CHAPTER III

3.1 Manual vs. Automation Testing

Manual Testing:

Manual testing is the process where testing of an application is done manually by human action without any support from tools or any scripts. It is the practice of testing an application manually to find bugs and errors. In manual testing, the tester checks all the essential features of the given application or software. It means that the tester evaluates numerous components like design, functionality, and performance by clicking through multiple elements of the applications. Manual testing is done by preparing test cases covering all the testing scenarios and executing it or in an ad-hoc process without preparing test cases.

Below diagram shows the process life cycle of manual testing. The cycle contains:

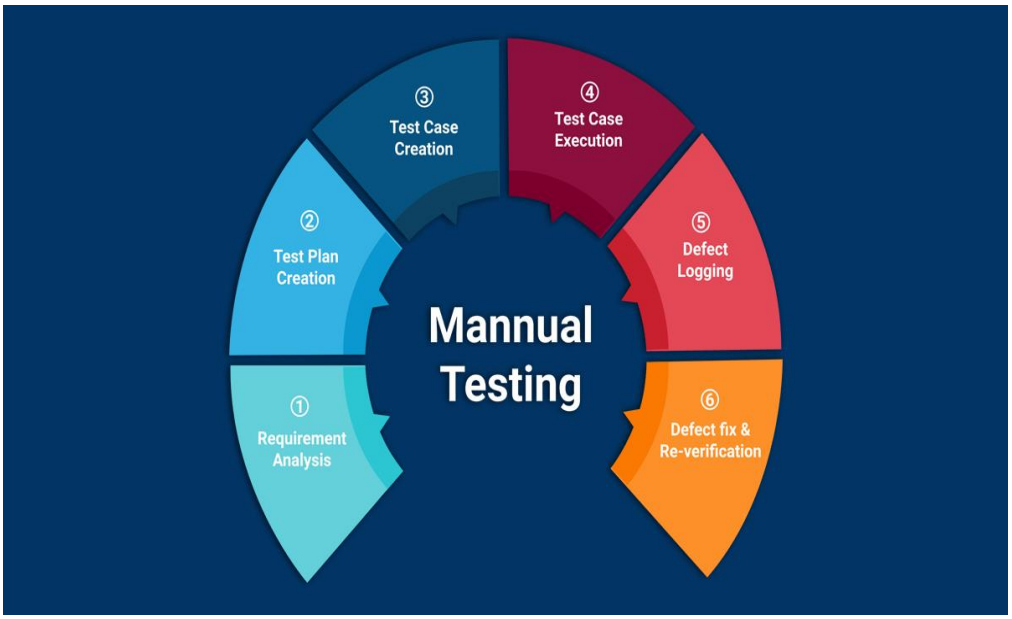


Image 3.1.1 Manual Testing process life cycle. *Image Source: esds.co.in*

1. **Requirement Analysis:** In this phase, testing team study and analyze the requirements from a testing point of view. Brainstorming of the requirements is done and in case of any confusion consult with product owner or stakeholders.
2. **Test Plan Creation**: It's a document developed by test manager. It is set of ideas that guides and is created to inform all the managers, testers, developers

about testing process. Test environment, test limitations, test resources and testing schedule are determined during this phase.

3. **Test Case Creation**: During this phase, assigned QA prepares test cases including all the possible criterion/cases. Each case defines test inputs, procedures, execution conditions and expected results. Test cases should be transparent, efficient and adaptable.
4. **Test Case Execution**: During this phase, features are tested in deployed environment using established test cases. Expected test results are compared to actual and final results are reported back to development team.
5. **Defect logging:** While executing test cases if expected result does not match with actual then it is traced as defect and those cases are reported to developer as bug.
6. **Defect fix and Re-verification:** Once reported bugs are fixed by developer and deployed in test environment, tester must verify it and should confirm that no further issues exist.

Pros of Manual Testing

1. Manual testing is less extensive and cost- efficient in comparison to automation testing.
2. Easier to prepare test data and can be easily updated according to project movement.
3. Manual testing covers all most all cases while it's not possible to cover every cases in automation script. 100% automation is not possible.
4. With manual testing, we can get exact replication of the real user experience.
5. We can get accurate user interface feedback in testing.

Cons of Manual Testing

- Running test manually is very time consuming.
- Testing like load and performance are not possible in manual testing.
- Less reliable testing results as test are performed by humans. Especially during peaks and under pressure.

Automation Testing: Automation testing is the practice of running tests automatically using a dedicated tool. It is used to automate repetitive tasks and other testing tasks which are difficult to perform manually, such as performance and load testing. Key objective of automation testing is obviously same as of manual testing, to compare the actual outcome with the expected outcome. In automation testing tester engineer or developer prepares scripts and execute it using automated tool. Selenium, TestComplete, Katalon Studio, Ranorex are some example of popular automation tools.

Below diagram shows the life cycle of Automation Testing. The cycle contains:

1. Determining the scope of test automation: It is the first stage of automation testing life cycle and its objective is to identify the feasibility of automation. Application modules that can be automated need to be identified from the ones which cannot. Important factors such as cost of the automation tool, the size of testing team, test coverage percentage that can be maintained and expertise related to the automation process need to be identified and considered. Following feasibility checks should be done before starting test automation:

   -Test Case Automation Feasibility

   -AUT (Application Under Test) Automation Feasibility

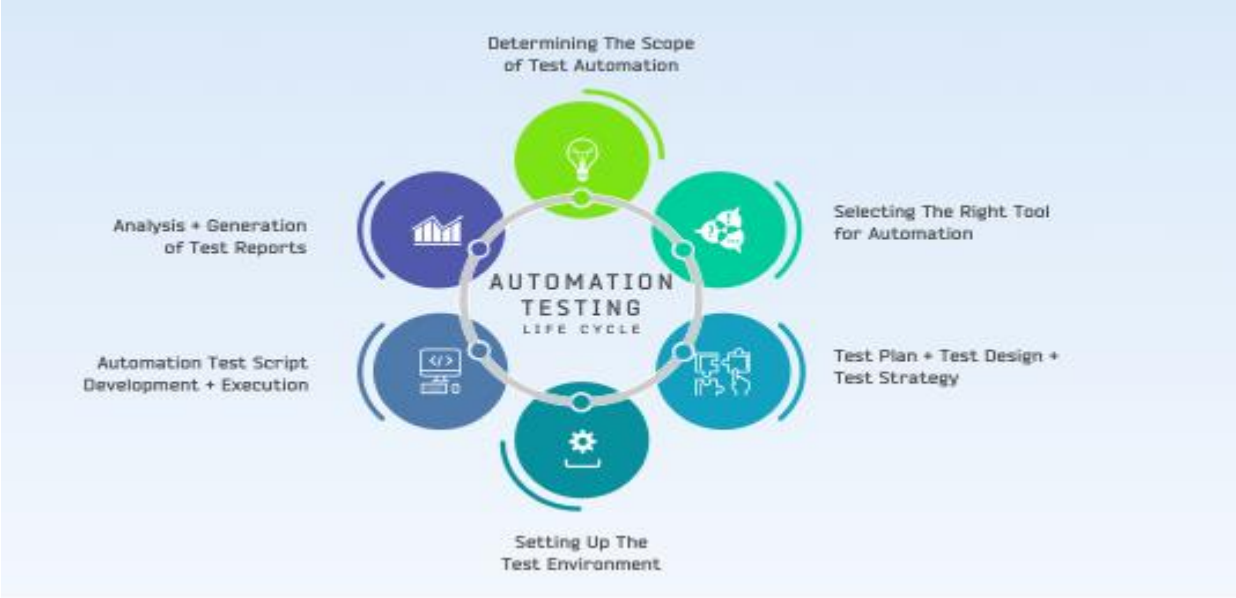


Image 3.1.2 Automation Testing life cycle. *Source: lambdatest.com*

2. Selecting the right tool for automation: Since automation testing is highly dependent on the tool used, identifying the right automation testing tool is critical for the automation testing. Choose a tool that provides you with a support team who can take care of queries or issues. Example of selecting right

tool for automation is, if you are looking for an automated browser compatibility testing tool than you need to keep in mind the variety of browser offered.

3. Test Plan + Test Design + Test Strategy: During this phase, testing team determines the test standards and guidelines as well as the software, hardware and network system to support the testing environment. It also determines a preliminary testing schedule, the data requirements for the test and error tracking system, an associated tracking tool and a method to control the configuration.
4. Setting up the test environment: Testing team need to track and schedule the test environment, setup and install test environment software as well as link network resources and hardware.
5. Automation test script development + Execution: During this phase, test scripts must be created and including all possible functional aspects. To execute test scripts and procedures, the test team has to comply with a schedule decided for procedure execution. Evaluation for test outcomes are executed and test results documentation is prepared during this phase. If the failure occurs due to some functionality, bug report must be written.
6. Analysis + Generation of Test Reports: After completion of all tests, the testing team analyzes to identify particular functionality or components that experience a relative number of problem reports. Analysis indicates any requirement for additional test or procedures and can confirm if the test scripts executed have errors or not. This is the end phase of automation testing and all test reports must be shared with clients at this stage.

Pros of Automation Testing:

- In automation testing, most of the process is automatic which makes it into an efficient and fast testing process.
- Reduces the dependability of testing on the availability of the test engineers.
- Reliability testing, Stress testing, load and performance testing which are not possible with manual testing can be easily carried out through automation testing.
- Automated testing has less chances of error hence more reliable

Cons of Automation Testing:

1. Initial costs are very high. Automation tools can be expensive.
2. Even though a minor change occurs in the GUI, the test scripts has to be recorded or replaces by a new nest script.
3. 100% automation is not possible.

3.2 Types of Testing

3.2.1. Functional Testing

Functional testing is performed to verify that a software application performs and functions correctly according to design specifications. During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines etc.

Its main purpose is to insure that the application is fully functioning.

Some Testing Types:

1. **Unit Testing:**

   A unit test is a way of testing a unit, the smallest piece of code that can be logically isolated in a system. The purpose is to validate that each unit of the software code performs as expected. Unit testing is usually performed by developer during coding phase.

   Unit Testing Advantages:

   - Reduce cost of testing as defects are captured in very early stage.
   - Improves design and allows better refactoring of code.
   - Unit testing helps simply the debugging process. If a test fails, then only the latest changes made in the code need to be debugged.
   - Helps to improve quality of code.

2. **Integration Testing**

After completion of unit testing, each unit or module are integrated. The objective of integration testing is to verify the functionality, performance and reliability between the modules after different units are merged together. It mainly focuses on checking data communication amongst the modules.

Integration Testing Approach

- Big Bank Approach: All components or modules are integrated at once and then tested as a unit. If all of the components in the module are not completed, the integration process will not execute.
- **Incremental Approach**
    - **Top Down Approach**: Approach in which testing is done by integration or joining two or more module by moving down from top to bottom through control flow of architecture structure. In these, higher level modules are tested first, and then low level modules are tested. Then finally, integration is done to ensure that system is working properly.

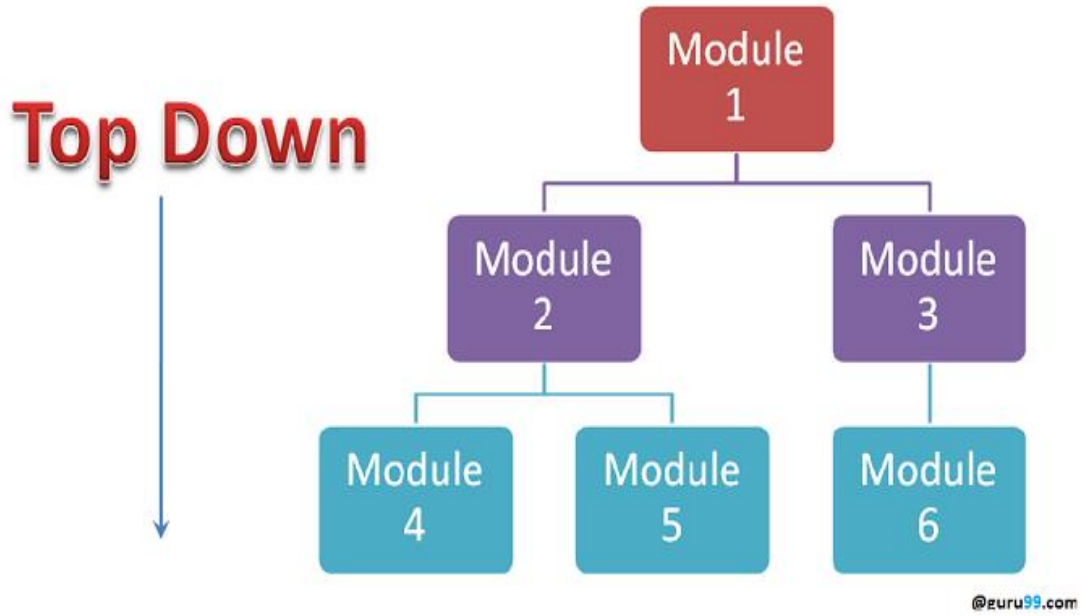


Image: 3.2.1.1: Top Down Approach *Source: guru99.com*

    - **Bottom Up Approach**: In Bottom Up Integration Testing, testing takes place from bottom to up. Lowest level modules are tested first and then high-level modules.

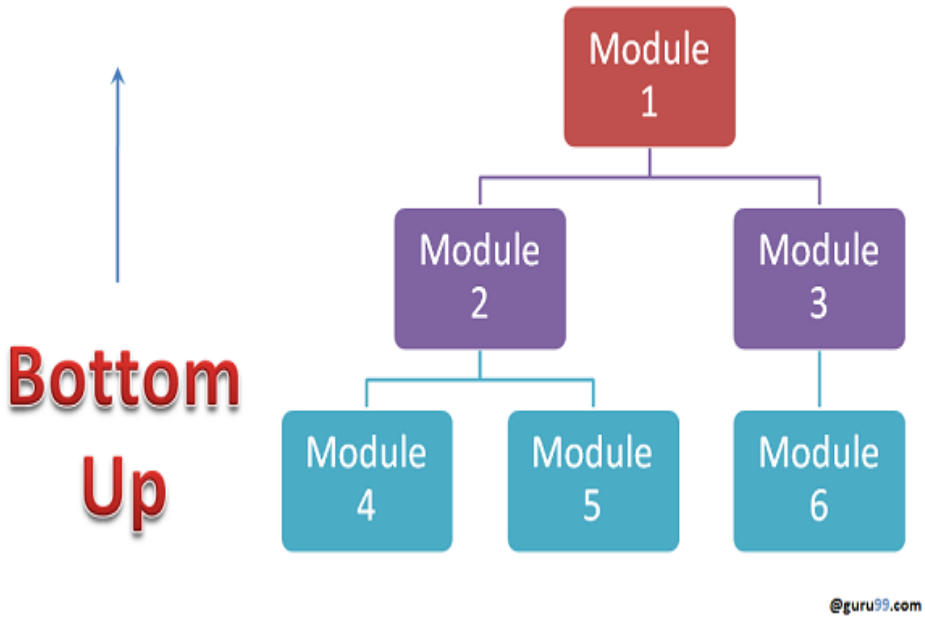


Image:3.2.1.2: Bottom Up Approach *Source: guru99.com*

- Sandwich Approach: It is also called hybrid integration of Top Down Approach and Bottom Up Approach. In this approach, top level modules are tested with lower level modules and at the same time lower level modules are integrated with top modules and tested as a system.

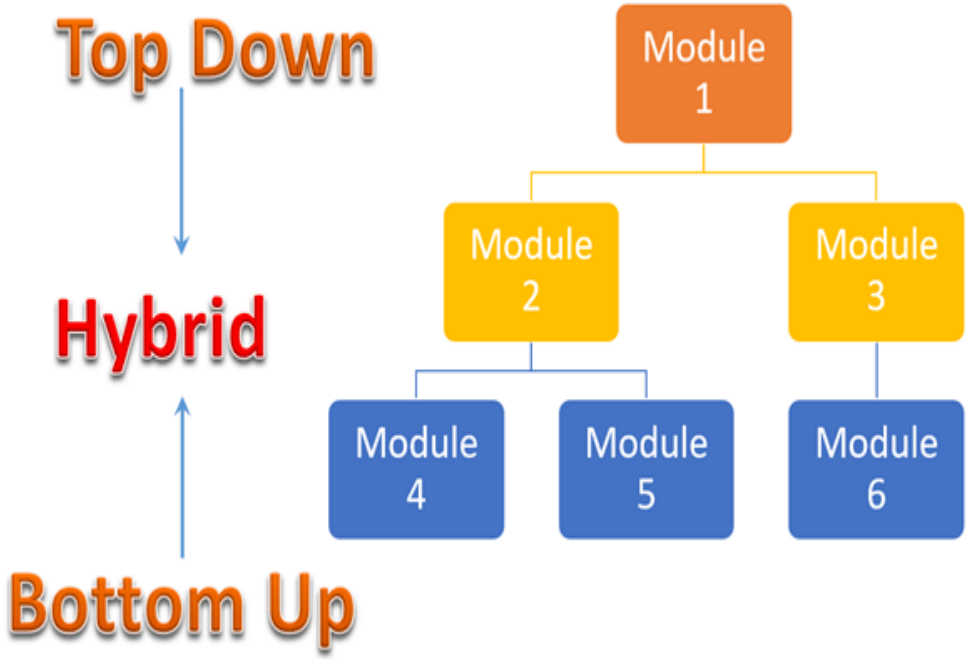


Image: 3.2.1.3: Sandwich Approach *Source: guru99.com*

Stubs and Drivers:

Stubs and Drivers are computer programs which act as a substitute of some other modules which are not available for testing.

Stubs are piece of code that simulates the activities of missing components. In other words, stubs give outputs that are similar to that of the actual product. Stubs are used to test modules during the process of Top-Down integration testing. There are basically four types of stubs used in top-down approach of integration testing:

- Display trace message.
- Values of parameter is displayed
- Returns the values that are used by the modules.
- Returns the values selected by the parameters that were used by modules being tested.

Drivers are program that control devices. Drivers act as translators for programs and a variety of devices. Since every device has its specific commands, the driver is responsible for translating them. They are developed during Bottom-Up approach of integration testing. It can be utilized to test the lower level of the code, when the upper levels of the codes or modules are not developed.

Integration Testing Advantages:

- It makes sure that integrated modules work properly as intended.
- It detects error related to the interface between modules.
- Helps module interact with APIs and other third party tools.
- Increase the test coverage and improves reliability of tests.

3. **System Testing:**

   System testing is the testing of the software system by combining all its individual components together as a complete system. Tester test system treating it as one complete entity. They give input to the system and validate the output against the predefined outputs.

   System Testing Advantages:

   - It is the first testing level in which the whole system is under test from end to end. So, it helps in finding important defects which unit and integration testing could not detect.
   - It uses the test environment which is similar to production environment.
   - Helps to verify system against business, functional and technical requirements of the end user.

4. **Sanity Testing:**

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

Sanity Testing Advantages:

- It helps in saving the time of QA by rejecting the build with faulty code changes in the initial stage itself.
- It helps to verify that application is properly functioning after changes.

5. Smoke Testing

Smoke testing is a software testing technique performed post software build to verify that the critical functionalities of software are working fine. It is executed before any detailed functional or regression tests are executed. The main purpose of smoke testing is to reject a software application with defects so that QA team does not waste time testing broken software application.

Smoke testing advantages:

- Helps to identify critical functionality issues.

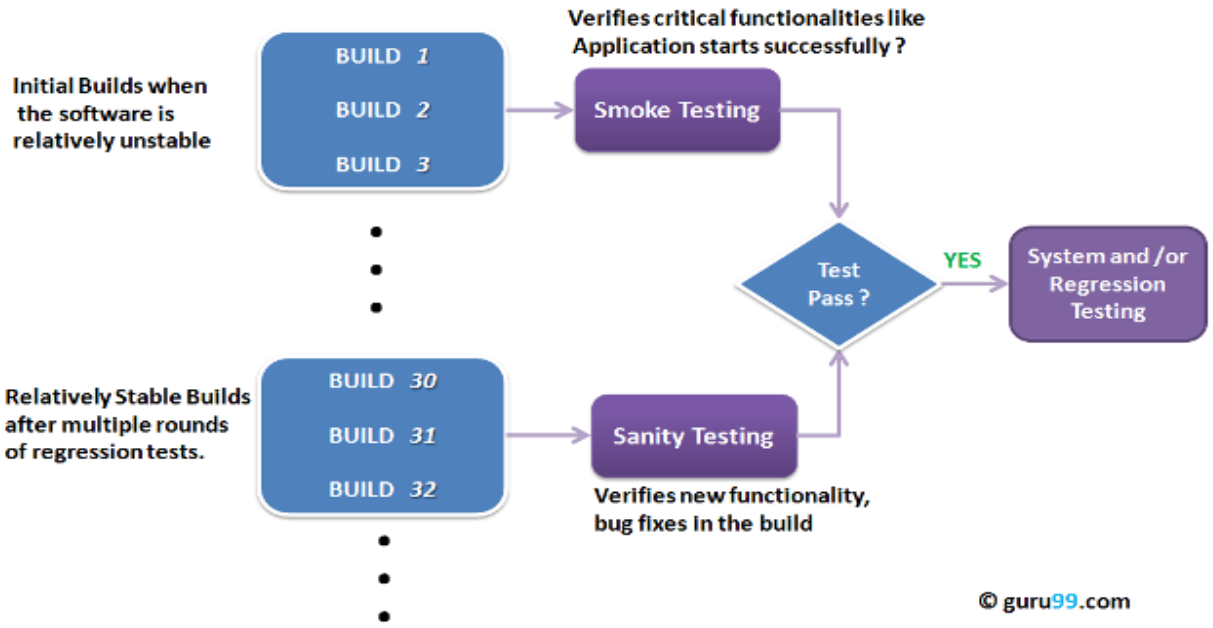


Image: 3.2.1.3: Diagram that display difference between sanity and smoke testing.

6. **Interface testing:**

When a software is developed, in embraces different component in it, such as database, server etc. The connection that helps user to establish communication amount these components is termed as an Interface. Interface testing is a type of testing, which verifies the communication between different software systems.

Interface Testing Advantages:

- Helps to verify the communication between the systems are done correctly.
- Verify if all supported hardware/software has been tested.
- Verify if all linked documents be supported on all platform.
- Verify the security requirements or encryption while communication happens between systems.

7. **Regression testing**

Regression testing is testing existing software applications to make sure that a changes made it system has not broken any existing functionality. Its purpose is to catch bugs that may have been accidentally introduced into a new build. Carrying out regression testing using automated tools will reduce the test consumption time.

Regression Testing Advantages:

- It verifies that the modifications do not impact the correct work of already tested code.
- It promotes the improvement of product quality.

8. **Beta/Acceptance Testing**

It is a formal type of software testing which is carried out by the customer. It is performed in real environment before releasing the product to the market for the actual end users. Beta Testing is carried out to ensure that there are no major failures in the software and it satisfies the business requirements from end user prospective. It is final testing done before releasing an application for commercial purpose.

Beta Testing Advantages:

- It provides an additional level of testing to the development life cycle before releasing the product.
- It helps in uncovering unexpected error that are not noticed by the QA while testing in test server.

9. White Box Testing:

White box testing is based on an analysis of the internal structure of the component or system. Internal system flow and code knowledge must be required for performing this type of testing. Under this tests are based on the coverage of code statements, branches, paths, conditions, etc. White Box testing method is applicable to the following levels of software testing:

- Unit Testing: For testing path within a unit.
- Integration testing: For testing paths between units
- System testing: For testing paths between subsystems.

White Box Testing Advantages:

- White box testing is a complete testing as the entire code and structures are tested.
- It results in the optimization of code removing errors and helps in removing extra lines of codes.

10. Black Box testing

Black box testing is a type of software testing in which the internal knowledge of the product is not necessary. Tests are based on requirements and functionality. Black box testing is mainly applicable to higher level of testing like acceptance testing and system testing.

Note: Black box testing technique are described in section 3.3.3

Black Box testing Advantages:

- Testing can be started once the development of the application is done.
- Defects and inconsistencies can be identified at the early stage of the testing
- It is more effective for large and complex applications

Difference between black box testing and white box testing:

Criteria	**Black Box testing**	**White Box testing**
Definition	It is a testing method without having knowledge about the actual code or structure of the application.	It is a testing method having knowledge about the actual code and internal structure of the application
Level Applicable To	Mainly applicable to higher levels of testing: Acceptance testing and system testing.	Mainly applicable to lower levels of testing: Unit testing and Integration testing
Responsibility	Generally, carried out by Software tester	Generally, carried out by Software Developers

Table: 3.2.1.10: Difference between black box and white box testing

11. End to End Testing

End-to-end testing is a Software testing methodology to test an application flow from start to end. It is performed from start to finish under real-world scenarios like communication with hardware, network, database and other applications. The main reason for carrying out this testing is to determine various dependencies of an application as well as ensuring that accurate information is communicated between various system components.

End-to-end testing advantages:
- Reduce future Risks
- Reduces cost and effort
- Decreases Repetitive Efforts

Differences between End to End Testing and System Testing:

S.NO.	End to End Testing	System Testing
1.	Validates both the main software system as well as all the interconnected sub-systems.	As per the specifications provided in requirement document, it just validates the software system.
2.	The main focus is on verifying the end to end testing process flow.	The main focus is on verifying and checking features and functionalities of the software system.
3.	All the interfaces including backend processes of the system is taken under consideration.	Only functional and non-functional areas and their features are considered during testing.

Table: 3.2.1.11: Difference between end to end testing and system testing.

12. Dependency Testing

Dependency Testing is a testing technique in which an application's requirements are pre-examined for an existing software. In dependency testing impacted areas of the application are also tested when testing the new features or existing features.
Example of dependency testing: If you are creating profile for an admin, testing is not completed after profile is successfully created. You should assign created profile to some admin and test if admin activities are allowed based on profile assigned or not. Dependency while editing the profile must also be tested.
Dependency testing advantages:

- There will be high chances of covering all the cases.

13. Static Testing and Dynamic Testing

Static Testing is a type of software testing method which is performed to check the defects in software without actually executing the code of the software application. It is a system of white box testing where developers verify or check code to find fault.

Dynamic Testing is a type of software testing which is performed to analyze the dynamic behavior of the code. It includes the testing of the software for the inputs values and output values that are analyzed. It is completed by walking the real application with valid entries to verify the expected results.

Differences between static and dynamic testing:
- Static testing is about prevention whereas dynamic testing is about cure.
- Static testing is done in verification stage whereas dynamic testing is done in validation stage.
- In static testing code is being examined without being executed whereas in dynamic testing, code is being executed and tested.
- Static testing involves checklist for testing process. Dynamic testing involves test cases for testing process.

14. Maintenance Testing

Maintenance testing stands for test that needs to be carried out after the modification and updates done after the delivery of the software product or sometimes it might be after system migration to other hardware.
Usually maintenance testing is consisting of two parts: First one is, testing the changes that has been made. Second one is regression tests that prove that the rest of the system has not been affected by the maintenance work.
Maintenance testing advantages:
- Ensure the reliability and efficiency of the equipment's.
- Helps to determine critical areas or equipment that needs immediate action.
- Helps to upgrade product quality.

3.2.2 Non-Functional Testing

Non-Functional Testing is a type of Software testing that checks non-functional aspects of a software application. It is designed to test the readiness of a system as per non-functional parameters that are never addressed by functional testing.

For instance, non-functional test would be to check how many people can simultaneously login into software.

Purpose

The sole purpose of this type of testing is to ensure that the non-functional aspects of the application are tested and the application works well in context to the same.

It also covers the testing of all the characteristics of the application which helps an application to meet the business expectation.

Functional vs. Non-Functional Testing

There are several differences but let's state few of them:

Parameters	Functional Testing	Non-Functional Testing
Functionality	Describes what the product does.	Describes how the product works.
Objective	Carried out to validate software actions.	Done to validate the performance of the software
Requirement	Carried out using the functional specification.	Carried out by performance specifications.
Focus Area	Based on customer's requirements.	Focuses on customer's expectation.
Execution	Performed before non-functional testing.	Performed after the functional testing.
Example	Check login functionality.	Page should load in 2 seconds.

Table 3.2.2: Difference between Functional & Non-Functional Testing

3.2.2.1. Non-Functional Testing Parameters

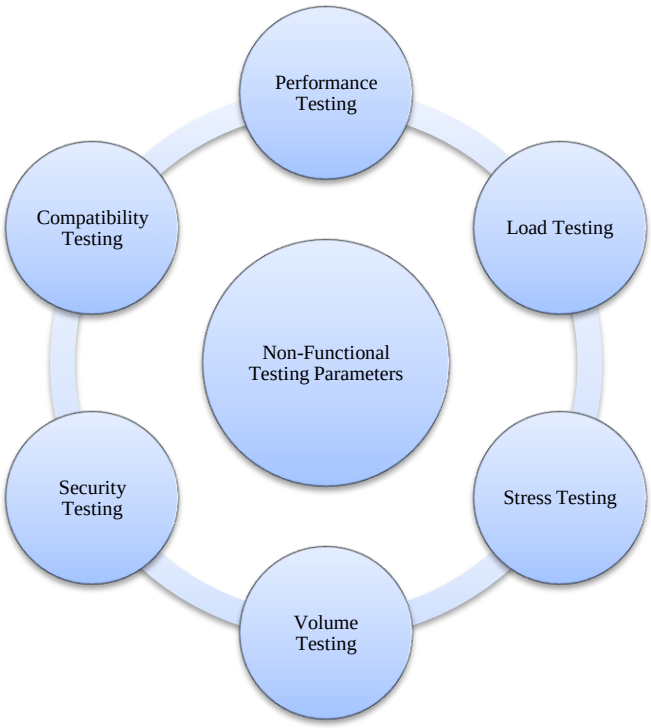


Figure 3.2.2.1: Non-Functional Testing Parameters

1. **Performance Testing**

   Performance Testing also knows as ‘Perf Testing’ is a type of testing performed to check how application or software performs under workload in terms of responsiveness and stability.

   This test is mainly performed to check whether the software meets the expected requirements for application speed, scalability, and stability.

   Performance Test Tool:

   Jmeter, Load Runner, Open STA, Web Load

   Key Points:

   - Validates that the system meets the expected response time.
   - Evaluates that the significant elements of the application meet the desired response time.
   - It can also be conducted as a part of integration testing and system testing.

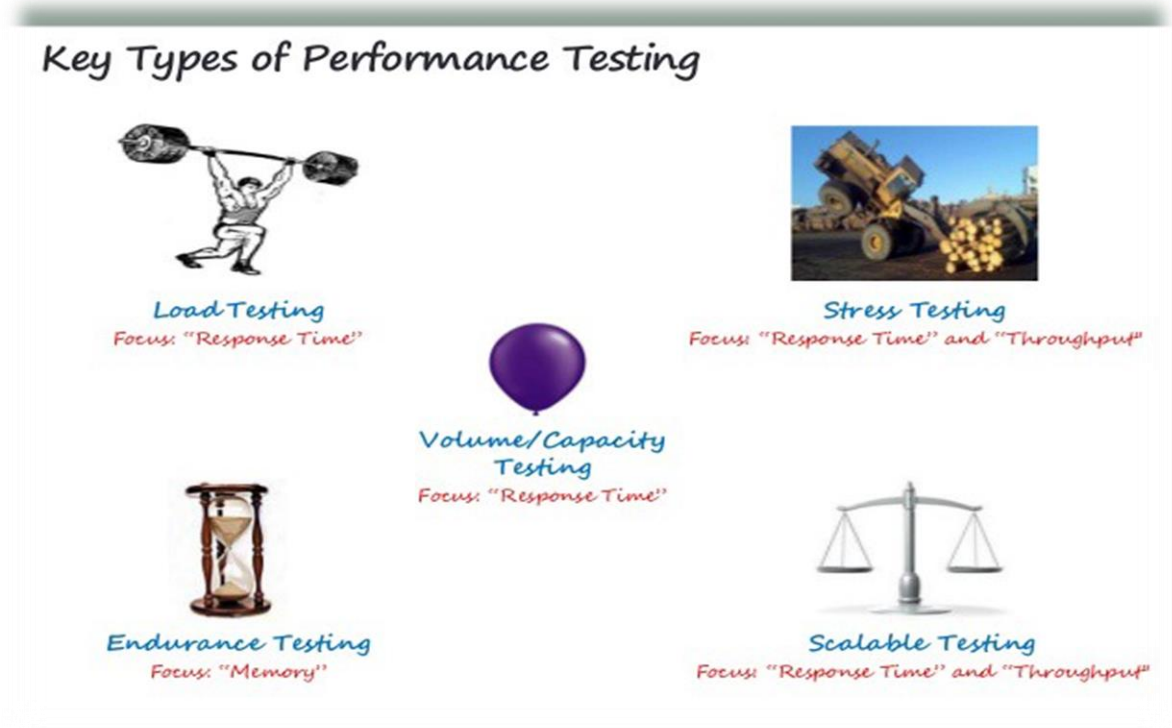


Figure 3.2.2.2: Performance Testing

2. Load Testing

Load Testing is a type of performance test where the application is tested for its performance on normal and peak usage. The performance of an application is checked with respect to its response to the user request and its ability to respond consistently within an accepted tolerance on different user loads.

Key Points:

- Validates that the system performs as expected when concurrent users access the application and get the expected response time.
- This test is repeated with multiple users to get the response time and throughput.
- At the time of testing, the database should be realistic.
- The test should be conducted on a dedicated server which stimulates the actual environment.

3. Stress Testing

Stress Testing is used to find ways to break the system. The test also provides the range of maximum load the system can hold.

Generally, Stress Testing has an incremental approach where the load is increased gradually. The test is started with a load for which the application has already been tested. Then, more load is added slowly to stress the system. The point at which we start seeing servers not responding to the requests is considered the breaking point.

Key Points:

- Test on low memory or low disc space on clients/servers that reveal the defects which cannot be found under normal conditions.
- Multiple users perform the same transactions on the same data.
- Multiple clients are connected to the servers with different workloads.
- Reduce the Time to "Zero" to stress the servers to their maximum stress.

4. Volume Testing

Volume Testing is to verify that the performance of the application is not affected by the volume of data that is being handled by the application. In order to execute a Volume Test, a huge volume of data is entered into the database. This test can be an incremental or steady test. In the incremental test, the volume of data is increased gradually.

Generally, with the application usage, the database size grows, and it is necessary to test the application against a heavy database. A good example of this could be a website of a new school or a college having small amounts of data to store initially, but after 5-10 years, the data stores in the database of the website is much more.

Key Points:

- When the software is subject to large amounts of data, checks the limit where the software fails.
- Maximum database size is created and multiple clients query the database or create a larger report.

5. Security Testing

Security Testing is done to ensure that the application has no loopholes which could lead to any data loss or threats. It is one of the important aspects of non-functional testing and if not performed properly, it can lead to security threats.

It aims at verifying 6 basic principles: Confidentiality, Integrity, Authentication, Authorization, Availability, and Non-repudiation

Few common security testing techniques:

SQL injection, Security Misconfiguration, Cross Site Scripting (XSS), Sensitive Data Exposure, Broken Authentication and Session Management and so on.

Note: Security Testing Techniques will be covered in Security chapter.

6. Compatibility Testing

Compatibility testing is a non-functional testing conducted on the application to evaluate the application's compatibility within different environments.

- Operating system Compatibility Testing - Linux , Mac OS, Windows
- Database Compatibility Testing - Oracle SQL Server
- Browser Compatibility Testing - IE , Chrome, Firefox
- Other System Software - Web server, networking/ messaging tool, etc.

Key Points:

- Test each hardware with minimum and maximum configuration.
- Test with different browsers.

Types of Compatibility Testing:

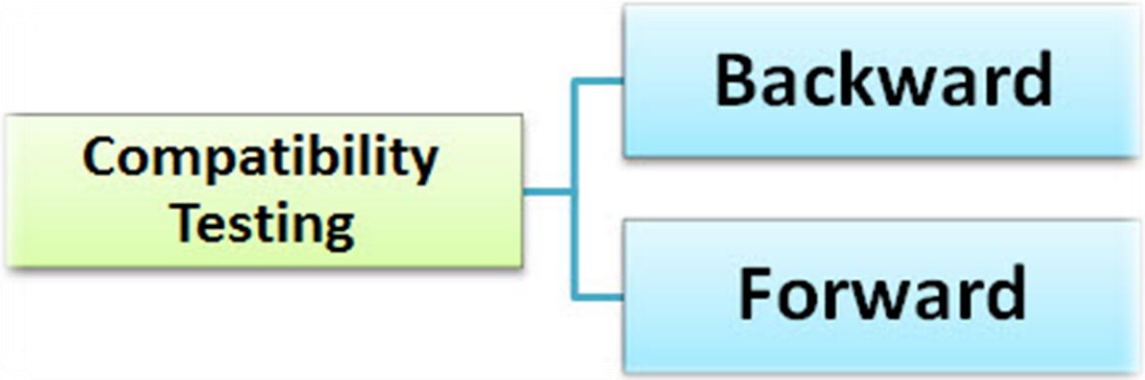


Figure 3.2.2.3: Types of Compatibility Testing

1. **Backward Compatibility Testing**

   A technique to verify the behavior and compatibility of the developed hardware or software with their older versions of the hardware or software.

   Backward compatibility testing is much predictable as all the changes from the previous versions are known.

2. **Forward Compatibility Testing**

   A process to verify the behavior and compatibility of the developed hardware or software with the newer versions of the hardware or software.

   Forward compatibility testing is a bit hard to predict as the changes that will be made in the newer versions are not known.

Tools for Compatibility Testing

- **Browser Stack** - Browser Compatibility Testing: This tool helps a Software engineer to check application in different browsers.
- **Virtual Desktops** - Operating System Compatibility: This is used to run the applications in multiple operating systems as virtual machines. Number of systems can be connected and compare the results.

7. **Other**

   Other non-functional testing techniques performed at all test levels can be: Compliance Testing, Endurance Testing, Localization Testing, and Usability Testing.

3.3 Test Design Techniques

In software engineering, test design technique is a procedure for determining test conditions, test cases and test data during software testing.

3.3.1. What is Test Design Technique?

Software test design refers to the process of designing different tests that validate the eligibility of software before its release in the market. The process of test designing assumes high priority, as a poorly designed test will lead to improper testing of an application. This, in turn, leads to failure in identifying defects and consequently, an application containing errors may be released.

Purpose
The main purpose is to ensure that the products meet the expectations of clients and their businesses, these techniques allow testers to execute the test effortlessly based on various risk factors.

3.3.2. Why are Test Design Techniques important?

Test design techniques are applied to satisfy the goals of every individual in software development projects, including testers. Here is a checklist of standards that a smooth testing process meets:

- Gather information to understand users' requirements
- Derive all important business scenarios
- Design test scenarios for every derived critical business scenarios
- Assign all planned test scenarios to different test cases

Two of main advantages of techniques lie on their consistency and repeatability:

- **Possibility to reproduce a test:**

  At several testing phases, testing techniques are considered as a set of rules help ensure a minimum level of consistency. Testers, indeed, would work much more efficiently with a base, thereby reducing a significant amount of effort in later fixing.

- **Increasing of the found bugs:**

  Test design techniques can also be used as analytical tools. When applying techniques to elements, we often see problems in the definition of those elements.

3.3.3. Types of Test Design Techniques

Each test design techniques are suitable for identifying a certain type of error. Software Testing includes the following techniques:

1. Specification-based or Black-box techniques

- Equivalence Partitioning:

  The idea of this approach is grouping the inputs with the same attributes to partitions. Code is not visible to testers. Your task is to pick one condition out of each partition, which covers all possible scenarios, to execute test cases. If a condition of a partition is valid, other conditions are valid too. Likewise, if a condition in a partition is invalid, other conditions are also invalid. This helps reduce the number of test cases.

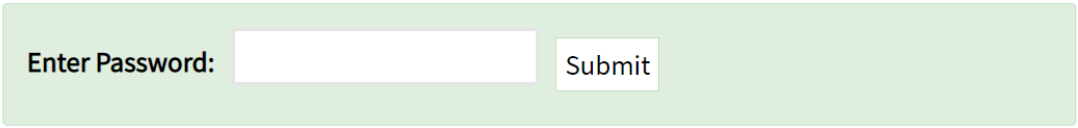
- Boundary Value Analysis:

  This is one of the software testing techniques in which test cases are designed to include values at the boundaries. If the input is within the boundary value, it is considered 'Positive testing.' If the input is outside of the boundary value, it is considered 'Negative testing.' The goal is to select test cases to execute boundary values. In other words, the behavior of Negative testing is more likely to be incorrect than the behavior of Positive testing; and boundaries are an area in which testing is more likely to yield defects.

Example: Equivalence & Boundary Value

Following password field accepts minimum 6 characters and maximum 10 characters

That means results for values in partitions 0-5, 6-10, 11-14 should be equivalent



Test Scenario #	Test Scenario Description	Expected Outcome
1	Enter 0 to 5 characters in password field	System should not accept
2	Enter 6 to 10 characters in password field	System should accept
3	Enter 11 to 14 character in password field	System should not accept

Source: guru99.com

Figure 3.3.3.1: Example of Equivalence & Boundary Value

Why Equivalence & Boundary Analysis Testing?

- This testing is used to reduce a very large number of test cases to manageable chunks.
- Very clear guidelines on determining test cases without compromising on the effectiveness of testing.
- Appropriate for calculation-intensive applications with a large number of variables/inputs

- Decision Table Testing:

  This technique can be used in test design because it helps testers explore the effects of combining different input values when adhering business rules. A Decision Table is a tabular representation of conditions versus test actions. Conditions are considered as inputs, while actions are considered as outputs.

  Example: Decision Table Testing

  The condition is simple if the user provides correct username and password the user will be redirected to the Account Page. If any of the input is wrong, an error message will be displayed. (Where T=True, F=False)

Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

Figure 3.3.3.2: Decision Table Testing

- State Transition Diagrams:

  Using this approach, the tester analyzes the behavior of an application under test (AUT) for different input conditions in a sequence. You can provide both positive and negative input test values and record the system behavior. Any system in which you get a different output for the same input is a finite state system.

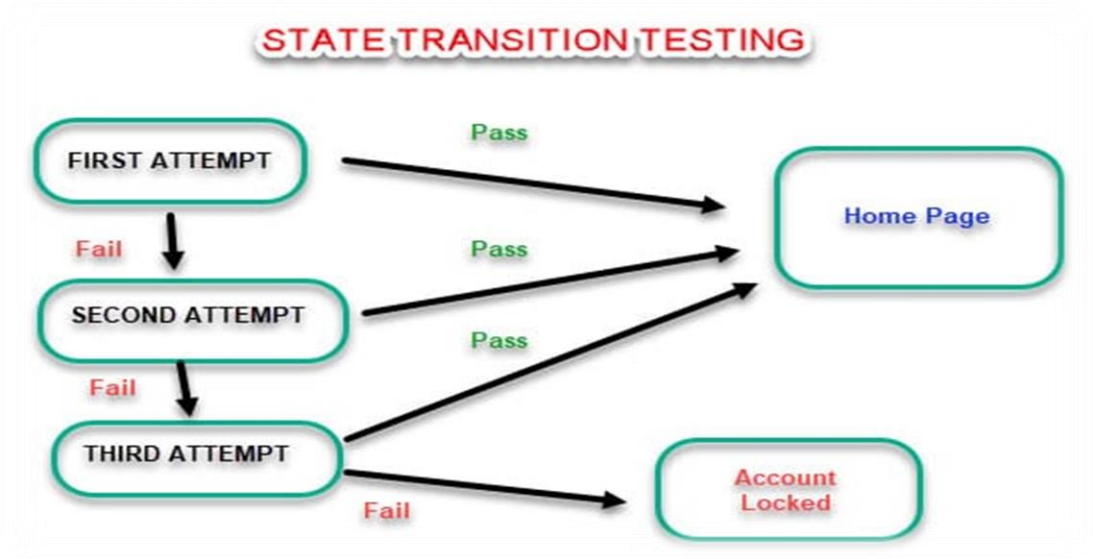
  When to Use State Transition?

  - This can be used when a tester is testing the application for a finite set of input values.
  - When the tester is trying to test sequence of events that occur in the application under test. i.e., this will allow the tester to test the application behavior for a sequence of input values.
  - When the system under test has a dependency on the events/values in the past.

  When to Not Rely On State Transition?

- When the testing is not done for sequential input combinations.
- If the testing is to be done for different functionalities like exploratory testing.

Example: State Transition Testing



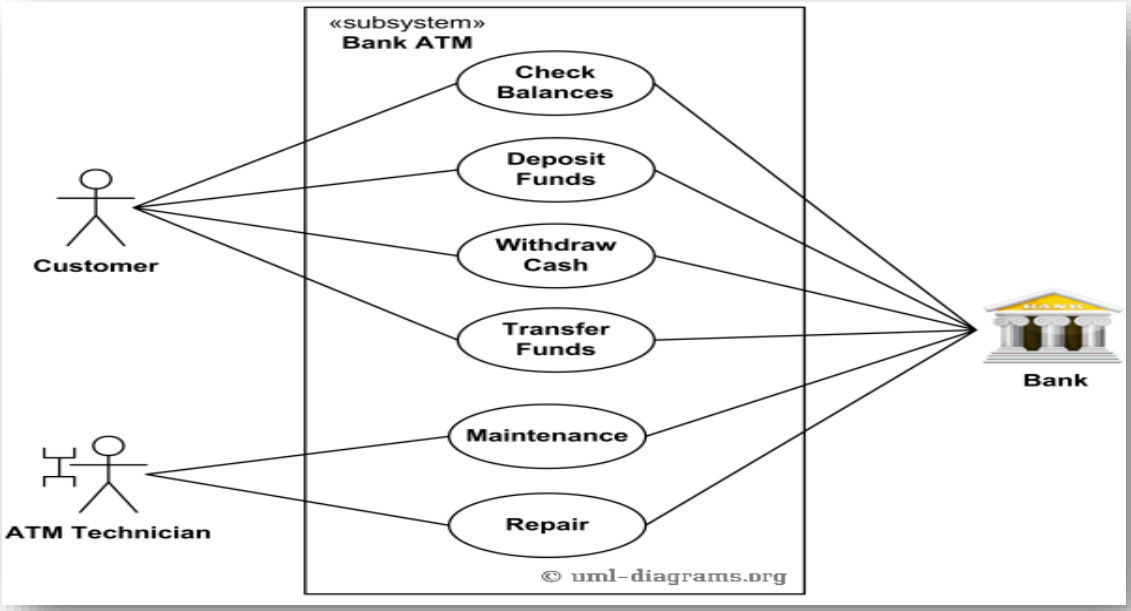
Source: softwaretestinghelp.com

Figure 3.3.3.3: State Transition Testing Diagram

- Use Case Testing:

  Use case testing is a functional testing technique, meaning programming skill is not required. It helps the tester determine which test scripts are executed on the entire system from the beginning to the end of each transaction.

  Example: Use Case Testing Diagram



Source: uml-diagrams.org

Figure 3.3.3.4: Use Case Testing Diagram

2. Structure-based or White-Box techniques

- Statement Coverage or Line Coverage:

  In this technique, every statement in the source code is executed at least once. Thereby, we can check what the source code is and is not expected to do. However, we cannot test the false condition in the source code. Statement coverage = (No. of statements Executed/Total no. of statements in the source code)*100

- Condition Coverage or Predicate Coverage:

  Condition coverage is seen for Boolean expression. Condition coverage ensures whether all the Boolean expressions have been covered and evaluated to both TRUE and FALSE.

- Decision Coverage or Branch Coverage:

  Test coverage criteria require enough test cases so that each condition in a decision takes on all possible outcomes at least once, and each point of entry to a program or subroutine is invoked at least once. That is, every branch (decision) is either true or false. It is helpful to invalidate all branches in the code to make sure that no branch leads to any abnormal behavior.

- Multiple Condition Coverage:

  Every combination of ‘true’ or ‘false’ for the conditions related to a decision has to be tested in this technique.

3. Experience-based technique

- Exploratory Testing:

  Usually, this process is carried out by domain experts. They perform testing just by exploring the functionalities of the application without having the knowledge of the requirements. Testers can explore and learn the system while using these techniques. High severity bugs are found very quickly in this type of testing.

- Error Guessing:

  Error guessing is one of the testing techniques used to find bugs in a software application based on the tester's prior experience. In Error guessing, no specific rules are applied.

Choosing the right technique

As mentioned, choosing the technique is the most decisive step. This step is dependent on these factors:

- Type of system or software application:

  Testing techniques are mainly determined based on requirements for the domain of the application. Moreover, these techniques are applied differently between mobile and web applications.

- Regulatory standards:

  It is evident that your selection of techniques must follow conventional rules of, and approved by the IT industry.

- Customer's requirements:

  Selection of techniques varies and is based on the customer requirements. If your customer does not provide any provision, you have to choose the experience-based approaches.

- Level and type of risk:

  Risks may include lack of requirement, equipment or anything related to quality assurance. Both high-level and low-level design techniques can be applied.

- Test objectives:

  Test objectives are important because it narrows down the scope of testing activities. Based on that, you can select the most suitable techniques for your project.

- Tester's skill and knowledge:

  This plays a very crucial role in the selection of techniques since it depends on the availability of test documents like requirement document, analysis report, design

document etc. The perception of testers about the application and experience on test execution help to figure out defects quickly, and make the product a quality one.

- Time and budget:

  Some projects are short-term, and some are long-term. Based on the project you need to choose techniques. You need to cleverly calculate the budget for each project. For small budgets, cost-effective approaches should be taken.

- Application development life cycle:

  The application development life cycle has different stages, parallel to testing stages. Different stages of development and testing require different techniques.

- Previous experience in types of defects tracked:

  This is a type of experience-based techniques on defects that you may encounter in the testing life cycle. What's more, you can replicate the same situation and catch the errors that met previously.

References

https://www.guru99.com/non-functional-testing.html

https://www.softwaretestinghelp.com/what-is-non-functional-testing/

https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/

https://www.invensis.net/blog/software-test-design-techniques-static-and-dynamic-testing/

https://testautomationresources.com/software-testing-basics/software-test-design-techniques/#:~:text=The%20importance%20of%20test%20design%20techniques&text=Although%20the%20main%20purpose%20is,based%20on%20various%20risk%20factors.

https://hackernoon.com/qa-engineering-roles-skills-tools-and-responsibilities-in-a-testing-team-7c499adc8057

https://www.lambdatest.com/blog/all-you-need-to-know-about-automation-testing-life-cycle/

https://www.esds.co.in/blog/manual-testing-process-lifecycle/#sthash.Lgd0Venn.tQQ2dwaH.dpbs