A Study on Various Testing Techniques Used In Software Organizations

Mohammed Rafiq

Research Scholar (Computer Science), CMJ University, Shillong, Meghalaya

Abstract— Software testing is a atsk which is targeted for estimating capability or an attribute of a program and assures that it attains the needed results. There are numerous approaches to software testing but efficient complex product testing is significantly an examination process not merely a matter of making and following route process. It is always not possible to predict all the mistakes in the prpgram. This mistakes can be resolved using various techniques of software testing. The benefits, challenges and best practices are also dicussed to develop software testing in future. Thus the selection of appropriate strategy at appropriate time will make software testing effective and efficient.

Index Terms— Software testing, neeed for software testing, techniques of software testing,. benefits of software testing, challenges of software testing, best practices of software testing

---♦-------------------------------------

1. INTRODUCTION TO SOFTWARE TESTING

According to Sommerville (2000) testing is a procedure used to recognize the quality, correctness and completeness of developed software of a computer. Testing can never set up the computer software correctness completely. Software testing is a group of tasks organized with the purpose of finding mistakes in software. It also validates and verifies whether the program is operating appropriately with no errors or not. It identifies the software for predicting errors. According to specification software testing is not only used for predicting and fixing errors but it also assures that the system is operating. Software testing is a group of process which is configured to make sure that the computer code does what it was configured to perform. Software testing is a destructive method of attempting to predict the mistakes. The major intention of testing can be assurance, quality, validation, reliability, verification or estimation. Bertolina (2003) has mentioned some of the features of software testing which involves:

- More the testing can be optimized and automated better the software can be controlled.
- The better it performs more effectively it can be checked.
- Fewer the modifications fewer the disruption to testing.
- Testing is a method to recognize the completeness and correctness of the software.
- A successful test is the one that discloses an undetected mistake.
- The software testing's general objective is to declare the software system quality by exercising the software systematically in carefully managed situations.

Whittaker (2000) has described that software testing is always used in connection with the terms validation and verification. Validation is the checking process that what has been specified is what the user needed actually. Verification is the testing or checking of items including consistency with a linked specification. Software testing is one type of verification which uses techniques such as analysis, reviews, walkthroughs and inspections. For instance:

Verification: Are we performing the work properly?

Validation: Are we performing the proper work?

According to Barr (2004) the term bug is always used to define to a fault or an issue in a computer. There are hardware bugs and software bugs. With debugging software testing must not be confused. Debugging is the method of locating and analyzing bugs when software does not act as regarded. Although some bug identification will be evident from playing with software a methodological process of software testing is a much more thorough means of recognizing bugs. Therefore debugging is a task which aids testing but cannot change testing. However no value of testing can be assured to discover all bugs. Other tasks which are always linked with software testing are dynamic analysis and static analysis. Dynamic analysis views at software behavior while it is performing to offer information such as timing profiles, test coverage information and execution traces. Static analysis examines the software's source code, viewing for issues and collecting metrics without actually performing the code.

2. WHY IS SOFTWARE TESTING NEEDED

A mistake is an action done by human beings that generates improper outcomes. A mistake is an error manifestation in software also referred as a bug or a defect. A mistake is encountered which may induce a fault which is a software derivation from its regarded service or delivery. Reliability is the probability that software will not induce the systems failure for a particular time under specified situations. Mistakes exist because human beings are not perfect and even if they are perfect they are operating under restrictions such as deadlines of delivery. Testing recognizes the mistakes whose removal develops the quality of software by enhancing the essential reliability of software. Testing is the software quality measurement. A human being can estimate how closely they have gained quality by testing similar factors such as reliability, correctness, reusability, usability, testability, maintainability, etc (Binder, 2000).

Burnstein (2004) has mentioned other factors that may decide the testing executed may be legal needs or contractual needs defined usually based on agreed best practice or industry specific standards.. Though it is very critical to decide how much testing is enough because sometimes an individual failure can cost a lot or nothing. In safety critical systems software can induce injury or death if it fails so the failure cost in such a system may be in the lives of human being. The amount of testing executed relies on the risks involved. Risk must be used as a foundation for allotting test time that is possible and for choosing where to fix emphasis and what to test. Software testing is essential as it may induce failure in mission, influence on reliability and operational performance if not performed appropriately. Efficient software testing delivers quality products of software fulfilling user's needs, expectations and requirements. Thus software testing is very much essential for any development organization to be assured of the quality that will be provided to customers.

3. SOFTWARE TESTING TECHNIQUES

According to Loveland et al (2004) software testing is a procedure which is used to estimate the developed software quality. It is also a method of disclosing mistakes in a program and makes it a reliable task. It is a useful procedure of occurring program with the purpose of predicting mistakes. The below figure indicates some of the most predominant software testing techniques which are categorized by purpose:

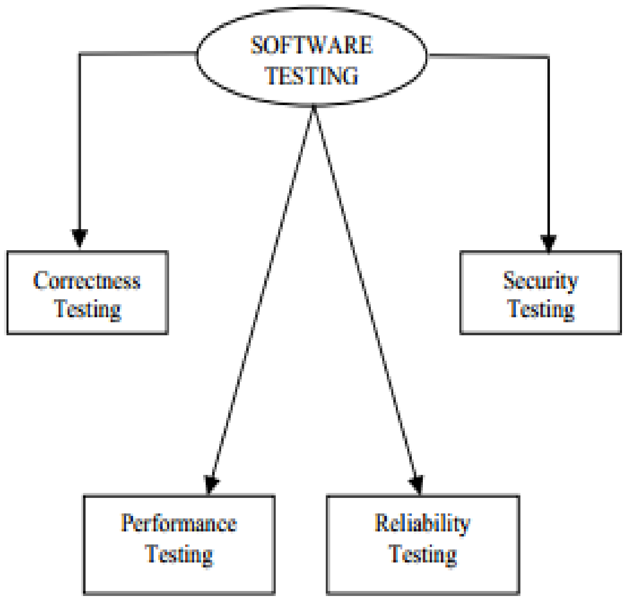

Figure 1: Software Testing Techniques categorized by purpose

Source: Whittaker J A (2000), "What is Software Testing? And Why Is It So Hard?" IEEE Software, pp. 70-79
Correctness Testing:

Correctness is the most significant testing purpose which is the minimum need of software.Correctness testing explains the proper system behavior from the incorrect one for which it will require some kind of Oracle. Either a black box view point or a white box view point can be included in software testing as a tester may or may not know the inside descriptions of the software module under test. For instance, Control flow, data flow, etc. The notion of black box, gray box or white box testing is not restricted to correctness testing (Kaner, Bach and Pettichord, 2002). The below figure shows the different forms of correctness testing:

]

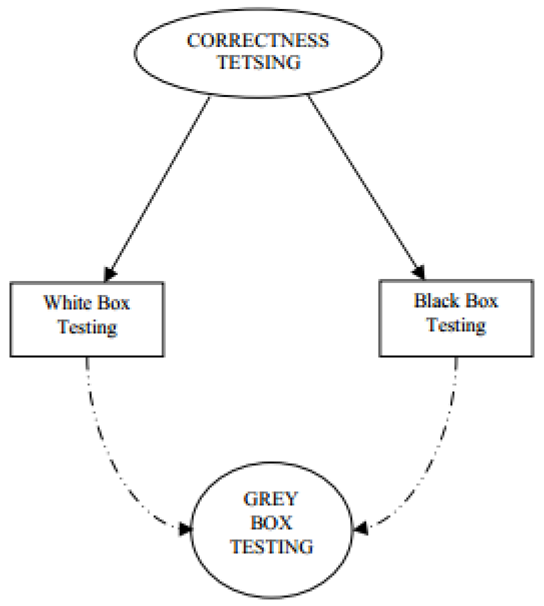

Figure 2: Different forms of correctness testing

Source: Burnstein I (2003), Practical Software Testing, Springer, New York

The different forms of correctness testing are described below:

Black Box Testing:

Black (2002) has mentioned that black Box testing is a major form of correctness testing but its notions are not restricted to only correctness testing. Correctness testing is a process which is categorized by software testing purpose. Black box testing is based on the specification analysis of a software piece without reference to its internal operation.

White Box Testing:

According to Copeland (2004) White box testing is based on internal working analysis and structure of a software piece. White box testing is the method of giving input to the system and verifying how the processes of system produces the needed result. It is important for a tester to have the complete source code knowledge. White box testing is possible at unit, system and at integration levels of software testing process.

Gray Box Testing:

The techniques of gray box testing integrate the testing methodology of black box and white box. The technique of gray box testing is used for testing a software piece against its specifications but using some internal working knowledge as well.

Performance Testing:

Rakitin (2001) has mentioned that performance testing includes all phases as the mainstream life cycle of testing as an independent discipline which includes strategy such as design, plan, analysis, reporting and execution. This testing is organized to estimate the system compliance or tool with the particular needs of performance. Performance evaluation of any software system involves throughput, stimulus response time and resource usage. The below figure shows the types of performance testing:

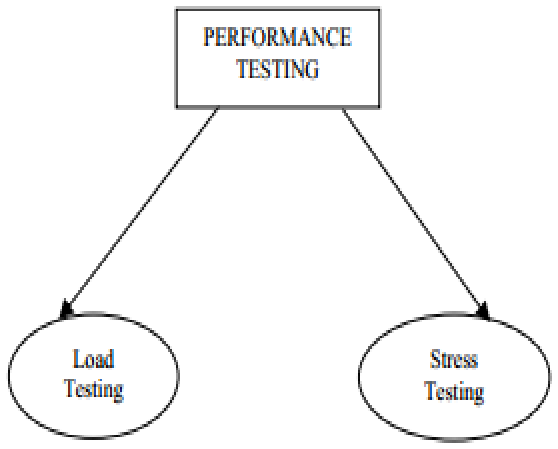

Figure 3: Two types of performance testing

Source: Jorgensen P C (2002), Software Testing: A Craftsman's Approach, 2nd Edition, CRC, USA

The two types of performance testing are described below:

Load Testing:

Load testing is an industry phrase for the attempt of performance testing. The load testing is used for verifying an application against heavy inputs or load such as website testing to predict at what point the application or website fails or at what point its performance degrades (Craig and Jaskiel, 2002).

Stress Testing:

According to Schulmeyer and Mackenzie (2000) stress testing is referred as an operating random sequence of operations at bigger than normal volume at quicker than usual speed and for bigger than the usual time period as a process to develop the rate of predicting defects and check the product's robustness. Stress testing decides the system behavior as the base of the user develops.

Reliability Testing:

Reliability testing is very essential as it discover all system failures and eliminates them before the system is deployed. reliability testing is similar to numerous perspectives of software in which the process of testing is involved. This process of testing is an efficient method of sampling to estimate the reliability of software. In reliability testing estimation model is prepared which is used to identify the data to predict and evaluate future software reliability. Depending on that evaluation the developers can determine whether to release the software or not and the end user will determine whether to acquire that software or not. The risk of using the software can be assessed based on reliable information. The two forms of reliability testing are stress testing and robustness testing (Dasso and Funes, 2006).

Security Testing:

Jorgensen (2002) has mentioned that security testing assures that only standardized personnel can access program and only standardized personnel can access available functions to their security level. Any developed system security testing is about predicting major weaknesses and loopholes of a system which can affect major damage to the system by a standardized user. Security testing is very useful for testerto to predict and fix software issues. It assures that the system will perform for a long time without any major issue. It also assures that systems used by organizations are protected from any unauthorized attack. In this way security testing is advantageous for organizations in all perspectives.

4. BENEFITS OF SOFTWARE TESTING

Haddox et al (2004) has mentioned that software testing is consigned to one phase of the software development Lifecycle. Some of the benefits of software testing are described below:

1. As manual testing uses a big deal of time in both the software development process as well as during the testing of software application, automated components are a quicker choice as big as the scripts which required to be performed are non complex and standard.

2. Lewis (2000) has described that test script execution automation removes human error possibility when a common sequence of actions is repeated again and again. This can be normally essential as users would be surprised to learn how several defects of test development are in fact affected by tester error, This specifically exists when similar boring test scripts have to be performed repeatedly as well as when at opposite spectrum actually complex testing has to be performed.

3. Software testing tools can be speed up the process of testing drastically. Automated tools of software are able to perform 100 or even 1000 times quicker.

4. Software testing tools can eliminate human factors such as boredom or carelessness. The test component will operate similar tests and verify the outcomes perfectly every time it is operated.

5. Software testing can aid code testing in a live surroundings. Test components are always used to replace software or hardware which the user plan to use their product on. This application can help to answer to software problems that might be critical to gain in a controlled test surroundings (Hutcheson, 2003).

5. CHALLENGES OF SOFTWARE TESTING

According to Mosley and Posey (2002) all areas of software engineering face numerous challenges during execution. So a tester would ever get astonished when the user faces challenges in software testing. The first challenge faced by software testing or lack of testing culture in software organizations. Management needs to play an essential role to build a testing culture. Second main challenge in software testing is the lack of skilled testers. Again the main cause of this challenge is incorrect decisions of management at the time of testers selection. As usual management does not need to spend in testers. This outcomes into inadequate, adhoc and incomplete testing throughout the Lifecycle of the project. It is also real that sometimes testers may add complexities in project testing due to their unqualified way of operation. Myers et al (2004) has mentioned some other challenges that are listed below:

1. Complete testing is not possible at all and testing is often conducted on the basis of sampling. Test data selection requires better proficiency in the selection processes of test data.

2. Developers did not interact what the construct is about and if developers did not interact about what the construct is about then them before testing they must ask.

3. Regression testing becomes a challenge as the project keeps on developing. For these types of circumstances both the development and testing teams required to operate together to perform an appropriate influence analysis.

4. Sometimes developers and testers collide with each other on mistakes and take things personally rather than professionally. It becomes a main challenge for stakeholders of the project (Perry, 2000).

5. If requirements are not communicated properly to testers and developers then they must ask for brief needs.

6. According to Pinkster et al (2006) in several software organizations in the effort evaluation time testing team is not included. Testing team is often asked to verify the construct in a specific area. If the tester or developer is facing this challenge then they must increase their voice to similar stakeholders.

7. Lastly organizations must often expect greater gains on automation investment.

Thus the above described challenges are resolved and software testing becomes the future for all industries.

6. BEST PRACTICES FOR SOFTWARE TESTING

According to Drabick (2003) when applications of software are designed there are best practices that must be used to verify the software. There are numerous best practices to assure risk management and compliance. Some of the best practices for software testing are;

1. Understanding the purpose or scope of the project will be useful to judge the level or extent of testing needed.

2. Before writing test cases testers must go through the needs briefly without missing any points given by the client.

3. Once the client gets new needs or changes the needs the test cases must be updated immediately.

4. Each test case explanation must be written clearly after perceiving the module or the context of the explanation. After executing them manual steps must be written. Expected outcomes must not have any ambiguity. Prerequisite situations must be described if needed (Aggrawal, Singh and Kaur, 2004).

5. Planning and creating test plan document are important for all software projects/

6. Planning of test/development/staging surroundings must be performed clearly.

7. Based on the test cases test execution must be performed carefully and it is very essential to use proper test data.

8. Whittaker (2002) has mentioned that the bug report must be prepared clearly with all important details particularly with the test data or steps for regenerating the bug. The report of bug must support the developers to regenerate the bug and to fix it.

9. Performing little regression test and re-test is important whenever a reported mistake is fixed.

10. It is not better if the tester performs all the testing manually as manual testing will take much effort/time and it is critical to handle and also it is repeatable or consistent. So it is better to develop test cases using the tools of the test such as quick test professional (Pressman, 2000).

Thus the above described software testing best practices are very efficient when software companies prefer to develop and design their own applications of software.

7. CONCLUSION

Software testing is an essential technique for the measurement and improvement of a quality software system. But it is really not feasible to predict all the mistakes in the program. For resolving these mistakes some of the most predominant and similarly used software testing strategies are described such as correctness testing, performance testing, reliability testing and security testing. The successful use of these techniques in software organizations will verify the outcomes of the research and direct future research.

8. REFERENCES

1. Sommerville I, (2000): Software Engineering, Addison-Wesley, 6th edition, August 2000.

2. Bertolina A (2003), ‖Software Testing Research and Practice‖, Proceedings of the abstract state machines 10th international conference on Advances in theory and practice, p 1-21.

3. Whittaker J A (2000), "What is Software Testing? And Why Is It So Hard?" IEEE Software, pp. 70-79.

4. Barr A (2004), Find the Bug: A Book of Incorrect Programs, Addison-Wesley Professional, USA.

5. Binder R (2000), Testing Object-Oriented Systems: Models Patterns and Tools, Addison Wesley, USA.

6. Burnstein I (2003), Practical Software Testing, Springer, New York.

7. Loveland S, Miller G, Prewitt R and Shannon M (2004), Software Testing Techniques: Finding the Defects that Matter, Charles River Media, USA.

8. Kaner C, Bach J and Pettichord B (2002), Lessons Learned in Software Testing, John Wiley & Sons, New York.

9. Black R (2002), Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing, 2nd edition, Wiley, New York.

10. Copeland L (2004), A Practitioner's Guide to Software Test Design, Artech House Publishers, Boston.

11. Rakitin S R (2001), Software Verification and Validation for Practitioners and Managers, 2nd Edition, Artech House Publishers, New Delhi.

12. Craig R D and Jaskiel S P (2002), Systematic Software Testing, Artech House Publishing, Norwood, MA.

13. Schulmeyer G G and Mackenzie G R (2000), Verification and Validation of Modern Software-Intensive Systems, Prentice Hall PTR, New York.

14. Dasso A and Funes A (2006), Verification, Validation and Testing in Software Engineering, Idea Group Publishing, UK.

15. Jorgensen P C (2002), Software Testing: A Craftsman's Approach, 2nd Edition, CRC, USA.

16. Haddox J, Kapfhammer G M, and Michael C C, (2002): An approach for understanding and testing third-party software components, In 48th Reliability and Maintainability Symposium, January 2002.

17. Lewis W E (2000), "Software Testing and Continuous Qualify Improvement" CRC Press LLC, USA.

18. Hutcheson M L (2003), Software Testing Fundamentals: Methods and Metrics, Wiley Publishing, Inc., Indianapolis, IN, New York.

19. Mosley D J and Posey B A (2002), Just Enough Software Test Automation, Prentice Hall PTR, Upper Saddle River, New Jersey.

20. Myers G J, Sandler C, Badgett T and Thomas T M (2004), The Art of Software Testing Second Edition, John Wiley & Sons, New Jersey.

21. Perry W E (2000), Effective Methods for Software Testing, 2nd Edition, John Wiley & Sons, Inc., New York.

22. Pinkster I, Van de Burgt B, Janssen D and Van Veenendaal E (2006), Successful Test Management: An Integral Approach, Springer, Germany.

23. Drabick R D (2003), Best Practices for the Formal Software Testing Process: A Menu of Testing Tasks, Dorset House Publishing Company, New York.

24. Aggrawal, K. K., Singh Y and Kaur A (2004), "Code coverage based technique for prioritizing test cases for regression testing,"SIGSOFT Software Engineering Notes,29(5): 1–4.

25. Whittaker J A (2002), How to Break Software: A Practical Guide to Testing, Addison Wesley, New York.

26. Pressman R (2001), Software Engineering: A Practitioner's Approach, McGraw Hill, Boston.