Chapter – 3

Software Testing Techniques and Strategies

Introduction:

Testing is the process of evaluating the system or its components with the intent to find errors whether it satisfies the specified requirements or not. It is the process of executing a system in order to identify gaps, errors or missing requirements in contrary to the actual desire or requirements.

Software Testing Fundamentals:

1. **Testing Objectives:**
   - To demonstrate that faults are not present
   - To find errors
   - To ensure that all the functionalities are implemented
   - To ensure that customer will be able to get his/her work done

2. **Testing Life cycle:**
   - Never ending process
   - Start from early development of project to the late delivery of product
   - An early start of testing reduces the cost, time and rework then provides error free software that is delivered to client
   - In SDLC, software testing can be started from requirement gathering phase and last till the deployment of software.
   - However, it also depends on the development model we preferred for our project.
   - It is difficult to determine when to stop testing, as testing is never ending process and no one can say that any software is 100% tested.
   - When to stop testing:
     - ✓ If testing deadline meet
     - ✓ Completion of test case execution
     - ✓ Buy rate falls below certain level
     - ✓ Management decision, etc.

3. **Test Cases:**

A test case is a document that describes an input, action or event and an expected response, to determine if a feature of application is working correctly.

A test case should contain particulars such as test case name, objective, test conditions, input data requirements and expected results.

In order to design test cases, we use two approaches:

a. **Black Box Testing:**

Knowing the specified function, that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function.

b. **White Box Testing:**

Knowing the internal workings of a product, tests can be conducted to ensure that all the internal operations are performed according to specifications and all the internal components have been adequately exercised.

LEVELS OF TESTING:

1. **Unit Testing:**

The unit testing focuses verification effort on the smallest unit of software design, the software component or module. During unit testing, the modules are tested in isolation i.e. the overall module is splitted into small units and the testing is performed over the small units.

Advantages:

- Reduces debugging effort
- If all modules were to be tested together, it may not be easy to determine which module has the error, so unit testing is necessary to find errors quickly.
- Testing can be carried out immediately
- In general, the goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionally.

Disadvantages:

- Time consuming
- Cost exceeds
- Does not removes bugs completely

2. **Integration Testing:**

Once all the modules have been unit tested, it needs to put them together i.e. interfacing is required. However, there may arise problems like data loss across an interface. One module can have inadvertent, adverse effect on another, sub-functions when combined may not produce the desired function, and global data structures can prevent problems and so on.

Therefore, the integration testing produces a systematic technique for constructing the problem structure while at the same time conducting tests to uncover errors associating with interfacing.

Two methods for integration testing:

- **Bottom Up Integration:** Test follows from lower to higher level modules.
- **Top Down Integration:** Test follows from higher level to lower level modules.

3. System Testing:

This testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested to see that it meets quality standards. The testing is performed by the specialized testing team.

The system testing is the first testing in SDLC where the system is tested as a whole. The application is tested thoroughly to verify that it meets the functional and technical specifications. The application is tested on the environment which is very close to product environment where the application will be implemented.

4. Acceptance Testing:

Conducted by quality assurance team, who will gauge whether the application meets the intended specifications and satisfies the clients' requirements. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application. More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Not done to find errors, but it is important to point out the system crash and major error issues in the application.

5. Alpha and Beta Testing:

Most software project builders use a process called alpha and beta testing to uncover errors that only end users seem able to find.

The alpha test is conducted at the developer's site by a representative group of end users. Unit testing, integration testing and system testing when combined are known as alpha testing.

Beta test is conducted at one or more end user's site. Unlike, alpha testing the developer is not present. Therefore, beta test is a "live" application in an environment that cannot be controlled by the developer.

6. Manual V/s Automatic Testing:

Manual Testing	Automated Testing
❖ Manual testing is not accurate at all times due to human error, hence it is less reliable.	❖ Automated testing is more reliable, as it is performed by tools and/or scripts.
❖ Manual testing is time-consuming, taking up human resources.	❖ Automated testing is executed by software tools, so it is significantly faster than a manual approach.
❖ Investment is required for human resources.	❖ Investment is required for testing tools.
❖ Manual testing is only practical when the test cases are run once or twice, and frequent repetition is not required.	❖ Automated testing is a practical option when the test cases are run repeatedly over a long time period.
❖ Manual testing allows for human observation, which may be more useful if the goal is user-friendliness or improved customer experience.	❖ Automated testing does not entail human observation and cannot guarantee user-friendliness or positive customer experience.

7. Static V/s Dynamic Testing:

Static Testing	Dynamic Testing
❖ Static Testing is white box testing which is done at early stage if development life cycle. It is more cost effective than dynamic testing	❖ Dynamic Testing on the other hand is done at the later stage of development lifecycle.
❖ Static testing has more statement coverage than dynamic testing in shorter time	❖ Dynamic Testing has less statement stage because it is covers limited area of code
❖ It is done before code deployment	❖ It is done after code deployment
❖ It is performed in Verification Stage	❖ It is done in Validation Stage
❖ This type of testing is done without the execution of code.	❖ This type of execution is done with the execution of code.
❖ Static testing gives assessment of code as well as documentation.	❖ Dynamic Testing gives bottlenecks of the software system.
❖ In Static Testing techniques a checklist is prepared for testing process	❖ In Dynamic Testing technique the test cases are executed.
❖ Static Testing Methods include Walkthroughs, code review.	❖ Dynamic testing involves functional and nonfunctional testing

8. Tester Workbench:

The tester workbench is the process used to verify and validate the system structurally and functionally. To understand the testing methodology it is necessary. The tester workbench is one of the part of SDLC, which is comprised of many workbenches.

Example: The programmer's workbench for one of the step to build a system is:

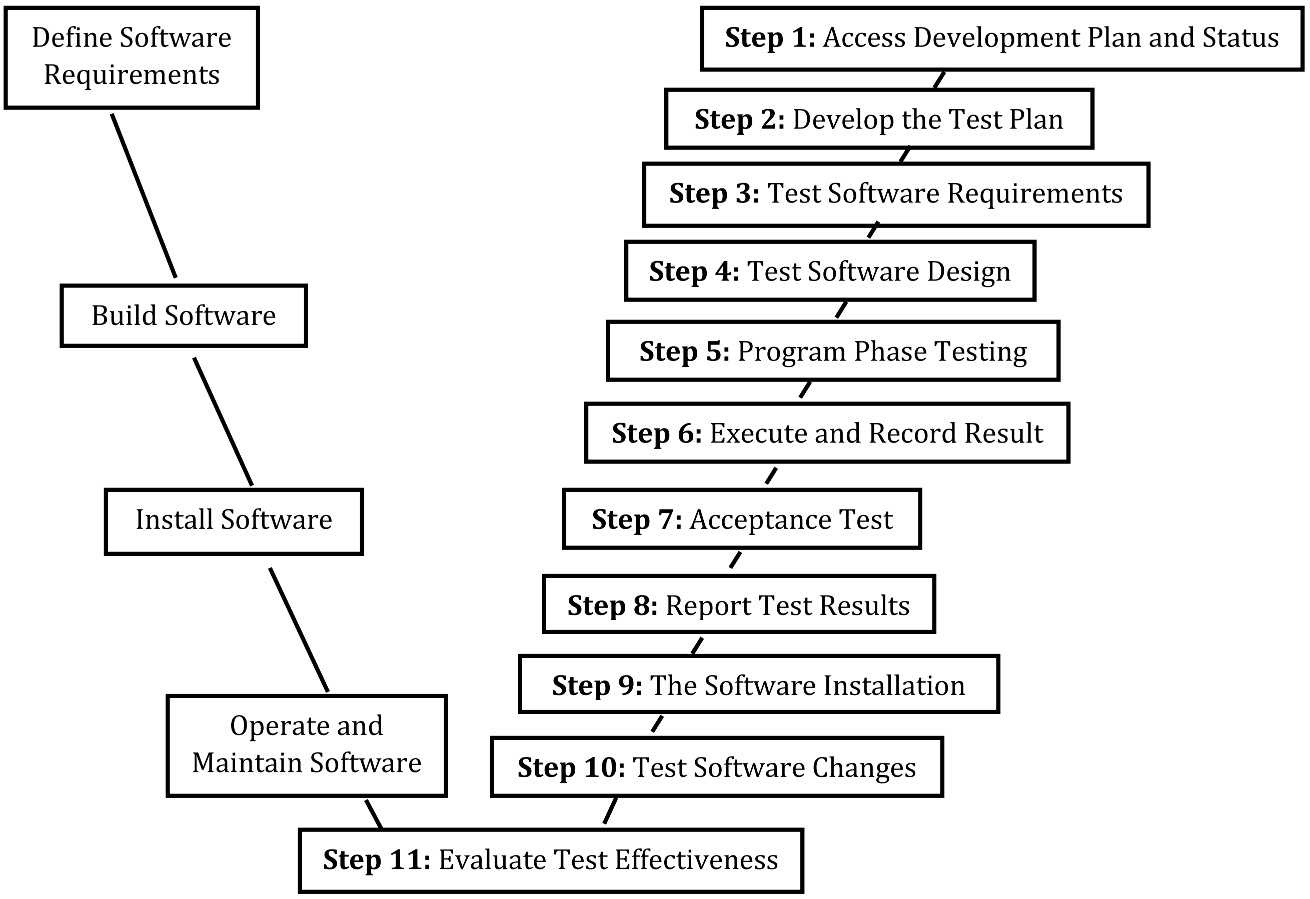
- Input (programmer specification) is given to the producer (programmer)
- Work (example: coding and debugging) is performed; a procedure is followed and a product is developed (produced i.e. methodology is used)
- Work is checked to ensure the product meets the specifications and standards. If check finds no problem, the problem is released to the next workbench. If the check finds a problem, the product is sent back for rework.

A project team uses the workbench to guide them through unit test of computer code. The programmer takes the following steps:

- Given input products (example: program code) to the tester
- Perform work (example: execute unit test) follow a procedure and produce a product deliverables (example: the test result)
- Check work to ensure test results meets test specifications and standards and that test procedure was followed. If the check finds no problem, release the product (i.e. test result) to the next workbench. If the check process finds a problem, the product is sent back for rework.

9. 11 – Steps of Testing Process:

An 11 – step testing process follows the "V" concept of testing. The "V" represents both the software development process and the 11 – step software testing process. The first five steps use verification as the primary means to evaluate the correctness of the interim development deliverables. Validation is used to test the software in an executable mode. Left side of "V" is for verification and right side of "V" shows the 11 steps of testing processes.



Define Software Requirements
Build Software
Install Software
Operate and Maintain Software
Step 1: Access Development Plan and Status
Step 2: Develop the Test Plan
Step 3: Test Software Requirements
Step 4: Test Software Design
Step 5: Program Phase Testing
Step 6: Execute and Record Result
Step 7: Acceptance Test
Step 8: Report Test Results
Step 9: The Software Installation
Step 10: Test Software Changes
Step 11: Evaluate Test Effectiveness

Fig: 11 – Steps of Testing Processes

DIFFERENT TYPES OF TESTING:

1. Installation Testing:

Installation testing is also known as deployment testing. In many cases, software must execute on a variety of platforms and under more than one operating system environments. This testing examines all installation procedures and specialized installation software (example: installers) that will be used by users and all documentation that will be used to introduce the software to end users. Example: A web application is necessary to test with all browsers.

2. Usability Testing:

Usability testing ensures that a good and user friendly GUI is designed and is easy to use for the end users. According to Nielson, usability can be defined in terms of 5 factors:

- Efficiency of use
- Learnability
- Memorability
- Safety

- Satisfaction

Usability testing deals with HCI (Human Computer Interaction)

3. Regression Testing:

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by the change. To verify that a fixed bug hasn't result in another functionality violation is regression testing.

4. Performance Testing:

Performance testing is done to identify any bottlenecks or performance issues rather than finding bugs in software. The different causes that result in performance are:

- Network Delay
- Client Side Processing
- Database Transaction Processing
- Loading Balancing Between Servers

Performance testing is done in terms of following aspects:

- Speed
- Capability
- Stability/Durability
- Scalability

5. Load Testing:

Load testing is the process of testing the behavior of software by applying maximum load in terms of software accessing and manipulating large input data. It can be done at both normal and peak load conditions. It identifies capacity of software by using different tools such as Load Runner, Apache Jmeter, Visual Studio Load Tester, etc.

6. Stress Testing:

Testing of software under abnormal condition such as taking away the resources, apply load beyond the actual load limit, etc.

7. Security Testing:

Under security testing we can include following criteria:

- Confidentiality
- Integrity
- Authentication
- Authorization
- Vulnerabilities
- Session Management
- Input Validation

BLACK BOX AND WHITE BOX TESTING:

1. **Functional Testing (Black-Box):**

This type of testing is called black box testing which is based on specification 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. Functional testing of the software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. The steps that are involved when testing an application for functionality are:

- The determination of the functionality that the intended application is meant to perform.
- The creation of test data based on the specifications of the application.
- The output based on the test data and the specifications of the applications.
- The writing of test scenarios and execution of test cases.
- The comparison of actual and expected results based on the executed test cases.

Black Box Approach:

The technique of testing without having any knowledge of the interior workings of the application. The tester is unclear about the system architecture and does not have access to source code. Typically when performing a black box test, the tester will interact with the system's user interface by providing inputs and examining the outputs without knowing how and where the inputs are worked upon.

2. **Structural Testing (White Box):**

Structural testing is also known as glass box or open box testing. It is the detailed investigation of the internal logic and structure of the code. In order to perform white box test on an application, the tester needs to have knowledge of the internal workings of the system. Inappropriateness of the code/module can be identified. It uses different path tests (cyclometer complexity). Expertise are necessary to conduct this test.

3. **Domain Testing (Gray Box):**

It is a technique to test the application with limited knowledge of the internal workings of an application.

In software testing, the term the more we know the better carries a lot of weight when testing and application.

4. **Non – Functional Testing:**

This testing is based upon the testing of the application from its non-functional attributes. Non-functional testing involves testing the software from the requirements which are non-functional in nature. Non-functional testing also give importance on fields such as performance, security, user interface, reliability, maintainability, usability, etc.

5. **Validation Testing and Validation Testing Activities:**

The set of activities that ensure that the software that has been built is traceable to customer requirements. When the software is completely assembled as a package and interfacing errors have been uncovered and corrected validation is necessary. The different activities of validation testing are:

- Test Criteria
- Review
- Alpha Test and Beta Test

6. **Black Box V/s White Box:**

The Differences between Black Box Testing and White Box Testing are listed below.

Criteria	Black Box Testing	White Box Testing
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
Levels Applicable To	Mainly applicable to higher levels of testing: Acceptance Testing System Testing	Mainly applicable to lower levels of testing: Unit Testing Integration Testing
Responsibility	Generally, independent Software Testers	Generally, Software Developers
Programming Knowledge	Not Required	Required
Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design