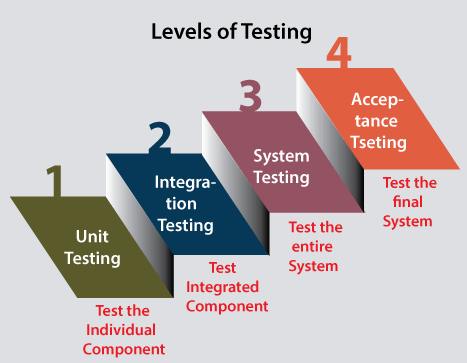
Software testing

Different Levels of Testing

The levels of software testing involve the different methodologies, which can be used while we are performing the software testing.

In [software testing](https://www.javatpoint.com/software-testing-tutorial), we have four different levels of testing, which are as discussed below:

1. **Unit Testing**
2. **Integration Testing**
3. **System Testing**
4. **Acceptance Testing**



As we can see in the above image that all of these testing levels have a specific objective which specifies the value to the software development lifecycle.

For our better understanding, let's see them one by one:

Level1: Unit Testing

**Unit testing** is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not.

The first level of testing involves **analyzing each unit or an individual component** of the software application.

Unit testing is also the first level of [**functional testing**](https://www.javatpoint.com/functional-testing). The primary purpose of executing unit testing is to validate unit components with their performance.

A unit component is an individual function or regulation of the application, or we can say that it is the smallest testable part of the software. The reason of performing the unit testing is to test the correctness of inaccessible code.

Unit testing will help the test engineer and developers in order to understand the base of code that makes them able to change defect causing code quickly. The developers implement the unit.

Level2: Integration Testing

The second level of software testing is the **integration testing.** The integration testing process comes after **unit testing**.

It is mainly used to test the **data flow from one module or component to other modules.**

In this testing,  **engineer** tests the units or separate components or modules of the software in a group.

The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units.

When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as **integration testing**.

We only go for the integration testing when the functional testing has been completed successfully on each application module.

In simple words, we can say that **integration testing** aims to evaluate the accuracy of communication among all the modules.

.

Level3: System Testing

The third level of software testing is **system testing**, which is used to test the software's functional and non-functional requirements.

It is **end-to-end testing** where the testing environment is parallel to the production environment. In the third level of software testing, **we will test the application as a whole system.**

To check the end-to-end flow of an application or the software as a user is known as **System testing**.

In system testing, we will go through all the necessary modules of an application and test if the end features or the end business works fine, and test the product as a complete system.

In simple words, we can say that System testing is a sequence of different types of tests to implement and examine the entire working of an integrated software computer system against requirements.

.

Level4: Acceptance Testing

The **last and fourth level** of software testing is **acceptance testing**, which is used to evaluate whether a specification or the requirements are met as per its delivery.

The software has passed through three testing levels (**Unit Testing, Integration Testing, System Testing**). Some minor errors can still be identified when the end-user uses the system in the actual scenario.

In simple words, we can say that Acceptance testing is the **squeezing of all the testing processes that are previously done.**

The acceptance testing is also known as **User acceptance testing (UAT)** and is done by the customer before accepting the final product.

Usually, UAT is done by the domain expert (customer) for their satisfaction and checks whether the application is working according to given business scenarios and real-time scenarios.

Principles of software testing

There are seven principle of software testing

1. Testing shows the presence of defects

The test engineer will test the application to make sure that the application is bug or defects free. While doing testing, we can only identify that the application or software has any errors.

1. Exhaustive Testing is not possible

Sometimes it seems to be very hard to test all the modules and their features with effective and non- effective combinations of the inputs data throughout the actual testing process.

Hence, instead of performing the exhaustive testing as it takes boundless determinations and most of the hard work is unsuccessful.

1. Early Testing

early testing means that all the testing activities should start in the early stages of the software development life cycle's **requirement analysis stage** to identify the defects because if we find the bugs at an early stage, it will be fixed in the initial stage itself,

1. Defect Clustering

defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules.

1. Pesticide Paradox

if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application

1. Testing is context-dependent

we have multiple fields such as e-commerce websites, commercial websites, and so on are available in the market. There is a definite way to test the commercial site as well as the e-commerce websites because every application has its own needs, features, and functionality.

1. Absence of error fallacy

The absence of error fallacy means identifying and fixing the bugs would not help if the application is impractical and not able to accomplish the client's requirements and needs.

White Box Testing

* white box testing is also known as glass box is testing, structural testing, clear box testing, open box testing and transparent box testing.
* It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs.
* It is based on inner workings of an application and revolves around internal structure testing.
* In this type of testing programming skills are required to design test cases.
* The goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.
* In this, the developer will test every line of the code of the program.

In white box testing there are five steps:

* Path testing:- we will write the flow and test all independent paths.
* Loop testing:-  we will test the loops such as while, for, and do-while, etc. and also check for ending condition if working correctly and if the size of the conditions is enough.
* Condition testing:- we will test all logical conditions for both **true** and **false** values; that is, we will verify for both **if** and**else** condition.
* Testing based on the memory perspective:- we have different programs of the same application, and the first 15 lines of the program are similar. We can write these 15 lines as a discrete function, and it should be accessible by the different programs as well. And also, if any bug is there, we can modify the line of code in the function rather than the entire code.
* Test performance of the program:- developer  understand that the code is running slow, or the performance of the program is also getting deliberate. And the developer cannot go manually over the program and verify which line of the code is slowing the program.

Black box testing

* Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.
* In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

steps of black box testing

* The black box test is based on the specification of requirements, so it is examined in the beginning.
* In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
* In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
* The fourth phase includes the execution of all test cases.
* In the fifth step, the tester compares the expected output against the actual output.
* In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

White box testing

Data flow :- checking the flow data.

e.g.

input(x)  
2. if(x>5)  
3.     z = x + 10  
4. else  
5.     z = x - 5  
6. print("Value of Z: ", z)

if we enter the value of x is greater than 5 then order should be 1 2 3 6

if we enter the value of x is less than equals to 5 then order should be 1 4 5 6

control flow

1

4

2

5

6

3

**Difference between validation and verification**

| ***Verification*** | ***Validation*** |
| --- | --- |
| 1. Verification is a static testing of verifying documents, design, code and program. | 1. Validation is a dynamic mechanism of validating and testing the actual product. |
| 2. It does not involve executing the code. | 2. It always involves executing the code. |
| 3. It is human based checking of documents and files. | 3. It is computer based execution of program. |
| 4 it is a step by step process | 4. it checking whole process |
| 5. Verification is to check whether the software conforms to specifications. | 5. Validation is to check whether software meets the customer expectations and requirements. |
| 6 It is low level exercise. | 6. . It is High Level Exercise. |
| 7. are we building product right? | 7. are we building right product |
| 8. Verification is done by QA | 8. Validation is carried out with the involvement of testing team. |
| 9. It generally comes first-done before validation. | 9. It generally follows |

**Differences between Black Box Testing vs White Box Testing:**

| S. No. | Black Box Testing | White Box Testing |
| --- | --- | --- |
| 1. | It check external behaviour of the software | It check the internal code. |
| 2. | Implementation of code is not needed for black box testing. | Code implementation is necessary for white box testing. |
| 3. | It is mostly done by software testers. | It is mostly done by software developers. |
| 4. | No knowledge of implementation is needed. | Knowledge of implementation is required. |

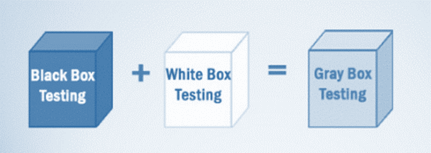
|  |  |  |
| --- | --- | --- |
| 5. | No knowledge of programming is required. | It is mandatory to have knowledge of programming. |
| 6. | It is a functional test of the software. | It is also called as clear box testing. |

**Gray box testing**

It is a combination of white box and black box testing

 On the one hand, tests are performed from the user’s perspective. On the other hand, testers do use some inside information to focus on the most important issues and identify the weaknesses of the system.

With gray box testing, you can ensure that applications work as expected for authenticated users. You can also verify that malicious users cannot access data or functionality you don’t want them too



**Step of gray box testing**

* Identify and select Inputs from white and black box testing methods.
* Identify probable outputs from these inputs.
* Identify key paths for the testing phase.
* Identify sub-functions for deep-level testing.
* Identify inputs for sub-functions.
* Identify probable outputs from sub-functions.
* Execute sub-function test cases.
* Assess and verify outcomes.
* Repeat steps 4-8.
* Repeat steps 7 and 8.

**Matrix Testing**

it is a technique that examines all variables in an application. In this technique, technical and business risks are defined by the developers and a list of all application variables are provided. Each variable is then assessed according to the risks it presents. You can use this technique to identify unused or un-optimized variables.Matrix technique is a method to remove unused and uninitialized variables by identifying used variables from the program.

**Orthogonal Array Testing**

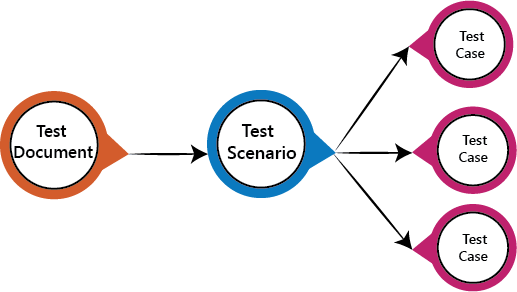
Orthogonal array testing is a technique you can use when your application has only a few inputs that are too complex or large for extensive testing. This technique enables you to perform test case optimization, where the quality and number of tests performed balance test coverage with effort. This technique is systematic and uses statistics to test pair-based interactions.

**Pattern Testing**

Pattern testing is a technique that evaluates past defects to identify patterns that lead to defects. Ideally, these evaluations can highlight which details contributed to defects, how the defects were found, and how effective fixes were. You can then apply this information to identifying and preventing similar defects in new versions of an application or new applications with similar structures.

**Test scenario**

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.



**Test basis**

Test basis is defined as the source of information or the document that is needed to write test cases and also for test analysis.

Test basis should be well defined and adequately structured so that one can easily identify test conditions from which test cases can be derived.

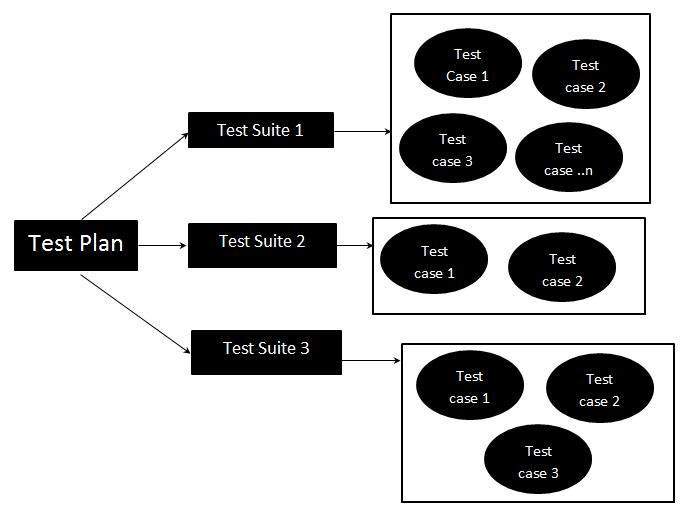
**Test case**

A **Test Case** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

**Test suite**

Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status. It can take any of the three states namely Active, Inprogress and completed.

A Test case can be added to multiple test suites and test plans. After creating a test plan, test suites are created which in turn can have any number of tests.



**Functional testings**

**Exploratory testing**

Exploratory testing is a type of software testing where test cases are not created but tester check the system.

Tester note down ideas about what to test before test execution.

The focus of exploratory testing is more on testing as a thinking activity.

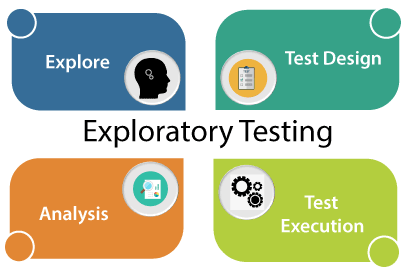
It is directed from requirements and exploring during testing

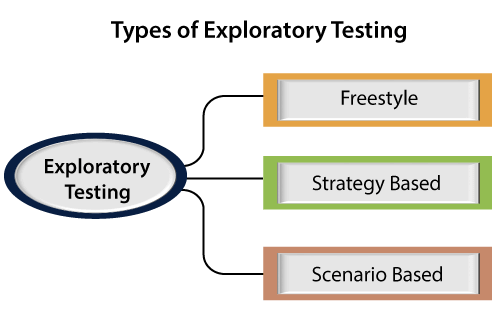
Exploratory testing is Investigation of system or application

It is about Improvement of test design

if requirement does not exist, then we do of exploratory testing.

In exploratory testing, we will be exploring the application in all possible ways, understanding the flow of the application, preparing a test document and then testing the application, this approach is known as exploratory testing.





* **Freestyle : i**n freestyle testing, we did not follow any rules,
* **Strategy based:** Strategy based exploratory testing can be performed with the help of multiple testing techniques such as risk-based, boundary value analysis, and equivalence partitioning. It is done by experience tester.
* **Scenario-based:-** Scenario-based exploratory testing is performed with the help of multiple scenarios such as end-to-end, [test scenarios](https://www.javatpoint.com/test-scenario), and real user scenarios.

**Sanity testing :-**

Sanity testing is performed on stable builds and it is also known as a variant of regression testing.Sanity testing was performed when we are receiving software build (with minor code changes) from the development team. It is a checkpoint to assess if testing for the build can proceed or not.

In other words, we can say that sanity testing is performed to make sure that all the defects have been solved and no added issues come into the presence because of these modifications**.**

**Gorilla testing:**

Gorilla testing is a software testing technique that repeatedly applies inputs on a module to ensure it is functioning correctly and that there are no bugs.

**Monkey Testing** is a software testing technique in which the tester enters any random inputs into the software application without predefined test cases and checks the behavior of the software application, whether it crashes or not. The purpose of Monkey testing is to find the bugs and errors in the software application using experimental techniques

**Smoke Testing** is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as “Build Verification Testing” or “Confidence Testing.”

**Mutation Testing** is a type of software testing in which certain statements of the source code are changed/mutated to check if the test cases are able to find errors in source code. The goal of Mutation Testing is ensuring the quality of test cases in terms of robustness that it should fail the mutated source code.

**Non-functional testing**

**Load Testing** is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.

**Stress testing** (sometimes called torture testing) is a form of deliberately intense or thorough testing used to determine the stability of a given system, critical infrastructure or entity. Stress testing involves testing the application under varying load. Extremely large numbers of concurrent users try to log into the application. Database linked to the website shuts down when the website tries to reach it from the front end. Data in added in extremely large quantity in the database.Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn’t crash under crunch situations.

**Usability testing**

to check the usability or ease of using a software product. Checking the user-friendliness, efficiency, and accuracy of the application is known as Usability Testing.

Parameters are Efficiency

Memorability

Accuracy

Learnability

Satisfaction

Errors

**Benchmark performance**

A Benchmark in Performance Testing is a metric or a point of reference against which software products or services can be compared to assess the quality measures. In other words, Benchmark means a set standard that helps to determine the quality of a software product or service.

**Performance testing** is in general a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Types of Performance Testing:

• Load

• Stress

• Spike

• Endurance

• Scalability

**Volume Testing** is a type of software testing which is carried out to test a software application with a certain amount of data. . In volume testing a huge volume of data is acted upon the software. It is basically performed to analyze the performance of the system by increasing the volume of data in the database. Volume testing is performed to study the impact on response time and behavior of the system when the volume of data is increased in the database.

**scalability testing** which comes under the non-functional testing of software testing.It is used to check an application's performance by increasing or decreasing the load in particular scales known as scalability testing. It is executed at a hardware, software, or database level

**security testing**

types of securityVulnerability Scanning. ...

Security Scanning. ...

Penetration Testing. ...

Security Audit/ Review. ...

Ethical Hacking. ...

Risk Assessment. ...

Posture Assessment. ...

Authentication.