Improvement of Key Problems of Software Testing in Quality Assurance

Ali Arman

Department of Computer Science and Technology

Weifang University of Science and Technology

aliarman787878@gmail.com

Abstract:

Quality assurance ensures that the project will be finished in accordance with the functionality, standards, and specifications that have already been accepted. It must be perfect and free from any potential issues. From the beginning of the project, it keeps an eye on and works to advance the development process. The entirety of the software development process, including the design of software, coding, source code control, code review, change management, configuration management, and release management, is referred to as software quality assurance, or SQA. The main issues with software testing in quality assurance are addressed in this study. Some issues with the current software methods include testing procedures, user attitudes, and organizational cultures. Each of these tree problems has a few related issues, such testing shortcuts, testing time reductions, inadequate documentation, etc. In this essay, we offer solutions to the aforementioned issues through recommended ways. Software quality [2] is gaining much more interest these days as well as much more importance is being given to the production of high quality software products. The process of developing software is complex and calls for the careful integration of many academic fields, technical tasks, project management, etc. The majority of software is created by numerous designers and programmers working together over several man-years. No single person can fully comprehend the final output. No matter how [2] well-designed methods used to test the final product, how complete the documentation, how structured the methodology, the development plans, the project reviews, the walkthroughs, the database management, the configuration control, no matter how advanced the tools and techniques - all will come to nothing and the project will fail if the quality management system is not effective.

Keywords: Software Quality Assurance, Testing, planning, documentation.

1. Introduction

Developing a [1] good software system is a very difficult task. Plenty of indicators of the quality of software must be considered in order to create a high-quality software product. Because it typically affects software quality features including software dependability, software testability, and software maintainability, the system complication dimension is crucial to the control and supervision of software quality. Thus, software quality assurance (SQA) [1] needs to be addressed keeping in view the new strategies, tool, methodologies and techniques applicable to software development life cycle.

2. Related Work

In this paper the author [1] describe that insecurely tested software system lowers down the system reliability that afterward negatively affects ‘Software Quality’. This article discusses "Software Reliability Measurement" and the applicability of the ISO approach to software quality assurance (SQA). Software houses need to shift toward a better software culture in order to maximize testing efficiency and software quality. Instead of focusing solely on finding and fixing flaws in the product that has been deployed, testing should prioritize increasing customer happiness. The variables influencing software quality management have been examined in this study, and the author has made several recommendations for potential enhancements. The findings of this study could be very helpful to the researchers in terms of quantifying the particular instruments used to measure certain features of software quality. In their paper [2] J Barrie Thompson and Helem M Edwards says that given the fact that time is very limited in the course the authors believe that in the Systems Engineering module they have been able to provide an appropriate balance between master’s level research activities and those of a more practical nature. Nonetheless, the writers think they've been able to provide the pupils some incredibly beneficial real-world experiences. The following are some specific advantages of the writers' approach: Students are given an early overview of the subject matter thanks to the planned arrangements, especially those that take into account the front-loading of formal lectures. They then have enough time to thoroughly investigate the research topic or areas they have selected. The students got the opportunity to use some of the software engineering concepts they had learned about and started to comprehend from their research projects during the practical portion of the program. In the [3] modern years an increasing number of software organizations have launched initiatives to improve their software process. Most of them have failed to go beyond diagnosis and plan of action, converting those plans into actual and workable activities. This paper presents a collection of fundamental tools to support the implementation of specified practices for Software Configuration Management (SCM) and Software Quality Assurance (SQA), two software process areas that are the focus of this work. A modified version of SQUID (software Quality In the Development Process) is used to define, track, and assess the software product quality as it is being developed. The authors present the outcomes of an application, demonstrating how the suggested adaption aids in formalizing and standardizing the implantation process, establishing measurable objectives, and more precisely assessing the outcomes. The authors [4] illustrate that software quality assurance is faced with many challenges starting with the method of defining quality for software. The final definition of high-quality software is typically impacted by the software consumption environment, but it must be well understood. SQA has several facets, some of which are included within software development life cycle phases and others of which span many phases. SQA is a very challenging field that has a significant impact on a project's overall performance. It also calls for a very broad range of abilities. The core set of necessary abilities is currently being expanded to include new information domains like software safety and reliability. For SQA to be effective, it needs to be separate from development groups.

Massood Towhiddnejad [5] describes an experiment which involved students in the undergraduate computer science senior project capstone design course, and students in the graduate software testing course. Senior project participants are graduating seniors who have finished all but the most of the two mandatory computer science courses. Their focus was on software development life cycle and software processes during their one-semester software engineering course, which they have already finished. Prior to enrolling in the software testing course, students may have taken additional courses in software design and architecture, project management, software engineering, or requirement engineering at the graduate level. Students in graduate classes functioned as the software quality assurance team, while those in undergraduate classes worked as the development team, all of them working on the same project. The authors [6] have addressed a practical drawback of software metrics based quality classification models based on Boolean Discriminant Functions. More precisely, BDFs have proven to be quite good at predicting modules that are prone to faults, but they do so at a very high inspection cost. It should be mentioned, nonetheless, that in some circumstances software development companies might be willing to bear somewhat high inspection charges as long as all subpar modules are examined and improved. This paper applies [7] Lehman's theory of software evolution to analyze the characteristics of web-based applications and identifies the essences and incidents that cause difficulties in developing high quality web-based applications. Lehman's eight rules of software evolution are said to be satisfied by them since they are considered to be part of his E-type systems. The questions that guide the creation of web apps are examined, and their consequences are talked about. The authors suggested using a cooperative multi-agent system strategy to support both development and maintenance tasks in order to enable the long-term, sustainable evolution of such systems. A reported prototype system prioritizes testing and quality assurance. This paper describes [8] the one of the most important things that students can learn in a course in software engineering is how to effectively work in a team to develop software that is too large for a single individual to produce. Students must also be taught the importance of ensuring software quality throughout the whole development process. This study also demonstrates how an object-oriented software engineering course might incorporate a UML-based team project. The project gives students actual experience in software development and quality assurance at each stage of the software lifecycle, including analysis, design, implementation, and integration. The author of this document outlines a methodology and an example project that contains the problem specifications, a deliverables schedule, and sample deliverables. This paper has [9] shown that the software quality has been advanced, and Software Process Improvement (SPI) has been also improved since the execution of system based SQA.

Like many other businesses, Samsung Electronics Semiconductor Business (SESB) is working to achieve optimal process and higher software quality. In order to improve software development productivity and streamline the process, SESB built IT-Workplace and ITPM. The SQA department has carried out initiatives to improve software quality, such as audit and software process improvement, based on IT-Workplace and IT Product Management. The Authors observed [9] that software quality has been increased and stabilized by appraising audit score and total time of software development process in the software project has taken less time than before. To ensure high software quality, the SQA department must continuously look for areas for improvement, consult with the project team, consider their feedback, and make process improvements. The authors conclude, in summary, that the adoption of system-based SQA has a significant impact on the advancement of SQA and SPI. This paper describes [10] quality assurance (QA) methods such as software testing and peer review are very important to reduce the adverse effects of defects in software engineering. The authors of this paper examine current quality assurance (QA) procedures and potential applications for them in open source software (OSS) initiatives. The authors identified major challenges for future work [10]: a) how to better formulate such indicators as the basis of meaningful notifications about the status of OSS product quality for different stakeholders, b) how much effort seems reasonable to spend on creating, maintaining and monitoring the indicators in an OSS context; and c) the need for empirical evaluation of the concept using larger set of OSS projects.

3. Hypothesis

Our paper's hypothesis is that we offer solutions for major issues like testing shortcuts, testing time reductions, a "let's deliver now, correct errors later" mentality, poor planning and coordination, low user involvement, subpar documentation, a lack of management support, insufficient application environment knowledge, inappropriate staffing, and subpar testability. To improve the aforementioned factors, we are concentrating on them in this work.

4. Strategy for improvement of key problems

In this paper we give strategy for the improvement of key problems [1] which are being faced in the software quality assurance during testing.

4.1 Shortcuts in testing

Many software project managers and software houses view testing as a challenging responsibility. Software testing is a complex and creative process that calls for knowledgeable, motivated staff members in the software development industry. The actions listed below must be taken in order to prevent testing shortcuts. Acquire the essential documentation, including the functional design, internal design specifications, and requirements. Additionally, we need schedule requirements that outline the roles and duties of project people as well as reporting specifications, necessary standards, and procedures including release and modification processes. The testing team must prioritize, define the scope and limitations of tests, and identify the higher-risk components of the application. Choose test strategies and techniques, such as functional, system, load, unit, integration, and usability testing. It is also necessary to determine the hardware, software, communications, and other requirements for the test environment. Ascertain the necessary testware, including record/playback tools, coverage analyzers, problem/bug tracking, test tracking, etc. It is necessary to ascertain the requirements for the test input data that will be used during testing. Determine which people are in charge of what, when, and how much labor is needed. We have to establish timetables, milestones, and estimations for the testing procedure. A test plan document must be prepared before beginning any testing. Writing test cases is required before beginning any testing. Establish test tracking procedures, logging and archiving procedures, test environment and test ware preparation, obtaining necessary user manuals, reference materials, configuration instructions, and installation guides, and setting up or obtaining test input data. To check for errors, testers need to download and install the program that was created by the developers. The software is tested, and evaluated, and findings are reported after installation. Retest as necessary and keep track of issues/bugs and their remedies. Throughout the test life cycle, keep test environments, test cases, test plans, and test ware up to date.

4.2 Reduction in testing time

In this, we must follow the following steps. In order to give appropriate time to testing, software engineers must follow the schedule. The time required for each phase of development must be followed; in reality, testing is often estimated inadequately. Design and coding generally take more time than estimated or planned therefore proper management must be done in order to avoid from reduction in testing time.

4.3 Let go-deliver now, correct errors laterattitude

Potential areas for development consist of The testing crew's need to participate fully in the process. Every member of the testing team has to pay attention to the testing guidelines as stated by the software house. The testing crew and development team need to coordinate more effectively and prepare ahead. The testing team needs to take feedback and the pursuit of ongoing improvement into consideration.

4.4 Poor planning and co-ordination Planning

Software development must take testing planning into account early on; testing is not given enough time until the very end of the project. Use this checklist at every level of the planning process. Gather key papers, including the documentation plan's previous iteration, the specifications and requirements documents, and the documentation proposal quality plan. Planning takes into account the following elements. It is necessary to carefully arrange for the provision of workers and equipment for the software development process. Assign duties to various documentation-related tasks. The team leader must project the financial expenses associated with software development. Schedule preparation is essential throughout the planning stage. It takes careful planning to determine which prototypes to utilize and when. Reviews of the documentation are also necessary to identify any weaknesses in earlier initiatives. To ensure the success of the project and establish an approval process for the documentation, developers, and customers must work together in perfect harmony. Decide updates and upcoming changes. If required, rewrite the documentation plan.

Coordination:

To prevent any harm to the project, the test crew and design team must work together in perfect harmony. To ensure total customer satisfaction, there must be established customer coordination between the design and test teams.

4.5 Lack of user involvement

The user is crucial to the testing process. In software development, the ideas of group support systems (GSS) and joint application design (JAD) are becoming more widely accepted as ways to incorporate users. They enable dynamic, active communication between developers and users. It is the responsibility of developers to focus users' attention during testing and encourage their participation in acceptability, system, and test planning.

4.6 Poor documentation

During the software development process, these two forms of documentation user and system documentation are crucial. Improvement must be made to the following factors to avoid inadequate documentation. Verify the information that is missing throughout. Uncertainty and poor writing often lead to major issues; this aspect has to be improved. A failure to anticipate the reader’s problem, questions, and environment. Documents are written for the writers and their surroundings not for the readers and their environment documents must be suited for the readers. Concentrate on raising the incorrect technical level. In the documentation, formatting and structural design are crucial. Documentation must be properly indexed as well since, although it contains valuable information, it can be difficult to find. a polished look that conceals subpar content and a lack of adaption to product changes. Poor documentation might also be caused by inadequate planning.

4.7 Lack of management support

The only way to implement excellence principles is through the use of an efficient quality management system. Software products are defined and implemented with quality, time, and budget compliance guaranteed by technical and managerial methods. Software upgrade technologies come in a variety of forms, and one of their main goals is software engineering. Few examples of the important technologies are [1] requirement definitions, defect prevention, defect detection and defect removal.

4.8 Inadequate knowledge of application environment

The testing team needs to be well-informed about the features of the software under test, its intended users, and the platform on which it will run. Without this, the testing team may focus on the wrong areas of the software and overlook others that are more important to the user. Without environmental knowledge, significant user requirements might be overlooked.

4.9 Improper staffing

Testing requires teamwork, and the team as a whole must contribute to its success. Testing success is significantly influenced by selecting the right team member for the job. We require team members with development and testing experience for testing. The ability to manage and solve problems, lead a team, and coordinate with clients are all necessary for the role of team leader.

4.10 Poor testability

Software testing requires planning, work, and patience. Software must be tested using methodologies for software validation and verification to prevent testability issues. Following development, we advise conducting validation tests using black box, white box, unit, integration, system, and acceptance testing. Peer and group evaluations of software between developers and customers at different stages of development are required for verification testing. Verification testing requires formal technical evaluation as part of the software quality assurance process. Developers must create software that possesses the following qualities: simplicity, stability, observability, controllability, decomposability, and understandability.

5. Conclusions

The process of running a software implementation using test data and analyzing the program's output is known as software testing. Techniques, processes, tools, and principles can all be supported while testing software. Effective testing is the management's responsibility. Members of the testing team must concentrate on the customer agreement. In this article, we offer recommendations for strategies to address the main issues with software testing in quality control. The following issues are taken into consideration: testing shortcuts, testing time reductions, a "let's deliver now, correct errors later" mentality, inadequate planning and coordination, low user involvement, inadequate documentation, a lack of management support, insufficient application environment knowledge, inappropriate staffing, and poor testability.

Reference

[1] N. S. Gill, "Factors Affecting Effective Software Quality Management Revisited," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 2, pp. 1-4, March 2005.

URL: https://dl.acm.org/doi/abs/10.1145/1050849.1050862

[2] J. Thompson and H. Edwards, "How to teach practical software quality assurance. An experience report," *Quality Software, 2000. In Proceedings. First Asia-Pacific Conference on 30-31,* pp. 181-187, Oct 2000.
URL: https://ieeexplore.ieee.org/abstract/document/883791

[3] L. G. M Visconti, "A Measurement Based Approach for Implanting SQA & SCM Practices," *In Proceedings of the XX International Conference of the Chilean Computer Science Society (SCCC'00),* pp. 126-134, November 2000 IEEE.
URL: https://ieeexplore.ieee.org/abstract/document/890400

[4] G. A. M. J. Rosenberg L H, "Software Quality Assurance Engineering at NASA," *In Proceeding Aerospace Conference,* vol. 5, pp. 5-2569 - 5-2575, 2002 IEEE.
URL: https://ieeexplore.ieee.org/abstract/document/1035438

[5] M. Towhidnejad, "Incorporating software quality assurance in computer science education: an experiment," *In Proceeding 32nd ASEE/IEEE Frontiers in education conference*, vol. 2, pp. F2G-1 - F2G-4, November 6-9, 2002.
URL: https://ieeexplore.ieee.org/abstract/document/1158172

[6] T. Khoshgoftaar and N. Seliya, "Improving usefulness of software quality classification models based on Boolean discriminant functions," *Software Reliability Engineering, 2002. ISSRE 2002. In Proceedings. 13th International Symposium on 12-15,* pp. 221-230, November 6-9, 2002.
URL: https://ieeexplore.ieee.org/abstract/document/1173256

[7] H. Zhu, "Cooperative agent approach to quality assurance and testing Web software," *Computer Software and Applications Conference, 2004. In Proceedings of the 28th Annual International,* vol. 2, pp. 110-113, 2004.
URL: https://ieeexplore.ieee.org/abstract/document/1342688

[8] P. Doerschuk, "Incorporating team software development and quality assurance in software engineering education," *Frontiers in Education, 34th Annual,* vol. 2, pp. F1C-7-12, October 20-23, 2004.
URL: https://ieeexplore.ieee.org/abstract/document/1408560

[9] J. Lee, S. Jung, S. Park, L. Y.J. and Y. Jang, "System based SQA and implementation of SPI for successful projects," *Information Reuse and Integration, Conf, 2005 IEEE International Conference on. 15-17,* pp. 494-499, August 2005.
URL: https://ieeexplore.ieee.org/abstract/document/1506522

[10] D. Wahyudin, A. Schatten, D. Winkler and S. Biffl, "Aspects of Software Quality Assurance in Open Source Software Projects: Two Case Studies from Apache Project," *Software Engineering and Advanced Applications, 2007. 33rd EUROMICRO Conference on 28-31*, pp. 229-236, August 2007.
URL: https://ieeexplore.ieee.org/abstract/document/4301084