ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Issue 4, April 2016

Software Testing Strategies and Methodologies

Poonam, Priya Dhir
Assistant Prof, A.S.College, Khanna

Abstract: Software Quality is the heart and soul of software engineering. Testing is the single most widely used approach to ensuring software quality. Software testing is as old as the hills in the history of digital computers. The testing of software is an important means for ensuring the quality of software. Testing consumes 40 to 50% of development efforts, and as the need of reliability increases, the need of testing increases many fold. Software testing is a very broad area, which involves many other technical and non-technical areas, such as specification, design, implementation, maintenance, and many management issues in software engineering. Our study focuses on the state of the art in testing techniques, as well as the latest techniques which represent the future direction of this area.

Keyword: Software Testing, Level of Testing, Testing Techniques, Testing Tools, Reliability.

I. Introduction

Today, testing is the most challenging and dominating activity in software industry. The purpose of testing is quality assurance, verification, and validation or reliability estimation. Numerous software development and testing methodologies, tools, and techniques have emerged over the last few decades promising to enhance the software quality.

Software Testing is an activity that is performed for evaluating software quality and also for improving it (Guide to the Software Engineering Body of Knowledge, Swebok – A project of the IEEE Computer Society Professional Practices Committee, 2004). Software Testing is a process of evaluating a system by manual or automatic means and verify that it satisfies specified requirements. Testing is basically a process to detect errors in the software product. Software testing is a process that can be systematically planned and specified. Test case design can be constructed and results can be evaluated against prescribed specification. The primary purpose for testing is to detect software failures so that defects can be corrected. Testing is an important part of quality control of software engineering process to assure functionality and reliability.

According to ANSI/IEEE 1059 standard, Testing can be defined as "A process of analyzing of software item to detect the differences between existing and required conditions and to evaluate the features of software item".

Verification is the process of evaluating a system to determine whether the software should conform to its specification. Verification involves reviews and meetings to evaluate documents, plans, requirements and specifications. To verify if system behaves as specified. It is the checking and testing of items, which includes software, for conformance and consistency of software by evaluating the results against pre-defined requirements. In verification we ask a question, are we building the product right?

Validation In this we check the system correctness which is the process of checking that what has been specified by user and what the user actually wanted. In validation we ask a question: Are we building the right system? It **is** the process of evaluating a system during at the end of development process to determine whether it satisfies the specified requirements.

Testing = Verification + Validation

Software Testing principles

1. Testing shows presence of defects: Testing can show the defects are present but cannot prove that there are no defects. Testing always reduces the number of undiscovered defects remaining in the software but even if no defects are found it is not a proof of correctness.

2. Early testing: In the software development life cycle testing activities should start as early as possible and should be focused on defined objectives.

3. Exhaustive testing is impossible: Testing everything including all combinations of inputs and preconditions is not possible. For this reason it is impossible to execute every combination of paths during testing. However it is possible to adequately cover program logic and to ensure all conditions in the component level design have been exercised.

4. Defect clustering: A small number of modules contains most of the defects discovered during pre-release testing or shows the most operational failures.

5. Pesticide paradox: If the same kinds of tests are repeated again and again, eventually the same set of test cases will no longer be able to find any new bugs. To overcome this Pesticide Paradox it is really very important to review the test cases regularly and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

6. Testing is context depending: Testing is basically context dependent. Different kinds of sites are tested differently.

7. Absence of errors fallacy: If the system built is unusable and does not fulfill the user's needs and expectations then finding and fixing defects does not help.

Objectives of Testing

The basic purpose for testing is to find problems and fix them to improve quality. To perform testing, test cases are designed. So a good test is one that finds undiscovered errors. The main objectives of testing are:

1. Detection of bugs: The main objective of testing is to uncover as many bugs as possible. A good test is one that has a high probability of finding errors. It discovers defects, errors and deficiencies which improve the quality of components and the system.

2. Prevention of errors: A good test is one which provides information to prevent or reduce the number of errors to clarify system specification and performance. It also identifies ways to avoid risk and problems in the future.

3. Improve Quality: Testing is a process of examining of code as well as quality concept of system. By doing effective testing we can minimize errors and hence improve the quality of software.

4. Reduce the risk of failure: Most of the complex software systems contains faults which cause the system to fail from time to time. Thus a high level objective of performance test is to bring down the risk of failing to an acceptable level.

Software Testing Strategies

Testing is a set of activities that can be planned in advance and conducted systematically. A test strategy is an outline that describes the testing approach to software development cycle. A number of software testing strategies have been proposed The test strategies describe the test level to be performed. The different levels of strategies are:

Unit testing: Unit testing focuses verification effort on the smallest unit of software design. It tests the basic unit of software which can be a module or component. Using the component level design important control paths are tested to uncover errors within the boundary of the module. A unit is the smallest testable software component and this testing is performed by the testers which require the detailed knowledge of the internal program code. The Unit testis a white box oriented, and the step can be conducted in parallel for multiple components.

Integration Testing: Integration testing is done at two or more tested units are combined into larger structure. It is a systematic testing for constructing the program structure. The objective is to take unit tested components and build a program structure that has been dictated by design. This strategy of integration testing not only affects the integration time but also the cost of the testing.

System Testing: System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team. System testing is based upon the functional and requirement specification of the system. Once modules are integrated then the entire system is tested as per requirements. It is completely black box testing in which no source code should be visible anywhere. System test can occur parallel with integration test especially with the top down method.

Function Testing: The system testing begins with the function testing. It is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for.

Performance Testing: Performance testing is designed to test the run time performance of software within the context of an integrated system.. It occurs throughout all steps in testing process. Performance testing occurs throughout all steps in the testing process. Even at the unit level, the performance of an individual module may be assessed as white box tests are conducted.

Acceptance Testing: It is the final testing stage. The aim of acceptance testing is to give assure that the system is working rather than to find errors. In acceptance testing, customers and users check that the system fulfills their actual needs.

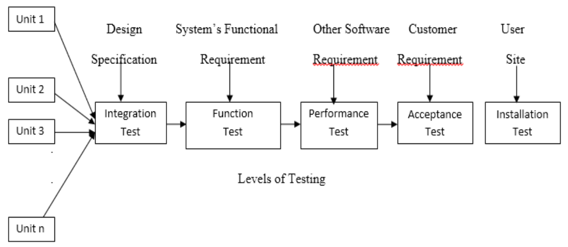
Alpha Testing: Alpha testing is a type of acceptance testing which is performed to identify all the possible bugs before releasing the product to public. The focus of this testing is to simulate real users by using black box and white box techniques. The aim of this testing is to carry out the task that a typical user might perform. Alpha testing is performed by the testers who are usually internal employees of an organization.

Beta Testing: Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Beta version of the software is released to a limited number of end users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation. It is final test before shipping the product to the customers.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Issue 4, April 2016

Installation Testing: It is considered as the final round of testing which involves installation of the system at the user site. Installation testing begins with the configuring the



Levels of Testing

system to the user environment and establishing the connection between them. If the acceptance test having performed at the user site only then installation testing may not be required.

Testing Methodologies

Test cases are developed using various test techniques to achieve more effective testing. These methodologies can be classified into three types-

1. Black box testing
2. White box testing
3. Gray Box testing

White box testing: White box testing is also known as glass box testing that runs on an application with the knowledge of the internal working of the code base. In this testing, the internal detail is made visible. So it is highly efficient in detecting and resolving problems, because bugs can often be found before they cause trouble. It is a strategy for finding errors in which the tester has complete knowledge of how the program components interact. The major advantages of this testing include, All the independent paths with in the module can be tested at least once, this testing helps in optimizing the code. This testing is to be performed by skilled testers, which makes this testing costly. This is an exhaustive testing technique which requires every nook and corner to find out hidden errors that may create problems as many paths will go untested.

Black Box Testing: Black box testing is also known as functional testing. Black box testing focus on testing software based on output requirements and without any knowledge of the internal structure or coding in the program. Black Box testing tests the overall functional requirements of a product. Inputs are supplied to product and outputs are verified. If output obtained are same as the expected ones then the product meet the functional requirements. Black Box testing has little or no knowledge to the internal structure of the system. Thus it only examines the fundamental aspect of the system. This testing strategy is well suited and efficient for large code segments and it clearly separates user's perspective from the developer's perspective through visibly defined roles. This technique takes the strategy of blind coverage, since the tester cannot target specific code segments or error prone areas. In this strategy, the test cases are difficult to design.

Gray Box Testing: It is a combination of White Box testing and Black Box testing. The aim of this testing is to search for defects if any due to improper structure or improper usage of applications. It uses internal data structures and algorithms for designing the test cases. This is particularly important when conducting integration testing between two modules of code written by two different developers. Gray box testing may also include reverse engineering to determine boundary values or error messages. This technique offers combined benefits of black box and white box testing wherever possible. Based on the limited information available, a grey box tester can design excellent test scenarios. The biggest advantage of this technique is that the testing is done from the point of view of the user and not the designer. Since the access to source code is not available, the ability to go over the code and test coverage is limited. The tests can be redundant if the software designer has already run a test case.

Comparison between three testing types:

	Black Box Testing	**Grey Box Testing**	**White Box Testing**
1.	The internal workings of an application are not required to be known.	Someone knowledge of the internal working are known.	Tester has full knowledge of the internal working of the application.
2.	Also known as closed box testing, data driven testing and functional testing.	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application.	Also known as clear box testing, structural testing or code base testing
3.	Performed by end users and also by testers and developers.	Performed by end users and also by testers and developers.	Normally done by testers and developers.
4.	This is the least time consuming	Partly time consuming and	The most exhaustive and

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Issue 4, April 2016

	and exhaustive	exhaustive	time consuming type of testing.
5.	Testing is based on external expectations – internal behavior of the application is unknown.	Testing is done on the basis of high level database diagrams and data flow diagrams.	Internal workings are fully known and tester can design test data accordingly.
6.	No suited to algorithm testing	No suited to algorithm testing	Suited for algorithm testing.
7	This can only be done by trial and error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.

Software Testing Tools

There are a number of tools available in the 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 functionality. The various researchers have produced sets of testing tools. **Miller** described various categories for test tools:

1. **Static Analyzers**: This program-analysis supports "proving" of static allegations-weak statements about program architecture and format.

2**. Code Auditors**: These special-purpose filters are used to examine the quality of software to ensure that it meets the minimum coding standards.

3**. Assertion processors**: These systems tell whether the programmer supplied assertion about the program is actually met.

4**. Test Data Generators**: These selections assist the user with selecting the appropriate test data.

5. **Output Comparators**: This tool allows us to contrast one set of outputs from a program with another set to determine the difference among them.

II. Conclusion

Software testing is an important element of software quality assurance and represents the ultimate review of specification, design and code generation. In this paper we discuss software testing fundamentals and methodologies for software test case design. A series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test techniques that assist in the design of test cases.To accomplish the objective of software testing different techniques are used. White box tests focus on the program control structure while Black box focuses on internal working of a program. Gray box testing uses internal data structures and algorithms for designing the test cases. Experienced software developers say that testing never ends and it gets transferred from software engineer to customers. By applying test case design the software engineer can achieve more complete testing and therefore correct the highest number of errors.

References

[1]. An Integrated Approach to Software Engineering by Pankaj jalote, Narosa Publishing House New Delhi.

[2]. Shivkumar Hasmukhrai Trivedi, "Softwre Testing Techniques", International Journal of Advanced Research in Computer Science and Software Engineering

[3]. Rasneet Kaur Chauhan and Iqbal Singh," Latest research and development on Software Testing Techniques and Tools", International Journal of Current Engineering and Technology.

[4]. Software Engineering: A Practitioner's Approach by R.S. Pressman.

[5]. P.B.Selvapriya," Different Software Testing Strategies and Techniques", International Journal of Science and Modern Engineering.