International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Demystifying the Obscurities of Debugging in Software Quality Assurance

J. Angala Parameswari[1], Dr. M. Sulthan Ibrahim[2]
[1]MPhil Scholar (Full Time), Department of Computer Science, Government Arts College (Autonomous), Karur, Tamilnadu, India-639005
[2]Assistant Professor and Head, P.G. and Research, Department of Computer Science, Government Arts College (Autonomous), Karur, Tamilnadu, India-639005

***Abstract*—** *Software Quality Assurance (SQA) is the basic foundation of the software development, which includes design of software, hardcoding, review of code, configuration management and release management. This paper deals with the probable solutions for the vital problems present in software testing to ensure utmost quality assurance for the clients. Presently there are plethora of problems related to testing practices, approach of users and organization principles. Apart from these difficulties, there are many snags in dealing with shortcuts in testing, reduction in testing time, deprived documentation. This paper recommends essential strategies to provide solution for the aforesaid glitches. The foremost objective of this paper is to provide suggestions to improve the software quality.*

***Keywords*—** *Software quality, quality assurance, software testing, planning.*

I. INTRODUCTION

Developing a bug free software system is a tedious and cumbersome task as it deals with lot of computations, vigorous testing and customized implementation. To createa perfect software product with flawless nature, ample metrics for software quality attributes need to be considered [1].The size and the complication of the software system play an imperative partin controlling the software quality since it usually affects the major attributes of software quality like reliability, testability and maintainability. This is the foremost reason the software quality assurance (SQA) must be carefully addressed keeping by adopting new strategies in testing, new tools to evolve, new methodologies and novel techniques applicable to software development life cycle. Now a day's software quality assurance is gaining much attentionand more significance is being given to the development of high quality software products. Software development is a complex task entailingcautious assimilation of manydepartments, technical activities, many teams and project management. Usually the software are designed and developed by the combined effort of many designers and programmers involving their expertisefor years together. But the pathetic thing is the developed software product is a mystery to many and they are not easily understood even the methodologies and techniques adopted in development is advanced and superior.

II. SOFTWARE QUALITY ASSURANCE

Software quality assurance (SQA) is a systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines [3]. The main objective of quality assurance is to minimize the cost of guaranteeing quality by a variety of activities performed throughout the development and manufacturing processes/stages. These activities prevent the causes of errors, and detect and correct them early in the development process. As a result, quality assurance activities substantially reduce the rate of products that do not qualify for shipment and, at the same time, reduce the costs of guaranteeing quality in most cases [3]. SQA governs the procedure to build the preferred quality into the products. Quality assurance (QA) techniques can be categorized into two ways, one is static and another is dynamic**.** Nothing like dynamic techniques, static techniques do not engage the execution of code. Static techniques engage examination of documentation by individuals or groups. This inspection is assisted by software tools, e.g., examination of the requirements specification and technical reviews of the code. Testing and simulation are dynamic techniques. Sometimes static techniques are used to support dynamic techniques and vice versa.

TABLE I. Software quality factors.

Quality of Design	**Description**
Correctness	Extent to which software conforms to its specifications and its objectives
Maintainability	Ease of effort for locating and fixing a software failure within a specified time period
Verifiability	Ease of effort to verify software features and performance based on its stated objectives
Quality of Performance	**Description**
Efficiency	Extent to which the software is able to do more with less system (hardware, operating system, communications, etc.) resources
Integrity	Extent to which the software is able to withstand intrusion by unauthorized users or software within a specified time period
Reliability	Extent to which the software will perform (according to its stated objectives) within a specified time period
Testability	Ease of testing to program to verify that it performs a specified Function
Quality of Adaptation	**Description**
Expandability	Relative effort required to expand soft ware capabilities and/or performance by enhancing current functions or by adding new functionality
Reusability	Ease of effort to use the software (or its components) in another software systems and applications

J. Angala Parameswari and Dr. M. Sulthan Ibrahim, "Demystifying the obscurities of debugging in software quality assurance," *International Research Journal of Advanced Engineering and Science*, Volume 2, Issue 3, pp. 31-34, 2017.

Interoperability	Relative effort needed to couple the software on one platform to another software and/or another platform
Portability	Ease of effort to transport software to another environment and/or platform.
Intra-operability	Effort required for communications between components in the same software system

III. Related Works

It is a known fact that doubtfully tested software system affects the system reliability and the quality of the software to a large extent. The author [1] introduced 'Software Reliability Measurement' and ISO approach applicable to software quality assurance (SQA) to improve the quality of the deliverables. To increase the testing effectiveness and efficiency software companies must make elevate them to higher software culture. Instead of detecting and debugging, testing need to concentrate more on augmenting customer satisfaction. The author discussed and suggested many new techniques and improvement related to the factors affecting the software quality.

The authors [2] describes that software quality assurance is facing many hitches regarding the method of describing quality for software. There must be a broad awareness regarding what high quality software is, but the decisive portrayal is usually persuaded by the milieu of the software usage. SQA is a very complicated vertical that is essential to the ultimate success of a project and also SQA is the one that requires diverse set of skill and expertise in software safety and reliability are now being added to the primary set of required skills. SQA must function independently from development organizations to be successful. The authors [4] have focusedon a practical shortcoming of software metrics based on Boolean Discriminant Functions. The BDFs have great adeptness to predict and discover bug prone modules but not at an affordable inspection cost. In future a situation might arise where almost all software development firms agree to deal with relatively high inspection costs and eradicating the low quality modules completely.

The authors [5] applies Lehman's theory of software evolution to analyze the characteristics of web-based applications and detects the criteria that cause complications in developing high quality web-based applications. The skeptics underlying the development of web applications are analyzed and their implications are discussed. In order to support viable long term evolution of such systems, authors proposed a cooperative multi-agent system approach to support both development and maintenance activities.

The Authors [6] perceived that the software quality is increased and stabilized by appraising audit score and total time of software development process in the software project is reduced considerably. The authors suggested that to maintain good software quality, SQA department should perpetually search for the improvement point, get the opinion from project team, and reflect on that view and improve the process.

IV. Motivation

Numerous facets are responsible for the lack of a good SQA and the factors that hinders the quality of software developments are poor planning, improper co-ordination, poor documentation, lack of users involvement and interest, improper hiring of staff, and poor testability. The main reason to focus on this paper is to suggest and plan a methodology to overcome the aforesaid snags and bring improvement in them.

V. Approaches to Improve of the Snags

There are many vital complexities and problems being faced in software quality assurance during the testing phase and the methods to overcome them are illustrated here.

5.1 Shortcuts in Testing

Software testing is considered as a hard but creative and complicated task requiring expert and energetic staffsby many software project managers and software companies.

Acquire functional design, and internal design specifications and other necessary documents before starting the testing to understand the system completely.

Acquire schedule requirements that determine project related employees and their responsibilities, reporting requirements, required standards and processes such as release processes, change processes, etc.

Identify higher-risk tasks, set priorities, and determine scope and limitations of tests.

Determine test approaches and methods - unit, integration, functional, system, load, usability tests along with the test environment requirements such as hardware and software.

Determine the Test input data that are to be used during testing.

Identify individual tasks, employees responsible for the tasks, and the total manpower requirements.

In testing process we must set schedule estimates, timelines, and milestones.

Preparation of test plan document and Test cases are to be written before starting testing.

Prepare user manuals/reference documents, configuration manuals, and installation manuals.

5.2 Reduction in Testing Time

The software engineers who do the coding must follow the agenda to provide necessary time to the testing department employees. Each and every phase in the software development should be completed within the deadline and in reality testing is often estimated inadequately. Design and coding the software usually take more time than estimated or planned therefore proper management must be done in order to avoid the reduction in testing time.

5.3 Let Go-Deliver Now, Correct Errors Later- Attitude

This is the most important negative attitude in quality assurance and this attitude should be avoided and the software should be completely tested, debugged and executed numerous times to avoid errors after implementation on the client side. Each member of testing team must focus on rules of testing as defined by the software companies. There must be better planning and effective coordination among the testing team and development team.

J. Angala Parameswari and Dr. M. Sulthan Ibrahim, "Demystifying the obscurities of debugging in software quality assurance," *International Research Journal of Advanced Engineering and Science*, Volume 2, Issue 3, pp. 31-34, 2017.

5.4 Lack of User Involvement and Interest

The concepts of joint application design (JAD) and the group support system (GSS) can be used for user involvement and are getting better acceptance in software development. They provide good interaction between users and developers to understand their role and provide better work culture. The developers need to draw the attention of users in testing and support their involvement in test planning, system testing and acceptance testing.

5.5 Poor planning and co-ordination

A good plan before any task completes half the task. Planning should be properly handled and a checklist should be used at each planning stage.

Collect important documents like previous version of the documentation plan,

Specifications requirements documents SRS, documentation proposal quality plan.

Planning is done to the personnel and equipment to be used during the software development.

Assign responsibilities to Team leader to estimate the financial costs during software development.

Preparation of schedules is very necessary at planning phase. Complete planning is required to know which prototypes are to be used when.

Documentation reviews are also needed to check the weakness in the previous projects. There must be complete coordination between developers and customer to make project successful and approval mechanism for the documentation.

5.6 Poor Documentation

Two types of documentation are used during the software development stages, user documentation and system documentation. Much improvement must be carried out to avoid poor documentation.

Check the missing information throughout.

Poor writing and vagueness often create huge problems and improvement is required to this factor.

Formatting and design of structure plays very important role in documentation.

Proper indexing of documentation is necessary

5.7 Lack of Management Support

Great success for companies can only be achieved by employing effective quality management structure. The management and technical procedures imbue quality into software product that are defined and implemented provide highest quality software.

5.8 Insufficient Knowledge of Application Environment

Testing team should possess completer knowledge of the functionalities of the software being tested, its users and the platform in which it is going to work and without this knowledge the testing team will miss many imperative points and produce inferior software with bugs.

5.9 Improper Hiring of Staffs

Software testing is a team work and everyone in the team should strive to achieve the highest standard in testing. Appointment of correct team member for testing has great control on the achievement of testing. In testing we need team members who possess experience in development and in testing. The team leader should have the qualities of problem solving and management skills and capacity to supervise a team and interact with clients.

5.10 Poor Testability

Software validation and verification techniques must be employed to test software in order to avoid from testability. After development it is recommended to conduct black box testing, white box testing, unit testing, integration testing, system testing and acceptance testing.

VI. IMPORTANT POINT FOR THE TESTERS

Most experienced professionals will understand that software testing is not an odd approach, in a broader sense it refers to a collection of tests and evaluations that points to determine whether the software application works as it should be and if it can continue working as it should in real world scenarios. The following are the important points that should be noted,

- Nurture a testing culture – software testing is not the last phase in development process as it should continuously work to provide a quality software.
- Inspire clarity in bug reporting – A decent bug report can save time by evading miscommunication or additional communication but a bad bug report can lead to a quick sacking by a developer.
- Handle testing like a team effort – when the testers are exposed to a greater amount in the project, they will feel much more comfy and relaxed in what their goals should be.
- Use effective tools to make testing easier –Find an appropriate tool to ensure perfect testing environment.
- Keep open line communication between testing teams – Opening up communication lines between the testing team can do wonders for making the testing smooth.
- Automation is good but it does not fix poor test design – Testing should ascertain the high risk areas or specific areas where test automation would add the most values rather than leaving such decision would affect the overall performance in the latter stages.
- Think outside the box – Testers sometimes have to deals with the most unthinkable scenarios and they have to think out of the box to cope up with the quality.
- Develop “Rule of Thumb” and document them to achieve the foremost goals of testing.
- Conduct code reviews often.
- Engage the end user – The most important person in the entire process is the end user, but many times we are tempted to keep them away from the scheme of things. Engage them.

J. Angala Parameswari and Dr. M. Sulthan Ibrahim, “Demystifying the obscurities of debugging in software quality assurance,” *International Research Journal of Advanced Engineering and Science*, Volume 2, Issue 3, pp. 31-34, 2017.

- Bug summaries should be thorough – Most customers, managers and the developer will read the summaries first when they review a bug report.
- Use flexible test tools that can adapt your needs.

VII. CONCLUSION

This paper has provided a clear insight about the necessary steps to be taken during testing to maintain the quality in software. The management, the testing team and the developer should consider the points enumerated in this paper and the do's and don'ts presented for every factors which influences and hinders the effective software testing.

REFERENCES

[1] N. S. Gill, "Factors affecting effective software quality management revisited," *ACM SIGSOFT Software Engineering Notes*, vol. 30, issue 2, pp. 1-4, 2005.

[2] L. H. Rosenberg and A. M. Gallo, "Software quality assurance engineering at NASA," in IEEE *Proceeding Aerospace Conference*, vol. 5, pp. 5-2569–5-2575, 2002.

[3] D. Galin, *Software Quality Assurance*, First published 2004.

[4] T. M. Khoshgoftaar and N. Seliya, "Improving usefulness of software quality classification models based on Boolean discriminant functions," *In Proceedings of the 13th International Symposium on Software Reliability Engineering*, pp. 221-230, 2002.

[5] H. Zhu, "Cooperative agent approach to quality assurance and testing Web software," *In Proceedings of the 28th Annual International Computer Software and Applications Conference*, vol. 2, pp. 110–113, 2004.

[6] J. W. Lee, S. H. Jung, S. C. Park, Y. J. Lee, and Y. C. Jang, "System based SQA and implementation of SPI for successful projects," *IEEE International Conference on Information Reuse and Integration, Conf.*, pp. 494–499, 2005.