International Journal of Computer Trends and Technology
ISSN: 2231 – 2803 / https://doi.org/10.14445/22312803/IJCTT-V69I7P107

Volume 69 Issue 7, 48-51, July 2021
© 2021 Seventh Sense Research Group®

Original Article

Quality Assurance Factors: Key Aspects for Software Quality Control and Testing

Kishan N Milan

University of Florida, Florida, USA.

Received Date: 05 June 2021
Revised Date: 08 July 2021
Accepted Date: 18 July 2021

Abstract *- In the modern technology era, the increased complexity of software involves a recognizable cost; thus, the demand for the best quality software is obvious. Quality is basically the extent to which a system, component, or process meets specified requirements, customer or user needs, or expectations. Hence a well-defined plan is to be followed to avoid any of the quality sufferings. The purpose of the paper is to suggest a complete quality assurance plan aimed at giving the confidence to the developers and the customers that the specified requirements will be met and the final product would be of satisfactory quality standards. This document outlines the different standards and methodologies going to be used and the various metrics and tests that are employed for ascertaining the quality of the deliverables at various stages of the development process. In addition, it describes the manner in which the various quality assurance activities are going to be distributed among the project members and identifies any training needs and walkthroughs to carry on these activities.*

Keywords *- SQAP(Software Quality Assurance Plan), SDLC (System Development Life Cycle).*

I. INTRODUCTION

The quality assurance plan outlines the methods and procedures to be employed throughout the software development process. The QAP is a tool that helps to achieve the product of the highest quality using the limited resources and in accordance with the project schedule. The set of systematic activities providing evidence of the ability of the software process to produce a software product that is fit to use (G. Schulmeyer and J. McManus, *Software Quality Handbook*, Prentice Hall, 1998.) is the quality assurance plan itself. Need for software quality assurance is everywhere and in every activity step, and then only we can get a fully desired qualitative product. But the point arises how one can ensure the qualitative ness of the product. Quality can't be quantifiable, but surely one can draw a conclusion of qualitative ness by comparing the final product with the needs that were specified by the user during the requirement analysis. Measure of quality is not just done after the completion of software but has to be done at each and every phase of development. Monitoring processes and products throughout the software development lifecycle to ensure the quality of the delivered product(s) Monitoring the processes provides management with objective feedback regarding process compliance to approved plans, procedures, standards, and analyses Monitoring the products focus on the quality of the product within each phase of the SDLC with the objective to identify and remove defects throughout the lifecycle, as early as possible. The goal of the SQA plan is to monitor processes throughout the [1] Software Development Life Cycle (SDLC) that ensures the quality of the delivered product. Hence to make the software safer with respect to quality, follow a good S/w Development Process, perform System and Software Safety Analyses & Software Development Analyses, design appropriate safety features, error handling including recovery, self-checks, and above follow healthy software quality assurance plan.

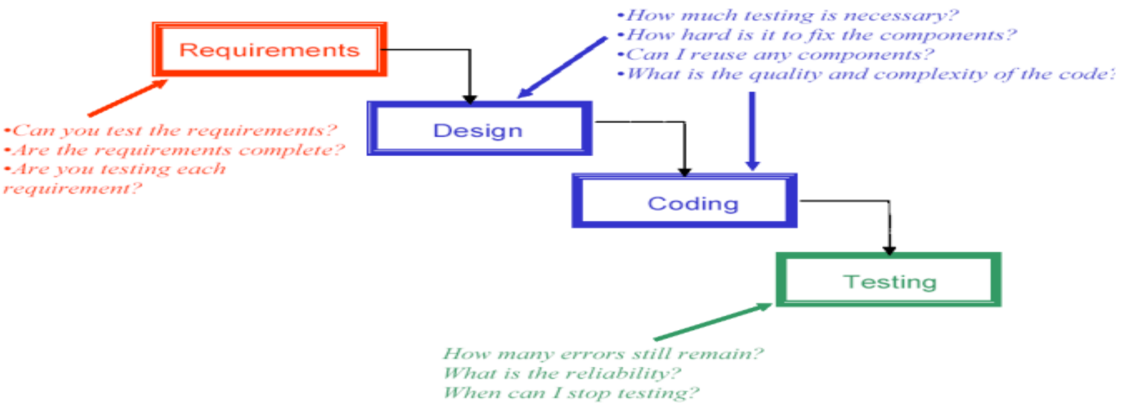
Requirements
•Can you test the requirements?
•Are the requirements complete?
•Are you testing each requirement?
Design
•How much testing is necessary?
•How hard is it to fix the components?
•Can I reuse any components?
•What is the quality and complexity of the code
Coding
Testing
How many errors still remain?
What is the reliability?
When can I stop testing?

Fig. 1 Measure software quality

II. SOFTWARE QUALITY ASSURANCE

Software quality is conformance to explicitly state functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software (Pressman, 1987). Software Quality Assurance (SQA) is a ‘planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements. [2] SQA does this by checking that:

- Plans are defined according to standards
- Procedures are performed according to plans
- Products are implemented according to standards.

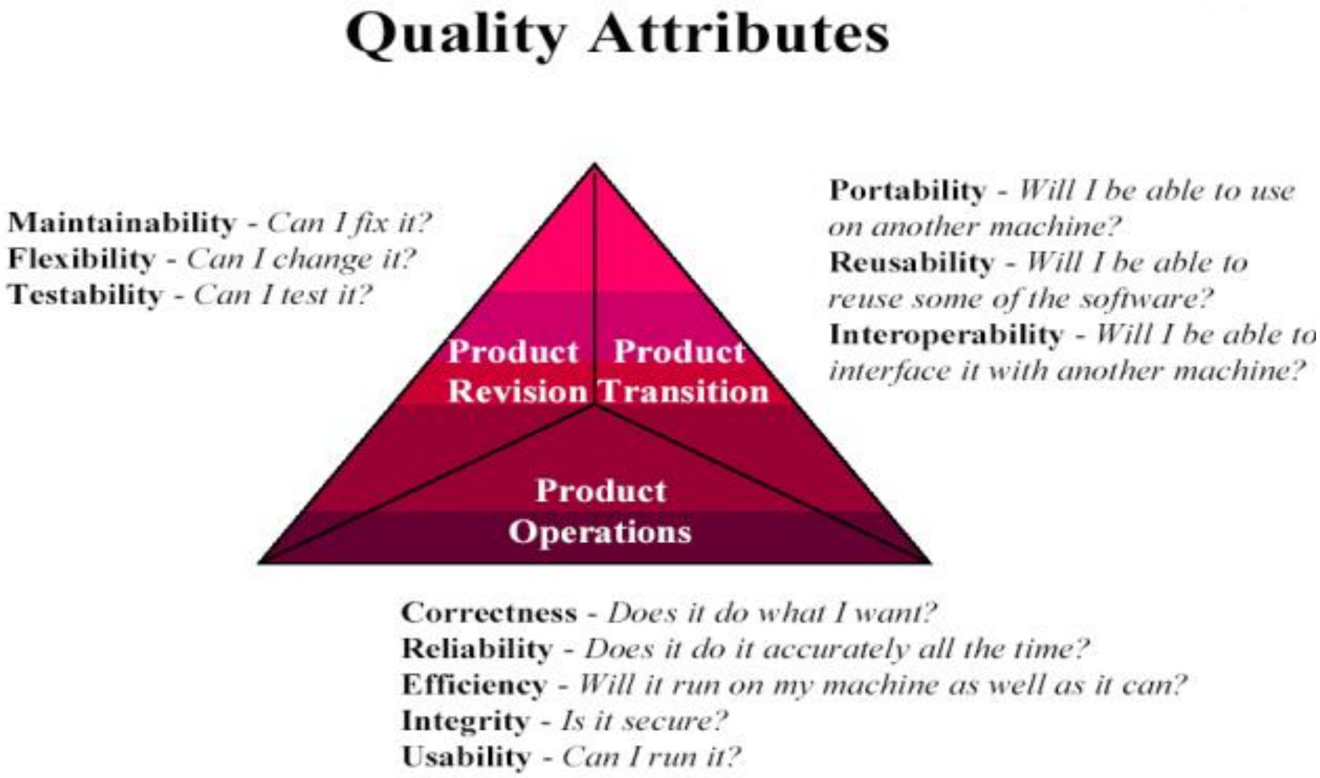
Quality Attributes
Maintainability - Can I fix it?
Flexibility - Can I change it?
Testability - Can I test it?
Portability - Will I be able to use on another machine?
Reusability - Will I be able to reuse some of the software?
Interoperability - Will I be able to interface it with another machine?
Product Revision
Product Transition
Product Operations
Correctness - Does it do what I want?
Reliability - Does it do it accurately all the time?
Efficiency - Will it run on my machine as well as it can?
Integrity - Is it secure?
Usability - Can I run it?

Fig 2. Quality attributes

III. PURPOSE AND SCOPE

SQA is part of a management discipline that must be practical, results-oriented, and aligned with evolving technology to produce quality products. SQA ensures the following:

- [3]To define quality and required quality standards for each deliverable and the final products and ensure compliance with established standards and procedures.
- To describe methodologies and procedures to be employed to verify that the deliverable has the defined quality standards and minimize defects.
- To describe a timeline for the quality verification.
- To fix the roles of the project members in carrying out the quality tests.
- To define an organizational structure for quality reporting and subsequent rectification procedures.
- To identify the tools and procedures required.
- To identify the training required by individual group members to perform their assigned tasks and perform both product and process assessment/audits.
- Follow up of well-defined authorized quality assurance practices.

IV. CHARACTERISTICS

The SQAP should be plain and concise. The document should be clear, consistent, and modifiable. [4]The author of the SQAP should be familiar with the purpose of the software.

- Reliability: precision with which program performs as expected
- Modifiability: the effort required to modify
- Maintainability: the effort required to locate and fix errors
- Security: extent to which access is controlled
- Efficiency: the number of computing resources required
- Usability: the effort required to learn, operate, and interpret

V. A ROADMAP TO SOFTWARE QUALITY ASSURANCE PLAN

A. Management

[5]The first step in SQAP is structuring of the organization for quality assurance, and the associated responsibilities. The SQAP should define the roles to be carried out, allocate people to roles, and also define the

effort and schedule. ANSI/IEEE Std 730-1989, ‘Standard for Software Quality Assurance Plans' [Ref 3] recommends that the following structure be used

- Identify the organizational roles that control and monitor software quality
- Describe the relationships between the organizational roles;
- Describe the interface with the user organization.

B. Tasks: Tools, techniques, and methods

Define the [6]SQA tasks that are to be carried out during the life cycle to which this SQAP applies. This involves sequencing of the selected tasks and identification the SQA tasks for which each organizational role it is responsible. Identification of the tools, techniques, and methods used to develop the software. This section should describe how the use of the tools, techniques, and methods would be monitored.

C. Documentation and Methods

Identification of the documents to be produced in all the phases of SDLC along with the standards, practices, and conventions that will be used to produce the documents. The SQAP normally contains (or references) a documentation plan listing all the documents to be produced in all the phases. The statement of how the documents will be checked is prepared.

D. Standards, Practices, Conversions, and methods

It involves identification of the standards, practices, conventions, and metrics used to specify software quality, and full explanation of how SQA will check that the required quality will be achieved.

a) Design standards

It includes the identification of the standards, practices, and conventions that will be used in the design phase of the software**.** Design standards are normally defined or referenced in the ADD(Architectural Design Document).

b) Coding standards

It involves the identification of the standards, practices, and conventions that will be used in the phase to write code. Coding standards are normally defined or referenced in the DDD (Development Design Document)**.**

c) Testing standards and practices

It includes the identification of the standards, practices, and conventions that will be used in the phase to test the software. Testing standards are normally defined in the SRD (Software Requirement Document) and SVVD (Software Verification and Validation Document).

d) Selected software quality assurance metrics

Quality assurance metrics include the identification of the metrics that will be used in the phase to measure the quality of the software. Metrics are normally defined in the project standards and plans.

E. Reviews and Audits

It is one of the important phases and makes arrangements for the technical reviews, inspections, walkthroughs, and audits that will be held during each phase. It should describe how adherence to the review and audit procedures will be monitored and the role of SQA personnel in the review and audit process.

F. Test

Two types of documents are prepared in this phase. First is a Test Document that includes test reports, and second is an Error Document that contains errors detected (like: incorrect functions, interface problems, and database errors) and their solutions.

G. Problem Reporting and Corrective Action

This section should describe how adherence to the problem reporting procedures would be monitored. This section may describe the metrics that will be applied to the problem reporting process to estimate the software quality. Problem reporting, tracking, and corrective actions are a process by which any problem in the product is reported, logged into a common repository and tracked through closure.

H. Media Control

Identification of the procedures used to maintain, store, secure, and document controlled versions of the physical media on which the identified software resides. These procedures should be defined in the SCMP (Software Configuration Management Plan).

I. Supplier Control

A supplier is an external organization that develops or provides software to the project. This section should identify the standards that will be applied by suppliers. This section should describe how adherence of the suppliers to the applicable standards will be monitored, identification of the procedures that will be applied to goods supplied, such as commercial software and hardware.

J. Records Collection, Maintenance, and Retention

It should describe where the records are kept, and for how long.

K. Training and Risk Management

Identification of training programs, defined for the project staff and explanation of how SQA will check that they have been implemented. Identification of the risk management procedures used in the project (which should be described in the SPMP (Software Project Management Plan).

VI. WALKTHROUGHS & INSPECTIONS, REVIEWS AND SQA REPORTING

Walkthrough is a peer review of a software product that is conducted by walking through the product sequentially (line by line) to judge the quality of the product under review. The advantage that walkthrough offers is that it is a great opportunity to cover more material in a single session when compared with a formal inspection. But one might not find defects of the nature as one is with formal inspections. Hence we would conduct inspections to overcome the above disadvantage. Inspection is useful in dealing with more complex defects that cannot be dealt with during a walkthrough. Inspections not only detect faults, but they also provide for correcting them. The document (specification, design, or code) is carefully checked by a team of two, not including the developers. The advantage of a review is that the different skills of the team members greatly increase the chances of finding a fault. The motivating factor behind reviews is that others are much more likely to uncover faults than the original developer. Software Configuration Management integrates the technical and administrative actions of identifying, documenting, changing, controlling, and monitoring the change to software product throughout its life cycle.

- It is essential that the reporting process for SQA activities is clearly defined with the project structure, and the reporting should target senior project management.
- SQA reporting should not be under the software development manager.
- SQA should report to someone in management high enough to have some chance of influencing priorities. However, with lower-level reporting, working relationships are generally better.
- Whenever possible, SQA should report to someone who has a vested interest in software quality.

VII. SQA STANDARDS AND METHODOLOGY: A BRIEF SUMMARY

- Standards and Methodology listed below are a typical set of standards:
- Pro forma the quality assurance plans.
- List the SQA responsibilities of the developing and the reviewing team.
- Criteria for testing the quality of the product, which the reviewer should adopt.
- Ensure that the reviewers collect all the defects and report them.
- The developing team analysis of the defects
- Criteria to determine which software element require the closest SQA attention.
- Reviewing team producing the feedback report of the delivered product.

VIII. SQA STANDARDS AND METHODOLOGY: A BRIEF SUMMARY

- Heterogeneity: developing techniques for building software that can cope with heterogeneous platforms and execution environments.
- Trust: developing techniques that demonstrate that its users can trust software.
- Ensure SQA skills are current and customer-driven
- Delivery: developing techniques that lead to faster delivery of software.
- SQA must remain independent

IX. CONCLUSION

The true success of the software depends upon the quality of the software obtained. Satisfying customer needs, making an effective, reliable, and a quality product should be the aim of software engineering. Simple, easy software can be developed, but the difference between the good and the best lies in developing average software and a quality software. To develop quality software, it is important to make a healthy software quality assurance plan. This plan is not limited to one phase but all the phases of software engineering. Following a good and effective software quality assurance plan guarantees a reliable and a quality product. Need for software quality assurance is everywhere and in every activity step, and then only one can get a fully desired qualitative product.

REFERENCES

[1] Madduri, A. Human Gait Recognition using Discrete Wavelet and Discrete Cosine and Transformation Based.

[2] Ayyagari, MaruthiRohit., Efficient Driving Forces to CMMI Development using Dynamic Capabilities. International Journal of Computer Applications. 178. 24-29. 10.5120/ijca2019919024., (2019).

[3] Ayyagari, MaruthiRohit., Classification of Imbalanced Datasets using One-Class SVM, k-Nearest Neighbors, and CART Algorithm. International Journal of Advanced Computer Science and Applications. 11. 10.14569/IJACSA.2020.0111101.

[4] Atoum, Issa&Ayyagari, MaruthiRohit., Effective Semantic Text Similarity Metric Using Normalized Root Mean Scaled Square Error. Journal of Theoretical and Applied Information Technology. 97 (2019) 3436-3447.

[5] Ayyagari, M. R., Cache Contention on Multicore Systems: An Ontology-based Approach.arXiv preprint arXiv:1906.00834., (2019).

[6] Ayyagari, M.R., iScrum: Effective Innovation Steering using Scrum Methodology. Int. J. Comput. Appl, 178(10) (2019) 8-13.