IMPORTANCE OF QUALITY ASSURANCE TO MAKE REALISTIC APPROACH OF SOFTWARE DEPLOYMENT

Radha Krishna Rambola, SVKM's NMIMS, Shirpur, India
Rakesh K. Sharma, University of Maryland Eastern Shore, USA

ABSTRACT

Software Quality Assurance (SQA) features entire software development process which helps in monitoring and improving the process that ensures the correct standards and procedures are followed and the issues and problems arised are also dealt with. Software Quality Assurance is a necessary factor to each and every software development process. With the help of Software Quality Assurance, the examination of the guidelines required in advancing the development process can be done as well as the stakeholders can be shown with the help of a study that continuous progress is being made on the project. The evaluation of the quality of a software product and maintain the level of quality high is a tedious process. Many factors such as performance, speed, efficiency and cost of software are needed to be maintained to ensure quality of a software product. This paper stresses on the need of software quality process and also describes the ways by which it can be achieved.

Keywords: Quality Factors, Software quality assurance, Quality factors, Determination factors of SQA, Reasons for software bugs, CMM.

INTRODUCTION

Software quality assurance (SQA) is a process that ensures that the software developed meets the standardized quality specification. In software development life cycle (SDLC), SQA is a ongoing process that regularly checks the software under development in order to ensure that defined quality measures are met by the software.

SQA ensures the quality of software by implementing various software testing methods to test the software, In SQA, instead of analyzing the quality after completion of the software development, quality in each phase of development is tested until the software is complete. The next phase of software development process is entered only once the current phase complies with the required quality standards (Way, Badawi, Aichouni, and Boujelbene, 2016).

QUALITY FACTORS WITH SOFTWARE PRODUCTS

- **Extensibility and Flexibility:**
  Extensibility is the ability of a software system to include any functionality without causing any loss to the system meanwhile flexibility refers to the ability of the software system to include any functionality or change or remove any existing functionality in the system without any damage being caused to the system (Azizi & Makmur, (2015).

- **Maintainability:**
  Maintainability refers to the ability of software system that is responsible for any changes occurring during the correction of errors and basic changes in the functions of the system (Azizi, and Makmur, (2015).

- **Efficiency and Performance:**
  Efficiency refers to the ability of the software to efficiently use its resources meanwhile performance is the ability of the software system that focuses on how quickly the software can respond to a particular task defined by the user (Azizi, and Makmur, 2015).

- **Scalability:**
  Scalability is an attribute of the software system that defines whether the system responds to the user actions in an acceptable amount of time even if that causes the load of the system to increase Azizi, and Makmur, (2015).

- **Accessibility and Usability:**
  Accessibility is the ability of the software that defines how easily the software can be accessed meanwhile usability is the ability of the software that defines the simplicity of the software Azizi, and Makmur, (2015).

- **Platform Compatibility and Portability:**
  An ideal software runs on any platform that it is being used on which refers to the platform compatibility. Portability is the ability of the software system that focuses on the adaptability of software on various platforms so as the software becomes platform compatible (Azizi, and Makmur, (2015).

- **Testability and Manageability:**
  A software is considered as a quality software when it satisfies all the customers' specifications which can be achieved by quality testing of the software system. After the deployment of the software, the software should be manageable (Azizi, and Makmur, (2015).

- **Security:**
  Security is the ability of the software system that refers how secure the software is from any intruder who is trying to corrupt the software (Azizi, and Makmur, (2015).

HOW TO DETERMINE THE SOFTWARE QUALITY ASSURANCE

There are two ways to determine the Software Quality Assurance. They are mentioned below:

I. The Defect Management Approach

The defects caused in a software are categorized depending upon the basis of the severity in the software. The number of the defects are calculated and analyzed and hence the actions are decided based on the analysis done on the occurrence of defects. These defects often come from very minor problems and usually cause the coding defects, the improper completion of the requirements of the customers and ultimately the software product end up not matching the expectations of the customers. This approach is based on some principles:

- The basic goal is preventing defect but complete prevention of defects is never possible and hence the purpose of this approach is to detect the defects as soon as possible and minimizing the impact that these defects can cause.
- In order to prevent the defects, alteration of some processes is required.
- To improve the process, the defect measurement processes should be integrated into the software development process.
- Defect information always aids in improving the processes (Son Nguyen, 2014).

II. The Software Quality Assurance Attribute Approach

The following mentioned attributes describe the Software Quality Assurance Attribute Approach:

- **Functionality:** The functionality attribute incorporates various functions that the software performs.
- **Suitability:** This attribute checks whether the functions performed by the software are appropriate or not.
- **Accuracy:** This attribute makes sure whether all the functions are being used in a correct manner or not.
- **Interoperability**: This attribute is responsible for ensuring the adequate interactions of the software with various other elements.
- **Security:** This attribute guarantees the capability of software to handle any security breach.
- **Reliability:** This attribute serves the purpose of ensuring the ability of the software to continue performing its task without causing any delays.
- **Maturity:** This attribute is responsible for guaranteeing the fact that there are very less possibility of the software failure while performing any activity.

- **Recoverability:** This attribute measures the rate at which the software can recover once it undergoes any type of failure.
- **Usability:** This attribute serves the purpose of ensuring the use of a particular function of software.
- **Understand ability:** This attribute measure the amount of effort required by the user in order for understanding of the functions of the software.
- **Efficiency:** This attribute defines how correctly the software can perform its task work.
- **Maintainability:** This attribute defines how much maintenance is required to analyze and correct the failure in the software.
- **Analyzability:** This attribute is responsible for figuring the main reason of the software failure.
- **Changeability:** This attribute measures the effects and responses of software when some change occurs (Son Nguyen, 2014).

WHY SOFTWARES ARE PRONE TO BUGS

- **Miscommunication of requirements introduces error in code:**
  Introduction of defects in the code is caused by the miscommunication of the requirement.it is one of the most common problems. Thus leading to errors and lack of miscommunication. Due to the miscommunication at the requirements stage, it lets the errors in the development process right from the first step of SDLC from gathering the resources information and requirements to the complete development of the software. Incomplete and vague requirements are one of the biggest issues for the developers while developing the software application which eventually causes issues in the testing process. Miscommunication of the requirements can be caused due to communication errors between different developers working on the same software (Colson, Jr., and Prell, 1992).

- **Unrealistic time schedule for development**:
  In today's world the wants for the new things to be implemented are far increasing even before the competitors start to implement those new things. Therefore with unrealistic deadlines, developing the software application can have an impact over the quality of the software only to meet the fast and difficult project deadlines. No software should be developed under high pressure of the schedules and deadlines or with limited or insufficient resources. All these defects only cause the way in for the errors in the software (Colson, Jr., and Prell, 1992).

- **Lack of designing experience:**
  For any software process, the main aspect of the whole process is in the designing phase. Based on the design we can estimate how good the software is and the overall development process. Research and requirement are always required for any design or implementation of the project. Sometimes while there is a time limit for completing the whole research, design and the development of the software which thus causes defects and introduction of the errors(Colson, Jr., and Prell, 1992).

- **Lack of coding practices experience:**
  Immature or bad coding leads to introduction of the defects and thus causing errors. For any development of the software, coding is very important. Unhandled errors and expressions, improper validation of the inputs or human errors in coding the process are all included in the bad coding. By using faulty tools for compiling or debugging, it adds errors (Colson, Jr., and Prell, 1992).

- **Human factors introduces errors in code:**
  Human errors are subjective and one of the biggest areas for the introduction of the errors. The system developed by the human is not perfect all the time and therefore we cannot expect the software to be error and bug free and thus there are many modifications and versions of the software that would keep removing the bugs and errors (Colson, Jr., and Prell, 1992).

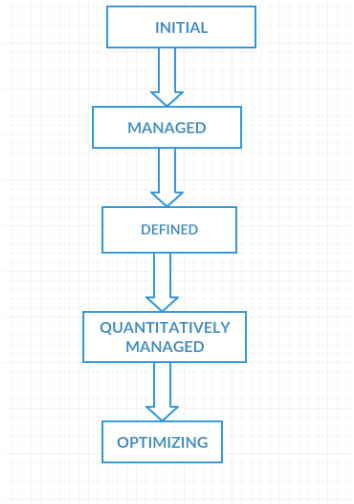
- **Last minute changes in the requirement introduce error:**
  Last minute changes in the requirement gathering for the development of the software is the biggest area for introducing the errors and causing defects. This could lead due to the difficult and impossible deadlines or due to the miscommunications (Colson, Jr., and Prell, 1992).

CAPABILITY MATURITY MODEL

The Capability Maturity Model (CMM) is a technique that is required to refine and develop the software development process of any organization. It elaborates the way of improving the maturity level of an organization through five levels. It was developed by the Software Engineering Institute (SEI) which was founded in 1984 to find solution for various issues and problems occurring in the field of software engineering and hence, advance the field of software engineering (Zakuan, Yusof, and Laosirihongthong, 2008).

CMM is used to evaluate an organization on the basis of its five levels that are based on certain Key Process Areas (KPA). CMM outlines the maturity of the organization based upon the project and the clients that the organization is working on. Measuring the maturity levels is done by achieving the specific and generic goals. Each level of CMM is responsible for ranking the organization as per the standardization of processes in the subject area that are being assessed (Rambola, Imam, and Ahmad, 2013).

There are five maturity levels in CMM as shown below:



The description of the characteristics of each maturity level is given below (Rambola, Imam, and Ahmad, 2013).

Maturity Level 1 – Initial: When the organization that has been assigned with the development of a software had not decided any procedure for the software development and doesn't even have a system that can track the project so that the developers can tell an estimate cost of development to the user or determine any finish dates (Rambola, Imam, and Ahmad, 2013).
Maturity Level 2 – Managed: The organization has installed the fundamental processes and controls required for software management but lacks the teamwork amongst the various groups (Rambola, Imam, and Ahmad, 2013).
Maturity Level 3 – Defined: The organization had gathered a fundamental set of the controls and procedures required so that the developers can switch projects easily and customers can have an interaction for their project with the various groups (Rambola, Imam, and Ahmad, 2013).
Maturity Level 4 – Quantitatively Managed: The organization has installed the systems that are required to assess the quality of numerous projects (Rambola, Imam, and Ahmad, 2013).
Maturity Level 5 – Optimizing: The organization has achieved all the above mentioned levels and can notice a change in the performance with time (Rambola, Imam, and Ahmad, 2013).

CONCLUSION

Software quality assurance's capability to confront the problems encountered during software development is extremely necessary. This paper mainly focuses on the methods through which achieving the goals becomes easier for software development teams and ensures all the requirements necessary for software quality assurance during development of software product.

REFERENCES

Way,Y., Badawi, I., Aichouni, M., & Boujelbene, M. (2016). A survey on the implementation of total quality management (TQM) at manufacturing industries in North Region, Kingdom of Saudi Arabia. 2016 2nd International Conference on information Management (ICIM)

Azizi, A., & Makmur, P. D. (2015). An integrated framework of critical techniques in implementation of Total Quality Management. The 2015 1nternational Conference on 1ndustrial Engineering and Operations Management Dubai, United Arab Emirates (UAE), March 3 - 5, 2015.

Son NGUYEN, D. (20114) Total Quality Management in product life cycle", 2014 IEEE International Conference on Industrial Engineering and Engineering Management.

Colson, Jr., J. S. & Prell, E. M. (1992) Total quality management for a large software project. 1992 AT&T Technical Journal.

Rambola, R. K., Imam, Z., & Ahmad, N. (2013) An Efficient Approach Concurrency Control In Database Management System: A Performance Analysis. International Journal of Computer Science and Network Security, 13(7), ISSN: 1738-7906.

Zakuan, N. M., Yusof, S. M. & Laosirihongthong, T. (2008) Reflective review of relationship between total quality management and organizational performance. 2008 4th IEEE International Conference on Management of Innovation and Technology.

de Klerk, A. M. (1994) Total quality in management decision making. Engineering Management Conference, 1994. 'Management in Transition: Engineering a Changing World', Proceedings of the 1994 IEEE International.

Tang, D., & Cai, S. (2011) An analysis of factors influencing the TQM implementation degree of maturity in enterprises. 2011 International Conference on Business Management and Electronic Information.

Md Kabir, A., Ur Rehman, M., & Majumdar, S. I. (2016) An Analytical and Comparative Study of Software Usability Quality Factors. 7th IEEE International Conference on Software Engineering and Service Science (ICSESS).

Nadia, D. A., Zakuan, N., Bahari, A. Z., Md Ariff, M. S., Chin, T. A., & Mat Saman, M. Z. (2016). Identifying critical success factors for TQM and employee performance in Malaysian automotive industry: A literature review. IOP Conference Series Materials Science and Engineering.

Rambola, R. K., & Jatkar, M. (2018). An effective synchronization of ERP in textile industries. In Proc. (IEEE Xplore) IEEE International Conference on Electronic, Communication and Aerospace technology (ICECA-2018) India during 29 & 30 March 2018.

Mehra, S., Anand, S., & Rambola, R. K. (2018) Synchronization of software quality management through testing. In Proc. (Springer) International Conference on Information Technology and Digital Applications (ICITDA-2018) India during 6& 7 April, 2018.

Rambola, R. K., Ahmad, N., & Sharma, B. K. (2012). Efficient data accessing through heterogeneous ERP solution. International Journal of Computer Applications (0975 – 8887) 53(6).

Rambola, R. K., Ahmad, N., & Deepak, A. (2012). An effective security management of database through DNA fingerprinting recognition using geometric parameters. International Journal of Computer Applications & Information Technology, I (II). ISSN: 2278-7720.

Radha Krishna Rambola, R. K., Md. Zafar Imam, Z. & N. Ahmad, N. (2013). An efficient approach concurrency control in database management system: A performance analysis. International Journal of Computer Science and Network Security, 13(7), ISSN: 1738-7906.