**Testing :**

Testing is the process of identifying defect ,where a defect is any variance between actual and expected result.

Mistake in coding is called error

Error found by tester called defect

Defect accepted by development team then it is called bug.

Build does not meet the requirement then it is failure.

#### ****BUG:****

A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. A bug is the terminology of Tester.

#### ****ERROR:****

An error is a mistake, misconception, or misunderstanding on the part of a software developer. In the category of the developer, we include software engineers, programmers, analysts, and testers. For example, a developer may misunderstand a de-sign notation, or a programmer might type a variable name incorrectly – leads to an Error. It is the one that is generated because of the wrong login, loop or syntax. The error normally arises in software; it leads to a change in the functionality of the program.

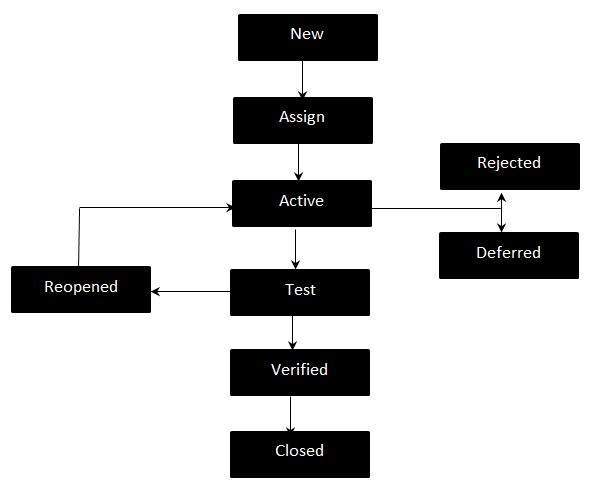
#### ****FAILURE:****

A failure is the inability of a software system or component to perform its required functions within specified performance requirements. When a defect reaches the end customer it is called a Failure. During development, Failures are usually observed by testers.

#### ****FAULT:****

An incorrect step, process or data definition in a computer program that causes the program to perform in an unintended or unanticipated manner. A fault is introduced into the software as the result of an error. It is an anomaly in the software that may cause it to behave incorrectly, and not according to its specification. It is the result of the error.

**BUG life cyle / workflow**



**New -**Potential defect that is raised and yet to be validated.

**Assigned -**Assigned against a development team to address it but not yet resolved.

**Active -**The Defect is being addressed by the developer and investigation is under progress. At this stage there are two possible outcomes; viz - Deferred or Rejected.

**Test -**The Defect is fixed and ready for testing.

**Verified -**The Defect that is retested and the test has been verified by QA.

**Closed -**The final state of the defect that can be closed after the QA retesting or can be closed if the defect is duplicate or considered as NOT a defect.

**Reopened -**When the defect is NOT fixed, QA reopens/reactivates the defect.

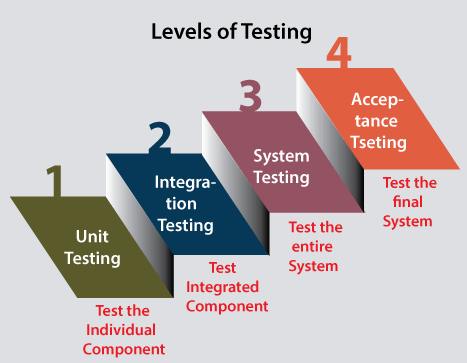
**Deferred -**When a defect cannot be addressed in that particular cycle it is deferred to future release.

**Rejected -**A defect can be rejected for any of the 3 reasons; viz - duplicate defect, NOT a Defect, Non Reproducible.

**Levels of 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**



**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.

**Integration testing**

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

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

**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.**

**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.**

**Levels of testing**

**Testing shows the presence of defects :**

The goal of software testing is to make the software fail. Software testing reduces the presence of defects.

**Exhaustive testing is impossible**:

Exhaustive testing is impossible means the software can never test at every test case. It can test only some test cases and assume that the software is correct and it will produce the correct output in every test case. If the software will test every test case then it will take more cost, effort, etc., which is impractical.

**Early testing:** testing should be start at early phase ,for better performance

**Defect clustering:** In a project, a small number of modules can contain most of the defects. Pareto Principle to software testing state that 80% of software defect comes from 20% of modules.

**Pesticide paradox:** Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.

**Testing is context-dependent:** The testing approach depends on the context of the software developed. Different types of software need to perform different types of testing.

**Absence of errors fallacy:** If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it is also mandatory to fulfill all the customer requirement.

**White box testing :**

The **clear box or white box, or transparent box** name denotes the ability to see through the software's outer shell into its inner workings.

It is performed by Developers,

the software will be sent to the testing team, where they perform black-box testing.

The main objective of white-box testing is to test the application's infrastructure. It is done at lower levels, as it includes unit testing and integration testing.

It requires programming knowledge, as it majorly focuses on code structure, paths, conditions, and branches of a program or software.

The primary goal of white-box testing is to focus on the flow of inputs and outputs

Types;

**Path testing, Loop testing,**and **Condition testing**.

**Black box testing :**

The primary source of black-box testing is a specification of requirements that are stated by the customer.

It is a software testing technique that examines the functionality of the software without knowing its internal structure or coding.

It does not require programming knowledge of the software.

All test cases are designed by considering the input and output of a particular function.

In this testing, the test engineer analyzes the software against requirements, identifies the defects or bugs, and sends it back to the development team.

black box testing is a process of checking the functionality of an application as per the customer's requirement.

Mainly, there are three types of black-box testing**: functional testing, Non-Functional testing,**and **Regression testing**.

**Difference between black box and white box testing**

|  |  |
| --- | --- |
| **Black Box Testing** | **White Box Testing** |
| The Black Box Test is a test that only considers the external behavior of the system; the internal workings of the software is not taken into account. | The White Box Test is a method used to test a software taking into consideration its internal functioning. |
| It is carried out by testers. | It is carried out by software developers. |
| This method is used in [System Testing](https://www.practitest.com/qa-learningcenter/resources/system-testing-vs-integration-testing/) or [Acceptance Testing](https://www.practitest.com/qa-learningcenter/best-practices/what-is-uat-testing/). | This method is used in [Unit Testing or Integration Testing](https://www.practitest.com/qa-learningcenter/resources/unit-test-vs-integration-test/). |
| It is the least time consuming. | It is most time consuming. |
| It is the behavior testing of the software. | It is the logic testing of the software. |
| It is also known as data-driven testing, [functional testing](https://www.practitest.com/qa-learningcenter/resources/what-is-functional-testing/), and closed box testing. | It is also known as clear box testing, code-based testing, structural testing, and transparent testing. |
| **Verification and validation** |  |

|  |  |
| --- | --- |
| **Verification** | **Validation** |
| Verification is the process of evaluating the artifacts (architecture,design) of software development in order to ensure that the product being developed will comply to the standards. | Validation is the process of validating that the developed software product conforms to the specified business requirements. |
| It is a static process of analysing the documents and not the actual end product. | It involves dynamic testing of a software product by running it. |
| Verification is a process oriented approach. | Validation is a product-oriented approach. |
| Answers the question – “Are we building the product right?” | Answers the question – “Are we building the right product?” |
| Errors found during verification require lesser cost/resources to get fixed as compared to be found during the validation phase. | Errors found during validation require more cost/resources. Later the error is discovered higher is the cost to fix it. |
| It involves activities like document review, test case review, walk-throughs, inspection etc. | It involves activities like functional testing, automation testing e |

**Grey box testing**

Grey box testing is a software testing technique to test a software product or application with partial knowledge of internal structure of the application. The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.

it is a combination of both white box testing and Black Box Testing method.

* In White Box testing internal structure (code) is known
* In Black Box testing internal structure (code) is unknown
* In Grey Box Testing internal structure (code) is partially known

While testing websites feature like links or orphan links, if tester encounters any problem with these links, then he can make the changes straightaway in HTML code and can check in real time.

### **Steps to perform Grey box Testing are:**

1. First, select and identify inputs from BlackBox and WhiteBox testing inputs.
2. Second, Identify expected outputs from these selected inputs.
3. Third, identify all the major paths to traverse through during the testing period.
4. The fourth task is to identify sub-functions which are the part of main functions to perform deep level testing.
5. The fifth task is to identify inputs for subfunctions.
6. The sixth task is to identify expected outputs for subfunctions.
7. The seventh task includes executing a test case for Subfunctions.
8. The eighth task includes verification of the correctness of result.

**Techniques**

**Matrix:**

It defines all the used variables of a particular program.

In any program, variable are the elements through which values can travel inside the program. It should be as per requirement otherwise, it will reduce the readability of the program and speed of the software.

Matrix technique is a method to remove unused and uninitialized variables by identifying used variables from the program.

**Orthogonal array testing (OAT) :**

The purpose of this testing is to cover maximum code with minimum test cases. Test cases are designed in a way that can cover maximum code as well as GUI functions with a smaller number of test cases.

**Pattern :**

Pattern testing is applicable to such type of software that is developed by following the same pattern of previous software. In these type of software possibility to occur the same type of defects. Pattern testing determines reasons of the failure so they can be fixed in the next software.

Test case:

test case is a detailed document, which provides information about the **testing strategy, testing process, preconditions, and expected output**.

It includes all the **positive and negative inputs, navigation steps, Expected results, pre and post condition, etc**.

It includes all the **positive and negative inputs, navigation steps, Expected results, pre and post condition, etc**.

The test case is work on the basics of **"How to be tested"**.

Test scenario :

The test scenarios are those derived from the **use case** and give **the one-line information about what to test**.

Test scenarios are one-liner statement, but it is connected with several test cases.

Writing the test scenario's main objective is a **cover end to end functionality of a software application**.

The test scenarios are work on the basic to **"What to be tested"**.

write **Test Scenarios** first and then write the **Test Case**

**Test suite**

A **TEST SUITE** is a collection of [test cases](https://softwaretestingfundamentals.com/test-case/). In automated testing, it can mean a collection of [test scripts](https://softwaretestingfundamentals.com/test-script/). In a test suite, the test cases / scripts are organized in a logical order. For example, the test case for registration will precede the test case for login.

An example of a test suite for purchasing a product could comprise of the following test cases:

* Test Case 1: Login
* Test Case 2: Add Products
* Test Case 3: Checkout
* Test Case 4: Logout

Test basis

Test Basis can also be defined as that data which is needed in order to start the analysis of the test. Usually, it will contain information like the business requirement, [**test plan**](https://www.professionalqa.com/test-plan), code specifications, interfaces etc. Hence, one can conclude that all relevant information that goes into the planning of test cases and its analysis is known as Test Basis.

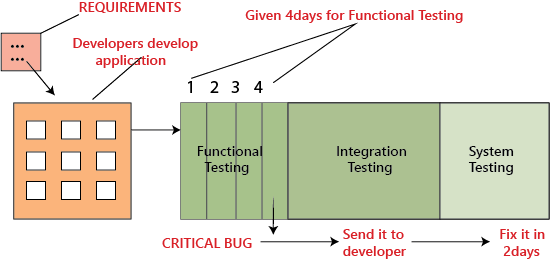
**SMOKE TESTING**

“Testing the basic & critical feature of an application before doing one round of deep, rigorous testing (before checking all possible positive and negative values) is known as smoke testing.”

Smoke Testing comes into the picture at the time of receiving build software from the development team.

The purpose of smoke testing is to determine whether the build software is testable or not. It is done at the time of "building software." This process is also known as "Day 0".

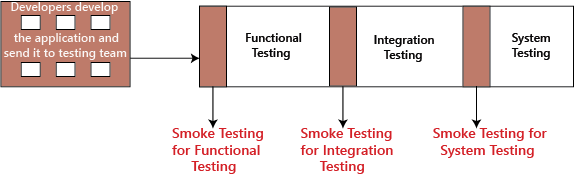
In the smoke testing, we only focus on the positive flow of the application and enter only valid data, not the invalid data. In smoke testing, we verify every build is testable or not; hence it is also known as **Build Verification Testing.**

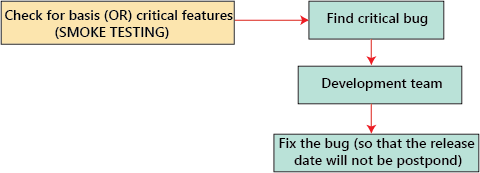


To overcome this problem, we perform **smoke testing**

, instead of the testing module by module thoroughly and come up with critical bug at the end,

it is better to do **smoke testing** before we go for functional, integration and system testing that is, in each module we have to test for essential or critical features, and then proceed for further testing





Quality assurance ([QA](https://www.techtarget.com/searchsoftwarequality/definition/quality-assurance)) testers perform smoke testing after the developers deliver every new build of an application.

**Scalability testing**

**scalability testing**, which comes under the [non-functional testing](https://www.javatpoint.com/non-functional-testing) of [software testing](https://www.javatpoint.com/software-testing-tutorial).

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**

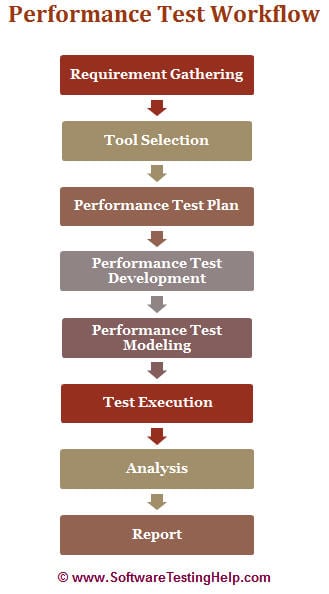
The main objective of performing the scalability testing is to control how the application balances with an increasing workload and at what point the software product or the system stops scaling and identify the reason behind it.

It is specified as the capacity of a **network, system, application, product, or process** to make the function correctly when modifications are made in the system's size or volume to meet an increasing need.

In this testing, the [**Test Cases**](https://www.javatpoint.com/test-case) are designed and implemented in a well-organized manner. It also analysis the **system, processes, or database's ability** to meet an upward need.

**For example**, a **web page** scalability testing depends on the number of users, CPU usage, network usage. In contrast, scalability testing of a **web server depends on the number of requests processed**

**Performance testing workflow**



**Requirement Analysis/Gathering**

The performance team interacts with the client for identification and gathering of requirements – technical and business. This includes getting information on the application’s architecture, technologies, and database used, intended users, functionality, application usage, [test requirement](https://www.softwaretestinghelp.com/how-to-test-software-requirements-specification-srs/), hardware & software requirements

.

**POC/Tool selection**

Once the key functionality is identified, POC (Proof Of Concept – which is a sort of demonstration of the real-time activity but in a limited sense) is done with the available tools.

**Performance Test Plan & Design**

Depending on the information collected in the preceding stages, test planning and designing are conducted.

Test Planning involves information on how the performance test is going to take place – test environment, workload, hardware, etc.

**Performance Test Development**

* Use cases are created for the functionality identified in the test plan as the scope of PT.
* These use cases are shared with the client for their approval. This is to make sure the script will be recorded with the correct steps.
* Once approved, script development starts with a recording of the steps in use cases with the performance test tool selected during the POC (Proof of Concepts) and enhanced by performing Correlation (for handling dynamic value), Parameterization (value substitution) and custom functions as per the situation or need. More on these techniques in our video tutorials.
* The Scripts are then validated against different users.
* Parallel to script creation, the performance team also keeps working on setting up the test environment (Software and hardware).
* The performance team will also take care of Metadata (back-end) through scripts if this activity is not taken up by the client.

**Performance Test Modeling**

Performance Load Model is created for the test execution. The main aim of this step is to validate whether the given Performance metrics (provided by clients) are achieved during the test or not. There are different approaches to create a Load model. “[Little’s Law](http://www.factoryphysics.com/principle/littleslaw.htm)” is used in most cases.

**Test Execution**

The scenario is designed according to the Load Model in Controller or Performance Center but the initial tests are not executed with maximum users that are in the Load model.

Test Execution is done incrementally. **For Example,** If the maximum number of users is 100, the scenarios are first run with 10, 25, 50 users and so on, eventually moving on to 100 users.

**Test Results Analysis**

Test results are the most important deliverable for the performance tester. This is where we can prove the ROI (Return on Investment) and productivity that a performance testing effort can provide.

**Report**

Test results should be simplified so the conclusion is clearer and should not need any derivation. Development Team needs more information on analysis, comparison of results, and details of how the results were obtained.

|  |
| --- |
|  |

Services