IJIEMR Transactions, online available on 06th Feb 2023. Link :http://www.ijiemr.org/downloads.php?vol=Volume-12&issue=ISSUE-02

DOI: 10.48047/IJIEMR/V12/ISSUE 02/25

Title **Review Paper on Software Testing: Techniques and Test Cases**

Volume 12, Issue 02, Pages: 155-161

Paper Authors
Kavya Muraharisetty, Sai Pallav Chitluri

[illegible]

International Journal for Innovative Engineering and Management Research

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

Review Paper on Software Testing: Techniques and Test Cases

Kavya Muraharisetty
QA Analyst, TEKSystems Global Services, kavyamuraharisetty@gmail.com
Sai Pallav Chitluri
Salesforce Analyst, Accenture, chitluripallav@gmail.com

Abstract
Software Testing has been considered as the most significant stage of the software development life cycle. Around 60% of resources and money are cast-off for the testing of software. Testing can be manual or automated. Software testing is an activity that focuses at assessing the capability of a program and dictates that it truly meets the quality results. Testing is broadly classified into three levels: Unit Testing, Integration Testing, and System Testing. Whenever we think of developing any software, we always concentrate on making the software bug free and most reliable. At this point of time Testing is used to make the software a bug free. There are many test cases that help in detecting the bugs so, in this paper we describe about the most used test cases and testing techniques for the error detection.

Keywords: **Software Testing, Software Testing Strategies, Testing Techniques, Test Cases.**

Introduction

Software Engineering: It is defined as a discipline for developing the high-quality system that allocates with the software development of the software product that uses the clear-cut methods, techniques, sub-routines and procedures. According to IEEE's definition, [1] Software Engineering can be defined as "The application of a systematic, well-defined, disciplined and quantifiable approach to the development, and maintenance of software and the study of these approaches that is considered as the application of engineering to software". Software Engineering is the procedure of making, testing and documentation of the programs ofcomputer.

Software Development Life Cycle: SDLC, Software Development Life Cycle is the task that is being used by the industry of software that helps to design, develop, and test the high-quality software. [2] This concept of SDLC is applied to the limit of both hardware and software configurations as we know that system is comprised of hardware only, software only and the combination of both the configurations. SDLC authorizes the set of various activities that are to be followed and designed to develop a software product effective and efficient. The substructure of this includes the list of steps:

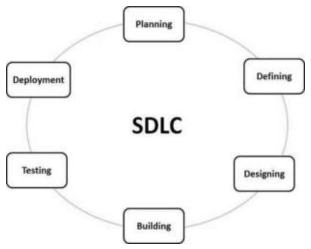


Fig 1: Stages of SDLC

Software Testing: Testing is defined as "Itis a process of gathering information by making observations and comparing them to the expectations". Software Testing plays a very important task in SDLC. It is an appraisal of the software that is against the requirements that are collected from the system and the user specifications. It is defined as the process of executing the program with the purpose of finding the bugs and a procedure to test the code of computer that it does for what it is designed for.

By whom Testing is done - Testing is being done by all those who are intricate to the software development. The various professionals are indulged in testing the software: Project Manager, Software Tester, Software Developer and End Users.

When Testing should be started - The first stage of SDLC is software testing. Starts from the requirement gathering (Planning) phase to the last stage i.e., Deployment phase. In waterfall model, Testing formally is being organized in the phase of testing. [3] Testing at the incremental model is implemented at the last of every increment/ iteration and the complete application is being tested at the last.

When Testing should be stopped - Testing the software is an everlasting process. No one can profess that the software is 100% bug free, instead of testing the software. As the Domain to the input is too large that we cannot verify every input.

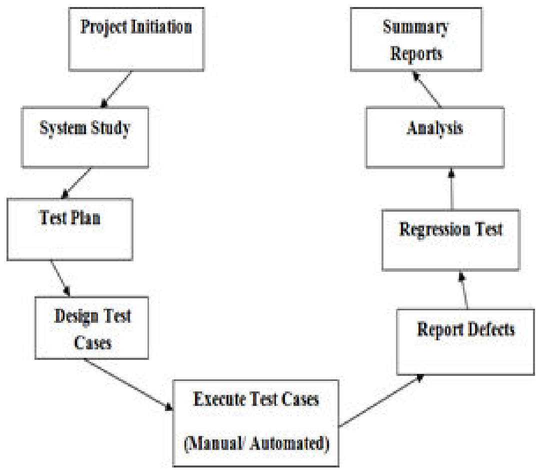


Fig 2: Software Testing Life Cycle

Software Testing Strategies:

There Are Various Testing Strategies That Are Being Used For The Purpose Of Testing:

Unit Testing - This type of testing is performed at the bottom level by the developers before it is moved to the team of testing to execute the test cases. It is the smallest module that can be tested and verified at each section or lines of code. In this output of one module becomes the input of another module. if the output of any one of the modules fails so, then the output to which we give the input also fails.

So, therefore it is nevertheless better to test each module differently so that there would be less chance of fails. In this, white box testing method is implemented.

When The Unit Testing Is Accomplished: It Is Being Accomplished Prior To Integration Testing.

By Whom Unit Testing Is Accomplished: I is accomplished by developers of software and their peers or very rarely by the testers those who are independent.

Integration Testing - Integration Testing is performed immediately after the Unit Testing. In this all the modules are merged to form a larger module and deter mine are they functioning in a proper way and then the testing is implemented on the modules. Testing is done so that in case if any bug remained in the Unit Testing it can be again tested in this testing so as to remove all the bugs.

The basic idea of integration testing is to test how different parts of the system are grouped or work together. For example, a unit test for database access code would not be able to talk to a real database but the integration testing would.
Testing is classified into two parts:
(i) Top-Down Testing
(ii) Bottom-Up Testing

When the Integration Testing is accomplished: It is being accomplished after Unit Testing and before System Testing.

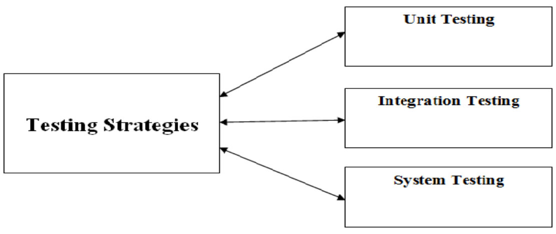
By whom Integration Testing is accomplished: It is accomplished by either the developers or by the Testers those who areindependent.

System Testing - This type of testing is conducted to test the entire system. It is needed to test all the integrated components to test and verify whether it meets the requirements and the standards of quality. The basic purpose of the testing is to assess the compliance of the system within the desired specific requirements. In this, black box testing method is implemented.

When the System Testing is accomplished: It is being accomplished after Integration Testing and before Acceptance Testing.

By whom System Testing is accomplished: It is accomplished by the Testers those who are independent.

Testing Techniques/ Methods:
There are various methods or techniques fortesting the software:



1. Black Box Testing
2. White Box Testing

Black Box Testing: In this type of testing, the intramural structure/ details of the data item are not known by or accessible to its user. In this test cases are generated or designed from the Input / Output value only and no knowledge of design/ code is being required. [1] The testers are only aware of knowing about what is assumed to do, and not to know how it does. These Types of tests can be functional or non-functional.

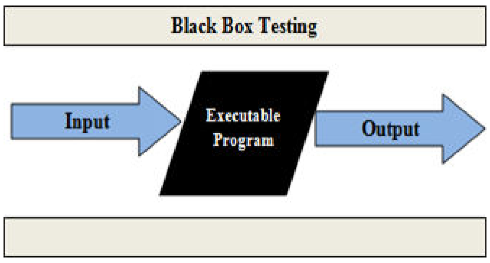
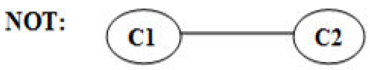


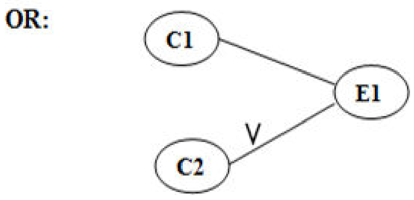
Fig 4: Black Box Testing

Black Box Testing is named so because as we know that in the tester's eyes it is named black box but inner side no one sees. Black Box Testing is also known as Functional testing, Specification, Behavioural, Data Driven or Input-Output Driven.
There are many test cases in Black Box Testing:

I. Equivalence Class Partitioning
II. Boundary Value Analysis
III. Cause Effect Graph
IV. Comparison Testing

Equivalence Class Partitioning: This type of technique partitions the program input domain into the set of equivalence classes from where we can derive the test cases. This partition is done in such a way that program's behaviour is same to every input data that is belonging to the similar equivalent class. [6] The main idea behind the defining of the equivalent class is to test the code with only one value that belongs to the equivalence class is as better as testing the software with some other value that belongs to that equivalence class.





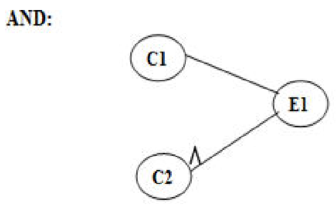
For Example:

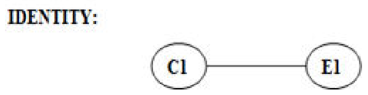
≤

1-500

501 and above

Boundary Value Analysis: It is complementary to partitioning the equivalence class instead of selecting the arbitrary input value to partition; the equivalence class chooses the values at the extreme end of the class.





Cause Effect Graph: It is technique of software test design that includes identifying the cases (Input conditions) and the effects (Output conditions). [2] A weakness of the above mentioned 2 methods are that they don't consider the potential combination of input and output condition. It connects the input classes (causes) to output classes (effects) yielding a directed graph. It utilizes 4 symbols: NOT, OR, AND, IDENTITY.

Comparison Testing: For critical applications that are required the fault tolerance, several independent versions of the software are developed for the similar specification if the output for each version is same then it is presumed that all the implementations are correct but if output is unique then each version is examined to check what is responsible for the different output.

Advantages of Black Box Testing

- Testing is being performed from the viewpoint of users.
- Tester and Programmer both are autonomous to each other.
- Test cases can be designed immediately after the completion of specifications.
- Testers don't know about the languages of programming or how the software has been accomplished

White Box Testing: In this type of testing, the intramural structure/ details of the data item is known by or accessible to its user. [3] In this, test cases are being made based on the code. Programming very well knows about how the implementation of knowledge is significant.

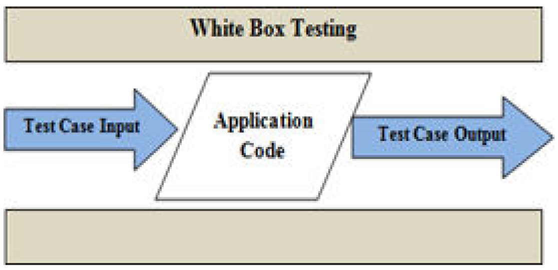


Fig 5: White Box Testing

White Box Testing is named so because as we know that in the tester's eyes it is named white box and inner side everyone sees perfectly.

White Box Testing is also known as Glass Box, Structural; Clear Box, Open Box, Logic Driven, or Path Oriented.

For Example: Basically, a tester and a developer study the code implemented of any field on a webpage, decides purposefully all the legal and the illegal inputs and verifies the output for the outcome that is expected. And also decides by studying the code that is implemented.

So, therefore we can say that white box testingis like the work of a mechanic who only needs to know why the car is not working correctly.

Strategies applied to white box testing are:
 Unit Testing: Within the units thepaths are tested.
 Integration Testing: Between theunits the paths are tested.
 System Testing: Between the subsystems the paths are tested.

For white box testing, unit testing is applicable. There are many test cases in White BoxTesting:

I. Statement
II. Branch
III. Condition
IV. Path
V. Data Flow
VI. Mutation
VII. Domain and Boundary Testing
VIII. Loop Coverage Testing
IX. Logic Based
X. Fault Based

Advantages of White Box Testing

- We need not to wait for the GUI to be implemented as testing is began at a very first stage.
- Helps in code optimization.

Enhancement In TestingProcesses

Test Suite Prioritisation does enhancement in the testing process by Combinational Criteria. [1] The major methodology behind such test case prioritising is the conversion of the weblogs into the test suites relevant with the user session, and further writing it down into an XML format. The Algorithm used for this approach should be accurately prioritised by the coverage based on combinatorial test suites. The usage of genetic algorithms (GAs) for the purpose of automated test data generation for testing the application is yet another enhancement in the testing process, as previously the dynamic means of test data generation remained a big issue in the software testing process, so the usage of Genetic Algorithm based testing is an effective of the test data generation, it also capable of handling the data generation keeping in line with the complexity of program.

Test Automation:

The major enhancement in the testing process leads the testing process towards the Test Automation, which is the use of particular software to carry out the testing process as well as it makes the comparison of actual results with the expected results. Test Automation technique is time effective, as it saves the time of manual testing which can be quite laborious.

In SDLC, [1] [2] Test Automation occurs during the implementation as well as the testing phase. Throughout the world, Test Automation is being practised instead of manual testing as it saves a great amount of time accomplishing the testing processes in shorter time span. Test automation has taken over the manual testing process by reducing its need as well as by exposing the number of errors, shortfalls that cannot be acknowledged via the manual testing process.

Regression Testing being one of the major testing types requires much time when done manually. It typically tests whether the software or the application works properly after the fixation of any bugs or errors.

Because sometimes after the error fixation, the code or application's error or bug ratio gets even higher. So, for the avoidance of the time taken for regression testing; a set of automated test suites is made to form a regression test suite for such purpose. Test Automation also helps in finding the problem at the much earlier stage, saving heaps of modification cost and energy at later stages.

The environment which caters a term typically knows the automation testing execution called Testing Framework. The testing framework is mainly responsible for executing the tests, as well as defining the format in which to express expectations and for the reporting of the results. The standout feature of Testing Framework that makes it widely applicable in various domains worldwide is its application independency. Testing Frameworks are of certain kinds, including Modular, Data Driven, Keyword Driven and Hybrid. The Modular Testing Framework is based on the principle of abstraction which involves the creation of different scripts for different modules of the software or application that is to be tested, thus abstracting each and every component from another level. This Modular division leads to the scalability as well as easier maintenance of the automated test suites. Also, once the functionality is available in the library, the creation of different driver scripts for different types of tests becomes easy and fast. The major con of such type of framework is to embed data within them, so when the modification or up gradation is requisite in the test data, the whole code of the test script needs to get modified. It was the major cause that served as a purpose for the invention of the Data Driven Testing Framework. [12] In this type of Framework the test data and the expected results are ideally stored within different files, helping in the execution of single driver script being able to execute all the test cases with multiple sets of data. This kind of Framework reduces the number of test scripts as well as minimises the amount of code desired for the generation of test cases, gives more flexibility in fixation of errors or bugs. Keyword driven testing Framework utilises self-explanatory keywords which are termed as Directives. Such type of framework is used to explain the actions that are expected to be performed by the software or application that is to be tested. This kind of testing is a basically extension of Data Driven Testing as the data as well as the directives are kept in separate data files. It encompasses all advantages of the data-driven testing framework. Also, reusability of the keywords is another major advantage. The ill factor of this kind of testing framework is that due to the usage of keywords, it adds complexity to the framework making test cases longer and more complex. Hence, to combine the strengths of all frameworks mitigating the ill factors being possessed by them. A hybrid approach is considered best for the usage as it is mainly a combination of all the three approaches and this combination integrates the advantages of all the testing frameworks, making it the most efficient one.

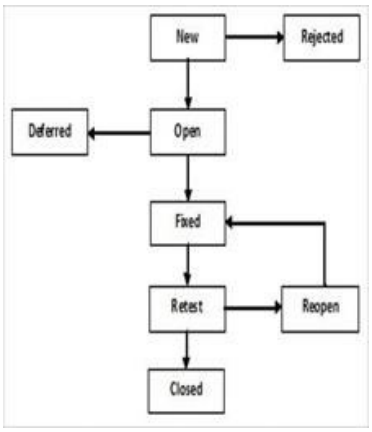
Testing Frameworks in the Agile:

The agile lifecycle is another innovation in software testing as it encompasses short and speedy test cycles with frequently modifying requirements. Thus, the agile environment can encompass any testing framework, but due to the frequent iterations and rapid change in specified requirements, its maintenance of test automation suite becomes quite difficult. Though testing frameworks remains a bad fit for the agile environment because achieving maximum code and functionality coverage remains difficult in it.

Test Driven Development:

It is a technique that makes use of automated unit tests for the purpose of driving the design of software and forcing the decoupling process of the dependencies. With the usual testing process, tester often finds one or more defects or errors, but TDD gives a crystal-clear measure of success when the test no longer fails, enhancing the confidence level about the system meeting its core specifications. Using the TDD approach a great amount of time can be save that might get wasted over thedebugging process.

[3] BDD (Behaviour Driven Development) is mainly an extension of Test-driven Development focusing on the behavioural aspects of the system rather than the implementation level aspects. Hence, giving a clear understanding of what exactly the system is supposed to do giving more efficiency to the testing process. Thus, BDD is mainly Test-driven Development incorporated with Acceptance testing, which typically refers to conducting a test to determine if the specified requirement of the product or software is met or not. If it is performed by the intended customer or user, then it is termed as User Acceptance Testing.



Bug Life Cycle:

A Defect is an error in an application that is restricting the normal flow of an application by mismatching the expected behaviour of an application with the actual one. [2] It occurs when any mistake is made by a developer during the building of an application and when this flaw is found by a tester, it is termed as a defect. It is the responsibility of a tester to do thorough testing of an application to find as many defects as possible to ensure that a quality product will reach the customer. It is important to understand the defect life cycle before moving to the workflow and different states of the defect.

1) New: This is the first state of a defect in the Defect Life Cycle. When any new defect is found, it falls in a 'New' state, and validations & testing are performed on this defect in the later stages of the Defect Life Cycle.
2) Assigned: In this stage, a newly created defect is assigned to the development team to work on the defect. This is assigned by the project lead or the manager of the testing team to a developer.
3) Open: Here, the developer starts the process of analysing the defect and works on fixing it, if required.
If the developer feels that the defect is not appropriate then it may get transferred to any of the below four states namely Duplicate, Deferred, Rejected, or Not a Bug-based upon a specific reason. We will discuss these

four states in a while.

- Rejected: If the defect is not considered a genuine defect by the developer, then it is marked as “Rejected” by the developer.
- Duplicate: If the developer finds the defect as same as any other defect or if the concept of the defect matches any other defect, then the status of the defect is changed to ‘Duplicate’ by the developer.
- Deferred: If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as ‘Deferred’.
- Not a Bug: If the defect does not have an impact on the functionality of the application, then the status of the defect gets changed to “Not a Bug”.

4) Fixed: When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as “Fixed”.
5) Pending Retest: After fixing the defect, the developer assigns the defect to the tester to retest the defect at their end, and until the tester works on retesting the defect, the state of the defect remains in “Pending Retest”.
6) Retest: At this point, the tester starts the task of retesting the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.
7) Reopen: If any issue persists in the defect, then it will be assigned to the developer again for testing and the status of the defect gets
8) changed to ‘Reopen’.
9) Verified: If the tester does not find any issue in the defect after being assigned to the developer for retesting and he feels that if the defect has been fixed accurately then the status of the defect gets assigned to ‘Verified’.
10) Closed: When the defect does not exist any longer, then the tester changes the status of the defect to “Closed”.

Conclusion

- Software testing is the basic activity of software engineering.
- It is an activity that executes the software with the aim of detecting errors or bugs in it.
- This paper describes in detail about the testing techniques, strategies of testing the software.
- Important stages in the process of testing are on the methods of designing the test cases. And it is impossible to find all the bugs from the software so for that we have designed the number of testing techniques that can be taken to analyse.

References:

[1] P. Ron. Software testing. Vol. 2. Indianapolis: Sam’s, 2001.
[2] S. Amland, "Risk-based testing:" Journal of Systems and Software, vol. 53, no. 3, pp. 287–295, Sep. 2000.
[3] Redmill and Felix, “Theory and Practice of Risk-based Testing”, Software Testing