Notes of Software Testing

What is Software

Software is a set of instructions or programs that tell a computer what to do. It can be divided into two main categeories ie., system software and application software.

What is Testing

Testing is the process of determining how effective something is.

What is Software Testing

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect-free in order to produce a quality product.

- The benefits of software testing:

1. Cost-effectiveness
2. Customer Satisfaction
3. Security
4. Product Quality

1. Cost-effectiveness

As a matter of fact, design defects can never be completely ruled out for any complex system.

It is not because developers are careless but because the complexity of a system is intractable.

If the design issues go undetected, then it will become more difficult to trace back defects and rectify it. It will become more expensive to fix it.

Sometimes, while fixing one bug we may introduce another one in some other module unknowingly. If the bugs can be identified in the early stages of development then it costs much less to fix them.

That is why it is important to find defects in the early stages of the software development life cycle.

One of the benefits of testing is cost-effectiveness.

It is better to start testing earlier and introduce it in every phase of the software development life cycle and regular testing is needed to ensure that the application is developed as per the requirement.

2. Customer Satisfaction

In any business, the ultimate goal is to give the best customer satisfaction. Yes, customer satisfaction is very important.

Software testing improves the user experience of an application and gives satisfaction to the customers. Happy customers mean more revenue for a business.

3. Security

This is probably the most sensitive and vulnerable part of testing. Testing (penetration testing & security testing) helps in product security.

Hackers gain unauthorized access to data. These hackers steal user information and use it for their benefit. If your product is not secured, users won't prefer your product. Users always look for trusted products. Testing helps in removing vulnerabilities in the product.

4. Product Quality

Software Testing is an art that helps in strengthening the market reputation of a company by delivering a quality product to the client as mentioned in the requirement specification documents.

Due to these reasons, software testing becomes a very significant and integral part of the Software Development process.

Seven Principles of Software Testing:

1. Testing Shows Presence of Defects:

Testing shows the presence of defects in the software. The goal of testing is to make the software fail. Sufficient testing reduces the presence of defects. In case testers are unable to find defects after repeated regression testing doesn't mean that the software is bug-free.

Testing talks about the presence of defects and don't talk about the absence of defects.

2. Exhaustive Testing is Impossible:

What is Exhaustive Testing?

Testing all the functionalities using all valid and invalid inputs and preconditions is known as Exhaustive testing.

Why it’s impossible to achieve Exhaustive Testing?

Assume we have to test an input field which accepts age between 18 to 20 so we do test the field using 18,19,20. In case the same input field accepts the range between 18 to 100 then we have to test using inputs such as 18, 19, 20, 21, …., 99, 100. It’s a basic example, you may think that you could achieve it using automation tool. Imagine the same field accepts some billion values. It’s impossible to test all possible values due to release time constraints.

If we keep on testing all possible test conditions then the software execution time and costs will rise. So instead of doing exhaustive testing, risks and priorities will be taken into consideration whilst doing testing and estimating testing efforts.

3. Early Testing:

Defects detected in early phases of SDLC are less expensive to fix. So conducting early testing reduces the cost of fixing defects.

Assume two scenarios, first one is you have identified an incorrect requirement in the requirement gathering phase and the second one is you have identified a bug in the fully developed functionality. It is cheaper to change the incorrect requirement compared to fixing the fully developed functionality which is not working as intended.

4. Defect Clustering:

Defect Clustering in software testing means that a small module or functionality contains most of the bugs or it has the most operational failures.

As per the Pareto Principle (80-20 Rule), 80% of issues comes from 20% of modules and remaining 20% of issues from remaining 80% of modules. So we do emphasize testing on the 20% of modules where we face 80% of bugs.

5. Pesticide Paradox:

Pesticide Paradox in software testing is the process of repeating the same test cases again and again, eventually, the same test cases will no longer find new bugs. So to overcome this Pesticide Paradox, it is necessary to review the test cases regularly and add or update them to find more defects.

6. Testing is Context Dependent:

Testing approach depends on the context of the software we develop. We do test the software differently in different contexts. For example, online banking application requires a different approach of testing compared to an e-commerce site.

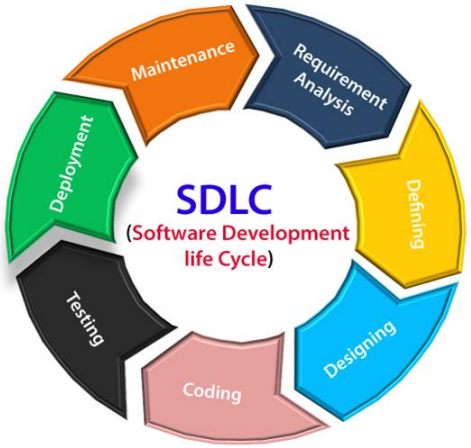
7. Absence of Error – Fallacy:

99% of bug-free software may still be unusable, if wrong requirements were incorporated into the software and the software is not addressing the business needs.

The software which we built not only be a 99% bug-free software but also it must fulfill the business needs otherwise it will become an unusable software.

What is SDLC?

Software Development Life Cycle (SDLC) aims to produce a high-quality system that meets or exceeds customer expectations, works effectively and efficiently in the current and planned information technology infrastructure, and is inexpensive to maintain and cost-effective to enhance..



What is SDLC Process:

SDLC is a process which follows in Software Projects to develop a product in a systematic way and to deliver a high-quality product. By following proper SDLC process, Software companies can react well to the market pressure and release high-quality software. This process involves different stages of SDLC right from the requirement stage to deployment and maintenance phase. These SDLC phases we will see later section of this post.

Why SDLC

Some of the reasons why SDLC is important in Software Development are as follows.

- It provides visibility of a project plan to all the involved stakeholders
- It helps us to avoid project risks
- It allows us to track and control the project
- It doesn’t conclude until all the requirements have been achieved

Requirement Phase

Requirement gathering and analysis is the most important phase in the software development lifecycle. Requirement phase is the first step of the SDLC. Business Analyst collects the requirement from the Customer/Client as per the clients business needs and documents the requirements in the Business Requirement Specification (document name varies depends upon the Organization. Some examples are Customer Requirement Specification (CRS), Business Specification (BS), etc., and provides the same to Development Team.

Analysis Phase

Once the requirement gathering and analysis is done the next step is to define and document the product requirements and get them approved by the customer. This is done through the SRS (Software Requirement Specification) document. SRS consists of all the product requirements to be designed and developed during the project life cycle. Key people involved in this phase are Project Manager, Business Analyst and Senior members of the Team. The outcome of this phase is the Software Requirement Specification.

Design Phase

It has two steps:
HLD – High-Level Design – It gives the architecture of the software product to be developed and is done by architects and senior developers
LLD – Low-Level Design – It is done by senior developers. It describes how each and every feature in the product should work and how every component should work. Here, only the design will be there and not the code
The outcome from this phase is High-Level Document and Low-Level Document which works as an input to the next phase

Development Phase

Developers of all levels (seniors, juniors, freshers) involved in this phase. This is the phase where we start building the software and start writing the code for the product. The outcome from this phase is Source Code Document (SCD) and the developed product.

Testing Phase

When the software is ready, it is sent to the testing department where Test team tests it thoroughly for different defects. They either test the software manually or using automated testing tools depends on the process defined in STLC (Software Testing Life Cycle) and ensure that each and every component of the software works fine. Once the QA makes sure that the software is error-free, it goes to the next stage, which is Implementation. The outcome of this phase is the Quality Product and the Testing Artifacts.

Deployment & Maintenance Phase

After successful testing, the product is delivered/deployed to the customer for their use. Deployment is done by the Deployment/Implementation engineers. Once when the customers start using the developed system then the actual problems will come up and needs to be solved from time to time. Fixing the issues found by the customer comes in the maintenance phase. 100% testing is not possible – because, the way testers test the product is different from the way customers use the product. Maintenance should be done as per SLA (Service Level Agreement)

Waterfall Model:

Waterfall Model is a traditional model. It is aka Sequential Design Process, often used in SDLC, in which the progress is seen as flowing downwards like a waterfall, through the different phases such as Requirement Gathering, Feasibility Study/Analysis, Design, Coding, Testing, Installation and Maintenance. Every next phase is begun only once the goal of previous phase is completed. This methodology is preferred in projects where quality is more important as compared to schedule or cost. This methodology is best suitable for short term projects where the requirements will not change. (E.g. Calculator, Attendance Management)

Advantages:

- Requirements do not change nor does design and code, so we get a stable product.
- This model is simple to implement. Requirements are finalized earlier in the life cycle. So there won't be any chaos in the next phases.
- Required resources to implement this model are minimal compared to other methodologies
- Every phase has specific deliverable's. It gives high visibility to the project manager and clients about the progress of the project.

Disadvantages:

- Backtracking is not possible i.e., we cannot go back and change requirements once the design stage is reached.
- Change in requirements leads to change in design and code which results defects in the project due to overlapping of phases.
- Customer may not be satisfied, if the changes they required are not incorporated in the product.
- The end result of waterfall model may not be a flexible product
- In terms of testing, testers involve only in the testing phase. Requirements are not tested in the requirement phase. It can't be modified even though we identify that there is a bug in the requirement in the testing phase. It goes on till the end and leads to lot of re-work.

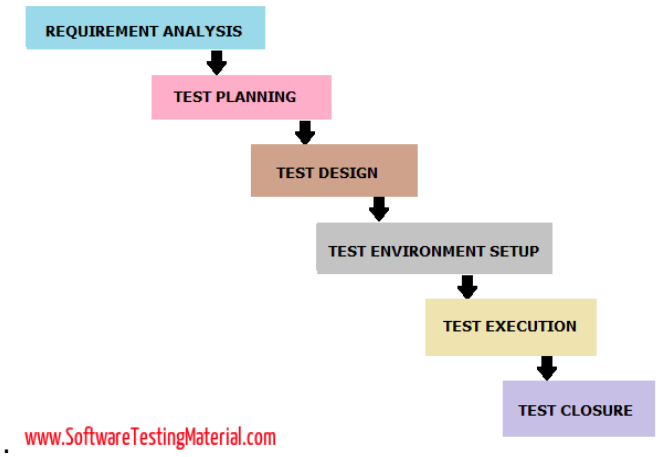
- It is not suitable for long term projects where requirements may change time to time
- Waterfall model can be used only when the requirements are very well known and fixed

V Model:

V-model is also known as **Verification and Validation (V&V) model**. In this each phase of SDLC must be completed before the next phase starts. It follows a sequential design process same like waterfall model.

What is STLC?

Software Testing Life Cycle (**STLC**) identifies what test activities to carry out and when to accomplish those test activities. Even though testing differs between organizations, there is a testing life cycle.



The different phases of the Software Testing Life Cycle Model (STLC Model) are:

1. Requirement Analysis
2. Test Planning
3. Test Design
4. Test Environment Setup
5. Test Execution
6. Test Closure

Every phase of STLC (Software Testing Life Cycle) has a definite Entry and Exit Criteria.

1. Requirement Analysis

The entry criteria for this phase is the BRS (Business Requirement Specification) document. During this phase, the test team studies and analyzes the requirements from a testing perspective.

This phase helps to identify whether the requirements are testable or not. If any requirement is not testable, the test team can communicate with various stakeholders (Client, Business Analyst, Technical Leads, System Architects, etc) during this phase so that the mitigation strategy can be planned.

Entry Criteria: BRS (Business Requirement Specification)

Deliverables: List of all testable requirements, Automation feasibility report (if applicable)

2. Test Planning:

Test planning is the first step in the testing process.

In this phase typically Test Manager/Test Lead involves determining the effort and cost estimates for the entire project. Preparation of the Test Plan will be done based on the requirement analysis.

Activities like resource planning, determining roles and responsibilities, tool selection (if automation), training requirements, etc., carried out in this phase.

The deliverables of this phase are Test Plan & Effort estimation documents.

Entry Criteria: Requirements Documents

Deliverables: Test Strategy, Test Plan, and Test Effort estimation document.

3. Test Design:

The test team starts with test case development activity here in this phase. Testers prepares test cases, test scripts (if automation), and test data.

Once the test cases are ready then these test cases are reviewed by peer members or team lead.

Also, the test team prepares the Requirement Traceability Matrix (RTM). RTM traces the requirements to the test cases that are needed to verify whether the requirements are fulfilled. The deliverables of this phase are Test Cases, Test Scripts, Test Data, Requirements Traceability Matrix

Entry Criteria: Requirements Documents (Updated version of unclear or missing requirement)

Deliverables: Test cases, Test Scripts (if automation), Test data.

In this phase, Selenium would be the most popular tool to use. However, its complexities and programming experience needed to Python or C# would definitely pose a problem to your manual QAs and automation freshers.

Here, Katalon Studio would be your go-to choice in simplifying Selenium's essential capabilities through codeless automation, built-in keywords, and pre-defined artifact templates for test suites.

4. Test Environment Setup:

This phase can be started in parallel with the Test design phase.

The test environment setup is done based on the hardware and software requirement list. In some cases, the test team may not be involved in this phase. The development team or customer provides the test environment.

Meanwhile, the test team should prepare the smoke test cases to check the readiness of the given test environment.

Entry Criteria: Test Plan, Smoke Test cases, Test Data

Deliverables: Test Environment. Smoke Test Results.

5. Test Execution:

The test team starts executing the test cases based on the planned test cases. If a test case result is Pass/Fail then the same should be updated in the test cases.

The defect report should be prepared for failed test cases and should be reported to the Development Team through a bug tracking tool for fixing the defects.

Retesting will be performed once the defect was fixed. Click here to see the Bug Life Cycle.

Entry Criteria: Test Plan document, Test cases, Test data, Test Environment.

Deliverables: Test case execution report, Defect report, RTM

6. Test Closure:

The final stage where we prepare Test Closure Report, Test Metrics.

The testing team will be called out for a meeting to evaluate cycle completion criteria based on Test coverage, Quality, Time, Cost, Software, Business objectives.

The test team analyses the test artifacts (such as Test cases, Defect reports, etc.,) to identify strategies that have to be implemented in the future, which will help to remove process bottlenecks in the upcoming projects.

Test metrics and Test closure report will be prepared based on the above criteria.

Entry Criteria: Test Case Execution report (make sure there are no high severity defects opened), Defect report

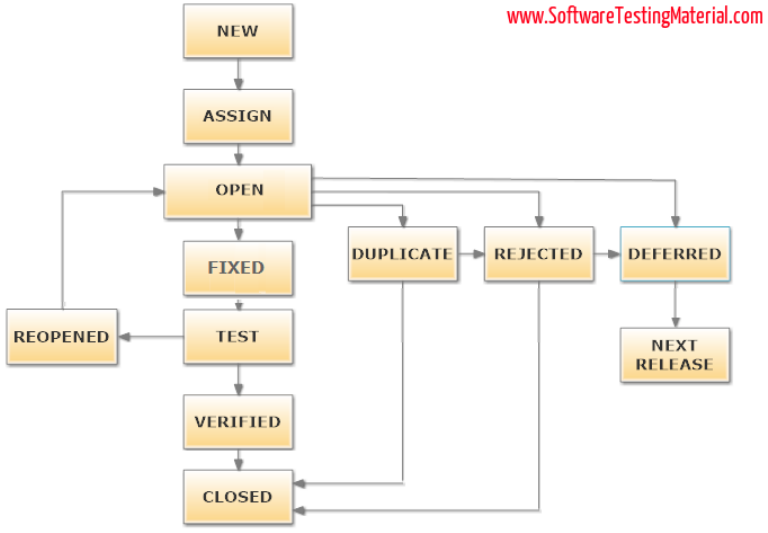
Deliverables: Test Closure report, Test metrics

Definition Bug Life Cycle

The bug life cycle is also known as the **Defect life cycle**. In the Software Development Process, the bug has a life cycle. The bug should go through the life cycle to be closed. The bug life cycle varies depends upon the tools (QC, JIRA, etc.,) used, and the process followed in the organization.

What is a Software Bug?

A software bug can be defined as the abnormal behavior of the software. The bug starts when the defect is found and ends when a defect is closed, after ensuring it is not reproduced.



#1. New

When a tester finds a new defect. He should provide a proper Defect document to the Development team to reproduce and fix the defect. In this state, the status of the defect posted by the tester is “New”

#2. Assigned

Defects that are in the status of New will be approved (if valid) and assigned to the development team by Test Lead/Project Lead/Project Manager. Once the defect is assigned then the status of the bug changes to “Assigned”

#3. Open

The development team starts analyzing and works on the defect fix

#4. Fixed

When a developer makes the necessary code change and verifies the change, then the status of the bug will be changed as “Fixed” and the bug is passed to the testing team.

#5. Test

If the status is “Test”, it means the defect is fixed and ready to do test whether it is fixed or not.

#6. Verified

The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”

#7. Closed

After verified the fix, if the bug is no longer exits then the status of the bug will be assigned as “Closed.”

#8. Reopen

If the defect remains the same after the retest, then the tester posts the defect using the defect retesting document and changes the status to “Reopen”. Again the bug goes through the life cycle to be fixed.

#9. Duplicate

If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate” by the development team.

#10. Deferred

In some cases, the Project Manager/Lead may set the bug status as deferred.

- If the bug found during the end of the release and the bug is minor or not important to fix immediately.
- If the bug is not related to the current build.
- If it is expected to get fixed in the next release.
- The customer is thinking to change the requirement.
- In such cases the status will be changed as “deferred” and it will be fixed in the next release.

#11. Rejected

If the system is working according to specifications and the bug is just due to some misinterpretation (such as referring to old requirements or extra features) then the Team lead or developers can mark such bugs as “Rejected”

Some other statuses are:

#12. Cannot be fixed

Technology not supporting, Root of the product issue, Cost of fixing a bug is more

#13. Not Reproducible

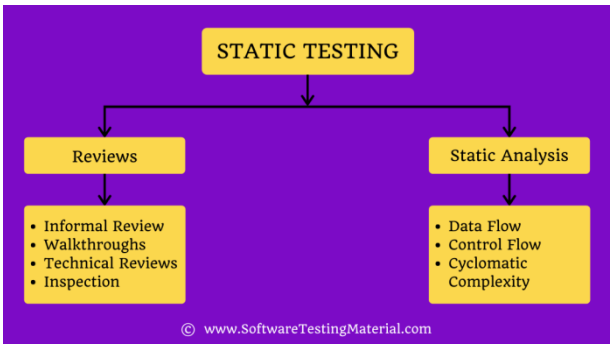
Platform mismatch, improper defect document, data mismatch, build mismatch, inconsistent defects

#14. Need more information

If a developer is unable to reproduce the bug as per the steps provided by a tester then the developer can change the status as “Need more information’. In this case, the tester needs to add detailed reproducing steps and assign bugs back to the development team for a fix. This won’t happen if the tester writes a good defect document.

What is Static Testing

Static Testing is a software testing method, which is used to check defects in software application without executing the code whereas dynamic testing is used to check defects by executing the code.



1. Reviews

Reviews are usually performed to identify and correct errors and uncertainties found in supporting documents such as requirement documents, design documents, test cases etc.,

Types of Static Testing Reviews

Static testing reviews are classified into four types:

#1. Informal Reviews

Informal reviews do not use any set method to detect problems. Coworkers review documents and give informal comments.

#2. Walkthrough

The document's author will explain the document to their team during a walkthrough. Participants would ask questions, and any notes are taken down.

#3. Technical Reviews

Peers examine technical specifications in order to detect any errors.

#4. Inspections

Moderator will do a comprehensive examination as part of the procedure to detect flaws.

Types of Participants in Review Process

Participants in the review process are as follows

#1. Author

Author's responsibility is to fix the errors found and improves the document's quality.

#2. Moderator

The Moderator is in charge of checking for entries, following up on reworks, coaching team members and to schedule the meetings.

#3. Scribe

Logs the error during a review.

#4. Reviewer

Examine the material for defects.

#5. Manager

Determine how evaluations will be carried out and make sure that the review process goals are achieved.

#2. Static Analysis

Static analysis is where the code developed by developers is evaluated. It is done to find structural defects that may lead to errors.

Static code analysis can be done both either manually or through automation with the use of various software testing tools.

Some of the tools which can be used for static analysis of code are:

- Checkstyle
- Clang
- Find Bugs
- JArchitect
- JTest
- Sonarqube
- Soot
- Source meter
- Thread Safe

Types of Static Analysis

Static Analysis is classified into three types:

#1. Data Flow

In static analysis, the data flow is related to the stream processing.

#2. Control Flow:

Control flow is used to specify how the statements or instructions are executed.

#3. Cyclomatic Complexity:

It is the measurement of the program's complexity that is basically related to the number of independent paths in the control flow graph of the program.

What is Dynamic Testing

Dynamic Testing is a software testing method, which is used to check defects in software application by executing the code where as static testing is used to check defects without executing the code.

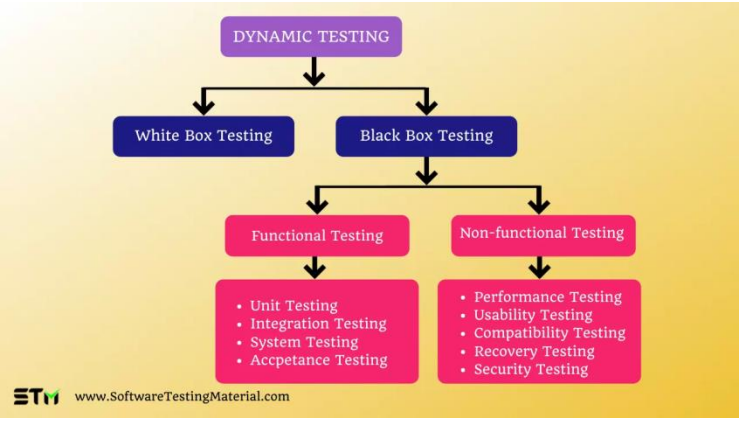
Verification is also known as **Static testing**.

Validation is also known as **Dynamic Testing**

Types of Dynamic Testing

Dynamic testing is divided into two categories, which are as follows:

- White Box Testing
- Black Box Testing



White Box Testing

White box testing is also known as clear box testing or glass box testing.

White box testing is a software testing method used to test how the application is performing based on the code.

White box testing is usually performed by the developers or white box testers who has knowledge on the internal structure/code/design.

Black Box Testing

Black box testing is also known as Behavioral/Specification-Based/Input-Output Testing

Black Box Testing is a software testing method used to evaluate the functionality of the software without looking at the internal code structure

This can be applied to every level of software testing such as Unit, Integration, System, and Acceptance Testing.

Black Box Testing is usually performed by testers who don't require any programming expertise.

Dynamic Testing Techniques

Dynamic Testing techniques are classified into two categories. They are

- Functional Testing
- Non-functional Testing

Functional Testing

Functional testing is done to verify that each function of the application behaves as specified in the requirement. Here testes test all the funcitionalities by providing appropriate input to validate the actual output with the expected output.

In simple words, what the system actually does is functional testing.

Learn more on Functional testing here

Non-functional Testing

Non-functional testing is done to improve the user experience on how fast the application responds to a request. it refers to various aspects of the software such as performance, load, stress, scalability, compatibility, security, etc.,

It verifies the attributes such as performance, memory leaks, or robustness of the system.

In simple words, how well the system performs is non-functionality testing.

Learn more on Non-functional testing here.

Levels of Dynamic Testing

Dynamic testing is done at both functional and non-functional levels.

Some of the levels in functional testing are as follows.

- Unit Testing
- Integration Testing
- System Testing
- Accpetance Testing

Unit Testing: Unit testing is also known as module testing or component testing. It is used to verify that each component of the source code, such as a unit or module, is functioning properly. Usually developers do unit testing in their own environment.

Integration Testing: The goal of integration testing is to check for any issues between two software components. There are numerous ways to do integration testing, including the Big Bang Approach, Top-Down Approach, Bottom-Up Approach, and Hybrid Integration approach.

System Testing: End-to-end testing of a fully integrated application to evaluate the system's compliance with its specified requirements is called System Testing. It is also known as End to End testing. Checking the completed system to ensure that the application performs as intended or not.

Accpetance Testing: Acceptance testing is the process of determining whether an application behaves as expected after it's released. It is carried out by end-users and testers to confirm that the application functions correctly.

Some of the levels in non-functional testing are as follows.

- Performance Testing
- Usability Testing
- Compatibility Testing
- Recovery Testing
- Security Testing

Performance Testing: This is the process of determining or validating the speed, scalability, and/or stability features of a system or application under test. The goal of performance testing is to produce response times, throughput, and resource-utilization levels that satisfy project or product performance specifications. Read more about Performance Testing here.

Usability Testing: To assess whether the application is easy to use or not. This testing seeks to determine if the end-user can easily understand and use the software. It should be self-explanatory, and no training should be required to run it. Read more about Usability Testing here.

Compatibility Testing: It's the process of deploying and measuring whether an application functions as intended in a variety of environmental factors.

Recovery Testing: Recovery testing determines how quickly the system can recover from a system crash or hardware failure.

Security Testing: Security testing is a method for assessing whether the system protects data and functions as intended. Read more about Security Testing here.

Performance Testing:

Performance testing determines or validates the speed, scalability, and/or stability characteristics of the system or application under test. Performance is concerned with achieving response times, throughput, and resource-utilization levels that meet the performance objectives for the project or product.

Types:

Capacity Testing:

Capacity Testing is to determine how many users the system/application can handle successfully before the performance goals become unacceptable. This allows us to avoid the potential problems in future such as increased user base or increased volume of data.

Load Testing:

Load Testing is to verify that the system/application can handle the expected number of transactions and to verify the system/application behaviour under both normal and peak load conditions (no. of users).

Volume Testing:

Volume Testing is to verify that the system/application can handle a large amount of data. This testing focuses on Data Base.

Stress Testing:

Stress Testing is to verify the behaviour of the system once the load increases more than the system's design expectations. This testing addresses which components fail first when we stress the system by applying the load beyond the design expectations. So that we can design more robust system.

Soak Testing:

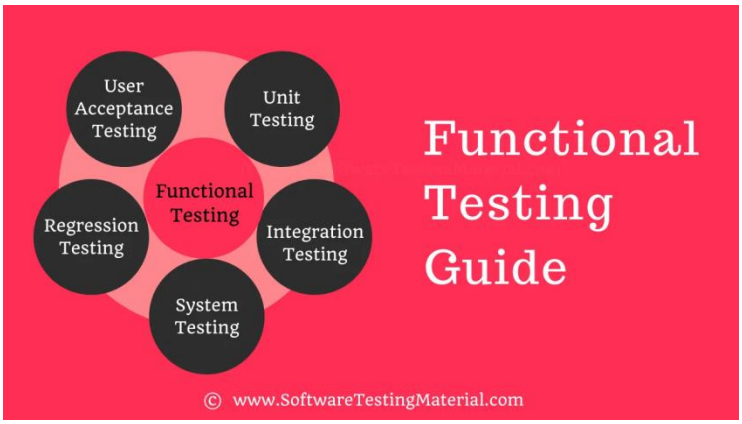
Soak Testing is aka Endurance Testing. Running a system at high load for a prolonged period of time to identify the performance problems is called Soak Testing. It is to make sure the software can handle the expected load over a long period of time.

Spike Testing:

Spike Testing is to determine the behaviour of the system under sudden increase of load (a large number of users) on the system.

What is Functional Testing?

In simple words, what the system actually does is functional testing. To verify that each function of the software application behaves as specified in the requirement document. Testing all the functionalities by providing appropriate input to verify whether the actual output is matching the expected output or not. It falls within the scope of black box testing and the testers need not concern about the source code of the application.



Functional Testing Types

There are several types of testing that are performed to ensure the quality of a software. Some of the most important types of functional testing are as follows:

Unit Testing

Unit Testing is also called Module Testing or Component Testing. It is done to check whether the individual unit or module of the source code is working properly. It is done by the developers in the developer's environment.

Integration Testing

Integration Testing is the process of testing the interface between the two software units. Integration testing is done by multiple approaches such Big Bang Approach, Top-Down Approach, Bottom-Up Approach, and Hybrid Integration approach.

System Testing

Testing the fully integrated application to evaluate the system's compliance with its specified requirements is called System Testing AKA End to End testing. Verifying the completed system to ensure that the application works as intended or not.

Regression Testing

Repeated testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software components.

User Acceptance Testing

It is also known as pre-production testing. This is done by the end users along with the testers to validate the functionality of the application. After successful acceptance testing. Formal testing conducted to determine whether an application is developed as per the requirement. It allows the customer to accept or reject the application. Types of acceptance testing are Alpha, Beta & Gamma.

What is Non-Functional Testing

Non-functional testing is a type of software testing which refers to various aspects of the software such as performance, load, stress, scalability, security, compatibility, etc., The main focus of non-functional testing is to improve the user experience on how fast the system responds to a request.

Non-Functional Testing Types:

#1. Performance Testing

Performance testing determines how quickly an application performs in various situations. Learn more Performance Testing Tutorial

#2. Load Testing

Load testing is a test that looks at an app's ability to function well when under peak conditions. Learn more Load Testing Tutorial

#3. Compatibility Testing

To test whether the application, website, or system under evaluation is suitable for a variety of environments such as web browsers, hardware platforms, databases, operating systems, networks, mobile devices, different versions and configurations. Learn more Compatibility Testing Tutorial

#4. Usability Testing

Usability Testing is a kind of testing that aims to determine how simple the software is to use. Learn more Usability Testing Tutorial

#5. Stress Testing

Stress testing subjects the system to severe conditions such as overloading it to see whether it can sustain the stress. Learn more Stress Testing Tutorial

#6. Volume Testing

Volume testing refers to the performance of a software application when it is exposed to a high amount of data. Learn more Volume Testing Tutorial

#7. Security Testing

Security testing is to ensure that the system protects data and continues to function as intended. Learn more Security Testing Tutorial

#8. Reliability Testing

Reliability testing is a kind of software testing procedure that validates whether the software application works correctly in a certain setting for a specific length of time. Learn more Reliability Testing Tutorial

#9. Endurance Testing

Endurance testing examines how a system behaves under a particular load for an extended period. We can assess the system's behavior to ensure that the software is capable of withstanding high loads without decreasing latency. Learn more Endurance Testing Tutorial

#10. Documentation Testing

Document testing is the process of verifying that the documented artifacts produced before, during, and after the product's testing are genuine. Learn more Documentation Testing Tutorial

#11. Localization Testing

Localization testing is to ensure that the application's design is appealing to a specific culture, region, or local. Learn more Localization Testing Tutorial

#12. Internationalization Testing

Internationalisation testing is to ensure whether the applications are designed in such a way that it appeals to any culture, region or local.

#13. Baseline Testing

Benchmark testing is a proactive form of testing that is used to set performance goals and track progress over time. Learn more Baseline Testing Tutorial

#14. Portability testing

Portability testing is used to test how an application transfer from one software to another software.

Non-Functional Testing Parameters

#1. Security

This parameter determines how a system is protected against planned and unpredictable attacks from both internal and external sources.

We perform Security Testing to validate this parameter.

#2. Reliability

The level to which any software system fulfills the required activities consistently and without error.

We perform Reliability Testing to validate this parameter.

#3. Survivability

The parameter ensures that the software system continues run smoothly and restores in the event of system failure.

We perform Recovery Testing to validate this parameter.

#4. Availability

The parameter determines how much the user can rely on the system during its operation.

We perform Stability Testing to validate this parameter.

#5. Usability

The convenience with which a user can learn, operate, input/output data through engagement with a system.

We perform Usability Testing to validate this parameter.

#6. Scalability

The term scalability refers to the software application's ability in expanding its processing capacity in order to fulfill growing demand.

We perform Scalability Testing to validate this parameter.

#7. Interoperability

Interoperability validates the connections between a software system and other software systems.

We perform Interoperability Testing to validate this parameter.

#8. Efficiency

The degree to which a software application can handles capacity, quantity and response time.

#9. Flexibility

Flexibility refers to how quickly and simply the program can work in different hardware and software configurations.

#10. Portability

The flexibility of the software to shift from its existing hardware or software configuration.

#11. Reusability

Reusability referes to a portion of software application that can be adapted for use in another software application.

Advantages of Non-functional testing

The following are some of the benefits of performing a non-functional test:

- It improves the performance of the application. Ensures the application runs smoothly and efficiently for large numbers of users simultaneously without any loading issues.
- It improves the security of the application. Protects the applicatoin from hacking.
- It includes testing that isn't possible or covered in functional testing.

Disadvantages of Non-functional testing

The following are some of the disadvantages of non-functional testing:

- Non-functional tests need to be executed again whenever the software is updated.
- Users must pay to re-examine the software as a result of software upgrades, making it prohibitively expensive.

What is Alpha Testing?

Alpha testing is a type of acceptance testing that is done by inhouse testers before the application goes live.

The objective of alpha testing is to perform a final round of testing and identify every type of issue which were not discovered in the previous rounds of testing.

What is Beta Testing?

In Software Testing, Beta Testing is a type of User Acceptance Testing. It is performed by a limited number of end-users (customers or real users) before delivery. Usually, it is done in the client's place.

What is Positive Testing

From the software testers' point of view, it is very important to verify that the software performs its basic functions as per the requirements. In Positive testing, the tester always checks for only valid sets of input values and verifies if the software under test or application behaves as it is designed to work. The word itself explains that positive testing is performed with a positive set of data.

The main goal of this testing is to check whether a software application is not showing an error when it is not supposed to and showing error when it is supposed to. Positive testing always tries to prove that a given product or project always meets the requirements and specifications.

What is Negative Testing

From the software testers' point of view, it is very important to verify that the software performs its basic functions as per the requirements but it is equally important to verify that the software is able to gracefully handle any abnormal situations or invalid input which helps to determine the stability of the software. Negative testing is performed to find a situation where there is the possibility of software to crash.

It is a negative approach, where testers try to design test cases to find the negative aspects of the application and validate against invalid input. Negative testing is also known as Failure testing or error path testing. The application's functional reliability can be measured only with designed negative scenarios.

What Is Maintenance Testing?

In software testing, there are more than 100 types of testing and this maintenance testing is one of them. As a tester, we do testing software during its pre-release stage. We also perform testing on the software after it is released. Performing testing after it is released is known as maintenance testing.

What is Regression Testing with example?

Regression testing is a type of software testing that seeks to identify bugs or errors in software code that was previously working correctly.

This testing is typically performed after new code has been added or modified, in order to ensure that the new code does not break existing functionality.

What is Retesting?

Retesting : To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build.

Retesting is running the previously failed test cases again on the new software to verify whether the defects posted earlier are fixed or not.

In simple words, Retesting is testing a specific bug after it was fixed.

What is Traceability Matrix?

Requirement Traceability Matrix (RTM) is used to trace the requirements to the tests that are needed to verify whether the requirements are fulfilled.

Requirement Traceability Matrix or **Traceability Matrix** or **Cross Reference Matrix**.

What is White Box Testing?

White Box Testing is a software testing technique that is based on the application's internal code structure. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. This testing usually is done at the unit level.

What is Grey Box Testing?

Gray Box testing is a combination of the black box and white box testing, in this technique the tester only needs limited knowledge about the internal workings of an application.

What is Ad hoc Testing

Adhoc testing is a type of unplanned testing which does not follow any formal process like requirement documents, test plan, test cases, etc.

What is Localization Testing?

Localization testing is a testing technique that verifies the accuracy and the suitability of the customized content of the target language, region, or local along with the functionality and usability of the application.

What do you mean by globalization testing?

Globalization Testing is **a type of software testing that is performed to ensure the system or software application can function independent of the geographical and cultural environment**.

What is Usability Testing?

Usability Testing is a testing technique used to evaluate how easily the user can use the software.

What is Accessibility Testing?

The accessibility testing is used to verify whether a web application is built in such a way that it is accessible i.e easy to use by any differently-abled individual.

What is Benchmark Testing?

The benchmark is just one reference point that is decided upon by measuring a repeatable set of quantifiable results. This reference point is considered as a point that serves to be the standard pointer used for further analysis.

Benchmark testing and performance testing have a very thin line of difference. The results against performance metrics are agreed upon in any business and are based on various industry standards.

What is Cross Browser Testing?

Cross Browser Testing is a type of non-functional test which helps us ensure that our website or web application works as expected in various web browsers.

What is ERP?

Enterprise Resource Planning or ERP is a comprehensive software that integrates the various functions of an organization into a single system. The software has a shared database containing all information pertaining to the various functions or units of an organization. The ERP system helps to streamline the processes and access to information across the organization 24×7.

Why is ERP software testing needed?

ERP software is cost-intensive and requires a lot of investment in time and effort. Every ERP software comes with multiple versions and requires customization to suit specific

business requisites. Moreover, since every element in the application is connected to some other module, upgrading them can be a challenging task. For example, creating a sales order would need access to the inventory management module. If any of the modules does not function to its optimum, the entire ERP application may be impacted. This can have a cascading effect on the performance of the company as well as create bad customer experiences. Hence, testing ERP applications shall ensure the correct implementation of the software and prevent crashes.

What Is Reliability Testing?

Reliability testing is a type of software testing process that verifies whether the software functions in an error-free way in the given environment for a particular time.

What is Volume Testing?

Volume testing validates the performance of a software application when it is subjected to a large volume of data, it is a type of non-functional testing, it comes under performance testing.

Volume Testing is also known as **Flood testing**.

Difference between SDLC & STLC

Criterion	SDLC	STLC
Origin	Development Life Cycle	Testing Life Cycle
Stands for	SDLC stands for Software Development Life Cycle	STLC stands for Software Testing Life Cycle
Definition	Software Development Life Cycle (SDLC) aims to produce a high-quality system that meets or exceeds customer expectations, works effectively and efficiently in the current and planned information technology infrastructure, and is inexpensive to maintain and cost-effective to enhance.	Software Testing Life Cycle (STLC) identifies what test activities to carry out and when to accomplish those test activities. Even though testing differs between Organizations,there is a testing life cycle.
Focus	On both development and testing process	On only testing process
Relationship	It is taken as the predecessor	It is taken as the successor
Phases	Requirement Gathering, Analysis, Design, Coding, Testing, Deployment & maintenance	Requirement Analysis, Test Planning, Test Design, Environment Setup, Test Execution, Test Closure

Criterion	SDLC	STLC
Requirement Gathering Phase	Business analyst gathers the requirements and create Development Plan	QA team analyses requirement documents and create System Test Plan
Design Phase	The development team develops the high and low-level design of the software based on the requirements	Test Architect or a Test Lead usually plan the test strategy
Coding Phase	The actual code is developed as per the designed document	The QA team prepares the test environment
Testing Phase	Actual testing is done in this phase. It includes Unit, Integration, System, Retesting & Regression testing etc., Also the development team involves in fixing the bugs reported	Actual testing is done in this phase. Defect reporting & retesting is done here
Deployment or Maintenance Phase	The development team involves in support and release updates	The QA team executes regression suites to check maintenance code deployed
When it is performed	The SDLC phases are performed before the STLC phases	The STLC phases are performed after the SDLC phases
Outcome	A good quality software product	A bug free software

Difference between Quality Control vs Quality Assurance (QC vs QA)

Difference between Manual Testing & Automation Testing

Automation Testing	Manual Testing
Automated testing is more reliable. It performs same operation each time. It eliminates the risk of human errors.	Manual testing is less reliable. Due to human error, m testing is not accurate all the time.
Initial investment of automation testing is higher. Investment is required for testing tools. In the long run it is less expensive than manual. ROI is higher in the long run compared to Manual testing.	Initial investment of manual testing is less than autom Investment is required for human resources. ROI is lo the long run compared to Automation testing.
Automation testing is a practical option when we do	Manual testing is a practical option where

Automation Testing	Manual Testing
regressions testing.	the test cases are not run repeatedly and only needs to run once or twice.
Execution is done through software tools, so it is faster than manual testing and needs less human resources compared to manual testing.	Execution of test cases is time consuming and needs more human resources
Exploratory testing is not possible	Exploratory testing is possible
Performance Testing like Load Testing, Stress Testing etc. is a practical option in automation testing.	Performance Testing is not a practical option in manual testing
It can be done in parallel and reduce test execution time.	Its not an easy task to execute test cases in parallel in manual testing. We need more human resources to do this and becomes more expensive.
Programming knowledge is a must in automation testing	Programming knowledge is not required to do manual testing.
Build verification testing (BVT) is highly recommended	Build verification testing (BVT) is not recommended
Human intervention is not much, so it is not effective to do User Interface testing.	It involves human intervention, so it is highly effective to do User Interface testing.

Differences between Smoke Testing and Sanity Testing:

SMOKE TESTING	SANITY TESTING
Smoke Test is done to make sure if the build we received from the development team is testable or not	Sanity Test is done during the release phase to check for the main functionalities of the application without going deeper
Smoke Testing is performed by both Developers and Testers	Sanity Testing is performed by Testers alone
Smoke Testing exercises the entire application from end to end	Sanity Testing exercises only the particular

SMOKE TESTING	SANITY TESTING
	component of the entire application
Smoke Testing, build may be either stable or unstable	Sanity Testing, build is relatively stable
It is done on initial builds.	It is done on stable builds.
It is a part of basic testing.	It is a part of regression testing.
Usually it is done every time there is a new build release.	It is planned when there is no enough time to do in-depth testing.

Difference between Static and Dynamic testing?

Static Testing	Dynamic Testing
Static testing is also known as verification testing.	Dynamic testing is also known as validation testing.
It is done without executing the source code.	It is done by executing the source code.
It prevents the defects.	It finds and fixes the defects.
It involves checklist, and process to be followed.	It involves source code and test cases for execution.
It consists of reviews, walkthrough, inspection etc.,	It consists of functional, non-functional testing and data/control flow analysis.
When compared to dynamic testing, there are a lot of meetings.	When compared to static testing, the number of meetings is fewer.
It is less expensive than dynamic testing.	It is more expensive than static testing.
It generally takes less time.	It takes longer time to complete the procedure because it involves executing several test cases.
It is performed in the early stage of the SDLC.	It is performed at the later stage of the SDLC.

Static Testing	Dynamic Testing
It is performed before code deployment.	It is performed after code deployment.
It covers the structural and statement coverage testing.	It covers the executable file of the code.
It includes requirement document, design document, program specifications etc.,	It includes unit testing, integration testing, system testing, performance testing, security testing etc.,
Some of the tools used for Static testing are: - Checkstyle - Clang - Eclipse - Sonarqube - Soot - Source meter	Some of the tools used for dynamic testing are: - BoundsChecker - DroidBox - Diakon - Procmon - ValGrind

Test Strategy vs Test Plan

Test Strategy	Test Plan
Test Strategy is a high level document which captures the approach on how we go about testing the product and achieve the goals.	Test plan document is a document which contains the plan for all the testing activities to be done to deliver a qu product.
Components of Test strategy includes Scope and overview, Test Approach, Testing tools, Industry standards to follow, Test deliverables, Testing metrics, Requirement Traceability Matrix, Risk and mitigation, Reporting tool, Test summary	Components of Test plan includes Test Plan Identifier, Features To Be Tested, Features Not To Be Tested, Approach, Pass/Fail Criteria, Suspension Criteria, Test Deliverables, Responsibilities, Staffing and Training Needs, Risks and Contingencies etc.,
It is developed by the project manager	It is prepared by test lead or test manager
It is derived from the Business requirement specifications (BRS)	It is derived from the Product Description, SRS, or

Test Strategy	Test Plan
	Use Case documents
It is a static document. Once defined, it cannot be changed	It is a dynamic document. It can be changed
It is defined at organization level and can be used for other projects of similar nature	It is defined at project level

Difference between Test Case and Test Scenario

TEST CASE	TEST SCENARIO
Test case consists of Test case name, Precondition, Test steps, Expected result and Post condition	Test scenario are one liner but it is associated with multiple test cases
Test case guide a user on “how to test”	Test scenario guide a user on “what to test”
Purpose of test case is to validate the test scenario by executing a set of steps	Purpose of test scenario is to test end to end unctionality of a software application
Creating test cases is important when working with testers off-site	Creating test scenarios helps you in a time-sensitive situation (especially working in Agile)
Software applications change often. It leads to redesigning the pages and adding new functionalities. It hard to maintain test cases	Test scenarios are easy to maintain due to its high level design
More time consumption compared to test scenarios	Less time consumption compared to test cases
Required more resources to create and execute test cases	Relatively less resources enough to create and test using test scenarios
It helps in exhaustive testing of application	It helps in agile way of testing end to end functionality
Test cases are derived from test scenarios	Test scenarios are derived from use cases
Test cases are low level actions	Test scenarios are high level actions

TEST CASE	TEST SCENARIO
Test cases are written by Testers	Test scenarios are written by Test Leads, Business Analysts, and Testers.
Test case may or may not be associated to multiple Test scenarios.	Test Scenarios have multiple test cases.

Retesting vs Regression Testing

REGRESSION TESTING	RETESTING
Regression Testing is performed to make sure the code changes have not affected existing features.	Re-testing is performed to make sure that the test cases which failed earlier are passed after the defects are fixed.
Regression testing is carried out to verify whether there is any existing features are affected or not on the basis of new code changes.	Re-testing is carried out on the basis of the Defect fixes.
Defect verification won't fall under regression testing.	Defect verification falls under retesting.
Priority of regression testing is lower than retesting. Regression testing is carried out in parallel with retesting.	Priority of retesting is higher than regression testing. Retesting is carried out before Regression testing.
We can automate regression test cases. Manual regression testing is more expensive and time consuming.	We cannot automate the test cases for Retesting due to uncertainty.
Regression testing is a generic testing.	Re-testing is a planned testing.
We do regression testing on passed test cases.	We do retesting only on failed test cases.

REGRESSION TESTING	RETESTING
Regression testing verifies unexpected side effects.	Re-testing verifies whether original defects has been fixed or not.
Regression test cases are derived from the functional specification.	Test cases that are failed earlier are used in retesting.

What is Severity?

Bug/Defect severity can be defined as the impact of the bug on the application. It can be Critical, Major or Minor. In simple words, how much effect will be there on the system because of a particular defect

What are the types of Severity?

Critical:

A critical severity issue is an issue where a large piece of functionality or major system component is completely broken and there is no workaround to move further.
For example, Due to a bug in one module, we cannot test the other modules because that blocker bug has blocked the other modules. Bugs which affects the customers business are considered as critical

Major:

A major severity issue is an issue where a large piece of functionality or major system component is completely broken and there is a workaround to move further.

Minor:

A minor severity issue is an issue that imposes some loss of functionality, but for which there is an acceptable & easily reproducible workaround.

For example, font family or font size or color or spelling issue

Trivial:

A trivial severity defect is a defect which is related to the enhancement of the system

What is Priority?

Defect priority can be defined as an impact of the bug on the customers business. Main focus on how soon the defect should be fixed. It gives the order in which a defect should be resolved. Developers decide which defect they should take up next based on the priority. It can be High, Medium or Low.

What are the types of Priority?

High:

A high priority issue is an issue which has a high impact on the customers business or an issue which affects the system severely and the system cannot be used until the issue was fixed. These kinds of issues must be fixed immediately. Most of the cases as per the user perspective, the priority of the issue is set to high priority even though the severity of the issue is minor.

Medium:

Issues which can be released in the next build comes under medium priority. Such issues can be resolved along with other development activities.

Low:

An issue which has no impact on the customer business comes under low priority.

Difference between Functional Testing and Non-functional Testing?

Functional Testing	Non-functional Testing
What the system actually does is functional testing	How well the system performs is non-functionality testing
To ensure that your product meets customer and business requirements and doesn't have any major bugs	To ensure that the product stands up to customer expectations
To verify the accuracy of the software against expected output	To verify the behavior of the software at various load conditions
It is performed before non-functional testing	It is performed after functional testing
Example of functional test case is to verify the login functionality	Example of non-functional test case is to check whether the homepage is loading in less than 2 seconds
Testing types are • Unit testing • Smoke testing • User Acceptance • Integration Testing • Regression testing • Localization • Globalization • Interoperability	Testing types are • Performance Testing • Volume Testing • Scalability • Usability Testing • Load Testing • Stress Testing • Compliance Testing • Portability Testing • Disaster Recover Testing
It can be performed either manual or automated way	It can be performed efficiently if automated

Difference between Unit Testing & Integration Testing:

UNIT TESTING	INTEGRATION TESTING
Unit testing is the first level of testing in the Software Testing	Integration Testing is the second level of testing in Software Testing
Considers each component, as a single system	Integrated components are seen as a single system
Purpose is to test working of individual unit	Purpose is to test the integration of multiple unit modules
It evaluates the each component or unit of the software product	It examines the proper working, interface and reliability, after the integration of the modules, and along with the external interfaces and system
Scope of Unit testing is limited to a particular unit under test	Scope of Unit testing is wider in comparison to Unit testing. It covers two or more modules
It has no further types	It is divided into following approaches • Bottom up integration approach • Top down integration approach • Big Bang approach • Hybrid approach
It is also known as Component Testing	It is also known as I&T or String Testing
It is performed at the code level	It is performed at the communication level
It is carried out with the help of reusable test cases	It is carried out with the help of stubs and drivers
It comes under White Box Testing	It comes under both Black Box and White Box Testing
It is performed by developers	It is performed by either testers or developers

Big Bang Approach?

Combining all the modules once and verifying the functionality after completion of individual module testing. In Big Bang Integration Testing, the individual modules are not integrated until all the modules are ready. Then they will run to check whether it is performing well. In this type of testing, some disadvantages might occur like, defects can be found at the later stage. It would be difficult to find out whether the defect arouse in interface or in module.

What is Top-Down Approach?

In Top Down Integration Testing, testing takes place from top to bottom. High-level modules are tested first and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.

In this type of testing, Stubs are used as temporary module if a module is not ready for integration testing.

What is Bottom-Up Approach?

It is a reciprocate of the Top-Down Approach. In Bottom Up Integration Testing, testing takes place from bottom to up. Lowest level modules are tested first and then high-level modules and finally integrating the high-level modules to a low level to ensure the system is working as intended. Drivers are used as a temporary module for integration testing.

What is a Stub?

It is called by the Module under Test.

What is a Driver?

These terms (stub & driver) come into the picture while doing Integration Testing. While working on integration, sometimes we face a situation where some of the functionalities are still under development. So the functionalities which are under development will be replaced with some dummy programs. These dummy programs are named as Stubs or Drivers.

What is the difference between Unit Testing and Integration Testing?

UNIT TESTING	INTEGRATION TESTING
Unit testing is the first level of testing in the Software Testing	Integration Testing is the second level of testing in Software Testing
Considers each component, as a single system	Integrated components are seen as a single system
Purpose is to test working of individual unit	Purpose is to test the integration of multiple unit modules
It evaluates the each component or unit of the software product	It examines the proper working, interface and reliability, after the integration of the modules, and along with the external interfaces and system
Scope of Unit testing is limited to a particular unit under test	Scope of Unit testing is wider in comparison to Unit testing. It covers two or more modules
It has no further types	It is divided into following approaches • Bottom up integration approach • Top down integration approach • Big Bang approach • Hybrid approach
It is also known as Component Testing	It is also known as I&T or String Testing
It is performed at the code level	It is performed at the communication level
It is carried out with the help of reusable test cases	It is carried out with the help of stubs and drivers

UNIT TESTING	INTEGRATION TESTING
It comes under White Box Testing	It comes under both Black Box and White Box Testing
It is performed by developers	It is performed by either testers or developers

What is the difference between Integration Testing and System Testing?

INTEGRATION TESTING	SYSTEM TESTING
It is a low level testing	It is a high level testing
It is followed by System Testing	It is followed by Acceptance Testing
It is performed after unit testing	It is performed after integration testing
Different types of integration testing are: • Top bottom integration testing • Bottom top integration testing • Big bang integration testing • Sandwich integration testing	Different types of system testing are: • Regression testing • Sanity testing • Usability testing • Retesting • Load testing • Performance testing • Maintenance testing
Testers perform functional testing to validate the interaction of two modules	Testers perform both functional as well as non-functional testing to evaluate the functionality, usability, performance testing etc.,
Performed to test whether two different modules interact effectively with each other or not	Performed to test whether the product is performing as per user expectations and the required

INTEGRATION TESTING	SYSTEM TESTING
	specifications
It can be performed by both testers and developers	It is performed by testers
Testing takes place on the interface of two individual modules	Testing takes place on complete software application
Here, we validate the interace between the individual modules.	Here, we validate the finished product.
Testers need to understand the interlinked modules and their interaction.	Testers need to understand the internal structure and programming language.
It covers only functional testing.	It covers both functional and non-functional testing.

Difference between Validation and Verification

#SoftwareTestingMaterial

VERIFICATION VS VALIDATION

	
Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not.	Validation is the process, whether we are building the right product i.e., to validate the product which we have developed is right or not.
As per IEEE-STD-610: The process of evaluation software to determine whether the products of a given development phase satisfy the conditions imposed at the beginning of that phase.	As per IEEE-STD-610: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements [IEEE-STD-610]
It answers the question, Am I building a product right?	It answers the question, Am I building a right product?
Activities involved here are Inspections, Reviews, Walkthroughs	Activities involved in this is Testing the software application by means of White box, Gray box, & Black box testing.
It's a Low-Level Activity.	It's a High Level Activity.
It is a static method of checking documents and files.	It is a dynamic process of testing the real product.
It doesn't involve code execution	It involves code execution
Low cost compared to validation tests	Costly compared to verification tests

Entry Criteria:

The prerequisites that must be achieved before commencing the testing process.

Exit Criteria:

The conditions that must be met before testing should be concluded.

Performance Testing	Load testing	Stress testing
It is a superset of load and stress testing	It is a subset of performance testing	It is a subset of performance testing
Goal of performance testing is to set the benchmark and standards for the application	Goal of load testing is to identify the upper limit of the system, set SLA of the app and check how the system handles heavy load	Goal of stress testing is to find how the system behaves under extreme loads and how it recovers from failure
Load limit is both below and above the threshold of a break	Load limit is a threshold of a break	Load limit is above the threshold of a break
The attributes which are checked in performance testing are speed, response time, resource usage, stability, reliability and throughput	The attributes which are checked in a load testing are peak performance, server throughput, response time under various load levels, load balancing requirements etc.	The attributes which are checked in a stress testing are stability response time, bandwidth capacity etc.,

What is a defect?

The variation between the actual results and expected results is known as defect.

If a developer finds an issue and corrects it by himself in the development phase then it's called a defect.

What is a bug?

If testers find any mismatch in the application/system in testing phase then they call it as Bug.

As I mentioned earlier, there is a contradiction in the usage of Bug and Defect. People widely say the bug is an informal name for the defect.

What is an error?

We can’t compile or run a program due to coding mistake in a program. If a developer unable to successfully compile or run a program then they call it as an **error**.

What is a failure?

Once the product is deployed and customers find any issues then they call the product as a failure product. After release, if an end user finds an issue then that particular issue is called as **failure.**