Specific with an eye to the users.

The Fifth Principle: Be Open to the Future
A System with a long lifeline has more value. In today's computing environments. Where specifications changes on a moment's notice and hardware platforms are obsolete when just a few months old, software lifelines are typically measured in months instead of years. However, true "industrial-strength" software systems must endure far longer. To do this successfully, these systems must be ready to ready to adapt to these and other changes.

The Sixth Principle: Plan Ahead for Reuse
Reuse saves time and effort. Achieving a high level if reuse is arguably the hardest goal to accomplish in developing a software system. The reuse of code and designs has been proclaimed as a major benefit of using object oriented technologies. However, the return on this investment is not automatic.

Planning ahead for reuse reduces the cost and increases the value of both the reusable components and the system into which they are incorporated.

The Seventh Principle: Think
This last principle is probably the most overlooked. Placing a clear, complete though before action almost produces better results. When you think about something, you are more likely to do it right. You also gain knowledge about how doing it right again. [4]

10. SQA METHODOLOGY
Software testing is as much an art as a science. In large, complex application, such as operating systems, it is practically impossible to iron out every single bug before releasing it both from the difficulty and time constraint points of view. Different software applications require different approaches when it comes to testing, but some of the most common tasks in software QA include:

- PPQA audits
- Peer Reviews
- Validation testing
- Data comparison
- Stress testing
- Conformance testing:
- Load testing
- Usability testing
- Robustness testing

PPQA audits:
Process and Product Quality Assurance is the activity of ensuring that the process and work product conform to the agreed upon process.

Peer Reviews:
Peer reviews of a project's work product is the most efficient defect removal quality control activity.

Validation testing:
Validation testing is the act of entering data pertaining to customer requirements and checking whether the function works as expected by the user. Whereas Verification testing is the act of entering data that the tester knows to be erroneous into an application. For instance, typing "Hello "Into an edit box that is expecting to receive a numeric entry.

Data comparison:
Comparing the output of an application with specific into parameters to a previously created set of data with the same input parameters that is known to be accurate.

Stress testing:
A stress test is when the software is used as heavily as possible for a period of time to see whether it copes with high levels of load. Often used for server software they will have multiple users connected to it simultaneously. This is also known as 'Destruction testing'. Fault injection is especially Useful for any software system with exposed interfaces, E.g., Protocol implementations.

Conformance testing:
Confirms that the software implementation complies with established standards.

Load testing:
Establishes the maximum amount of traffic that a target can accept, usually measured in packets-per-second (PPS) or calls-per-second or any other rate-oriented metric depending on the application. Such devices often have very fast hardware-optimized data plane processing, with software-based control and management plane functions. The SQA process is designed to ensure that the software portions of the system properly support the hardware functions.

Usability testing:
Sometimes getting users who are unfamiliar with the software to try it for a while and offer feedback to the developers about what they found difficult to do is the best way of making improvements to a user interface.

Robustness testing:
Software systems are presented with invalid or unexpected inputs to determine whether they have robust error-handling or input validation. Such systems pass the test(s) if they can tolerate a wide variety of invalid or unexpected inputs across the entire protocol interface specification. Commercially robustness testing products are available, including Dynamics' Service Analyzer. [2]

11. CONCLUSION
In this paper, the Software Quality Assurance concepts that are used to make the error-free Software and concentrate on complex activities and used to complete in time and in cost estimation is prevented. As discussed in the previous sections of this paper the Software quality Assurance activities, principles, factors and its methods are implemented in the early stages of software engineering development phases, because of this activity the software developer get the knowledge about the software what he is going to develop, it may reduce the rework and failures of the softwares. Nowadays all the software development industries are implementing the SQA component to get quality softwares

and to satisfy all the requirements of the customer and make a finite software.

12. REFERENCES

[1] Galin, Daniel (2004), Software Quality Assurance – From theory to implementation, Pearson –Addison Wesley, England.

[2] Chandramouli, pradiba, Software Quality Assurance

[3] Omer Korkmazsystem, Simulation for software Quality Assurance (SQA).

[4] David Hooker, Seven Principle of Software Development.

[5] Darrel Ince, ISO 9001 and Software Quality assurance

[6] Warden, R. (1992), Software Reuse and Reverse Engineering in Practice. London, England: Chapman & Hall, 283–305.

[7] Sametinger, Johannes (2006), Software Engineering with Reusable Components, Springer Verlag GmbH; 1st Edition, ISBN-13: 978-3540626954..

[8] B. W. Boehm. Software Engineering Economics. Prentice Hall, 1981.

[9] Felipe Ortega, Daniel Izquierdo, Pedro Coca, Introduction to software quality evaluation

[10] ESA Board for Software Standardization and Control , Guide to software quality assurance

[11] M. Jorgensen, Software quality measurement

[12] Norman f. Schneidewind, Knowledge Requirements for Software quality measurement

[13] NITS (2006), NIST/SEMATECH e-Handbook of Statistical Methods,

[14]The Software CrisisSource:http://www.unt.edu/benchmarks/archives/1999/july99 /crisis.htm)

[15] Balan, S. (2003), A Composite Model For Software Quality Assurance