International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 2, Issue 2, ISSN No. 2455-2143, Pages 113-117
Published Online December-January 2016 in IJEAST (http://www.ijeast.com)

SOFTWARE TESTING TECHNIQUES AND STRATEGIES

Rajendra Kumar

Abstract— **Software Testing is a process of finding errors while executing a program so that we will get a zero defect software. It is aimed at evaluating the capability or usability of a program. Software testing is an important feature of accessing quality of software. Software testing provides a means to reduce errors, decrees maintenance and overall software costs. One of the major problems within software testing area is how to get a suitable set of cases to test a software system. Though a lot of advancements have been done in formal methods and verification techniques, still we need software to be fully tested before it could be handled to the customer side. Thus there are a number of testing techniques and tools made to accomplish the task. Software testing is an important area of research and a lot of development has been made in this field. In this paper, testing techniques and tools have been described. Some typical latest researches have been summarized. Software testing is gaining more and more importance in the future.**

Keywords— **Software testing, Software Testing Goals, Software testing strategies, Software testing principles, Software Testing Methodologies**.

I. INTRODUCTION

Software testing is more than just error detection; Testing software is operating the software under controlled conditions, to verify that it behaves "as specified"; to detect errors, and to validate that what has been specified is what the user actually wanted. Software testing refers to process of evaluating the software with intention to find out error in it. Software testing is a technique aimed at evaluating an attribute or capability of a program or product and determining that it meets its quality. It is also used to test the software for other software quality factors like reliability, usability, integrity, security, capability, efficiency, portability, maintainability, compatibility etc. For many years now we are still using the same testing techniques .some of which is crafted method rather than good engineering methods. Testing can be costly but not testing software can be even more costly. Software testing aims at achieving certain a goals and principles which are to be followed.

II. SOFTWARE TESTING GOALS

Goals are the output of the software process. Software testing has following goals.

Verification and validation: Verification is a review without actually executing the process while validation is checking the product with actual execution. For instance, code review and syntax check is verification while actually running the product and checking the results is validation. Testing can also be used for verifying that the product or the software works as desired and validate whether the software fulfills condition laid down.

Priority Coverage: Testing should be performed in efficient and effective manner within the budget and schedule limits.

Balanced: Testing process must balance the requirements, technical limitation and user expectation.

Traceable: Documents should be prepared of both success and failures of testing process. So no need to test same thing again.

Deterministic: We should know what we are doing, what we are targeting, what will be the possible outcome.

III. SOFTWARE TESTING STRATEGIES

A software testing strategy integrates various software test case design methods into a well planned series of steps that result in successful testing of software. Software testing strategies are thus important for testing. Software testing strategy is generally developed by testing specialist, project managers and software engineer. There are four software testing strategies:

Unit testing: It is done at the lowest level. It tests the basic unit of software, which can be a module or component. Unit is the smallest module i.e. smallest set of lines of code which can be tested. Unit testing is just one of the levels of testing which contribute to make the big picture of testing a whole system. Unit testing is generally considered as a white box test class.

Integration Testing: It is done when two or more tested units are combined into a larger structure. This testing is often done on the interfaces that are between the components and the larger structure that is being constructed, if its quality property cannot be properly assessed from its components.

System Testing: It tends to test the end-to-end quality of the entire system. System test is often based on the functional and requirement specifications of the system. Non-functional quality attributes, such as security, reliability, and maintainability, are also checked.

International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 2, Issue 2, ISSN No. 2455-2143, Pages 113-117
Published Online December-January 2016 in IJEAST (http://www.ijeast.com)

Acceptance Testing : It is done when the complete system is handed over to the customers or users from developer side. The aim of acceptance testing is to give assure that the system is working rather than to find errors.

Unit testing :Unit is the smallest module i.e. smallest collection of lines of code which can be tested. Unit testing is just one of the levels of testing which go together to make the big picture of testing a system. IT complements integration and system level testing. It should also complement code reviews and walkthroughs. Unit testing is generally seen as a white box test class. That is it is biased to looking at and evaluating the code as implemented. Rather than evaluating conformance to some set of requirements.

Benefits of Unit Testing:-

1) Unit level testing is very cost effective.

2) It provides a much greater reliability improvement for resources expanded than system level testing. In particular, it tends to reveal bugs which are otherwise insidious and are often catastrophic like the strange system crashes that occur in the field when something unusual happens.

3) Be able to test parts of a project without waiting for the other parts to be available,

4) Achieve parallelism in testing by being able to test and fix problems simultaneously by many engineers,

5) Be able to detect and remove defects at a much less cost compared to other later stages of testing,

6) Be able to take advantage of a number of formal testing techniques available for unit testing,

7) Simplify debugging by limiting to a small unit the possible code areas in which to search for bugs,

8) Be able to test internal conditions that are not easily reached by external inputs in the larger integrated systems

9) Be able to achieve a high level of structural coverage of the code,

10) Avoid lengthy compile-build-debug cycles when debugging difficult problems.

Unit testing techniques: A number of effective testing techniques are usable in unit testing stage. The testing techniques may be broadly divided into three types:

1. Functional Testing
2. Structural Testing
3. Heuristic or Intuitive Testing

Integration testing: Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. Different Integration testing Strategies are discussed below:-

1) Top down Integration testing

2) Bottom up Integration testing

Top down Integration: Top-down integration testing is an incremental approach to construct program structure. Modules are integrated by moving downward through the structure, beginning with the main control module. Modules subordinate to the main control module are incorporated into the structure in either a depth-first or breadth-first manner. [4] The integration process is performed in a series of five steps:

- The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.
- Depending on the integration approach selected subordinate stubs are replaced one at a time with actual components.
- Tests are conducted as each component is integrated.
- On completion of each set of tests, another stub is replaced with the real component.
- Regression testing may be conducted to ensure that new errors have not been introduced. It is not as relatively simple as it looks. In this logistic problem can arise. Problem arises when testing low level module which requires testing upper level. Stub replace low level module at the beginning of top down testing. So no data can flow in upward direction.

Bottom up Integration: Bottom-up integration testing, as its name implies, begins construction and testing with atomic modules. Because components are integrated from the bottom up, processing required for components subordinate to a given level is always available and the need for stubs is eliminated. [4] A bottom-up integration strategy may be implemented with the following steps:

- Low-level components are combined into clusters that perform a specific software subfunction.
- A driver is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure.

Acceptance testing: Acceptance testing (also known as user acceptance testing) is a type of testing carried out in order to verify if the product is developed as per the standards and specified criteria and meets all the requirements specified by customer. [4] This type of testing is generally carried out by a user/customer where the product is developed externally by another party. Acceptance testing falls under black box testing methodology where the user is not very much interested in internal working/coding of the system, but evaluates the overall functioning of the system and compares it with the requirements specified by them. User acceptance testing is considered to be one of the most important testing by user before the system is finally delivered or handed over to the end user. Acceptance testing is also known as validation testing, final testing, QA testing, factory acceptance testing and application testing etc. And in software engineering, acceptance testing may be carried out at two different levels;

International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 2, Issue 2, ISSN No. 2455-2143, Pages 113-117
Published Online December-January 2016 in IJEAST (http://www.ijeast.com)

one at the system provider level and another at the end user level.

Types of Acceptance Testing User Acceptance Testing: User acceptance testing in software engineering is considered to be an essential step before the system is finally accepted by the end user. In general terms, user acceptance testing is a process of testing the system before it is finally accepted by user.

- **Alpha Testing & Beta Testing** Alpha testing is a type of acceptance testing carried out at developer‟s site by users.[4] In this type of testing, the user goes on testing the system and the outcome is noted and observed by the developer simultaneously. Beta testing is a type of testing done at user‟s site. The users provide their feedback to the developer for the outcome of testing. This type of testing is also known as field testing. Feedback from users is used to improve the system/product before it is released to other users/customers.
- **Operational Acceptance Testing** This type of testing is also known as operational readiness/preparedness testing. It is a process of ensuring all the required components (processes and procedures) of the system are in place in order to allow user/tester to use it.
- **Contact and Regulation Acceptance Testing** In contract and regulation acceptance testing, the system is tested against the specified criteria as mentioned in the contract document and also tested to check if it meets/obeys all the government and local authority regulations and laws and also all the basic standards.

System testing:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that system elements have been properly integrated and perform allocated functions. Some of Different types of system testing are as follows:-

- Recovery testing
- Security testing
- graphical user interface testing
- Compatibility testing

Recovery Testing : Recovery testing is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed. If recovery is automatic, re-initialization, check pointing mechanisms, data recovery, and restart are evaluated for correctness. If recovery requires human intervention, the mean-time-to-repair is evaluated to determine whether it is within acceptable limits.

Security testing: Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration. During security testing, the tester plays the role(s) of the individual who desires to penetrate the system. Anything goes! The tester may attempt to acquire passwords through external clerical means; may attack the system with custom software designed to breakdown any defenses that have been constructed; may overwhelm the system, thereby denying service to others; may purposely cause system errors, hoping to penetrate during recovery; may browse through insecure data, hoping to find the key to system entry.

Graphical user interface testing: Graphical user interface testing is the process of testing a product's graphical user interface to ensure it meets its written specifications. This is normally done through the use of a variety of test cases.

Compatibility testing: Compatibility testing, part of software non-functional tests, is testing conducted on the application to evaluate the application's compatibility with the computing environment.

Software Testing Principles: Different software testing principles are as follows:

Test a program so as to make it fail: Testing is the process of executing a program with the intent of finding bugs and errors. Testing becomes more effective when failures are exposed.

Start testing early: This helps in finding and fixing a number of errors in the early stages of development, thus reduces the rework of finding the errors in the later stages.

Testing is context dependent: Testing should be appropriate and different for different context and also at different points of time.

Test Plan: Test Plan usually describes test strategy, test scope, test objectives, test environment, deliverables of the test, risks and mitigation involved, schedule, levels of testing to be applied, techniques, methods and tools to be used. Test plan should accurately meet the needs of an organization and customer as well (IEEE(1990), IEEE Standard Glossary of Software Engineering Terminology, Los Alamitos, CA: IEEE Computer Society Press).

Effective Test cases: Effective test cases must be designed so that they can be measured and clear test results are produced. Test valid as well as invalid Conditions: In addition to valid test cases, test cases for invalid and unexpected inputs/conditions must also be checked. This form of testing is sometimes specified as regression testing.

Test at different levels: Different testing must be done at different level of testing so different people can perform testing differently using different testing techniques at all level.

End of Testing: Testing has to be stopped somewhere. It is stopped when risks are under some limit or if there is some limitation to it.

International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 2, Issue 2, ISSN No. 2455-2143, Pages 113-117
Published Online December-January 2016 in IJEAST (http://www.ijeast.com)

IV. SOFTWARE TESTING METHODOLOGIES

Correctness is the minimum requirement of software. Correctness testing will need some type of oracle, to tell the right behavior from the wrong one. The tester may or may not know the inside details of the software module under test. [3] Therefore either white box testing or black box testing can be used. Correctness testing has following three forms:-

1) White box testing
2) Black box testing
3) Grey box testing

Black box: it is testing strategy based solely on requirements and specifications. Black box testing requires no knowledge of internal paths, structures, or implementation of the software being tested.

White box: testing is a testing strategy based on internal paths, code structures, and implementation of the software being tested. White box testing generally requires detailed programming skills.

Gray box testing: In this we look into the "box" being tested just long enough to understand how it has been implemented. Then we close up the box and use our knowledge to choose more effective black box tests.

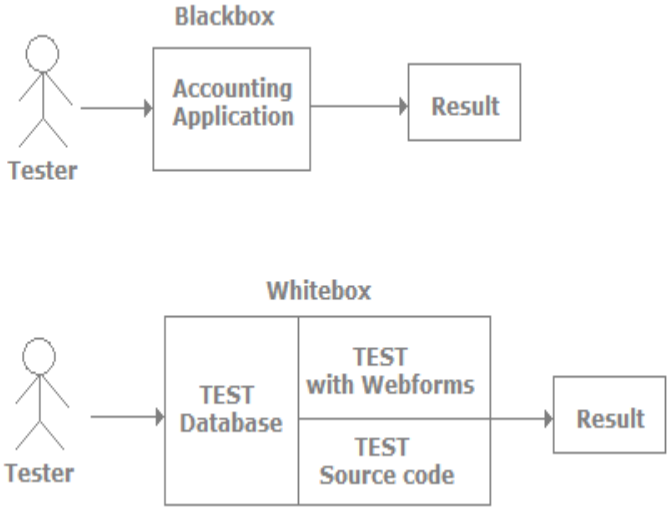

Fig.1

The above figure shows how both types of testers view an accounting application during testing. Black box testers view the basic accounting application. While during white box testing the tester knows the internal structure of the application. In most scenarios white box testing is done by developers as they know the internals of the application. In black box testing we check the overall functionality of the application while in white box testing we do code reviews, view the architecture, remove bad code practices, and do component level testing. There are a number of tools available in market for software testing. Some have been used from a very long time and some new tools have also been developed with a lot of new functionalities. Here, we are going to discuss few tools that are used for automated testing:

Ranorex: This is a simple, comprehensive and cost effective tool used for automatic testing. It is a better alternative to other testing tools because it tests applications from a user's perspective, using standard language and common programming techniques like C# and VB.net. It does not require understanding a scripting language, because it is coded in pure .net code. Any one of the three languages, VB.net, C# and Iron Python can be used. It is used by a lot of commercial software companies and enterprises around the globe. These simulation tools such can have same problems to the same record and playback methods, as the test plan and test cases are often tightly coupled to the code, and both methods still depend highly on experts to create the correct these tests to ensure full coverage. Future work for ranorex involves creating an easily accessible, open and highly documented interface for the clients to write their own plug-ins, which provides the maximum of recognition for their own applications. Some of the features of this tool are:

- It does image-based recognition
- It contains Record-Replay functionality which is called Ranorex Recorder
- It provides easy integration for 32 and 64 bit operating systems
- It is built on the .NET Framework
- It offers a standard and flexible test automation interface
- The Ranorex Recorder provides user code actions, which allows developers to provide special validation or automation methods for their testers with less experience in programming
- It targets to get everything flexible and automated
- It supports all the technologies via Ranorex Plug-Ins
- It allows user interface for managing test cases, plans and configurations
- It supports the use of data variables
- The test automation modules can be created with a standard .NET compiler.
- It provides the ability to do test automation in client's own environment
- It uses standard and modern programming techniques
- It allows testers with little programming knowledge to create professional test plans and cases and modules with Ranorex Recorder.

Rational Functional Tester (RFT): IBM developed this product in 1999. It is an object-oriented programming based automated testing tool. It includes regression and functional testing tools which note down the results of black box tests in a well scripted format. Once captured, these scripts can be

executed against future script builds of any application to verify that
new functionalities have not disabled any previous functionality. With the help of this tool, black box tests can be run as well as white box tests for code bottlenecks, memory leaks or measuring code coverage. In 2006, IBM made a major transition to its software development platform to better help companies build complex software and applications. The Baltic or IBM Rational 7 was developed in 2006. Some of the advantages of this tool are:

- It enables regression testing
- It frees up Quality Assurance departments from maintaining and executing basic tests plan and cases, and encourages the creation of additional, thorough tests
- It automates other non testing activities such as functional and test lab machine preparation.
- It reduces the probability of human error that can occur during activities such as test step execution and also test result recording

It works with Web based, Java, and Microsoft Visual Studio, .NET, SAP, terminal-based, Siebel and Web 2.0 applications. This product also uses a Object Code Insertion (OCI) technology where no source code is used. This technology looks at the executable files in an application. These tools when built into the software, including Pure Coverage and Purify Quantify, perform white box testing on a third party code. Some of the advantages of these tools are:

- It provides memory leak detection and run-time error
- It records the exact amount of time an application spends in a given block of code for the purpose of finding all inefficient code bottlenecks
- It pinpoints areas of application that have been and have not been executed
- When performing regression tests on a product, if the application changes, like, images in different locations, tests will not fail because the product uses robust

Janova : This tool is much similar to others as it enables some users to automate software testing solutions and with the help of this tool it is done in a cloud too. This tool does not require any scripts to be written i.e. only simple English-based tools are used that simplify the task of software implementation with efficient and easy to use tools. Other advantage of this tool is that its cost is very less i.e. $10 per month. There is no such software to download and thus no infrastructural investment is required. Since it is used in the cloud, it has a very quick and easy setup that includes no install. This cloud based software has an easy navigation to home page.

V. CONCLUSION

This paper on Software testing describes in detail about software testing, need of software testing, Software testing goals and principles. . Software testing is often less formal and rigorous than it should, and a main reason for that is because we have struggled to define best practices, methodologies, principles, standards for optimal software testing. To perform testing effectively and efficiently, everyone involved with testing should be familiar with basic software testing goals, principles, limitations and concepts. We further explains different Software testing techniques such as Correctness testing, Performance testing, Reliability testing, Security testing. Further we have discussed the basic principles of black box testing, white box testing and gray box testing. We have surveyed some of the strategies supporting these paradigms, and have discussed their pros and cons.

VI. REFERENCES

[1] Sahil Batra and Dr. Rahul Rishi,"IMPROVING QUALITY USING TESTING STRATEGIES," Journal of Gobal Research in Computer Science, Volume 2,No.6,June 2011.
[2] S.M.K Quadri and Sheikh Umar Farooq,"Software Testing-Goals,Principles and Limitations," International Journal of Computer Applications, Volume 6-No.9,September 2010.
[3] Mohd. Ehmer Khan,"Different Forms of Software Testing Techniques for Finding Errors,"IJCSI International Journal of Computer Science Issues,Vol. 7, Issue 3, No 1, May 2010.
[4] Ajay Jangra, Gurbaj Singh, Jasbir Singh and Rajesh Verma,"EXPLORING TESTING STRATEGIES,"International Journal of Information Technology and Knowledge Management, Volume 4, NO.1,January-June 2011.
[5] Jovanovic and Irena,"Software Testing Methods and Techniques," May 26,2008.
[6] Fu Bo (2007), Automatic Generation Method of Test Data Based on Ant Colony Algorithm, Computer Engineering and Applications.43(12).
[7] Stacey, D. A.(2004), Software Testing Techniques Guide to the Software Engineering Body of Knowledge, Swebok – A project of the IEEE Computer Society Professional Practices Committee.
[8] R.S. Pressman & Associates, Inc. (2005). Software Engineering: A Practitioner's Approach, 6/e; Chapter 14: Software Testing Techniques,
[9] ger S. Pressman, "Software engineering: A practitioner"s Approach," fifth edition, 2001.
[10]Myers, Glenford J.(1979), IBM Systems Research Institute, Lecturer in Computer Science, Polytechnic Institute of New York, The Art of Software Testing, by John Wiley & Sons, Inc.
[11] Redmill, Felix (2005), Theory and Practice of Risk-based Testing, Vol. 15, No. 1.
[12] IEEE(1990), IEEE Standard Glossary of Software Engineering Terminology , Los Alamitos, CA: IEEE Computer Society Press.