Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 6, June 2016, pg.218 – 223

Software Testing Principles and Strategies

Rajiv Sharma, Mohit Sangwan
Assistant Professor, Department of CSE, Shri Baba Mastnath Engineering College
M.Tech Student, Department of CSE, Shri Baba Mastnath Engineering College
mohitcse9@gmail.com, sonurajiv_007@rediffmail.com

Abstract- Software testing is an essential activity in software engineering, it is a discipline as well as an iterative process, which consists of Tests Designing, Tests Execution, Problems Identifying and Problem Fixing, for validating functionality and as well as for attempting the software break. Testing refers to the process of evaluating attributes like correctness, completeness, security, consistency, unambiguousness, quality etc. of a software and determine whether it meets its required functionality or not. Software testing doesn't only for fixing the bug (in the code), but also to check whether the program behaves according to its given specifications and requirements. Testing is an activity to find the bugs in software that may perform by tester or by applying strategies like white box or black box. So, the activities involved in the testing should be in planned way.
Keywords--- BVA, UAT, V&V, MCQRTT, HTTP

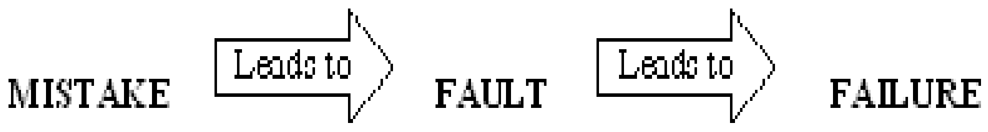
1. INTRODUCTION

Testing simply refers to the validation and verification specially to build good quality software. Testing also defined as ***"Software testing is technique of evaluating the attributes (i.e.correctness, completeness, security, consistency, unambiguousness, quality etc.) of software and determining that whether it meets its required functionality or not".*** There is some basic terms used in software testing that are as follows:-

- **Verification** – Are we building the product right? It refers to the correctness of the function specifications.
- **Validation** – Are we building the right product? It refers to the user expectation whether the product developed meets the user requirement or not.

Determining when to perform verification and validation relates to the development, acquisition, and maintenance of software.

- **Mistake** – something wrong done by the developer and shows different result.
- **Fault [or Defect]** – is a condition that causes a system to fail in performing its required functions.
- **Failure** – the inability of a system or component to perform its required function within the specified performance requirement. A failure is normally detected by a difference between expected and actual result.



(Fig: 1) Failure occurrence

2. PRINCIPLES OF TESTING

- **All tests should be traceable to customer requirements.** As we have seen, the objective of software testing is to uncover errors. It follows that the most severe defects are those that cause the program to fail to meet its requirements.
- **Tests should be planned long before testing begins.** Test planning can begin as soon as the requirements model is complete. Detailed definition of test cases can begin as soon as the design model has been solidified. Therefore, all tests can be planned and designed before any code has been generated.
- **The Pareto principle applies to software testing.** Stated simply, the Pareto principle implies that 80 percent of all errors uncovered during testing will likely be traceable to 20 percent of all program components. The problem, of course, is to isolate these suspect components and to thoroughly test them.
- **Testing should begin "in the small" and progress toward testing "in the large."** The first tests planned and executed generally focus on individual components. As testing progresses, focus shifts in an attempt to find errors in integrated clusters of components and ultimately in the entire system.
- **Exhaustive testing is not possible.** The number of path permutations for even a moderately sized program is exceptionally. For this reason, it is impossible to execute every combination of paths during testing. It is possible, however, to adequately cover program logic and to ensure that all conditions in the component-level design have been exercised.
- **To be most effective, testing should be conducted by an independent third party.** By most effective, we mean testing that has the highest probability of finding errors. The software engineer who created the system is not the best person to conduct all tests for the software.

Key Points:

- **To remove Defects-** The main objective of testing is to remove defects from the software.
- **To improve Quality-** The other main objective is to improve the quality of the software.
- **To increase Consistency-**The extent to which the product is consistent within itself and with other products.
- **To check Necessity-**The extent to which everything in the product is necessary.
- **Sufficiency-**The extent to which the product is complete.
- **Performance-**The extent to which the product satisfies its performance requirements.
- **Correctness-**The extent to which a program satisfies its specification and fulfills the customer's mission objectives.
- **Reliability-**The extent to which a program can be expected to perform its intended function with required precision.
- **Efficiency-** The amount of computing resources and code required by a program to perform its function.
- **Integrity-** Extent to which access to software or data by unauthorized persons can be controlled.
- **Usability-** To improve Effort required learning, operating, preparing input, and interpreting output of a program.
- **Maintainability-** To improve Effort required locating and fixing an error in a program.
- **Flexibility-** To improve Effort required modifying an operational program.
- **Testability-** To improve Effort required testing a program to ensure that it performs its intended function.
- **Portability-** To improve Effort required transferring the program from one hardware and/or software system environment to another.
- **Reusability-**Extent to which software can be reused in other software's, related to the packaging and scope of the functions that the program performs.
- **Interoperability-** To improve Effort required to couple one system to another.

3. TYPES OF TESTING

Static analysis is used to investigate the structural properties of source code. Dynamic test cases are used to investigate the behavior of the source code by executing the program on the test data. During the testing process, only failures are observed, by which the presence of faults is deduced.

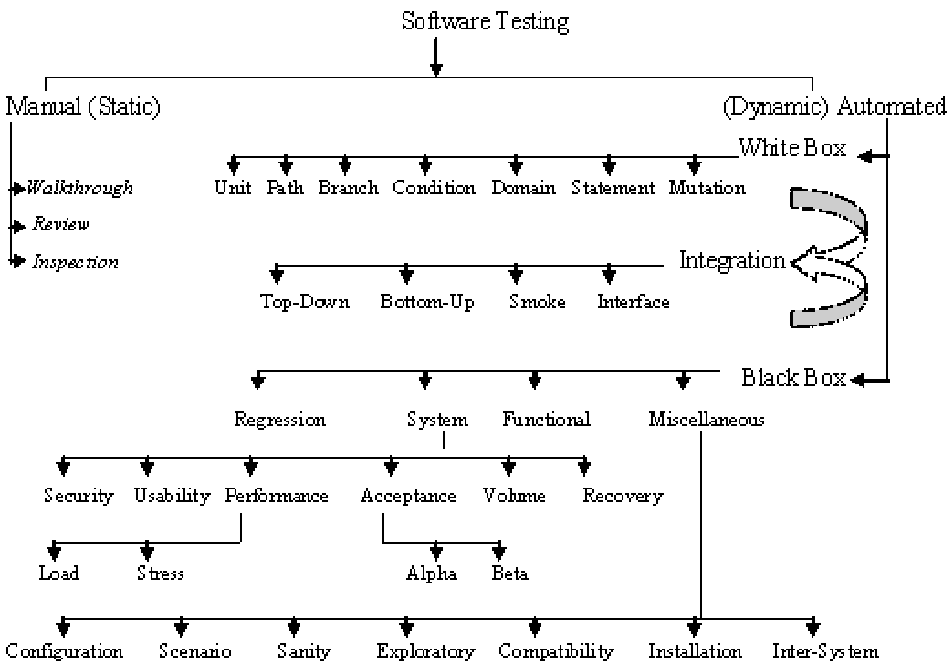


Fig 2. Testing Strategies

A. Manual testing : When the software is in early phase of software development life cycle like in analysis phase Manual Testing is to be done, because at that time everything is on paper nothing is for execution therefore Manual Testing is done at this stage. Manual testing can be done by

- ***Walkthrough:*** Walkthrough team (project manager & few concerned people) adapt informal method to review the code. The tasks to be completed should be based on the scenarios used during the design process, and should be representative of the tasks the interface is designed to support. It is important to exercise as many elements of the interface as possible. If there are any elements of the user interface that you suspect may be troublesome, make sure the tasks use those elements. Tasks should be worded in a simple and straightforward fashion, and should not describe any aspects of screen interactions. Participants should be representative users, although the ability to have many participants provides the opportunity to have other stakeholders participate. Ensure that screens contain appropriate data. For example, if a screen would contain a customer name, make sure that this appears on the relevant screen in the booklet.
- ***Review:*** In this, the product documentation is analyzed in details & the conflicts/error is detected, then suggestion and solution are proposed in review report.
- ***Inspection:*** It is a formal method, minutely checks all the procedures standards and maturity of working environment of the organization and generates an inspection report, on this basis the approval is given.

Advantages:

- Manual testing can be use in both small and big project.
- Easily we reduce and added our test case according to project movement.
- It is covered in limited cost.
- Easy to learn for new people who are entered in manual testing.
- Manual is more reliable then automated (in many cases automated not cover all cases)

- It allows the tester to perform more ad-hoc. In my experiences, more bugs are found via ad-hoc than via automation. And, the more time a tester spends playing with the feature of the software so he gets few user level bugs.

Disadvantages:

- GUI object size difference and color combination etc is not easy to find out in manual testing.
- Actual load and performance is not possible to cover in manual testing for large number of users.
- Running test manually is very time consuming job.

B. Automated testing: Automated testing is done when different modules are ready for execution. Individual/integrated modules are tested thoroughly using different scripts that are run on testing tool. **Automated software testing tool** is able to playback pre-recorded and predefined actions, compare the results to the expected behavior and report the success or failure of these manual tests to a test engineer. Once automated tests are created they can easily be repeated and they can be extended to perform tasks impossible with manual testing. Because of this, savvy managers have found that automated software testing is an essential component of successful development projects.

Advantages:

- **Automated Software Testing Saves Time and Money**

Software tests have to be repeated often during development cycles to ensure quality. Every time source code is modified software tests should be repeated. For each release of the software it may be tested on all supported operating systems and hardware configurations. Manually repeating these tests is costly and time consuming. Once created, automated tests can be run over and over again at no additional cost and they are much faster than manual tests. Automated software testing can reduce the time to run repetitive tests from days to hours. A time savings that translates directly into cost savings.

- **Automated Software Testing Improves Accuracy**

Even the most conscientious tester will make mistakes during monotonous manual testing. Automated tests perform the same steps precisely every time they are executed and never forget to record detailed results.

- **Automated Software Testing Increases Test Coverage**

Automated software testing can increase the depth and scope of tests to help improve software quality. Lengthy tests that are often avoided during manual testing can be run unattended. They can even be run on multiple computers with different configurations.

4. AUTOMATED TESTING TYPES

Automated testing is done by:

A .White box testing: In order to implement white box testing, the tester has to deal with the code and hence is needed to possess knowledge of coding and logic i.e. internal working of the code. Also known as glass, structural, open box or clear box testing. **White box testing helps to derive test cases to ensure:**

- All independent paths are exercised at least once.
- All logical decisions are exercised for both true and false paths.
- All loops are executed at their boundaries and within operational bounds.
- All internal data structures are exercised to ensure validity.

White box testing helps to:

- Traverse complicated loop structures
- Cover common data areas,
- Cover control structures and sub-routines,
- Evaluate different execution paths
- Test the module and integration of many modules
- Discover logical errors, if any.
- Helps to understand the code

Advantages of White box testing:-

- As the knowledge of internal coding structure is prerequisite, it becomes very easy to find out which type of input/data can help in testing the application effectively.
- The other advantage of white box testing is that it helps in optimizing the code
- It helps in removing the extra lines of code, which can bring in hidden defects.

Disadvantages of white box testing :-

- As knowledge of code and internal structure is a prerequisite, a skilled tester is needed to carry out this type of testing, which increases the cost.

- And it is nearly impossible to look into every bit of code to find out hidden errors, which may create problems, resulting in failure of the application.

B. Black box testing: This testing ignores the inspection of internal mechanism of a system and focuses solely on the outputs generated in response to inputs and execution conditions. Tester does not have to deal with the code. It is also called as **Behavioral Testing**.

Advantages of Black Box Testing:-

- Efficient when used on large systems.
- Since the tester and developer are independent of each other, testing is balanced and unprejudiced.
- Tester can be non-technical.
- There is no need for the tester to have detailed functional knowledge of system.
- Tests will be done from an end user's point of view, because the end user should accept the system. (This testing technique is sometimes also called Acceptance testing.)
- Testing helps to identify vagueness and contradictions in functional specifications.
- Test cases can be designed as soon as the functional specifications are complete.

Disadvantages of Black Box Testing:-

- Test cases are challenging to design without having clear functional specifications.
- It is difficult to identify tricky inputs if the test cases are not developed based on specifications.
- It is difficult to identify all possible inputs in limited testing time. As a result, writing test cases may be slow and difficult.
- There are chances of having unidentified paths during the testing process.
- There is a high probability of repeating tests already performed by the programmer.

5. CONCLUSION

Software Testing consumes large amounts of resources in development projects. Hence, it is of general interest to assess the effectiveness and efficiency of current test methods and compare these with new or refinements of existing test methods to find possible ways of improving the testing activity. Software activity like testing is essential for software quality but also consume a large part of software development costs. Therefore efficient and cost effective software testing are both a priority and an essential considering the demands to decrease time-to-market and intense competition faced by many. It is then perhaps not unexpected that decisions related to software quality, when to stop testing, testing schedule and testing resource allocation needs to be as accurate as possible. Although software testing is crucial to build a good quality software. This dissertation elaborates all testing strategies, their interrelations, when to perform and how.

REFERENCES

[1] Robert Nilsson and Jeff Offutt “Automated Testing of Timeliness: A Case Study” Published in IEEE 2007.

[2] Glenford J. Myers ”The Art of Software Testing, Second Edition” published in 2004. Antonia Bertolino” SOFTWARE TESTING” published in *IEEE – Trial (Version 0.95) – May 2001.*

[3] R. Boddu, G. Lan, G. Mukhopadhyay, B. Cukic, “RETNA: from requirements to testing in a natural way,” 12th IEEE International Requirements Engineering Conference, pp.262-271, 2004.

[4] Beck, K. (2003). Test Driven Development -- by Example. Boston, Addison Wesley.

[5] E. van Veendaal (2008), Test Improvement Manifesto, in: *Testing Experience*, Issue 04/08, December 2008.

[6] X. Qu, M. B. Cohen, and G. Rothermel. Configurationaware regression testing: an empirical study of sampling and prioritization. In ISSTA ’08, 2008.

[7] R. Lutz, I.C Mikulski, “Requirements discovery during the testing of safety-critical software,” Software Engineering 25th IEEE International Conference, pp.578-583, 2003.

[8] Wagner, S. & Seifert, T. 2005. Software Quality Economics for Defect Detection Techniques Using Failure Prediction. In Proceedings of the 3rd Workshop on Software Quality (3-WoSQ). ACM Press.

[9] S. Berner, R. Weber, R.K. Keller, "Requirements & testing: Observations and lessons learned from automated testing," Proceedings of 27th international conference on software Engineering, pp.571-579, 2005.

[10] Boehm, B., Huang, L., Jain, A. & Madachy, R. 2004. The ROI of Software Dependability: The iDAVE Model. *IEEE Software*, 21(3).

[11] K. R. Walcott, M. L. Soffa, Gregory M. Kapfhammer and Robert S. Roos. Time-aware test suite prioritization. In *Proceedings of the 2006 International Symposium on Software testing and Analysis*, page 1-12, July 2006.

[12] Peter Sestoft" Systematic software testing" published in IT University of Copenhagen, Denmark Version 2, 2008-02-25.