**Module -2 (Manual testing)**

**What is exploratory testing ?**

Exploratory Testing is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a “thinking” activity.

Exploratory Testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

**What is traceability matrix ?**

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.

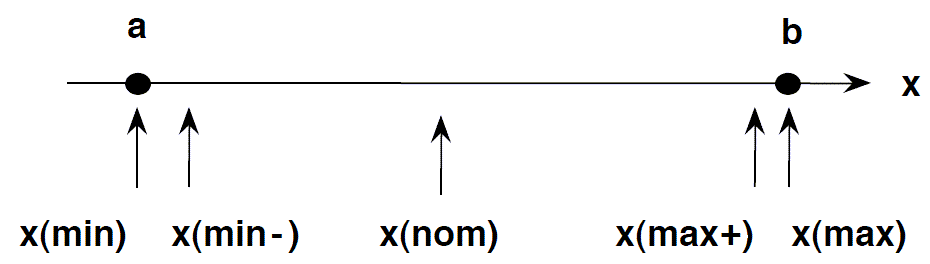
It is used to track the requirements and to check the current project requirements are met.

**What is boundary value testing ?**

Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.

* So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called “boundary testing”.
* The basic idea in normal boundary value testing is to select input variable values at their:

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum



* In Boundary Testing, Equivalence Class Partitioning plays a good role
* Boundary Testing comes after the Equivalence Class Partitioning.

**What is equivalence partitioning testing ?**

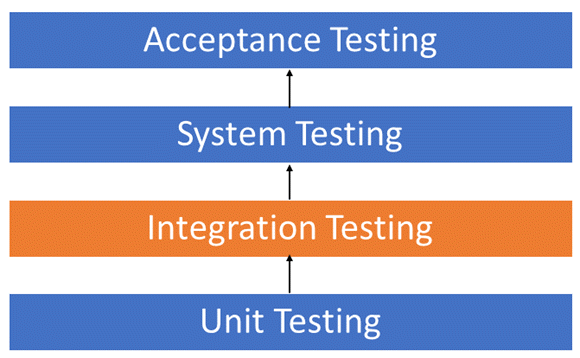
Equivalence Partitioning or Equivalence Class Partitioning is type of black box testing technique which can be applied to all levels of software testing like unit, integration, system, etc. In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduces time required for testing because of small number of test cases.

* It divides the input data of software into different equivalence data classes.
* You can apply this technique, where there is a range in the input field.

**What is integration testing ?**

Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as ‘I & T’ (Integration and Testing), ‘String Testing’ and sometimes ‘Thread Testing’.



Although each software module is unit tested, defects still exist for various reasons like

* A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity
* At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
* Interfaces of the software modules with the database could be erroneous
* External Hardware interfaces, if any, could be erroneous
* Inadequate exception handling could cause issues.

**What determines the level of risk ?**

The likelihood of an adverse event and the impact of the event.

**What is alpha testing ?**

Alpha Testing is a type of software testing performed to identify bugs before releasing the software product to the real users or public. It is a type of acceptance testing. The main objective of alpha testing is to refine the software product by finding and fixing the bugs that were not discovered through previous tests.

This testing is referred to as an alpha testing only because it is done early on, near the end of the development of the software, and before Beta Testing. Check Differences between Alpha testing and Beta testing

Alpha testing is typically performed by in-house software engineers or QA staff. It is the final testing stage before the software is released into the real world.

**What is beta testing ?**

Beta Testing is performed by “real users” of the software application in “real environment” and it can be considered as a form of external User Acceptance Testing. It is the final test before shipping a product to the customers. Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer’s environment.

Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

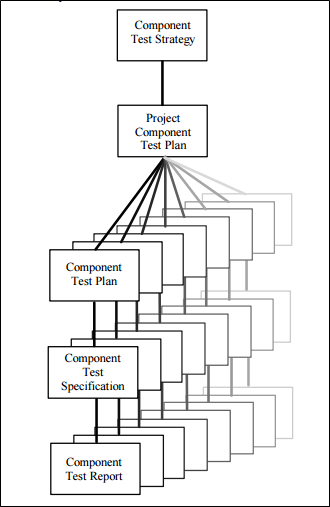
**What is Component Testing ?**

Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components. It’s also referred to as Module Testing when it is viewed from an architecture perspective. Component Testing is also referred to as Unit Testing, Program Testing or Module Testing.

Generally, any software as a whole is made of several components. Component Level Testing deals with testing these components individually.

It’s one of most frequent black box testing types which is performed by QA Team.

As per the below diagram, there will be a test strategy and test plan for component testing. Where each and every part of the software or application is considered individually. For each of this component a Test Scenario will be defined, which will be further brought down into a High Level Test Cases -> Low Level detailed Test Cases with Prerequisites.



**What is functional system testing?**

FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

**What is Non-Functional testing ?**

Non-Functional Testing is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

An excellent example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

**What is GUI testing ?**

Graphical User Interface Testing (GUI) Testing is the process for ensuring proper functionality of the graphical user interface (GUI) for a specific application.GUI testing generally evaluates a design of elements such as layout, colors and also fonts, font sizes, labels, text boxes, text formatting, captions, buttons, lists, icons, links, and content. GUI testing processes may be either manual or automatic and are often performed by third-party companies, rather than developers or end users.

**What is Adhoc testing ?**

Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.



Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the Software testing technique called Error Guessing. Error guessing can be done by the people having enough experience on the system to “guess” the most likely source of errors.

This testing requires no documentation/ planning /process to be followed. Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. This means that, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

**What is Load 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.



This testing usually identifies –

* The maximum operating capacity of an application
* Determine whether the current infrastructure is sufficient to run the application
* Sustainability of application with respect to peak user load
* Number of concurrent users that an application can support, and scalability to allow more users to access it.

**What is Stress testing ?**

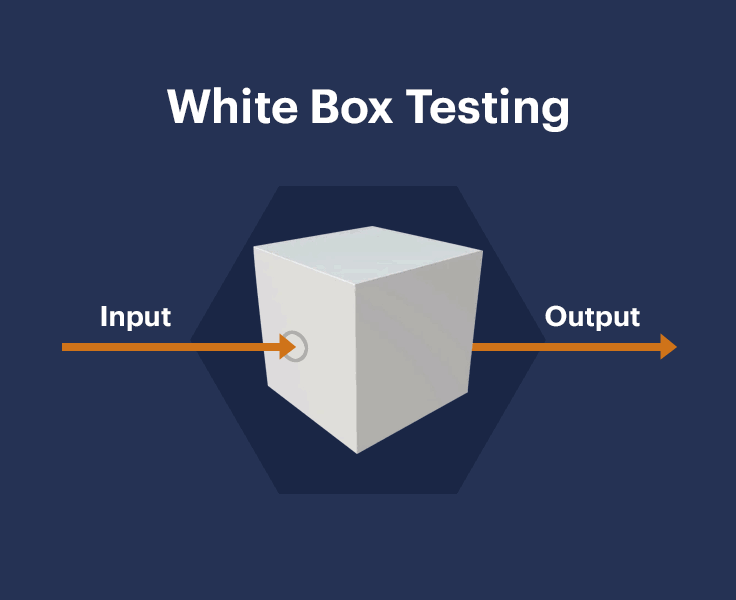
Stress Testing is a type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn’t crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.



In Software Engineering, Stress Testing is also known as Endurance Testing. Under Stress Testing, AUT is be stressed for a short period of time to know its withstanding capacity. A most prominent use of stress testing is to determine the limit, at which the system or software or hardware breaks. It also checks whether the system demonstrates effective error management under extreme conditions.

**What is white box testing and list the types of white box testing?**

White Box Testing is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.



It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term “WhiteBox” was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings.

**7 Different types of white-box testing -**

1. **Unit Testing**
2. **Static Analysis**
3. **Dynamic Analysis**
4. **Statement Coverage**
5. **Branch testing Coverage**
6. **Security Testing**
7. **Mutation Testing**

**Unit Testing**

Unit Testing is one of the basic steps, which is performed in the early stages. Most of the testers prefer performing to check if a specific unit of code is functional or not. Unit Testing is one of the common steps performed for every activity because it helps in removing basic and simple errors.

**Static Analysis**

As the term says, the step involves testing some of the static elements in the code. The step is conducted to figure out any of the possible defects or errors in the application code.

The static analysis is an important step because it helps in filtering simple errors in the initial stage of the process.

**Dynamic Analysis**

Dynamic Analysis is the further step of static analysis in general path testing. Most of the people prefer performing both static and dynamic at the same time.

The dynamic analysis helps in analyzing and executing the source code depending on the requirements. The final stage of the step helps in analyzing the output without affecting the process.

**Statement Coverage**

Statement coverage is one of the pivotal steps involved in the testing process. It offers a whole lot of advantages in terms of execution from time to time.

The process takes place to check whether all the functionalities are working or not. Most of the testers use the step because it is designed to execute all the functions atleast once. As the process starts, we will be able to figure out the possible errors in the web application.

**Branch Testing Coverage**

The modern-day software and web applications are not coded in a continuous mode because of various reasons. It is necessary to branch out at some point in time because it helps in segregating effectively.

Branch coverage testing gives a wide room for testers to find quick results. It helps in verifying all the possible branches in terms of lines of code. The step offers better access to find and rectify any kind of abnormal behavior in the application easily.

**Security Testing**

It is a known fact that security is one of the primary protocol, which needs to be in place all the time. Most of the companies prefer having a regular security testing activity because of obvious reasons. It is essential to have a process in place to protect the application or software automatically.

Security testing is more like a process because it comes with a lot of internal steps to complete. It verifies and rectifies any kind of unauthorized access to the system. The process helps in avoiding any kind of breach because of hacking or cracking practices.

Security testing requires a set of techniques, which deal with a sophisticated testing environment.

**Mutation Testing**

The last step in the process and requires a lot of time to complete effectively. Mutation testing is generally conducted to re-check any kind of bugs in the system.

The step is carried out to ensure using the right strategy because of various reasons. It gives enough information about the strategy or a code to enhance the system from time to time.

**What is black box testing? What are the different black box testing techniques?**

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users “don’t care” how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.



**Black Box Testing Techniques -**

**Equivalence Partitioning**

Testers can divide possible inputs into groups or “partitions”, and test only one example input from each group. For example, if a system requires a user’s birth date and provides the same response for all users under the age of 18, and a different response for users over 18, it is sufficient for testers to check one birth date in the “under 18” group and one date in the “over 18” group.

**Boundary Value Analysis**

Testers can identify that a system has a special response around a specific boundary value. For example, a specific field may accept only values between 0 and 99. Testers can focus on the boundary values (-1, 0, 99 and 100), to see if the system is accepting and rejecting inputs correctly.

**Decision Table Testing**

Many systems provide outputs based on a set of conditions. Testers can then identify “rules” which are a combination of conditions, identify the outcome of each rule, and design a test case for each rule.

**State Transition Testing**

In some systems, significant responses are generated when the system transitions from one state to another. A common example is a login mechanism which allows users to authenticate, but after a specific number of login attempts, transition to a different state, locking the account.

If testers identify a state transition mechanism, they can design test cases that probe the system when it transitions states. For example, for a system that locks the account after five failed login attempts, a test case can check what happens at the sixth login attempt.

**Error Guessing**

This technique involves testing for common mistakes developers make when building similar systems. For example, testers can check if the developer handled null values in a field, text in a numeric field or numbers in a text-only field, and sanitization of inputs—whether it is possible to submit user inputs that contain executable code, which has security significance.

A specific type of error guessing is testing for known software vulnerabilities that can affect the system under test.

**Mention what are the categories of defects?**

Following are some of the basic types of defects in the software development:

**Arithmetic Defects:**

It include the defects made by the developer in some arithmetic expression or mistake in finding solution of such arithmetic expression. This type of defects are basically made by the programmer due to access work or less knowledge. Code congestion may also lead to the arithmetic defects as programmer is unable to properly watch the written code.

**Logical Defects:**

Logical defects are mistakes done regarding the implementation of the code. When the programmer doesn’t understand the problem clearly or thinks in a wrong way then such types of defects happen. Also while implementing the code if the programmer doesn’t take care of the corner cases then logical defects happen. It is basically related to the core of the software.

**Syntax Defects:**

Syntax defects means mistake in the writing style of the code. It also focuses on the small mistake made by developer while writing the code. Often the developers do the syntax defects as there might be some small symbols escaped. For example, while writing a code in C++ there is possibility that a semicolon(;) is escaped.

**Multithreading Defects:**

Multithreading means running or executing the multiple tasks at the same time. Hence in multithreading process there is possibility of the complex debugging. In multithreading processes sometimes there is condition of the deadlock and the starvation is created that may lead to system’s failure.

**Interface Defects:**

Interface defects means the defects in the interaction of the software and the users. The system may suffer different kinds of the interface testing in the forms of the complicated interface, unclear interface or the platform based interface.

**Performance Defects:**

Performance defects are the defects when the system or the software application is unable to meet the desired and the expected results. When the system or the software application doesn’t fulfill the users’s requirements then that is the performance defects. It also includes the response of the system with the varying load on the system.

**Mention what bigbang testing is?**

Big Bang Testing is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a unit. This combined set of components is considered as an entity while testing. If all of the components in the unit are not completed, the integration process will not execute.

**Advantages:**

* Convenient for small systems.

**Disadvantages:**

* Fault Localization is difficult.
* Given the sheer number of interfaces that need to be tested in this approach, some interfaces link to be tested could be missed easily.
* Since the Integration testing can commence only after “all” the modules are designed, the testing team will have less time for execution in the testing phase.
* Since all modules are tested at once, high-risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

**What is the purpose of exit criteria?**

Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.

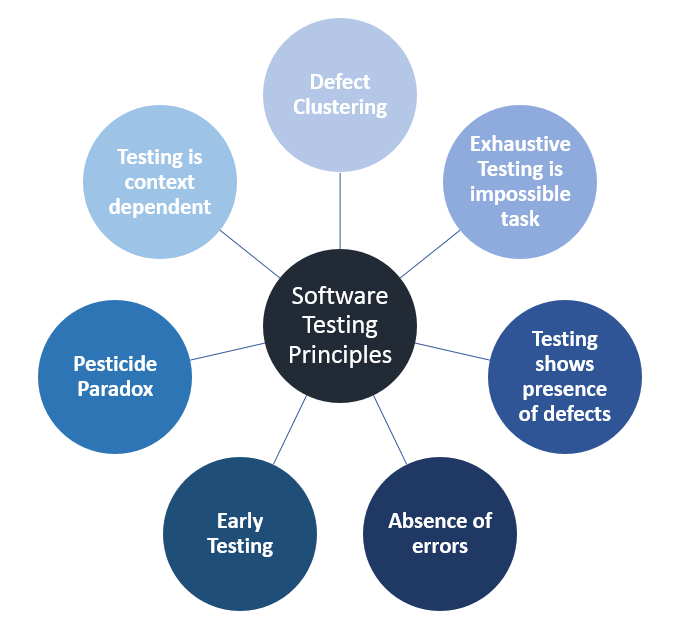
Exit criterion should be part of test plan and decided in the planning stage.

**When should "Regression Testing" be performed?**

Regression testing can be performed on a new build when there is a significant change in the original functionality. It ensures that the code still works even when the changes are occurring. Regression means Re-test those parts of the application, which are unchanged.

**What is 7 key principles? Explain in detail ?**

7 Principles of Software Testing



**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. The primary purpose of doing testing is to identify the numbers of unknown bugs with the help of various methods and testing techniques because the entire test should be traceable to the customer requirement, which means that to find any defects that might cause the product failure to meet the client's needs.

By doing testing on any application, we can decrease the number of bugs, which does not mean that the application is defect-free because sometimes the software seems to be bug-free while performing multiple types of testing on it. But at the time of deployment in the production server, if the end-user encounters those bugs which are not found in the testing process.

**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. So we can complete this type of variations according to the importance of the modules because the product timelines will not permit us to perform such type of testing scenarios.

**Early Testing**

Here 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, which may cost us very less as compared to those which are identified in the future phase of the testing process.

To perform testing, we will require the requirement specification documents; therefore, if the requirements are defined incorrectly, then it can be fixed directly rather than fixing them in another stage, which could be the development phase.

**Defect clustering**

The defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules. We have various reasons for this, such as the modules could be complicated; the coding part may be complex, and so on.

These types of software or the application will follow the Pareto Principle, which states that we can identify that approx. Eighty percent of the complication is present in 20 percent of the modules. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performing regularly, hence the same test will not able to identify the new defects.

**Pesticide paradox**

This principle defined that 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. To get over these pesticide paradoxes, it is very significant to review all the test cases frequently. And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

**Testing is context-dependent**

Testing is a context-dependent principle states that 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. To check this type of application, we will take the help of various kinds of testing, different technique, approaches, and multiple methods. Therefore, the testing depends on the context of the application.

**Absence of errors fallacy**

Once the application is completely tested and there are no bugs identified before the release, so we can say that the application is 99 percent bug-free. But there is the chance when the application is tested beside the incorrect requirements, identified the flaws, and fixed them on a given period would not help as testing is done on the wrong specification, which does not apply to the client's requirements. 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.

**Difference between QA v/s QC v/s Tester ?**

|  |  |  |
| --- | --- | --- |
| **Quality Assurance** | **Quality Control** | **Testing** |
| Quality assurance is process oriented. It is all about preventing defects by ensuring the processes used to manage and create deliverables works. Not only does it work, but is consistently followed by the team. Moreover, QA is about engineering processes that assure quality is achieved in an effective and efficient way.  For instance, if a defect is found and fixed, there is no guaranteeing it won’t pop back up. The role of QA is to identify the process that allowed the error to occur and re-engineer the system so that these defects won’t appear for the second time. The QA process verifies that the product will continue to function as the customer expects.  Though QC is absolutely necessary, QA is perhaps more important. By the time you reach the QC stage, for instance, fixing bugs becomes an expensive issue. Because of that, focusing efforts on improved QA processes is one of the best investments an organization can make.  Examples of QA include process definition and implementation, training, audits and selection of tools. | Quality control, alternatively, is product oriented. It is the function of software quality that determines the ending result is what was expected. Whereas QA is proactive, QC is reactive. QC detects bugs by inspecting and testing the product. This involves checking the product against a predetermined set of requirements and validating that the product meets those requirements.  Examples of QC include technical reviews, software testing and code inspections. | Testing is a subset of QC. It is the process of executing a system in order to detect bugs in the product so that they get fixed. Testing is an integral part of QC as it helps demonstrate that the product runs the way it is expected and designed for.  To summarize, think of everything as an assembly line. QA can be thought of as the process to ensure the assembly line actually works, while QC is when the products coming off the assembly line are checked to verify they meet the required specifications.  Ultimately, both QA and QC are required for ensuring a successful product. When used together, they can help detect inefficient processes and identify bugs in the product. Moreover, QA and QC can help to develop and deliver a consistently high-quality product to your customers. |

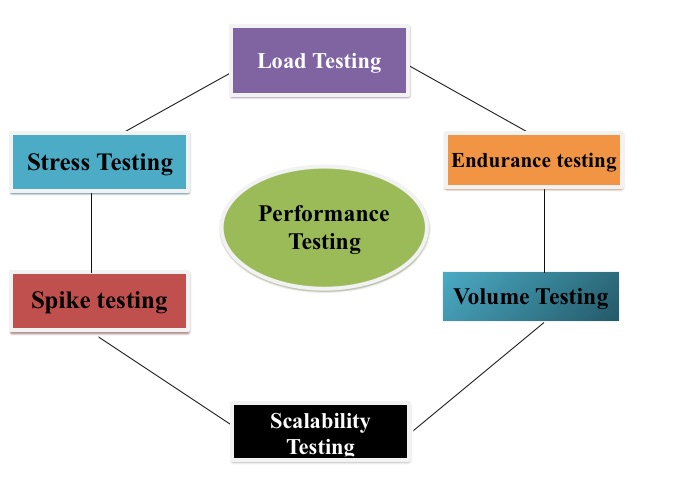
**Difference between Smoke and Sanity?**

|  |  |
| --- | --- |
| **Smoke testing** | **Sanity testing** |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality/bugs have been fixed |
| The objective of this testing is to verify the “stability” of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the “rationality” of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing in software testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of Regression Testing |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

**Difference between verification and Validation ?**

|  |  |  |
| --- | --- | --- |
|  | **Verification** | **Validation** |
| **Definition** | It is a process of checking if a product is developed as per the specifications. | It is a process of ensuring that the product meets the needs and expectations of stakeholders. |
| **What it tests or checks for** | It tests the requirements, architecture, design, and code of the software product. | It tests the usability, functionalities, and reliability of the end product. |
| **Coding requirement** | It does not require executing the code. | It emphasizes executing the code to test the usability and functionality of the end product. |
| **Activities include** | A few activities involved in verification testing are requirements verification, design verification, and code verification. | The commonly-used validation activities in software testing are usability testing, performance testing, system testing, security testing, and functionality testing. |
| **Types of testing methods** | A few verification methods are inspection, code review, desk-checking, and walkthroughs. | A few widely-used validation methods are black box testing, white box testing, integration testing, and acceptance testing. |
| **Teams or persons involved** | The quality assurance (QA) team would be engaged in the verification process. | The software testing team along with the QA team would be engaged in the validation process. |
| **Target of test** | It targets internal aspects such as requirements, design, software architecture, database, and code. | It targets the end product that is ready to be deployed. |

**Explain types of Performance testing.**

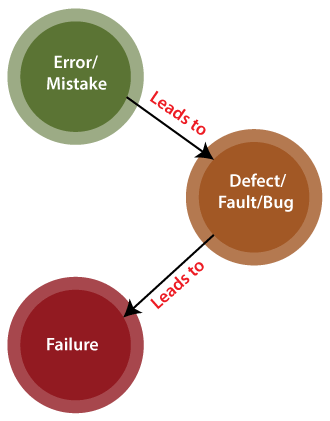


* **Load testing –** checks the application’s ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
* **Stress testing –** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
* **Endurance testing –** is done to make sure the software can handle the expected load over a long period of time.
* **Spike testing –** tests the software’s reaction to sudden large spikes in the load generated by users.
* **Volume testing –** Under Volume Testing large no. of. Data is populated in a database, and the overall software system’s behavior is monitored. The objective is to check software application’s performance under varying database volumes.
* **Scalability testing –** The objective of scalability testing is to determine the software application’s effectiveness in “scaling up” to support an increase in user load. It helps plan capacity addition to your software system.

**What is Error, Defect, Bug and failure?**

**Error**

The Problem in code leads to errors, which means that a mistake can occur due to the developer's coding error as the developer misunderstood the requirement or the requirement was not defined correctly. The developers use the term error.



**Defect**

When the application is not working as per the requirement is knows as defects. It is specified as the aberration from the actual and expected result of the application or software.

In other words, we can say that the bug announced by the programmer and inside the code is called a Defect.

**Bug**

In software testing, a bug is the informal name of defects, which means that software or application is not working as per the requirement. When we have some coding error, it leads a program to its breakdown, which is known as a bug. The test engineers use the terminology Bug.

If a QA (Quality Analyst) detect a bug, they can reproduce the bug and record it with the help of the bug report template.

**Failure**

Many defects lead to the software's failure, which means that a loss specifies a fatal issue in software/ application or in its module, which makes the system unresponsive or broken.

In other words, we can say that if an end-user detects an issue in the product, then that particular issue is called a failure.

**Difference between Priority and Severity ?**

|  |  |
| --- | --- |
| **Priority** | **Severity** |
| Defect Priority has defined the order in which the developer should resolve a defect | Defect Severity is defined as the degree of impact that a defect has on the operation of the product |
| Priority is categorized into three types   * Low * Medium * High | Severity is categorized into five types   * Critical * Major * Moderate * Minor * Cosmetic |
| Priority is associated with scheduling | Severity is associated with functionality or standards |
| Priority indicates how soon the bug should be fixed | Severity indicates the seriousness of the defect on the product functionality |
| Priority of defects is decided in consultation with the manager/client | QA engineer determines the severity level of the defect |
| Priority is driven by business value | Severity is driven by functionality |
| Its value is subjective and can change over a period of time depending on the change in the project situation | Its value is objective and less likely to change |
| High priority and low severity status indicates, defect have to be fixed on immediate bases but does not affect the application | High severity and low priority status indicates defect have to be fixed but not on immediate bases |
| Priority status is based on customer requirements | Severity status is based on the technical aspect of the product |
| During UAT the development team fix defects based on priority | During SIT, the development team will fix defects based on the severity and then priority |

**What is Bug Life Cycle?**

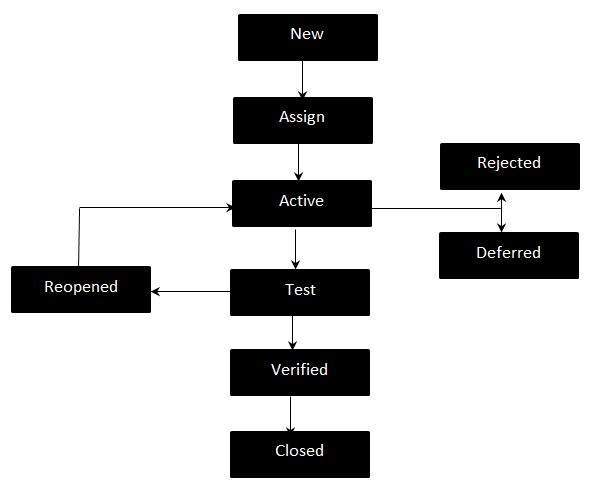
The Defect Life Cycle is also known as Bug Life Cycle. It is a process in which bug goes through different stages in its entire life. Consider the bug as a lively object that has different states throughout its life right from the state it’s opened by the tester to the state of getting closed after getting fixed.

There are two major people during any Software development process:

1. Software Developer: The one who designs and implements the code.
2. Software Tester: The one whose role is to find as many bugs as possible in the software and to open that bug for fixing it in the Defect Life Cycle.

A bug may arise in software, due to some mistake made by the Software developer either during the design or development of the product. It’s the tester’s or QA’s paramount responsibility to find all the bugs in software during the development phase itself so that they are fixed before the release of the software.

**Defect Life Cycle Workflow**



Let’s take a look at each of the state in this workflow and what does it mean in the realm of Defect Life Cycle :

**New:** This is the first state of a bug in the defect life cycle. In this stage, the tester finds a defect. Tester makes a complete specification document of what the bug is, how to reproduce it and what are the tests for which it fails. The tester then assigns the bug as a “New” bug.

Let us say, a software company is making a website for some client and has got a bug in it.

The tester tests the screen layout of the website in various screen sizes and finds out that for 5.5-inch screen size, the rendering is not according to the needs. The tester creates a bug report for the bug & that bug is in the new state.

Assign: In this state, the bug in the “New” state is assigned to a particular developer or a team by the project lead manager or some person of responsibility.

In the case of our bug, the screen size issue is assigned to the front-end development team.

Open: This is the state where the software developer starts analyzing the bug on the basis of the report developed by the tester in Step-1. The software developer starts working on fixing the code and reimplements the portions of the code that were causing trouble. There could also be a situation at this state when the developer feels like the bug is not appropriate, in that case, it is transferred to one of the following states in the defect life cycle i.e. Duplicate, Rejected, Deferred. These states would be discussed later in the blog.

In case of our bug, the front-end team analyses the reason for the bug, propose a solution for fixing it and apply the fixed patch to the codebase.

Fixed: This is the state when the developer fixes the “Open” bug after all the testing and

validation from his/her side.

In case of our bug, the front-end team fixes and tests/validates whether the issue is resolved or not.

Pending Retest: This is the state assigned to the bug, between the time the developer fixes the problem to the time the tester starts retesting whether the problem has been rectified or not.

In case of our bug, the front-end team marks the bug as “pending-retest”.

Retest: This is the state of the bug when the actual retesting of the problem is started by the software tester. The tester tests whether the problem has been fixed or not based on the requirements that were mentioned in the bug report of Step-1. Software tester also tests whether the current fix doesn’t affect any existing functionality in the software which in turn would lead to the origin of new bugs.

In the case of our bug, the tester tests whether the bug is properly fixed or not. He also checks whether any new bug has originated in the patched code.

Reopen: The state of a bug is changed to reopen if the tester during the retesting process finds that the bug is not completely resolved or some new bug has originated and should be again assigned to a developer for further fixation.

In case of our bug, if the tester after performing all the tests, finds that there is some component that is still not fixed, the tester reopens the bug and the bug again goes to the “Open” stage with an updated bug report.

Verified: If the tester during the Retest state finds that all the issues are resolved, then the tester

assigns the verified state to the bug in the defect life cycle.

In the case of our bug, if the tester becomes satisfied after performing all the tests, the tester changes the state of the bug as verified.

Closed: The tester changes the state of the bug to closed, once the bug no longer exists.

In case of our bug, the mobile screen rendering bug is marked as closed at this stage.

**Explain the difference between Functional testing and NonFunctional testing.**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Functional** | **Non-functional testing** |
| **Execution** | It is performed before non-functional testing. | It is performed after the functional testing. |
| **Focus area** | It is based on customer’s requirements. | It focusses on customer’s expectation. |
| **Requirement** | It is easy to define functional requirements. | It is difficult to define the requirements for non-functional testing. |
| **Usage** | Helps to validate the behavior of the application. | Helps to validate the performance of the application. |
| **Objective** | Carried out to validate software actions. | It is done to validate the performance of the software. |
| **Requirements** | Functional testing is carried out using the functional specification. | This kind of testing is carried out by performance specifications |
| **Manual testing** | Functional testing is easy to execute by manual testing. | It’s very hard to perform non-functional testing manually. |
| **Functionality** | It describes what the product does. | It describes how the product works. |
| **Example Test Case** | Check login functionality. | The dashboard should load in 2 seconds. |
| **Testing Types** | Examples of Functional Testing Types   1. Unit testing 2. Smoke testing 3. User Acceptance 4. Integration Testing 5. Regression testing 6. Localization 7. Globalization 8. Interoperability | Examples of Non-functional Testing Types   1. Performance Testing 2. Volume Testing 3. Scalability 4. Usability Testing 5. Load Testing 6. Stress Testing 7. Compliance Testing 8. Portability Testing 9. Disaster Recover Testing |

**What is the difference between the STLC (Software Testing Life Cycle) and SDLC (Software Development Life Cycle)?**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Software Testing Life Cycle (STLC)** | **Software Development Life Cycle (SDLC)** |
| **Origin** | Testing Life Cycle | Development Life Cycle |
| **Objective** | The only objective of the STLC phase is testing. | The main object of SDLC life cycle is to complete successful development of the software including testing and other phases. |
| **Requirement Gathering** | In STLC, the QA team analyze requirement documents like functional and non-functional documents and create System Test Plan | In SDLC the business analyst  gathers the requirements and  create Development Plan |
| **High & Low-Level Design** | In STLC, the test analyst creates the Integration Test Plan | In SDLC, the development team creates the high and low-level design plans |
| **Coding** | The testing team prepares the test environment and executes them | The real code is developed,  and actual work takes place as per the design documents. |
| **Maintenance** | Testers, execute regression suits, usually automation scripts to check maintenance code deployed. | SDLC phase also includes post-deployment supports and updates. |

**What is the difference between test scenarios, test cases, and test script?**

|  |  |  |
| --- | --- | --- |
| **Test scenario** | **Test case** | **Test script** |
| A Test Scenario is any functionality that can be tested. It is also called Test Condition or Test Possibility. | It is a document that contains the steps that has to be executed, it has been planned earlier. | It is written in a programming language and it's a short program used to test part of functionality of the software system. In other words a written set of steps that should be performed manually. |

**Explain what Test Plan is? What is the information that should be covered.**

A test plan is a detailed document which describes software testing areas and activities. It outlines the test strategy, objectives, test schedule, required resources (human resources, software, and hardware), test estimation and test deliverables.

The test plan is a base of every software's testing. It is the most crucial activity which ensures availability of all the lists of planned activities in an appropriate sequence.

The test plan is a template for conducting software testing activities as a defined process that is fully monitored and controlled by the testing manager. The test plan is prepared by the Test Lead (60%), Test Manager(20%), and by the test engineer(20%).

**Components of a Test Plan**

**Scope:** Details the objectives of the particular project. Also, it details user scenarios to be used in tests. If necessary, the scope can specify what scenarios or issues the project will not cover.

**Schedule:** Details start dates and deadlines for testers to deliver results.

**Resource Allocation:** Details which tester will work on which test.

**Environment:** Details the nature, configuration, and availability of the test environment.

**Tools:** Details what tools are to be used for testing, bug reporting, and other relevant activities.

**Defect Management:** Details how bugs will be reported, to whom and what each bug report needs to be accompanied by. For example, should bugs be reported with screenshots, text logs, or videos of their occurrence in the code?

**Risk Management:** Details what risks may occur during software testing, and what risks the software itself may suffer if released without sufficient testing.

**Exit Parameters:** Details when testing activities must stop. This part describes the results that are expected from the QA operations, giving testers a benchmark to compare actual results to.

**What are the different Methodologies in Agile Development Model ?**

Agile refers to the methods and best practices for organizing projects based on the values and principles documented in the Agile Manifesto. However, there’s no one right way to implement Agile and many different types of methodologies from which to choose. Here are some of the most common Agile frameworks.

**Kanban**

Kanban is a simple, visual means of managing projects that enables teams to see the progress so far and what’s coming up next. Kanban projects are primarily managed through a Kanban board, which segments tasks into three columns: “To Do,” “Doing,” and “Done.”

**Scrum**

Scrum is similar to Kanban in many ways. Scrum typically uses a Scrum board, similar to a Kanban board, and groups tasks into columns based on progress. Unlike Kanban, Scrum focuses on breaking a project down into sprints and only planning and managing one sprint at a time. Scrum also has unique project roles: Scrum master and product owner.

**Extreme Programming (XP)**

Extreme Programming (XP) was designed for Agile software development projects. It focuses on continuous development and customer delivery and uses intervals or sprints, similar to a Scrum methodology. However, XP also has 12 supporting processes specific to the world of software development:

1. Planning game
2. Small releases
3. Customer acceptance tests
4. Simple design
5. Pair programming
6. Test-driven development
7. Refactoring
8. Continuous integration
9. Collective code ownership
10. Coding standards
11. Metaphor
12. Sustainable pace

**Feature-driven development (FDD)**

Feature-driven development is another software-specific Agile framework. This methodology involves creating software models every two weeks and requires a development and design plan for every model feature. It has more rigorous documentation requirements than XP, so it’s better for teams with advanced design and planning abilities. FDD breaks projects down into five basic activities:

1. Develop an overall model
2. Build a feature list
3. Plan by feature
4. Design by feature
5. Build by feature

**Dynamic Systems Development Method (DSDM)**

The Dynamic Systems Development Method (DSDM) was born of the need for a common industry framework for rapid software delivery. Rework is to be expected, and any development changes that occur must be reversible. Like Scrum, XP, and FDD, DSDM uses sprints. This framework is based on eight fundamental principles:

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

**Crystal**

Crystal is a family of Agile methodologies that includes Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red, etc. Each has a unique framework. Your choice depends on several project factors, such as your team size, priorities, and project criticality.

**Lean**

Lean development is often grouped with Agile, but it’s an entirely different methodology that happens to share many of the same values. The main principles of the Lean methodology include:

1. Eliminating waste
2. Build quality in
3. Create knowledge
4. Defer commitment
5. Deliver fast
6. Respect people
7. Optimize the whole

**Explain the difference between Authorization and Authentication in Web testing.**

|  |  |
| --- | --- |
| **Authorization** | **Authentication** |
| Authorization determines what resources a user can access. | Authentication verifies who the user is. |
| Authorization works through settings that are implemented and maintained by the organization. | Authentication works through passwords, one-time pins, biometric information, and other information provided or entered by the user. |
| Authorization always takes place after authentication | Authentication is the first step of a good identity and access management process. |
| Authorization isn’t visible to or changeable by the user. | Authentication is visible to and partially changeable by the user. |
| Example: Once their level of access is authorized, employees and HR managers can access different levels of data based on the permissions set by the organization. | Example: By verifying their identity, employees can gain access to an HR application that includes their personal pay information, vacation time, and 401K data. |

**What are the common problems faced in Web testing?**

**Integration**

Integration testing exposes problems with interfaces among different program components before deployment. Additionally, integration testing can show the different issues an application may have when interacting with other applications, allowing the developer to tweak things. Environment and infrastructure inconsistency, different interaction models, and overall performance are just a few of the issues associated with integration testing.

**Interoperability**

Proving end-to-end functionality between communicating systems is always a challenging obstacle. Different users utilize different browsers and operating systems. To pull data, testing each one to confirm a clear information pathway is very important. Even if the browsers are similar, the web application may be rendered differently based on screen resolution and overall software configuration. This can present some serious issues for developers.

**Security**

In one of the most important tests, the developer must make sure that the continually evolving cyber threat can be countered and neutralized. Additionally, tests associated with data integrity before and after an attack are equally important when considering data breaches or lost information. Some of the challenges associated with security testing include dealing with unsecured communications, removing malicious files (if security firewalls have been breached), and the utilization (and integration) of different authentication procedures.

**Performance**

Slow applications are not successful. Developers understand that the speed of the app is defined by the need of the user, and with more users expecting more speed, the requirement of performance is non-negotiable. Testing large applications on minimal hardware, underestimating software requirements, and overextending application features are just a few of the issues associated with performance testing. Integration and interoperability issues can also have a direct effect on performance, and because of that, should be tested at the beginning.

**Usability**

Since web-based applications are dependent on different browsers, consistent usability is crucial. Additionally, since the app is the brand (or a component thereof), any inconsistency within the user experience may translate into a negative experience, affecting the brand and its potential growth. When testing usability, developers face issues with scalability and interactivity. Since every user is different, it is important for developers to utilize a representative group to test the application across different browsers, using different hardware.

The testing process is inherently full of issues, but each issue faced by the developer is one less that will be faced by the user. In app development, the first interaction with the app needs to be as flawless as possible, allowing the user to adopt and use the app in a way that is as effortless as possible. By testing integration, interoperability, security, performance, and usability, developers will be better able to identify and manage the actual issues associated with the app. In the end, it is about a positive user experience.