CHAPTER 3

A SURVEY ON SOFTWARE TESTING METHODS BASED ON COST REDUCTION

3.1 INTRODUCTION

Testing is the process of execution of a system or its component(s) with the motive of finding, whether it fulfils the specified requirements or not. This brings the effectiveness and distinction between their outcomes. Generally testing is, executing a system to discover any errors, leakages or missing prerequisites. Software testing is a vital part of the software development life cycle. One of the key challenges in software testing is, deploying and maintaining a test platform for a project. As a rule, experts are included in testing of a system within their individual limits: Software Developer, Software Tester, Project Lead/Manager and End User. Testing cost decreases when the testing begins as early as possible, it allows time to revamp and deliver failure free software to the customer. However in Software Development Life Cycle, testing might begin from the requirements gathering stage and keeps on contributing till the deployment of the software. It also relies on the development method used. For instance in Water fall model, formal testing begins from the testing phase only, yet in incremental model, testing is done at the end of each cycle and at the end the entire code is executed. Testing is carried out in various methods throughout SDLC, for example in requirement gathering phase, the investigation and verification of requirements are considered as testing. In design stage, reviewing the configuration with the intention to enhance the design is additionally acknowledged as testing. Testing performed by a code developer or system designer on assuming the code is sorted as Units. The decision on when to stop the testing can be recognized by the factors such as Management choice, Testing
3.2 TYPES OF TESTING

Different types of software testing are carried out to accomplish different objectives when testing a software or application. Schematic representation of the testing procedure is on the Figure 3.1 below:

![Schematic representation of the testing procedure](image)

**Figure 3.1 - Schematic representation of the testing procedure**

Based on its objectives, testing can be organized and classified like functional testing, non-functional testing, Structural testing, manual testing,
automated testing, unit testing, integration testing, regression testing, acceptance testing, system testing, stress testing, security testing, performance testing, load testing etc. The testing can be divided into two major parts: unit testing and system or integration testing. The unit testing concentrate on the lowest level components of the software whereas latter focuses on larger subsystems or system as a whole.

3.2.1 FUNCTIONAL TESTING

Functional testing is, testing the functions of component or system or the activities that checks a specific action or function of code. Functional testing will always tend to answer the questions such as ‘can the user do this? Or does this feature work?’ which used to be described in requirement specification or in functional specification. Few functional testing types are as below:

3.2.1.1 REQUIREMENT BASED TESTING

Requirement-based testing, in which the testing requirements are prioritized based on the risk criteria. Depending on their risk level, the tests are prioritized. This will help to make sure that the most important tests are considered in the testing effort.

3.2.1.2 BUSINESS PROCESS BASED TESTING

The second one is Business-process-based testing, in which the scenarios involved from the day-to-day business use of the system with the knowledge of business process.
3.2.1.3 UNIT TESTING

Unit testing is performed by developers before the software code setup is given to the testing team, to officially execute the test cases. In general it is performed by the developers who developed the individual units of source code. The developers use different set of test data from what the quality assurance team uses. The goal of unit testing is to segregate each part of the program and ensure that individual parts are working as expected in terms of functionality and requirements.

3.2.1.4 INTEGRATION TESTING

Integration testing is the testing of combined parts of an application to assess if they function correctly. Integration testing is done in two ways: Bottom-up integration testing and Top-down integration testing. Bottom-up integration testing starts from unit testing, followed by tests of higher-level combinations of modules or builds. In Top-down integration testing, the high level modules are tested first lower-level modules are tested then.

3.2.1.5 SYSTEM TESTING

System testing tests the whole system. After all the components are integrated, the system as a whole is tested thoroughly to check that it meets the mentioned Quality Standards. This testing is performed by a specialized System testing team.

3.2.1.6 REGRESSION TESTING

Whenever an update or change in a software application is done, there are possibilities that the other areas within the application to be impacted by this change. Regression testing is performed to check that a fixed defect has not resulted in any other business rule or functionality violation. The aim of
regression testing is to make sure that a change, such as defect fix should not result in any new fault that is uncovered in the application. Figure 1 represent the typical graphical representation of regression testing.

3.2.1.7 ACCEPTANCE TESTING

This is the most important type of testing, since it is done by the Quality Assurance Team or the internal or external customer who will measure whether the application meets the requirements and satisfies the intended specifications. A set of pre-written scenarios and test cases will be used by QA team to test the application. By performing acceptance tests on the software or application, the testing team can realize how well the application will perform in real production environment. There are legal and contractual requirements for acceptance testing.

3.2.1.8 ALPHA TESTING

This is the first stage of testing that will be done by the teams. The Unit testing, integration testing and system testing are combined together in a test which is called as alpha testing. During this testing, the spelling mistakes, broken lines, cloudy directions will be tested.

3.2.1.9 BETA TESTING

Beta testing will be performed after the successful completion of alpha testing. In this testing, a sample of the aimed audience, somebody from client or user side tests the application. This is also known as pre-release testing. Beta test versions of application or software are distributed to the web audience, partially giving the program to a real world test environment. In this phase, users will install and run the application. Then they send their feedback to the project team. With the help of those feedbacks, the project team will fix the issues before releasing the application or software to the actual users.
3.2.2 NON FUNCTIONAL TESTING

Non-functional indicates to the features of the software that is not related to a user action or function like security or scalability. Non-functional testing includes the below listed various types of testing.

3.2.2.1 RELIABILITY TESTING

Reliability testing is exercising an application to discover failures and eliminated before deploying the system. The intention of reliability testing is to verify product reliability, and to conclude whether the application or software meets the reliability requirements of the customer.

3.2.2.2 USABILITY TESTING

Usability testing is to tests the ease of application with the user interfaces that can be used. It tests whether the application developed is user-friendly or not. There are five components in usability testing. They are Learn-ability, Efficiency, Memorability, Errors, and Satisfaction. Learn-ability is how easy it is for users to complete basic tasks for the first time they see the design. Efficiency is how fast user can experience the task accomplishment. Memorability is when users come back to the design after a long period, does the user remember to use it effectively, or do they have to start from scratch learning everything. Errors are the number of errors users make, the severity of errors and how easily they can recover from the errors. And Satisfaction is how much the user like using the application or system does.

3.2.2.3 EFFICIENCY TESTING

Efficiency testing is to test the amount of code and resources needed for a program to perform a specific function. Software Test Efficiency is calculated by the number of test cases executed divided by the time unit.
3.2.2.4 LOAD TESTING

A load test is conducted to know the behaviour of the application under a particular expected load. Load testing is done to verify a system’s behaviour under normal and peak conditions. It helps to discover the maximum operating capacity and bottlenecks of an application and identify which component is causing degradation.

3.2.2.5 PERFORMANCE TESTING

Performance testing is testing done to check how fast some feature of a system performs under a specific level of workload. It can help in different purposes such as demonstrating that the system achieves performance criteria, compare two systems to check which performs better or measure which part of the system or workload makes the system to perform badly.

3.2.2.6 COMPATIBILITY TESTING

Compatibility testing is the testing of the product or application built with the computing environment. It checks whether the software application or the product built is compatible with the database, operating system, hardware.

3.2.2.7 SECURITY TESTING:

Security testing is to determine that whether the product or application is secured or not and can anyone hack the system or login without any authorization. It is a method to determine that the system protects data and also preserve functionality as it has to be.
3.2.2.8 SCALABILITY TESTING

It is testing a software product or application for measuring its capability in terms of any non-functional capability such as load, number of transactions, data volume.

3.2.2.9 VOLUME TESTING

Volume testing is testing a software product or application with certain amount of data. If the application has to be volume tested with a specific database size, then the database has to be expanded to that size and then test the performance of the application on it.

3.2.2.10 STRESS TESTING:

Stress testing involves testing above and beyond the normal operational capacity, very often to a breaking point to assess the results. It is used to observe the stability of the given system. The focus of this testing is on error handling under a heavy load, availability, robustness. The goals of this testing to ensure that the software or application does not crash under insufficient computational resource conditions.

3.2.2.11 SMOKE TESTING

Smoke Testing is a type of testing which is also known as Build Verification Testing, which comprises of a non-exhaustive set of tests to ensure that the most critical functions work as expected.

3.2.2.12 RECOVERY TESTING

Recovery testing is performed in order to determine how better and fast the application can recover after any type of hardware failure or crash. It is making
the application or software to fail forcefully in different ways to check that recovery is properly ensured.

3.2.3 STRUCTURAL TESTING

The structural testing is basically testing the structure of the component. Structural testing is referred as ‘white box’ or ‘clear-box testing’ or ‘glass box’ because the interest of structural testing is to know what is happening inside the application or system. It needs the testers to have the knowledge of the code implementations. During structural testing the tester is focusing on how actually the software does it. Structural testing can be done at all the levels of testing. In component testing and component integration testing, developers use structural testing especially for code coverage. It is also used in system and acceptance testing, with different structures.

3.2.4 BLACKBOX TESTING

The procedure of this testing is, the testers have to test the software or application without knowing the inside workings or coding. The tester is absent are not involved at all to the system construction modelling and does not have access to the source code. When performing a black box test, tester will coordinate with the system’s client interface by giving inputs and observing outputs without knowing how and where the inputs are cooked upon.

3.2.5 GRAYBOX TESTING

Grey box testing is a testing method which is a combination of white box testing and black box testing. In black box testing the internal structure of the software or application is unknown to the testers whereas in white box testing the tester knows the internal structure of the application. In grey box testing, the internal structure of the application is partially known to the testers. The user
level or black box level or access to the code or internal structure is given to the testers.

3.2.6 MANUAL TESTING

Manual testing is testing the software physically without using any automated tools or scripts. The testers assumes the control of an end client and test the application or software to discover any unexpected error or conduct or defect.

3.2.7 AUTOMATION TESTING

Testing automation is a process of writing scripts for the test cases to be executed to test the software. The testers write the scripts and use software to test the developed software. This is the process automating the manual method. This is used to re-run the test situations more than once such as regression testing situations.

3.3 SUGGESTIONS FOR EFFICIENT TESTING

There are few things that need to be considered for effective and smooth testing. Testers have to closely work with developers and participate in some parallel testing with them as the product or software is getting developed. Some of the non-testing activities such as processes, management documentation, metrics etc. can be identified and eliminated. All applications under construction have to be analysed and outlined to determine stable and well tested areas of the software. These areas may need the least or no testing effort. Testers can thoroughly examine the test suite and remove redundant cases. The aim should be to reduce the test suite or number of to be executed test cases as small as possible. Then these well tested or stable areas test cases can be reviewed and reduced from regression testing suites. Testing focus needs to be switched to
highly improvisational exploratory or rapid testing from resource intensive and highly scripted testing approach. Testing can be planned in small frequent cycles which help to reduce planning and management burdens. The tools and techniques used, have to be analysed and look for the possibilities to reduce the use of costly tool licenses, especially on the test management tools which does not directly help in testing. On the reports side, need focus to cut down on more number of reports, dashboards, lengthy test plans. The defect management processes can be simplified by reducing the number of defect life cycles. The communication on defects can be informal or quick. There are also few ways to reduce the software testing efforts without sacrificing the quality. Those are, the testing has to be managed by walking around and listening, the barriers to the high performers needs to be identified and resolve as early as possible, focus on possibilities to speed up the testing process and eliminate the rework or excess work in progress inventory.

3.4 CONCLUSION

In this study the field of software testing has been surveyed by analysing the methods which is towards to reduce the effort of software testing. By the overall observation, the software development process itself includes the testing process to ensure the quality. In this study a survey on software testing methods based on cost reduction is presented and also suggestions provided to increase the efficiency of the testing in general. Various testing types have been analysed and discussed for better understanding of software testing.