SUBJECT CODE: 604 **SUBJECT NAME: SOFTWARE TESTING**

Unit 1 : Software Testing

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 the quality product.

Definition:
Software Testing Definition according to **ANSI/IEEE 1059** standard – A process of analyzing a software item to detect the differences between existing and required conditions (i.e., defects) and to evaluate the features of the software item.

Software Testing Types:
Manual Testing: Manual testing is the process of testing software by hand to learn more about it, to find what is and isn't working. This usually includes verifying all the features specified in requirements documents, but often also includes the testers trying the software with the perspective of their end user's in mind. Manual test plans vary from fully scripted test cases, giving testers detailed steps and expected results, through to high-level guides that steer exploratory testing sessions. There are lots of sophisticated tools on the market to help with manual testing, but if you want a simple and flexible place to start, take a look at Testpad.
Automation Testing: Automation testing is the process of testing the software using an automation tool to find the defects. In this process, testers execute the test scripts and generate the test results automatically by using automation tools. Some of the famous automation testing tools for functional testing are QTP/UFT and Selenium.

Testing Methods:
1. Static Testing
2. Dynamic Testing

Static Testing: It is also known as Verification in Software Testing. Verification is a static method of checking documents and files. 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.
Activities involved here are Inspections, Reviews, Walkthroughs

Dynamic Testing: It is also known as Validation in Software Testing. Validation is a dynamic process of testing the real product. 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.

Testing Approaches:
There are three types of software testing approaches.

1. White Box Testing
2. Black Box Testing
3. Grey Box Testing

- **White Box Testing:** It is also called as Glass Box, Clear Box, Structural Testing. White Box Testing is based on 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 is usually done at the unit level.
- **Black Box Testing:** It is also called as Behavioral/Specification-Based/Input-Output Testing. Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure.
- **Grey Box Testing:** Grey box is the combination of both White Box and Black Box Testing. The tester who works on this type of testing needs to have access to design documents. This helps to create better test cases in this process.

Testing Levels:
1. Unit Testing
2. Integration Testing
3. System Testing
4. Acceptance Testing

- **Unit Testing:** Unit Testing is done to check whether the individual modules of the source code are working properly. i.e. testing each and every unit of the application separately by the developer in the developer's environment. It is AKA Module Testing or Component Testing. To learn about Unit Testing, check out our detailed Unit Testing Guide
- **Integration Testing:** Integration Testing is the process of testing the connectivity or data transfer between a couple of unit tested modules. It is AKA I&T Testing or String Testing. It is subdivided into the Top-Down Approach, Bottom-Up Approach, and Sandwich Approach (Combination of Top-Down and Bottom-Up). To learn about Integration Testing, check out our detailed Integration Testing Guide
- **System Testing (end to end testing):** It's a black box testing. Testing the fully integrated application this is also called as an end to end scenario testing. To ensure that the software works in all intended target systems. Verify thorough testing of every input in the application to check for desired outputs. Testing of the user's experiences with the application.
- **Acceptance Testing:** To obtain customer sign-off so that software can be delivered and payments received. Types of Acceptance Testing are Alpha, Beta & Gamma Testing.

Types of Black Box Testing:

1. Functionality Testing
2. Non-functionality Testing

- **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.

- **Non-functional testing:** In simple words, how well the system performs is non-functionality testing. Non-functional testing refers to various aspects of the software such as performance, load, stress, scalability, security, compatibility, etc., The Main focus is to improve the user experience on how fast the system responds to a request.

Testing Artifacts:
Test Artifacts are the deliverables that are given to the stakeholders of a software project. A software project which follows SDLC undergoes the different phases before delivering to the customer. In this process, there will be some deliverables in every phase. Some of the deliverables are provided before the testing phase commences and some are provided during the testing phase and rest after the testing phase is completed.

Some of the test deliverables are as follows:

- Test plan
- Traceability matrix
- Test case
- Test script
- Test suite
- Test data or Test Fixture
- Test harness

Some of the reasons why software testing becomes a very significant and integral part in the field of information technology are as follows.
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. One of the reasons why software testing is necessary is to provide the best user experience.

3. Security
This is probably the most sensitive and vulnerable part of software 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 the quality product to the client as mentioned in the requirement specification documents.

Principles of Software Testing:

Testing of software consists of some principles that play a vital role while testing the project.

The Principles of Software Testing are as follows :

1. Testing shows the presence of defects
2. Exhaustive testing is impossible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context-dependent
7. Absence of error

Difference between Defect, Error, Bug, Failure and Fault :

Testing is the process of identifying defects, where a defect is any variance between actual and expected results. "A mistake in coding is called Error, error found by tester is called Defect, defect accepted by development team then it is called Bug, build does not meet the requirements then it Is Failure."

DEFECT: It can be simply defined as a variance between expected and actual. Defect is an error found AFTER the application goes into production. It commonly refers to several troubles with the software products, with its external behavior or with its internal features. In other words Defect is the difference between expected and actual result in the context of testing. It is the deviation of the customer requirement.

Defect can be categorized into the following:

Wrong: When requirements are implemented not in the right way. This defect is a variance from the given specification. It is Wrong!

Missing: A requirement of the customer that was not fulfilled. This is a variance from the specifications, an indication that a specification was not implemented, or a requirement of the customer was not noted correctly.

Extra: A requirement incorporated into the product that was not given by the end customer. This is always a variance from the specification, but may be an attribute desired by the user of the product. However, it is considered a defect because it's a variance from the existing requirements.

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

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

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

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

Software Testing Fundamentals

Different types of Software Testing processes are described below:

- **Unit Testing**
  It is a method by which individual units of source code are tested to determine if they are fit for use.
- **Integration Testing**
  Here individual software modules are combined and tested as a group.
- **Functionality Testing**
  It is a type of black box testing that bases its test cases on the specifications of the software component under test.
- **Usability Testing**
  It is a technique used to evaluate a product by testing it on users.
- **System Testing**
  It is testing conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirements.
- **Performance Testing**
  It is testing that is performed, to determine how fast some aspect of a system performs under a particular workload.

- **Load Testing**
  It refers to the practice of modeling the expected usage of a software program by simulating multiple users accessing the program concurrently.
- **Stress Testing**
  It is a form of testing that is used to determine the stability of a given system or entity.

Debugging :

In the context of software engineering, debugging is the process of fixing a bug in the software. In other words, it refers to identifying, analyzing and removing errors. This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software. It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

Debugging Process: Steps involved in debugging are:

- Problem identification and report preparation.
- Assigning the report to software engineer to the defect to verify that it is genuine.
- Defect Analysis using modeling, documentations, finding and testing candidate flaws, etc.
- Defect Resolution by making required changes to the system.
- Validation of corrections.

Debugging Strategies:

1. Study the system for the larger duration in order to understand the system. It helps debugger to construct different representations of systems to be debugging depends on the need. Study of the system is also done actively to find recent changes made to the software.
2. Backwards analysis of the problem which involves tracing the program backward from the location of failure message in order to identify the region of faulty code. A detailed study of the region is conducting to find the cause of defects.
3. Forward analysis of the program involves tracing the program forwards using breakpoints or print statements at different points in the program and studying the results. The region where the wrong outputs are obtained is the region that needs to be focused to find the defect.

4. Using the past experience of the software debug the software with similar problems in nature. The success of this approach depends on the expertise of the debugger.

Debugging Tools:

Debugging tool is a computer program that is used to test and debug other programs. A lot of public domain software like gdb and dbx are available for debugging. They offer console-based command line interfaces. Examples of automated debugging tools include code based tracers, profilers, interpreters, etc. Some of the widely used debuggers are:

- Radare2
- WinDbg
- Valgrind

Difference Between Debugging and Testing:

Debugging is different from testing. Testing focuses on finding bugs, errors, etc whereas debugging starts after a bug has been identified in the software. Testing is used to ensure that the program is correct and it was supposed to do with a certain minimum success rate. Testing can be manual or automated. There are several different types of testing like unit testing, integration testing, alpha and beta testing, etc.

Debugging requires a lot of knowledge, skills, and expertise. It can be supported by some automated tools available but is more of a manual process as every bug is different and requires a different technique, unlike a pre-defined testing mechanism.

Unit 2 : Approaches to Testing -I

White Box Testing

WHITE BOX TESTING (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

- **White-box testing:** Testing based on an analysis of the internal structure of the component or system.
- **White-box test design technique:** Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system.

Example

A tester, usually a developer as well, studies the implementation code of a certain field on a webpage, determines all legal (valid and invalid) AND illegal inputs and verifies the outputs against the expected outcomes, which is also determined by studying the implementation code.

White Box Testing is like the work of a mechanic who examines the engine to see why the car is not moving.

White Box Testing method is applicable to the following levels of software testing:

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

However, it is mainly applied to Unit Testing.

Advantages

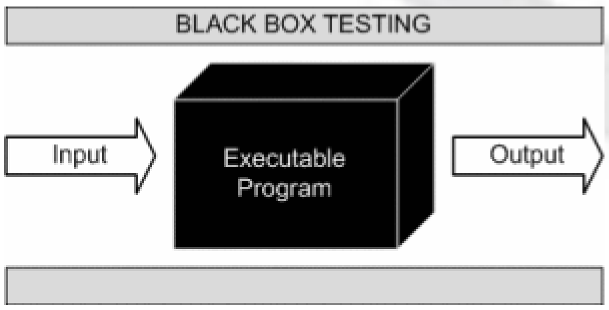
- Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
- Testing is more thorough, with the possibility of covering most paths.

Disadvantages

- Since tests can be very complex, highly skilled resources are required, with a thorough knowledge of programming and implementation.
- Test script maintenance can be a burden if the implementation changes too frequently.
- Since this method of testing is closely tied to the application being tested, tools to cater to every kind of implementation/platform may not be readily available.

BLACK BOX TESTING

BLACK BOX TESTING, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

  Black box testing: Testing, either functional or non-functional, without reference to the internal structure of the component or system.

  Black box test design technique: Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure.

Example

A tester, without knowledge of the internal structures of a website, tests the web pages by using a browser; providing inputs (clicks, keystrokes) and verifying the outputs against the expected outcome.

Black Box Testing method is applicable to the following levels of software testing:

- Integration Testing
- System Testing
- Acceptance Testing

The higher the level, and hence the bigger and more complex the box, the more black-box testing method comes into use.

Techniques

Following are some techniques that can be used for designing black box tests.

- ***Equivalence Partitioning***: It is a software test design technique that involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data.

- ***Boundary Value Analysis:*** It is a software test design technique that involves the determination of boundaries for input values and selecting values that are at the boundaries and just inside/ outside of the boundaries as test data.
- ***Cause-Effect Graphing:*** It is a software test design technique that involves identifying the cases (input conditions) and effects (output conditions), producing a Cause-Effect Graph, and generating test cases accordingly.

Advantages

- Tests are done from a user's point of view and will help in exposing discrepancies in the specifications.
- Tester need not know programming languages or how the software has been implemented.
- Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias.
- Test cases can be designed as soon as the specifications are complete.

Disadvantages

- Only a small number of possible inputs can be tested and many program paths will be left untested.
- Without clear specifications, which is the situation in many projects, test cases will be difficult to design.
- Tests can be redundant if the software designer/developer has already run a test case.

Gray Box Testing

GRAY BOX TESTING is a software testing method which is a combination of Black Box Testing method and White Box Testing method. In Black Box Testing, the internal structure of the item being tested is unknown to the tester and in White Box Testing the internal structure is known. In Gray Box Testing, the internal structure is partially known. This involves having access to internal data structures and algorithms for purposes of designing the test cases, but testing at the user, or black-box level.

Gray Box Testing is named so because the software program, in the eyes of the tester is like a gray/semi-transparent box; inside which one can partially see.

Example

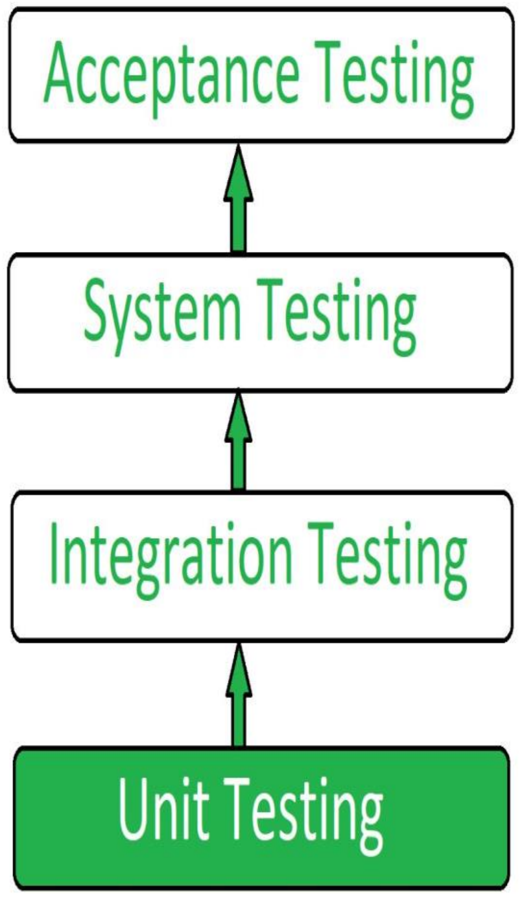
An example of Gray Box Testing would be when the codes for two units/modules are studied (White Box Testing method) for designing test cases and actual tests are conducted using the exposed interfaces (Black Box Testing method).

Unit Testing

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent modules are tested to determine if there are any issue by the developer himself. It is correlated with functional correctness of the independent modules.
Unit Testing is defined as a type of software testing where individual components of a software are tested.
Unit Testing of software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer.

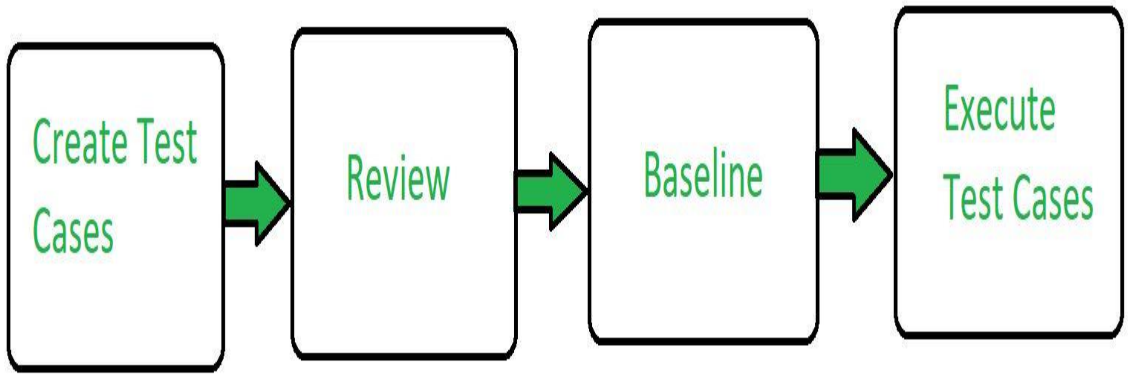
In SDLC or V Model, Unit testing is first level of testing done before integration testing. Unit testing is such type of testing technique that is usually performed by the developers. Although due to reluctance of developers to tests, quality assurance engineers also do unit testing.

Objective of Unit Testing:
The objective of Unit Testing is:
1. To isolate a section of code.
2. To verify the correctness of code.
3. To test every function and procedure.
4. To fix bug early in development cycle and to save costs.
5. To help the developers to understand the code base and enable them to make changes quickly.
6. To help for code reuse.



Types of Unit Testing:
There are 2 type of Unit Testing: **Manual**, and **Automated**.

Workflow of Unit Testing:



Unit Testing Tools:

Here are some commonly used Unit Testing tools:

1. Jtest
2. Junit
3. NUnit
4. EMMA
5. PHPUnit

Advantages of Unit Testing:

- Unit Testing allows developers to learn what functionality is provided by a unit and how to use it to gain a basic understanding of the unit API.
- Unit testing allows the programmer to refine code and make sure the module works properly.

Integration testing

The meaning of Integration testing is quite straightforward- *Integrate/combine the unit tested module one by one and test the behavior as a combined unit.*

The main function or goal of this testing is to test the interfaces between the units/modules.

We normally do Integration testing after "Unit testing". Once all the individual units are created and tested, we start combining those "Unit Tested" modules and start doing the integrated testing.

The main function or goal of this testing is to test the interfaces between the units/modules.

The individual modules are first tested in isolation. Once the modules are unit tested, they are integrated one by one, till all the modules are integrated, to check the combinational behavior, and validate whether the requirements are implemented correctly or not.

Here we should understand that Integration testing does not happen at the end of the cycle, rather it is conducted simultaneously with the development. So in most of the times, all the modules are not actually available to test and here is what the challenge comes to test something which does not exist!

Why Integration Test?

We feel that Integration testing is complex and requires some development and logical skill. That's true! Then what is the purpose of integrating this testing into our testing strategy?

Some reasons:

1. In the real world, when applications are developed, it is broken down into smaller modules and individual developers are assigned 1 module. The logic implemented by one developer is quite different than another developer, so it becomes important to check whether the logic implemented by a developer is as per the expectations and rendering the correct value in accordance with the prescribed standards.
2. Many a time the face or the structure of data changes when it travels from one module to another. Some values are appended or removed, which causes issues in the later modules.
3. Modules also interact with some third party tools or APIs which also need to be tested that the data accepted by that API / tool is correct and that the response generated is also as expected.
4. A very common problem in testing – Frequent requirement change! :) Many a time developer deploys the changes without unit testing it. Integration testing becomes important at that time.

Advantages

- This testing makes sure that the integrated modules/components work properly.
- Integration testing can be started once the modules to be tested are available. It does not require the other module to be completed for testing to be done, as Stubs and Drivers can be used for the same.
- It detects the errors related to the interface.
-

Challenges Listed below are few challenges that are involved in Integration Test.

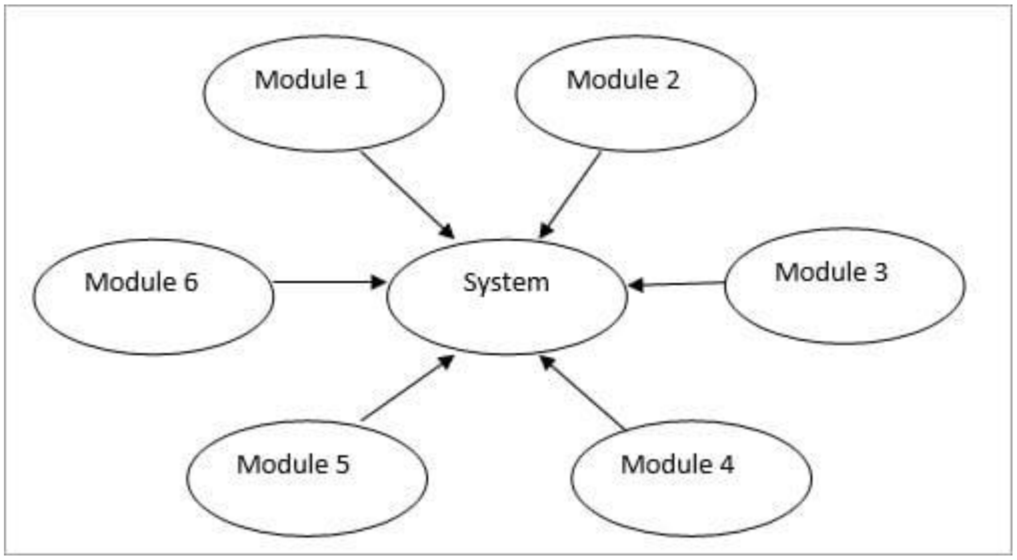
1) Integration testing means testing two or more integrated systems in order to ensure that the system works properly. Not only the integration links should be tested but an exhaustive testing considering the environment should be done to ensure that the integrated system works properly.

There might be different paths and permutations which can be applied to test the integrated system.

2) Managing Integration testing becomes complex because of few factors involved in it like the database, Platform, environment etc.
3) While integrating any new system with the legacy system, it requires a lot of changes and testing efforts. Same applies while integrating any two legacy systems.
4) Integrating two different systems developed by two different companies is a big challenge as for how one of the systems will impact the other system if any changes are done in any one of the systems is not sure.
In order to minimize the impact while developing a system, few things should be taken into consideration like possible integration with other systems, etc.

Types of Integration Testing
Big Bang Approach:
Big bang approach integrates all the modules in one go i.e. it does not go for integrating the modules one by one. It verifies if the system works as expected or not once integrated. If any issue is detected in the completely integrated module, then it becomes difficult to find out which module has caused the issue.

Big bang approach is a time-consuming process of finding a module which has a defect itself as that would take time and once the defect is detected, fixing the same would cost high as the defect is detected at the later stage.



Advantages of Big Bang approach:
- It is a good approach for small systems.

Disadvantages of Big Bang Approach:
- It is difficult to detect the module which is causing an issue.

- Big Bang approach requires all the modules all together for testing, which in turn, leads to less time for testing as designing, development, Integration would take most of the time.
- Testing takes place at once only which thereby leaves no time for critical module testing in isolation.

Integration Testing Steps:

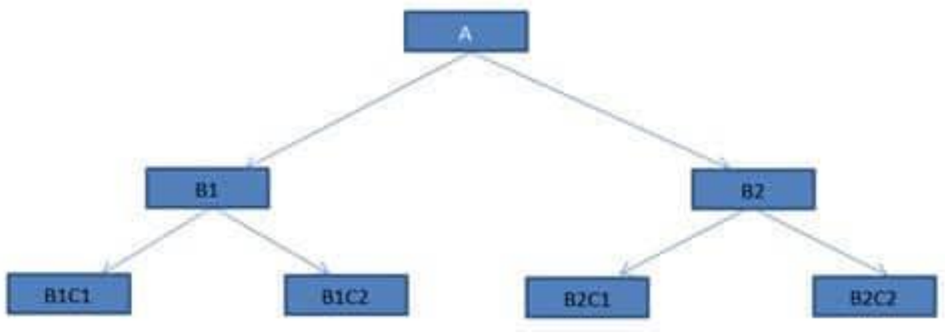
1. Prepare Integration Test Plan.
2. Prepare integration test scenarios & test cases.
3. Prepare test automation scripts.
4. Execute test cases.
5. Report the defects.
6. Track and re-test the defects.
7. Re-testing & testing goes on until integration testing is complete.

Test Integration Approaches

There are fundamentally 2 approaches for doing test integration:

1. **Bottom-up approach**
2. **Top-down approach.**

Let’s consider the below figure to test the approaches:



Bottom-up approach:

Bottom-up testing, as the name suggests starts from the lowest or the innermost unit of the application, and gradually moves up. The Integration testing starts from the lowest module and gradually progresses towards the upper modules of the application. This integration continues till all the modules are integrated and the entire application is tested as a single unit.

In this case, modules B1C1, B1C2 & B2C1, B2C2 are the lowest module which is unit tested. Module B1 & B2 are not yet developed. The functionality of Module B1 and B2 is that it calls the modules B1C1, B1C2 & B2C1, B2C2. Since B1 and B2 are not yet developed, we would need some program or a “stimulator” which

will call the B1C1, B1C2 & B2C1, B2C2 modules. These stimulator programs are called **DRIVERS**.
In simple words, **DRIVERS** are the dummy programs which are used to call the functions of the lowest module in a case when the calling function does not exist. The bottom-up technique requires module driver to feed test case input to the interface of the module being tested.
The advantage of this approach is that, if a major fault exists at the lowest unit of the program, it is easier to detect it, and corrective measures can be taken.

The disadvantage is that the main program actually does not exist until the last module is integrated and tested. As a result, the higher level design flaws will be detected only at the end.

Top-down approach
This technique starts from the topmost module and gradually progress towards the lower modules. Only the top module is unit tested in isolation. After this, the lower modules are integrated one by one. The process is repeated until all the modules are integrated and tested.

In the context of our figure, testing starts from Module A, and lower modules B1 and B2 are integrated one by one. Now here the lower modules B1 and B2 are not actually available for integration. So in order to test the topmost modules A, we develop "**STUBS**".
"Stubs" can be referred to as code a snippet which accepts the inputs/requests from the top module and returns the results/ response. This way, in spite of the lower modules, do not exist, we are able to test the top module.

In practical scenarios, the behavior of stubs is not that simple as it seems. In this era of complex modules and architecture, the called module, most of the time involves complex business logic like connecting to a database. As a result, creating Stubs becomes as complex and time taking as the real module. In some cases, Stub module may turn out to be bigger than the stimulated module.

Both Stubs and drivers are dummy piece of code which is used for testing the "non- existing" modules. They trigger the functions/method and return the response, which is compared to the expected behavior

Let's conclude some difference between Stubs and Driver:

Stubs	**Driver**
Used in Top down approach	Used in Bottom up approach
Top most module is tested first	Lowest modules are tested first.
Stimulates the lower level of components	Stimulates the higher level of components
Dummy program of lower level components	Dummy program for Higher level component

The only change is Constant in this world, so we have another approach called "**Sandwich testing**" which combines the features of both Top-down and bottom-up approach. When we test huge programs like Operating systems, we have to have some more techniques which are efficient and boosts more confidence. Sandwich testing plays a very important role here, where both, the Top down and bottom up testing are started simultaneously.
Integration starts with the middle layer and moves simultaneously towards up and down. In case of our figure, our testing will start from B1 and B2, where one arm will test the upper module A and another arm will test the lower modules B1C1, B1C2 & B2C1, B2C2.

Since both the approach starts simultaneously, this technique is a bit complex and requires more people along with specific skill sets and thus adds to the cost.

GUI application Integration Test
Now let's talk about how we can imply integration testing in Black box technique.

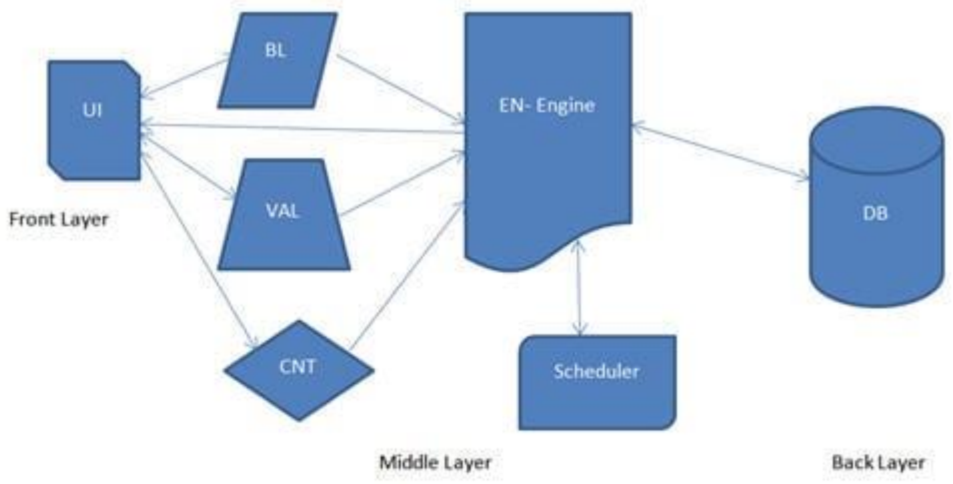
We all understand that a web application is a multitier application. We have a front end which is visible to the user, we have a middle layer which has business logic, we have some more middle layer which does some validations, integrate some third party APIs etc., then we have the back layer which is the database.

Integration testing example:

Let's check the below example:

I am the owner of an advertising company and I post ads on different websites. At the end of the month, I want to see how many people saw my ads and how many people clicked on my ads. I need a report for my ads displayed and I charge accordingly to my clients.

GenNext software developed this product for me and below was the architecture:



UI – User Interface module, which is visible to the end user, where all the inputs are given.
BL – Is the Business Logic module, which has all the all the calculations and business specific methods.
VAL – Is the Validation module, which has all the validations of the correctness of the input.
CNT – Is the content module which has all the static contents, specific to the inputs entered by the user. These contents are displayed in the reports.
EN – Is the Engine module, this module reads all the data that comes from BL, VAL and CNT module and extracts the SQL query and triggers it to the database.
Scheduler – Is a module which schedules all the reports based on the user selection (monthly, quarterly, semiannually & annually)
DB – Is the Database.
Now, having seen the architecture of the entire web application, as a single unit, Integration testing, in this case, will focus on the flow of data between the modules.

Steps to Kick off Integration Tests

1. Understand the architecture of your application.
2. Identify the modules
3. Understand what each module does
4. Understand how the data is transferred from one module to another.
5. Understand how the data is entered and received into the system (entry point and exit point of the application)
6. Segregate the application to suit your testing needs.
7. Identify and create the test conditions
8. Take one condition at a time and write down the test cases.

Entry/Exit Criteria for Integration Testing

Entry Criteria:

- Integration test plan document is signed off and approved.
- Integration test cases have been prepared.
- Test data has been created.
- Unit testing of developed modules/Components is complete.
- All the critical and high Priority defects are closed.
- The test environment is set up for integration.

Exit Criteria:

- All the integration test cases have been executed.
- No critical and Priority P1 & P2 defects are opened.
- Test Report has been prepared.

Integration Test Cases

Integration test cases focus mainly on the **interface between the modules, integrated links, data transfer** between the modules as modules/components that are already unit tested i.e. the functionality and the other testing aspects have already been covered.

So, the main idea is to test if integrating two working modules works as expected when integrated.

For Example Integration Test cases for Linkedin application will include:

- Verifying the interface link between the login page and the home page i.e. when a user enters the credentials and logs it should be directed to the homepage.
- Verifying the interface link between the home page and the profile page i.e. profile page should open up.
- Verify the interface link between the network page and your connection pages i.e. clicking accept button on Invitations of the network page should show the accepted invitation in your connection page once clicked.
- Verify the interface link between the Notification pages and say congrats button i.e. clicking say congrats button should direct towards the new message window.

Many integration test cases can be written for this specific site. The above four points are just an example to understand what Integration test cases are included in testing.

Is Integration a White box or Black box Technique?

Integration testing technique can be counted in both black boxes as well as white box technique. Black box technique is where a tester need not have any internal knowledge of the system i.e. coding knowledge is not required whereas white box technique needs internal knowledge of the application.
Now while performing integration testing it could include testing the two integrated web services which will fetch the data from the database & provide the data as required which means it could be tested using white box testing technique whereas integrating a new feature in the website can be tested using the black box technique.

So, it’s not specific that integration testing is a black box or white box technique.

Integration Testing Tools

There are several tools available for this testing.

Given below is a list of tools:

- Rational Integration Tester
- Protractor
- Steam
- TESSY

For more details on the above tools check this tutorial:

System Integration Testing

System Integration Test is done to test the **complete integrated system**.

Modules or components are tested individually in unit testing before integrating the components.

Once all the modules are tested, system integration testing is done by integrating all the modules and the system as a whole is tested.

Difference between Integration Testing & System Testing
Integration testing is a testing in which one or two modules which are unit tested are integrated to test and verification is done to verify if the integrated modules work as expected or not.

System testing is a testing where the **system as a whole** is tested i.e. all the modules/components are integrated together to verify whether the system works as expected and no issues occur because of the integrated modules.

Unit 3 : Testing for Specialized Environment

Test Environment for Software Testing

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured.

Test bed or test environment is configured as per the need of the Application Under Test. On a few occasion, test bed could be the combination of the test environment and the test data it operates.

For the test environment, a key area to set up includes

- System and applications
- Test data
- Database server
- Front-end running environment
- Client operating system
- Browser
- Hardware includes Server Operating system
- Network
- Documentation required like reference documents/configuration guides/installation guides/ user manuals

Process of Software Test environment setup

Tests are limited to what can be tested and what not should be tested.

Following people are involved in test environment setup

- System Admins,
- Developers
- Testers
- Sometimes users or techies with an affinity for testing.

The test environment requires setting up of various number of distinct areas like,

Setup of Test Server

Every test may not be executed on a local machine. It may need establishing a test server, which can support applications.

For example, Fedora set up for PHP, Java-based applications with or without mail servers, cron set up, Java-based applications, etc.

Network

Network set up as per the test requirement. It includes,

- Internet setup
- LAN Wifi setup
- Private network setup

It ensures that the congestion that occurs during testing doesn't affect other members. (Developers, designers, content writers, etc.)

Test PC setup

For web testing, you may need to set up different browsers for different testers. For desktop applications, you need various types of OS for different testers PCs.

For example, windows phone app testing may require

- Visual Studio installation
- Windows phone emulator
- Alternatively, assigning a windows phone to the tester.

Bug Reporting

Bug reporting tools should be provided to testers.

Creating Test Data for the Test Environment

Many companies use a separate test environment to test the software product. The common approach used is to copy production data to test. This helps the tester, to detect the same issues as a live production server, without corrupting the production data.

The approach for copying production data to test data includes,

- Set up production jobs to copy the data to a common test environment
- All PII (Personally Identifiable Information) is modified along with other sensitive data. The PII is replaced with logically correct, but non-personal data.
- Remove data that is irrelevant to your test.

Testers or developers can copy this to their individual test environment. They can modify it as per their requirement.

Privacy is the main issue in copy production data. To overcome privacy issues you should look into obfuscated and anonymized test data.

For Anonymization of data two approaches can be used,

- BlackList: In this approach, all the data fields are left unchanged. Except those fields specified by the users.
- WhiteList: By default, this approach, anonymizes all data fields. Except for a list of fields which are allowed to be copied. A whitelisted field implies that it is okay to copy the data as it is and anonymization is not required.

Also, if you are using production data, you need to be smart about how to source data. Querying the database using SQL script is an effective approach.

Test Environment Management

Test Environment Management deals with the maintenance and upkeep of the test bed.

List of activities by the Test environment management function include,

1. Maintenance of a central repository with all the updated version of test environments.
2. Test environment management as per the test team demands.
3. As per the new requirements creating new environments
4. Monitoring of the environments
5. Updating/deleting outdated test-environments
6. Investigation of issues on the environment
7. Co-ordination till an issue resolution.

Besides these, there are a few more questions to answer before setting up the test environment.

- Whether to develop an internal Test Environment or to outsource?
- Whether to follow an internal company standard or follow any External (IEE, ISO, etc.)?
- How long the test environment is required?
- Differences between the test and production systems and their impact on test validity must be determined.
- Can you re-use an existing setup for other projects in the company?

Challenges in setting up Test Environment Management

1. **Proper planning on resource usage**

   Ineffective planning for resource usage can affect the actual output. Also, it may lead to conflict between teams.

2. **Remote environment**

   It is possible that a Test environment is located geographically apart. In such a case, the testing team has to rely on the support team for various test assets. (Software, hardware, and other issues).

3. **Elaborate setup time**

   Sometimes test set up gets too elaborated in cases of Integration Testing.

4. **Shared usage by teams**

   If the testing environment is used by development & testing team simultaneously, test results will be corrupted.

5. **Complex test configuration**

   Certain test requires complex test environment configuration. It may pose a challenge to the test team.

Summary:

- A testing environment is a setup of software and hardware on which the test team will conduct the testing
- For the test environment, a key area to set up includes
  - System and applications

- Test data
- Database server
- Front-end running environment, etc.

- Few challenges while setting up a test environment include,
    - Remote environment
    - Combined usage between teams
    - Elaborate setup time
    - Ineffective planning for resource usage for integration
    - Complex test configuration

Client-Server And Web Based Testing

CLIENT / SERVER TESTING

This type of testing usually done for 2 tier applications (usually developed for LAN). Here we will be having Front-end and Backend.

The application launched on front-end will be having forms and reports which will be monitoring and manipulating data

For Example, applications developed in VB, VC++, Core Java, C, C++, D2K, PowerBuilder, etc., The backend for these applications would be MS Access, SQL Server, Oracle, Sybase, Mysql, Quadbase

The tests performed on these types of applications would be

- User Interface Testing
- Manual Support Testing
- Functionality Testing
- Compatibility Testing & Configuration Testing
- Intersystem Testing

WEB TESTING

This is done for 3 tier applications (developed for Internet / intranet / xtranet) Here we will be having Browser, web server and DB server.

The applications accessible in the browser would be developed in HTML, DHTML, XML, JavaScript, etc. (We can monitor through these applications)

Applications for the webserver would be developed in Java, ASP, JSP, VBScript, JavaScript, Perl, Cold Fusion, PHP, etc. (All the manipulations are done on the webserver with the help of these programs developed)

The DB server would be having Oracle, SQL Server, Sybase, MySQL, etc. (All data is stored in the database available on the DB server)

The tests performed on these types of applications would be

- User Interface Testing
- Functionality Testing
- Security Testing
- Browser Compatibility Testing
- Load/Stress Testing
- Interoperability Testing/Intersystem Testing
- Storage and Data Volume Testing

A Web Application is a Three-Tier Application

This has a browser (monitors data) [monitoring is done using HTML, Dhtml, XML, javascript]-> webserver (manipulates data) [manipulations are done using programming languages or scripts like adv java, asp, JSP, VBScript, javascript, Perl, ColdFusion, php] -> database server (stores data) [data storage and retrieval is done using databases like Oracle, SQL Server, Sybase, mysql].

The types of tests, which can be applied to this type of applications, are

- User Interface Testing for validation & user-friendliness
- Functionality Testing to validate behaviors, i/p, error handling, o/p, manipulations, services levels, the order of functionality, links, content of web page & backend coverage’s
- Security Testing
- Browser Compatibility
- Load/Stress Testing
- Interoperability Testing
- Storage & Data Volume Testing

A Client-Server Application is a Two-Tier Application

This has forms & reporting at front-end (monitoring & manipulations are done) [using vb, vc++, core java, c, c++, d2k, power builder etc.,] -> database server at the backend [data storage & retrieval) [using ms access, SQL Server, Oracle, Sybase, MySQL, quad base etc.,]

The tests performed on these applications would be

- User Interface testing
- Manual Support Testing
- Functionality Testing
- Compatibility Testing
- Intersystem Testing

Some more points to clear the difference between Client-Server, Web and Desktop applications:

Desktop Application:

- Application runs in single memory (Front end and Back end in one place)
- Single user only

Client/Server Application:

- Application runs in two or more machines
- Application is a menu-driven
- Connected mode (connection exists always until logout)
- A limited number of users
- Less number of network issues when compared to the web app.

Web Application:

- Application runs in two or more machines
- URL-driven
- Disconnected mode (stateless)
- Unlimited number of users
- Many issues like Hardware Compatibility, Browser Compatibility, Version Compatibility, Security Issues, performance issues, etc.

As per difference in both, the applications come where, how to access the resources. In Client-Server, once the connection is made it will be in the state on connected, whereas in case of web testing HTTP protocol is stateless, then there comes logic of cookies, which is not in Client-Server.

For Client-Server application users are well known, whereas for web application any user can log in and access the content, he/she will use it as per his intentions.

So, there are always issues of security and compatibility for a Web Application.

Testing Documentation

Documentation as part of process

Testing documentation is usually associated with the documentation of artifacts that should be developed before or during the testing of software.

Testing Documentation is an important part of the testing process.

Documentation for software testing is necessary for evaluating the testing effort needed, requirement tracking/tracing, test coverage etc.

Further are described commonly used documented artifacts concerning software testing:

- Test Plan.
- Test Case.
- Test Scenario.
- Traceability Matrix.

Test Plan

A **Test plan** outlines the common strategy that will be applied to test an application, the resources that will be used, the test environment in which testing will be performed, and the schedule of testing activities along with the limitations. Typically writing a Test Plan is the competence of the Quality Assurance Team Lead.

Test plan outlines the common strategy that will be applied to test an application.

A test plan includes the following:

- Introduction to the Test Plan document.
- Assumptions while testing the application.
- Approaches to be used while testing the software/application.
- List of test cases used during the testing process.
- Enumeration of features to be tested.
- List of deliverables to be tested.
- The resources allocated for testing the application.
- A schedule of tasks and milestones to be reached.
- Possible risks involved during the testing process.

Test Case

Test case is a complexity of inputs, serious of steps, and conditions that can be used during the process of testing. The key point of this activity is to find out whether a software is successful in terms of its functionality and other aspects. There are various types of test cases such as logical, functional, error, negative test cases, physical test cases, UI test cases, etc.

What's more, test cases are written to keep track of the software testing coverage. Basically, there are no formal templates that can be used while writing a test case.

Test case is a complexity of inputs, serious of steps, and conditions that can be used during the process of testing.

However, the following components are commonly accepted:

- Test case ID.
- Product module.
- Product version.
- Revision history.
- Purpose.
- Assumptions.
- Pre-conditions.
- Steps.
- Expected outcome.
- Actual outcome.
- Post-conditions.

A single test scenario can be applied to many test cases. In addition, sometimes numerous test cases are written for a single software which are collectively known as test suites.

Test Step – is the smallest unit of testing. This is the minimum action which is done by a tester during testing. As an example, a good test scenario is the wish to brush your teeth. Rather simple and clear wish, I may say. The test case can be performed by the process of squeezing a toothpaste from the tube. This process can easily be divided into smaller steps, such as: taking the tube, unscrewing the cap, pressing it until there is enough toothpaste etc.

Test Step – is the smallest unit of testing.

Test Scenario

Test Scenario - it is a one line statement which purpose is to inform what particular area in the application will be tested. Test scenarios are used to ensure that all process flows are thoroughly tested. For a particular area of an application a single test scenario may be enough, and for others the number may go up to a few hundred scenarios depending on the significance and complexity of the application.

Test scenarios are used to ensure that all process flows are thoroughly tested.

The terms 'test scenario' and 'test cases' are sometimes used interchangeably, however a test scenario usually consists of several steps, whereas a test case has a single step. Taking this into account, test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Moreover, each test relies on the output from the previous test.

Traceability Matrix

Traceability Matrix (also referred to as Requirement Traceability Matrix - RTM) is a table that is used to trace the requirements during the Software Development Life Cycle. It can be applied in various usages: forward tracing (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements). Different user-defined templates for RTM may be used.

Traceability Matrix is a table that is used to trace the requirements during the Software Development Life Cycle.

Each requirement in the RTM document is linked with its associated test case so that testing can be done as per the mentioned requirements. Moreover, Bug ID is also included and linked with its associated requirements and test case.

The main goals for this matrix are:

- Verification that the software is developed as per the mentioned requirements.
- Identification the root cause of any bug.
- Tracing the developed documents during different phases of SDLC.

What is meant by the term "Real Time Testing"?

In the field of software testing, real time testing refers to the process of testing the real time software product or system i.e. the system is constrained by some specific time limits to perform and complete the job. Air Traffic Control and space navigation system, may be seen as the examples of real time system, i.e. they responses in real time.

Generally, it involves, testing of the software product in its running/operational mode to evaluate its capability, to execute and finish a particular in a given stipulated time. The chief purpose behind the real time testing is to drastically minimize the probability of software's failure in real time environment at the user site. It is done to ensure that all the bugs and defects have been removed or fixed before its delivery to the client or user.

Difficulties in Real time testing

First of all, real time systems are associated with the time-constraints as such testing the system endorsed with the deadlines and real-time responses may seems to be a tedious & complex job. Moreover, the results being generated by the real time systems are non –predictable and have non-deterministic outcomes as such conventional methods may not prove to be effective to test the real time systems. Further, analyzing and testing the real time systems may be carried out in various ways as such it would not recommendable to rely on one strategy or approach, i.e. it would need exhaustive type of testing to evaluate the real time system and thereby, may prove to be a time-consuming and expensive process.

How to do it?

Real time testing technique follow a simple and basic strategy to execute the testing task, as described below:

- **Task Testing**

  Involves the implementation and usage of conventional static techniques to discover errors and flaws in the programming code such as syntax error. It, usually, overlooks the behavorial aspect of the software product along with the sequence of the action as it does not faces the time constraint.

- **Behavioral Testing**

  The simulated real environment is being created based on the design model with the help of automated tools to study the behaviour of the system under the impact of external forces.

- **Intertask Testing**

  After getting through the task of revealing errors in code and behavorial aspect of the system, time-constraints are being introduced through intertask testing.

- **System Testing**

  Testing the completely integrated system as a single whole system to explore the defects & issues occurred during the integration process and in the interfaces of the integrated software and hardware.

  Tools

Below mentioned are some of the testing tools to perform testing on real time systems.

- Messange Sequence Chart (MSC)
- Specification and Description Language (SDL)
- Testing and Test Control Notation (TTCN)
- TTCN-3

Unit 4 : Software Testing Strategies and Software Matrices

Strategies of Software Testing

- Testing is a set of activities which are decided in advance i.e before the start of development and organized systematically.
- In the literature of software engineering various testing strategies to implement the testing are defined.
- All the strategies give a testing template.

Following are the characteristic that process the testing templates:

- The developer should conduct the successful technical reviews to perform the testing successful.
- Testing starts with the component level and work from outside toward the integration of the whole computer based system.
- Different testing techniques are suitable at different point in time.
- Testing is organized by the developer of the software and by an independent test group.
- Debugging and testing are different activities, then also the debugging should be accommodated in any strategy of testing.

Verification	Validation
Verification is the process to find whether the software meets the specified requirements for particular phase.	The validation process is checked whether the software meets requirements and expectation of the customer.
It estimates an intermediate product.	It estimates the final product.
The objectives of verification is to check whether software is constructed according to requirement and design specification.	The objectives of the validation is to check whether the specifications are correct and satisfy the business need.
It describes whether the outputs are as per the inputs or not.	It explains whether they are accepted by the user or not.
Verification is done before the validation.	It is done after the verification.
Plans, requirement, specification, code are evaluated during the verifications.	Actual product or software is tested under validation.
It manually checks the files and document.	It is a computer software or developed program based checking of files and document.

Difference between Verification and Validation

Strategy of testing

A strategy of software testing is shown in the context of spiral.

Following figure shows the testing strategy:

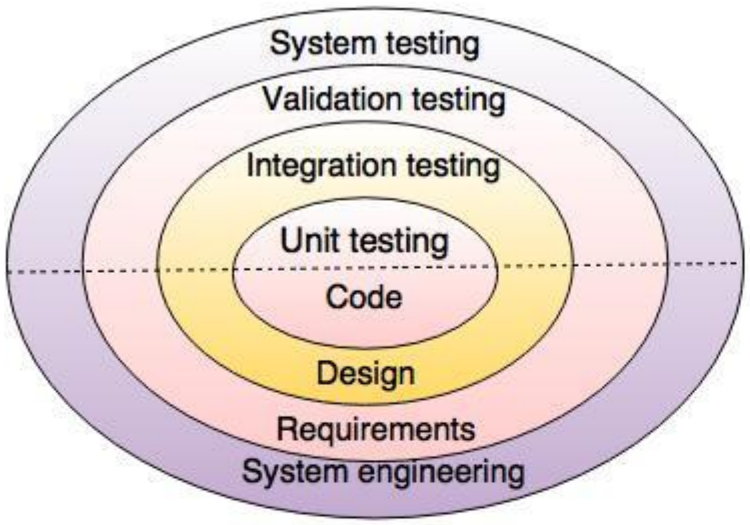


Fig. - Testing Strategy

Unit testing
Unit testing starts at the centre and each unit is implemented in source code.

Integration testing
An integration testing focuses on the construction and design of the software.

Validation testing
Check all the requirements like functional, behavioral and performance requirement are validate against the construction software.

System testing
System testing confirms all system elements and performance are tested entirely.

Testing strategy for procedural point of view

As per the procedural point of view the testing includes following steps.

1) Unit testing
2) Integration testing
3) High-order tests
4) Validation testing

These steps are shown in following figure:

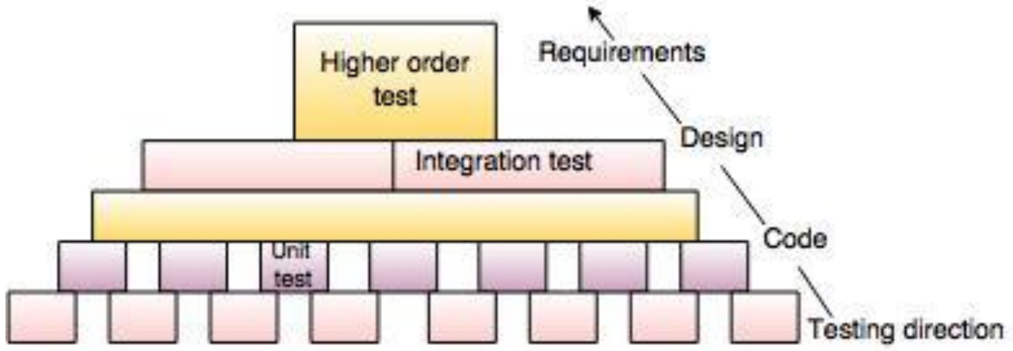


Fig.- Steps of software testing

Following are the issues considered to implement software testing strategies.

- Specify the requirement before testing starts in a quantifiable manner.
- According to the categories of the user generate profiles for each category of user.
- Produce a robust software and it's designed to test itself.
- Should use the Formal Technical Reviews (FTR) for the effective testing.
- To access the test strategy and test cases FTR should be conducted.
- To improve the quality level of testing generate test plans from the users feedback.

Test strategies for conventional software

Following are the four strategies for conventional software:
1) Unit testing
2) Integration testing
3) Regression testing
4) Smoke testing

1) Unit testing

- Unit testing focus on the smallest unit of software design, i.e module or software component.
- Test strategy conducted on each module interface to access the flow of input and output.
- The local data structure is accessible to verify integrity during execution.
- Boundary conditions are tested.

- In which all error handling paths are tested.
- An Independent path is tested.

Following figure shows the unit testing:

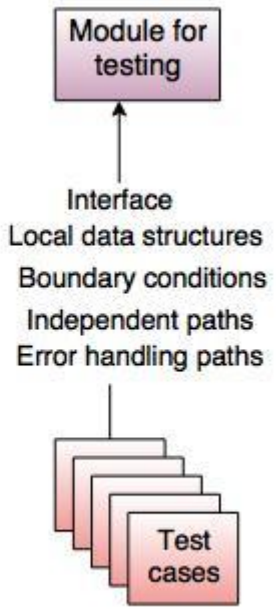


Fig. - Unit test

Unit test environment

The unit test environment is as shown in following figure:

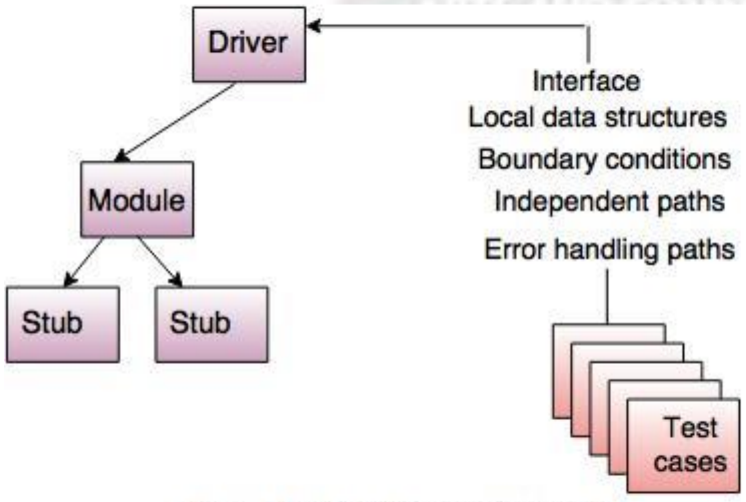


Fig. - Unit test environment

Difference between stub and driver

Stub	Driver
Stub is considered as subprogram.	It is a simple main program.
Stub does not accept test case data.	Driver accepts test case data.
It replace the modules of the program into subprograms and are tested by the next driver.	Pass the data to the tested components and print the returned result.

2) Integration testing

Integration testing is used for the construction of software architecture.

There are two approaches of incremental testing are:
i) Non incremental integration testing
ii) Incremental integration testing

i) Non incremental integration testing

- Combines all the components in advanced.
- A set of error is occurred then the correction is difficult because isolation cause is complex.

ii) Incremental integration testing

- The programs are built and tested in small increments.
- The errors are easier to correct and isolate.
- Interfaces are fully tested and applied for a systematic test approach to it.

Following are the incremental integration strategies:
a. Top-down integration
b. Bottom-up integration

a. Top-down integration

- It is an incremental approach for building the software architecture.
- It starts with the main control module or program.
- Modules are merged by moving downward through the control hierarchy.

Following figure shows the top down integration.

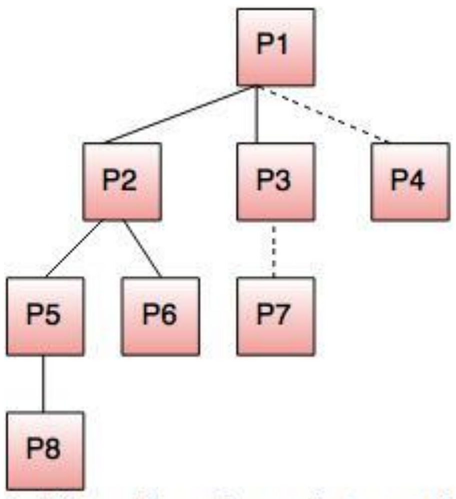


Fig. - Top-down integration

Problems with top-down approach of testing

Following are the problems associated with top-down approach of testing as follows:

- Top-down approach is an incremental integration testing approach in which the test conditions are difficult to create.
- A set of errors occur, then correction is difficult to make due to the isolation of cause.
- The programs are expanded into various modules due to the complications.
- If the previous errors are corrected, then new get created and the process continues. This situation is like an infinite loop.

b. Bottom-up integration

In bottom up integration testing the components are combined from the lowest level in the program structure.

The bottom-up integration is implemented in following steps:

- The low level components are merged into clusters which perform a specific software sub function.
- A control program for testing(driver) coordinate test case input and output.

- After these steps are tested in cluster.
- The driver is removed and clusters are merged by moving upward on the program structure.

Following figure shows the bottom up integration:

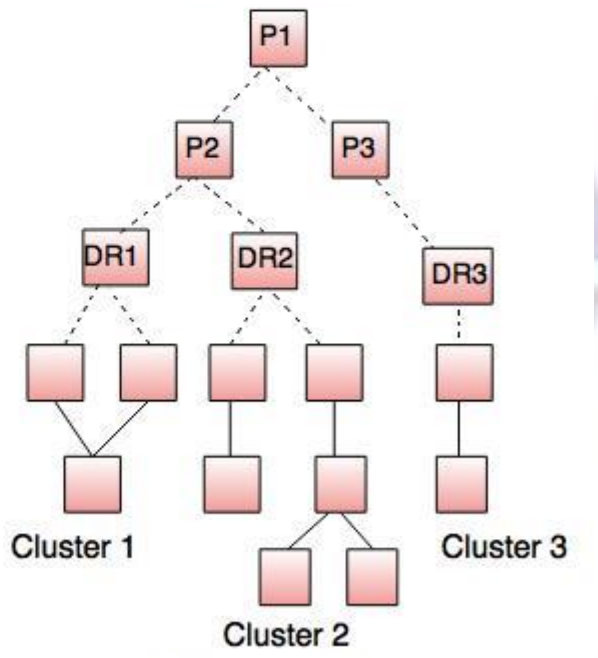


Fig. - Bottom-up integration

3) Regression testing

- In regression testing the software architecture changes every time when a new module is added as part of integration testing.

4) smoke testing

- The developed software component are translated into code and merge to complete the product.

Difference between Regression and smoke testing

Regression testing	**Smoke testing**
Regression testing is used to check defects generated to other modules by making the changes in existing programs.	At the time of developing a software product smoke testing is used.
In regression tested components are tested again to verify the errors.	It permit the software development team to test projects on a regular basis.
Regression testing needs extra manpower because the cost of the project increases.	Smoke testing does not need an extra manpower because it does not affect the cost of project.
Testers conduct the regression testing.	Developer conducts smoke testing just before releasing the product.

Alpha and Beta testing

Alpha testing	**Beta testing**
Alpha testing is executed at developers end by the customer.	Beta testing is executed at end-user sites in the absence of a developer.
It handles the software project and applications.	It usually handles software product.
It is not open to market and the public.	It is always open to the market and the public.
Alpha testing does not have any different name.	Beta testing is also known as the field testing.
Alpha testing is not able to test the errors because the developer does not known the type of user.	In beta testing, the developer corrects the errors as users report the problems.
In alpha testing, developer modifies the codes before release the software without user feedback.	In beta testing, developer modifies the code after getting the feedback from user.

Verification and Validation

Verification and Validation is the process of investigating that a software system satisfies specifications and standards and it fulfills the required purpose.

Verification:

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have.

Verification is **Static Testing**.

Activities involved in verification:

1. Inspections
2. Reviews
3. Walkthroughs
4. Desk-checking

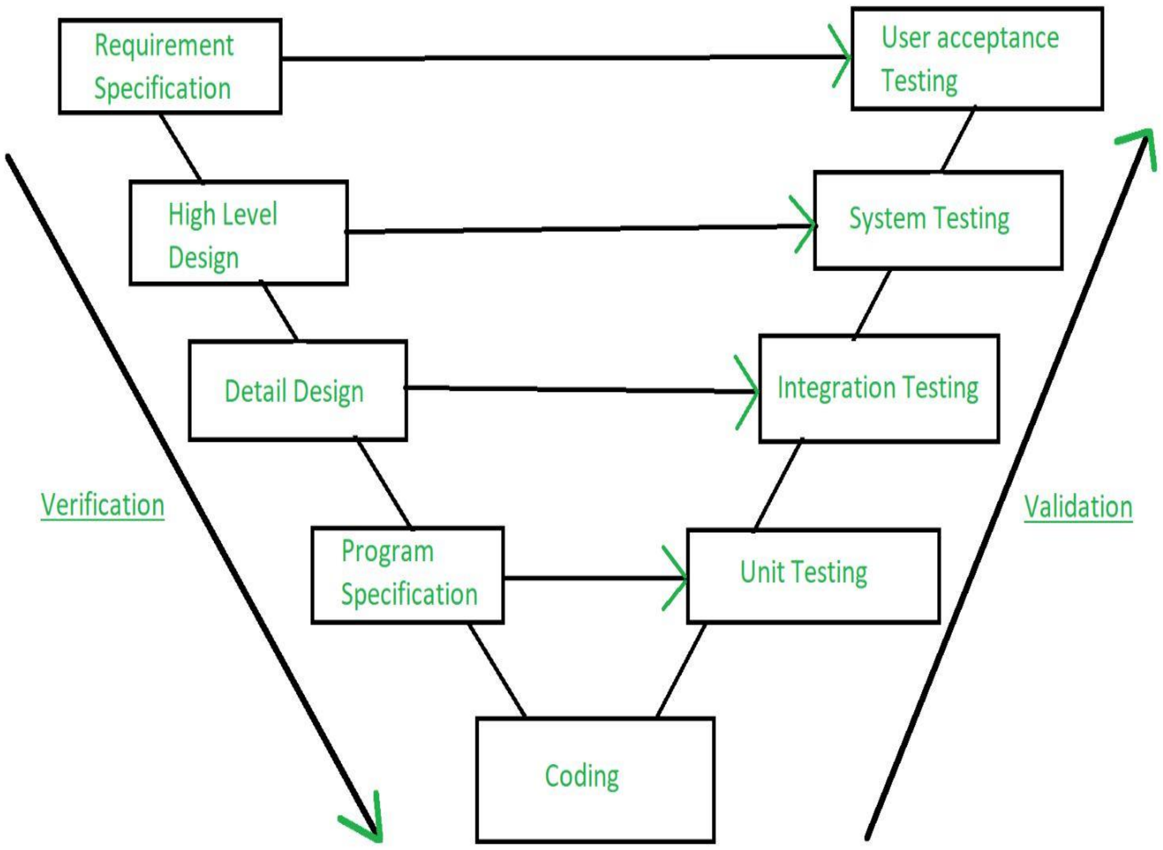
Validation:

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product.

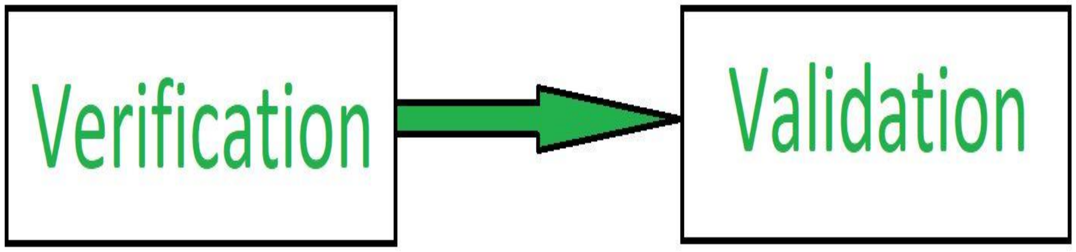
Validation is the **Dynamic Testing**.

Activities involved in validation:

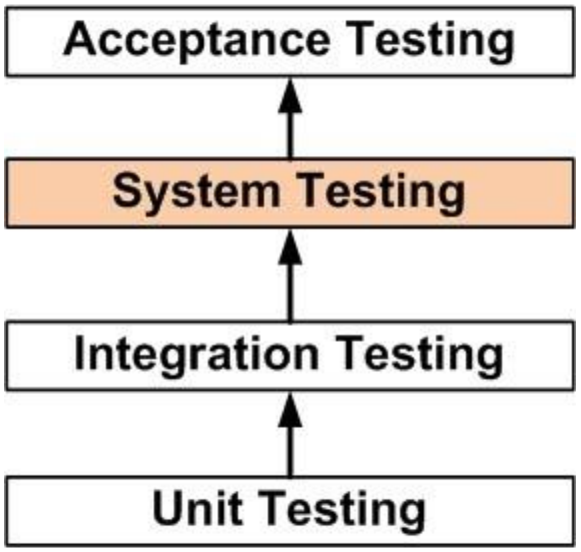
1. Black box testing
2. White box testing
3. Unit testing
4. Integration testing



Note: Verification is followed by Validation.



SYSTEM TESTING is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.



<u>Definition</u>

- **system testing:** The process of testing an integrated system to verify that it meets specified requirements.

During the process of manufacturing a ballpoint pen, the cap, the body, the tail, the ink cartridge and the ballpoint are produced separately and unit tested separately. When two or more units are ready, they are assembled and Integration Testing is performed. When the complete pen is integrated, System Testing is performed.

Method

Usually, <u>Black Box Testing</u> method is used.

Tasks

- System Test Plan
  - Prepare
  - Review
  - Rework
  - Baseline
- System Test Cases

-
    - Prepare
    - Review
    - Rework
    - Baseline
- System Test
    - Perform

When is it performed?

System Testing is the third level of software testing performed after Integration Testing and before Acceptance Testing.

Who performs it?

Normally, independent Testers perform System Testing.

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.
In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).

System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing.

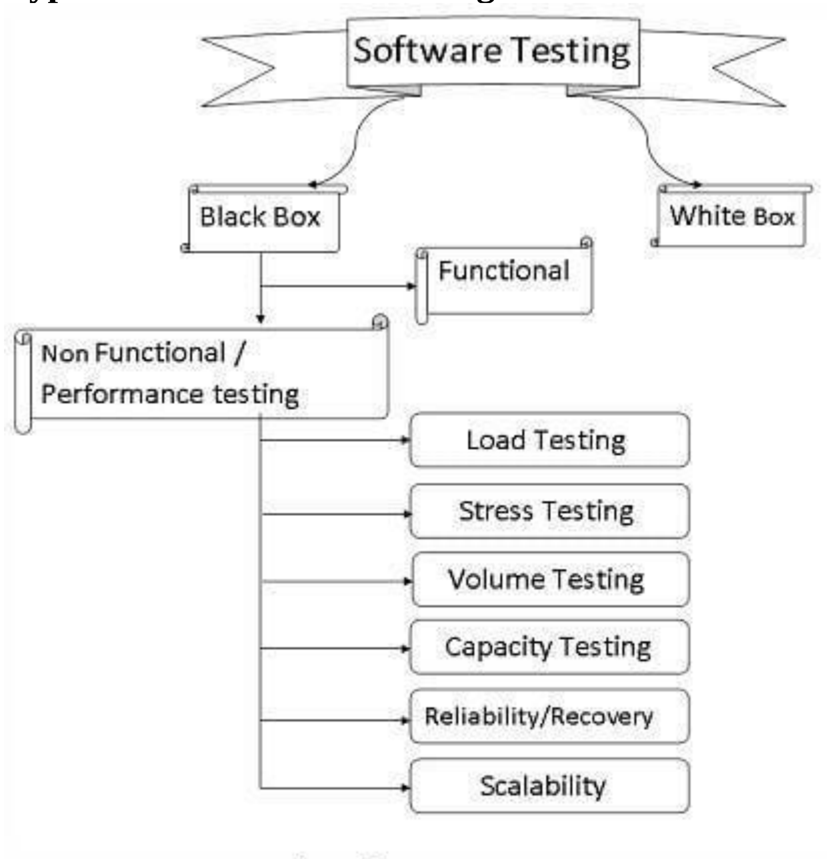
System Testing is a black-box testing.
System Testing is performed after the integration testing and before the acceptance testing.

Performance Testing

Performance Testing also knows as ‘Perf Testing', is a type of testing performed to check how application or software performs under workload in terms of responsiveness and stability. The Performance Test goal is to identify and remove performance bottlenecks from an application.

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

Types Of Performance Testing



Types of Performance Testing

© www.SoftwareTestingHelp.com

Load Testing

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

The key considerations are:

1. What is the maximum load the application is able to hold before the application starts behaving unexpectedly?
2. How much data the Database able to handle before the system slows or the crash is observed?
3. Are there any network related issues to be addressed?

Stress Testing

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

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

The following questions are to be addressed:

- What is the maximum load a system can sustain before it breaks down?
- How is the system break down?
- Is the system able to recover once it's crashed?
- In how many ways a system can break and which are the weak node while handling the unexpected load?

Volume Testing

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

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

Capacity Testing

=> Is the application capable of meeting business volume under both normal and peak load conditions?

Capacity Testing is generally done for future prospects. **Capacity Testing addresses the following:**

1. Will the application be able to support the future load?
2. Is the environment capable of standing for the upcoming increased load?
3. What are the additional resources required to make the environment capable enough?

Capacity Testing is used to determine how many users and/or transactions a given web application will support and still meet performance. During this testing, resources such as processor capacity, network bandwidth, memory usage, disk capacity, etc. are considered and altered to meet the goal.

Online Banking is a perfect example of where capacity testing could play a major role.

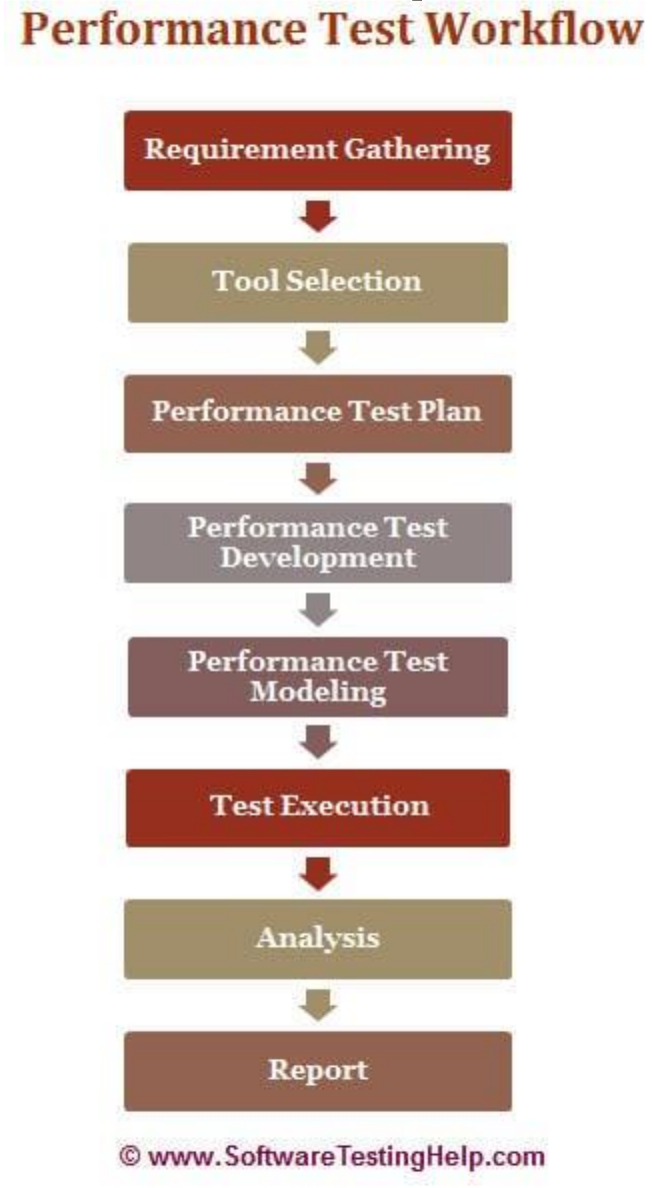
Reliability/Recovery Testing

Reliability Testing or Recovery Testing – is to verify whether or not the application is able to return back to its normal state after a failure or abnormal behavior and how long does it take for it to do so (in other words, time estimation).

If an online trading site experiences a failure where the users are not able to buy/sell shares at a certain point of the day (peak hours) but are able to do so after an hour or two, we can say the application is reliable or recovered from the abnormal behavior.

Performance Test Process

Here are all the activities performed in this testing:


Performance Test Workflow
Requirement Gathering
Tool Selection
Performance Test Plan
Performance Test Development
Performance Test Modeling
Test Execution
Analysis
Report
© www.SoftwareTestingHelp.com

1) Requirement Analysis/Gathering

The performance team interacts with the client for identification and gathering of requirements – technical and business. This includes getting information on the application's architecture, technologies, and database used, intended users, functionality, application usage, test requirement, hardware & software requirements, etc.

2) POC/Tool selection

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

The list of available tools depends on the cost of the tool, protocol that application is using, the technologies used to build the application, the number of users we are simulating for the test, etc. During POC, scripts are created for the identified key functionality and executed with 10-15 virtual users.

3) Performance Test Plan & Design

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

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

More on the Test Strategy document below.

4) Performance Test Development

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

5) Performance Test Modeling

Performance Load Model is created for the test execution. The main aim of this step is to validate whether the given Performance metrics (provided by clients) are achieved during the test or not. There are different approaches to create a Load model. "Little's Law" is used in most cases.

6) Test Execution

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

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

7) Test Results Analysis

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

Some of the best practices that help the Result Analysis process:

1. A unique and meaningful name to every test result – this helps in understanding the purpose of the test.
2. Include the following information in the test result summary:

- Reason for the failure/s
- Change in the performance of the application compared to the previous test run
- Changes made in the test from the point of application build or test environment.
- It's a good practice to make a result summary after each test run so that analysis results are not compiled every time test results are referred.
- PT generally requires many test runs to reach the correct conclusion.
- It is good to have the following points in result summary:
  - Purpose of test
  - Number of virtual users
  - Scenario summary
  - Duration of test
  - Throughput
  - Graphs
  - Graphs comparison
  - Response Time
  - Error occurred
  - Recommendations

8) Report

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

The test report is considered to be good if it is brief, descriptive and to the point.

How To Write Performance Test Strategy Document?

This tutorial will explain how to write a sample Performance Test Strategy for a Messaging Application.

Remember, that this is just an example and the requirements will differ from one client to another, we will also get to know the best practices for Performance Testing in this tutorial.

About ABC chat Application – Let's assume that this is a chat workbench that is used in a company by their customer support agent, this chat application uses XMPP protocol i.e, Extensible Messaging and Presence Protocol and Open fire server for sending and receiving Instant messages.

Some enhancements have been made to this existing chat client like Remote PC control, PC diagnosis, Repair tools, Online chat, etc., so this performance Test strategy is a sample of such applications.

For this application let's assume that the project team has decided to use JMeter for Performance Testing and JIRA for defect tracking.

The first page of the Performance Test Strategy document should contain Title of the Document and the Copyrights of the Company.

The second page should contain Document Control which includes, Document Version history, Reviewers & Approvers list and Contributors list.

The third page should contain the Table of contents, followed by the below topics.

1) Introduction

The purpose of this document is to define/explain how Performance Testing will be performed on the ABC chat application for the current and future state.

ABC chat application is an in-house remote support Agent workbench. This workbench will be used to fulfill customer requests. This Workbench has capabilities such as Online chat, Customer Identification, Remote PC control, PC diagnosis, and repair tools.

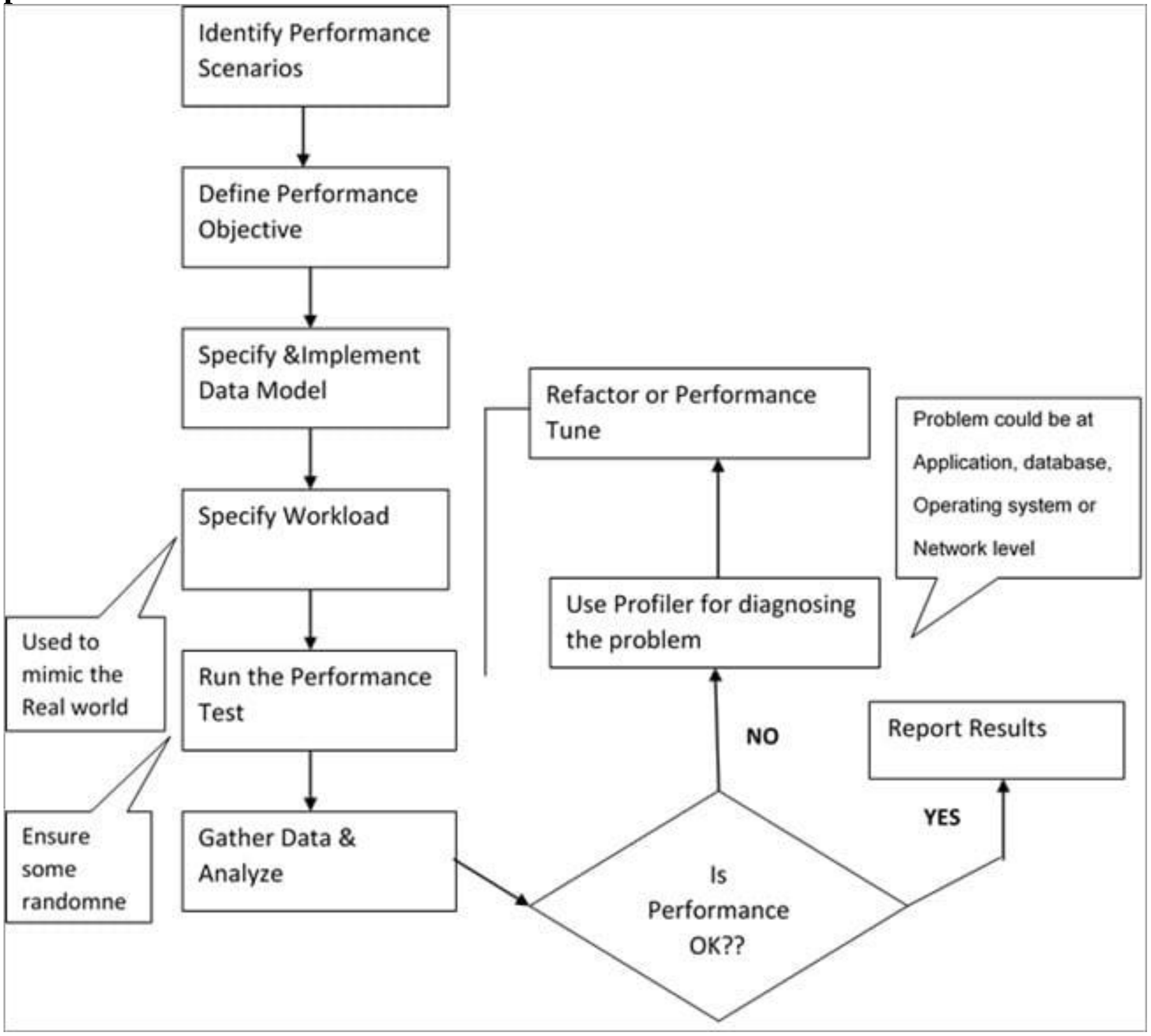
Objective

The key objectives of Performance Testing are as follows:

- To gain the confidence that the changes to the existing chat application are in line with the defined Service Level Agreement.

- To ensure that the application performance, service availability, and the stability of the application are not impacted as a result of the new enhancements.
- Transaction Response Times remain within the acceptable tolerance over the increasing Load profile.
- JVMs show stable memory usage over the increasing load profiles.

The below picture clearly explains Performance Testing & Optimization process:


Identify Performance Scenarios
Define Performance Objective
Specify &Implement Data Model
Specify Workload
Run the Performance Test
Gather Data & Analyze
Used to mimic the Real world
Ensure some randomne
Is Performance OK??
NO
YES
Use Profiler for diagnosing the problem
Refactor or Performance Tune
Problem could be at Application, database, Operating system or Network level
Report Results

Architecture

You need to incorporate the architecture diagram of your project in this session.

2) Scope

Below is the Performance Testing scope for ABC chat workbench:

- Knowledge acquisition of the key business transactions and build load distribution after a detailed study of the system.
- Identify the critical scenarios for performance testing with assistance from different project tracks.
- Use the previous release results as a baseline for future releases.
- Verify and validate the performance test environment and the Performance/Load test tool infrastructure for any additional Agent Machines.
- Preparation of performance test scripts using JMeter for the identified scenarios that mimic the identified peak load.
- Setup performance monitoring on the servers for monitoring the test in order to identify the bottlenecks during the test execution phase.
- Publish Performance test results.
- Coordinate with various stakeholders to resolve the identified performance issues.
- Baseline the performance level for future releases.

Out of Scope

- Functional Testing, UAT, System Testing & Security Testing.
- Performance testing/monitoring of any third-party interfaces.
- Performance Tuning. (Most of the times Tuning is done by a different team, if in case you have performance engineers to tune the system then you can add this in the Inscope).
- Code profiling / Hardware sizing / Capacity planning.
- Security / Vulnerability testing / UAT/ White box testing.
- Data generation for Performance Testing.
- Non-functional tests (**For Example,** failover, disaster recovery, back-up, usability) other than the performance tests.
- Testing of any mobile solution.
- Third-Party Application Performance Testing & Tuning.

- Realization of performance recommendations, Application code changes and the vendor-supported products/server configuration changes will be out of scope from the Performance Team perspective.
- Infrastructure Support / Build Deployment/ Environment Readiness/ Database Restore/ Network Support etc.

3) Approach

Performance testing for ABC chat will be conducted using Jmeter by writing custom XMPP plugins that use a smack library for XMPP connections. These libraries are used to set up connections, login and send chat messages to the XMPP server.

These libraries are bundled into a jar file that is deployed into the Jmeter and is designed based on the scenarios to be tested. The Jmeter Work Bench is installed in the local machine which connects to the JMeter server which has the Load Generators to generate the required load on the Chat server system to monitor the system behavior.

The test scenario will be scripted using the JMeter tool. The scripts would be customized as required. The schedule will be created with the required ramp-up to simulate the real-world scenarios.

The Test Scenario would be broken up and measured in the below aspects:

a) Baseline Test: To run each scenario with 1 Vuser and multiple iterations in order to identify whether the application performance meets the business Service Level Agreement or not.

b) Base Load Test: To meet the Business Benchmark under load test, the Performance Testing team will perform a baseload test which will help to identify any system performance issues with increasing load and creates the baseline for the next level of performance testing.

c) Peak Load / Scalability Test: Performance Testing team will perform multiple tests with increasing Vusers to meet the expected load and also to measure the application performance to establish the performance curve and identify whether the deployment can support the service level agreements under the peak user load.

It helps in tuning or capacity planning of the individual Java virtual machines (JVM), the total number of required JVMs, and the processors. This will be achieved by increasing the no of Vusers to 50%, 75%, 100% and 125% of peak capacity.

d) Endurance Test: Performance Testing team will run this test for a period of 8 Hours / 16 Hours /24 hours to identify memory leaks, performance issues over

time, and overall system stability. During endurance tests, the Performance Testing team monitors the key performance indicators, such as transaction response times and the stability of memory usage.
System resources like CPU, Memory, and IO need to be monitored with the help of the project team.

The Performance test environment is assumed to be a replica of the production environment. The tests will be run with an incremental load to identify where the application fails.

Performance Test Scenarios
Include the excel with the set of scenarios.

For Example,
Scenario 1: To validate the Agent and customer chat for X no. of concurrent sessions.
Performance Test Types
The table given below explains the various types of Performance Tests along with their objectives.

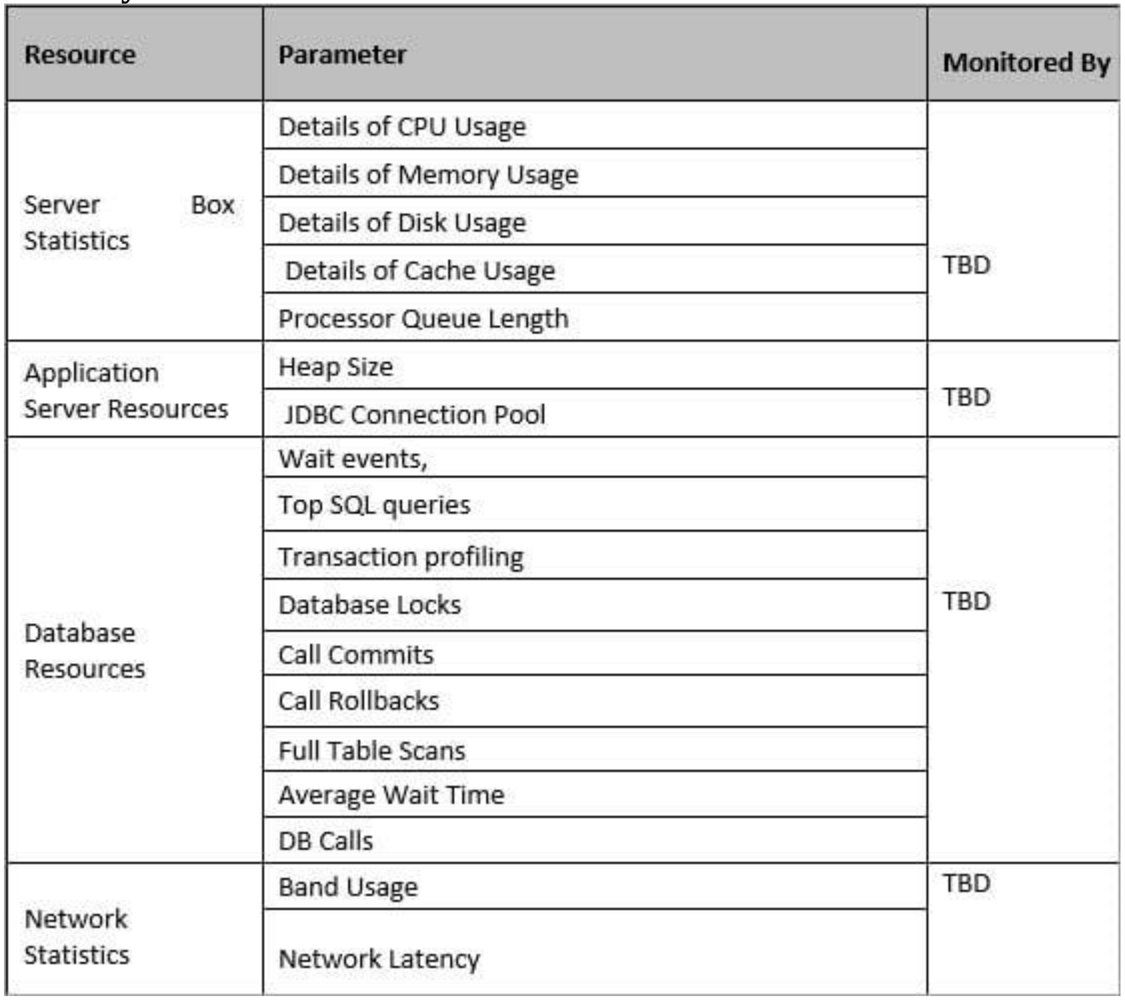
Test Type	**Objective**
Baseline Test	Establish the best performance under specific volumes which will be used as a reference for subsequent measurements.
Load Test	Measure the system performance under anticipated peak production load.
Endurance Test	Measuring the system stability under high volume for an extended period.
Stress Test	Measure the system performance under unfavorable conditions.

Performance Metrics

- **Client-side Metrics**

S.No	Metric	Description	Format
1	Transaction Response Time	Response time of pages during the steady state of the performance test	Graph
2	Throughput	The amount of data that the VUsers received from the server over time	Graph
3	Hits/second	The number of HTTP requests made by VUsers to the Web server during the scenario run	Graph
4	Number of Passed/Failed Transactions	Total number of transactions that Passed and Failed during the test execution	Excel
5	Transaction Error Rate	The Percentage of transactions that failed during the test execution	Graph

System & Network Performance Metrics



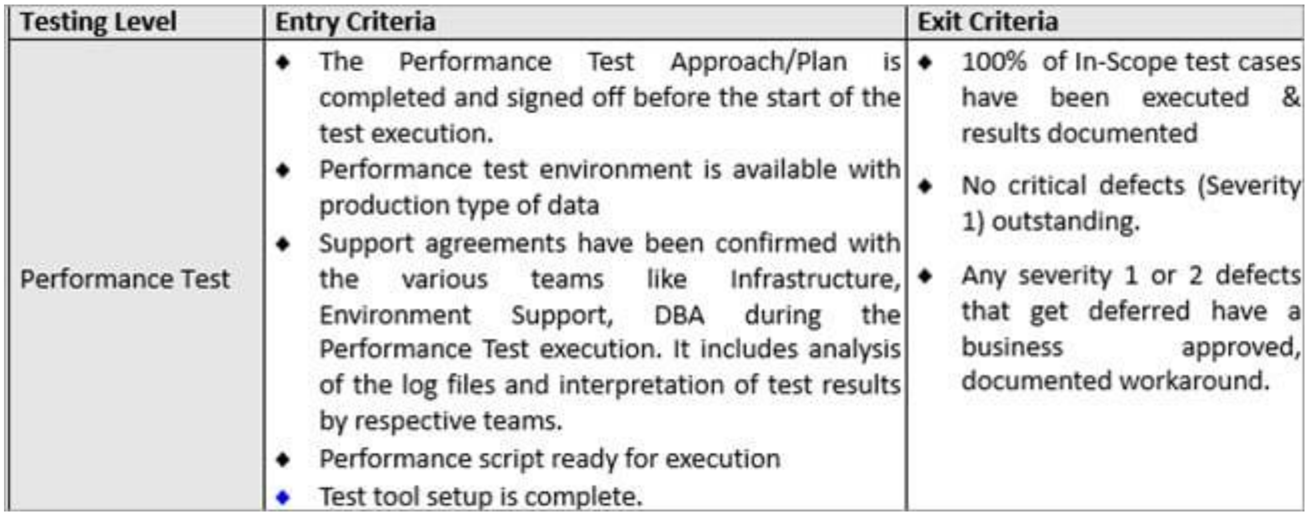
Resource	Parameter	Monitored By
Server Box Statistics	Details of CPU Usage	TBD
	Details of Memory Usage	
	Details of Disk Usage	
	Details of Cache Usage	
	Processor Queue Length	
Application Server Resources	Heap Size	TBD
	JDBC Connection Pool	
Database Resources	Wait events,	TBD
	Top SQL queries	
	Transaction profiling	
	Database Locks	
	Call Commits	
	Call Rollbacks	
	Full Table Scans	
	Average Wait Time	
	DB Calls	
Network Statistics	Band Usage	TBD
	Network Latency	

4) Test Data

It is being assumed that the Performance environment data will be a copy of the production data and the required test data will be provided by the project team.

5) Entry & Exit Criteria

- Access to all the applications in the environment.
- Environment readiness complete.
- Performance Test Data readiness.



Testing Level	Entry Criteria	Exit Criteria
Performance Test	◆ The Performance Test Approach/Plan is completed and signed off before the start of the test execution. ◆ Performance test environment is available with production type of data ◆ Support agreements have been confirmed with the various teams like Infrastructure, Environment Support, DBA during the Performance Test execution. It includes analysis of the log files and interpretation of test results by respective teams. ◆ Performance script ready for execution ◆ Test tool setup is complete.	◆ 100% of In-Scope test cases have been executed & results documented ◆ No critical defects (Severity 1) outstanding. ◆ Any severity 1 or 2 defects that get deferred have a business approved, documented workaround.

6) Defect Management

- The Defect Management module in JIRA will be used in the project for defect logging and for tracking to closure.
- Identification of defects that are found during the test execution phase will be captured in JIRA and these defects will be resolved by the development team according to the below severities.
- Defect review meetings would be held on a daily basis with the participation from the testing, development, Quality Analysts, and business teams.
- The criteria to fix defects would get stringent as the project approaches the Go Live date. Guidelines for defect fix criteria to be published in defect review meetings.

Defect Severity Definition

The definitions of severity codes are as follows:

Severity	Description for Development and Enhancement Problems
Blocker	System error, show stopper, Network issues
Critical	System errors, no clear workaround, interruption or missing business functionality
Major	A serious problem was detected for which the workaround exists that might not be c all the users, however, product should not be released without fixing

Severity	Description for Development and Enhancement Problems
Medium	Problem exists with easy/simple work around but this type of defect may be released approval by Business and/or Project Manager
Low	Cosmetic issues that do not interfere with business functionality or other intern problems that are not reproducible every time

#7) Testing Tools & Techniques

Tools	Purpose
Jmeter	To verify the Load and Performance of the ABC Chat application.

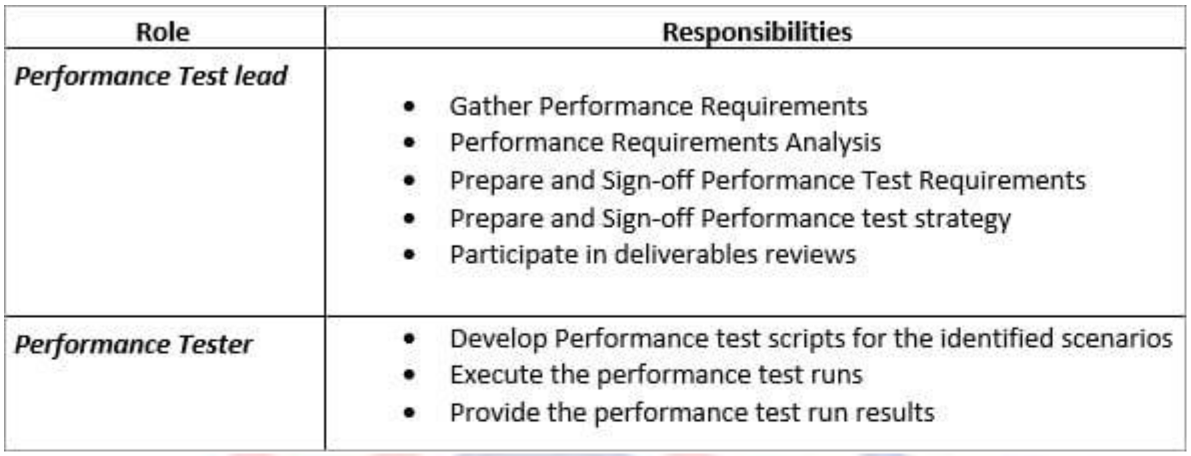
#8) Suspension and Resumption Criteria
Given below are the Critical Suspension and Resumption criteria which will impact the testing activities:

Suspension	Impact	Resumption
Environment not set up	Testing cannot proceed	Environment readiness
Application found to be unstable	Testing cannot proceed.	Issue resolved
Test Data not available	Testing cannot proceed.	Test Data ready

#9) Test Deliverables
The Performance Test Deliverables include:

- Performance Testing Strategy
- Performance Requirements Document
- Performance Test Scenario Document
- Performance Test Scripts
- Performance Test Results

#10) Roles & Responsibilities
Roles & Responsibilities are clearly explained in the table given below.



Role	Responsibilities
Performance Test lead	• Gather Performance Requirements • Performance Requirements Analysis • Prepare and Sign-off Performance Test Requirements • Prepare and Sign-off Performance test strategy • Participate in deliverables reviews
Performance Tester	• Develop Performance test scripts for the identified scenarios • Execute the performance test runs • Provide the performance test run results

Regression Testing

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change.

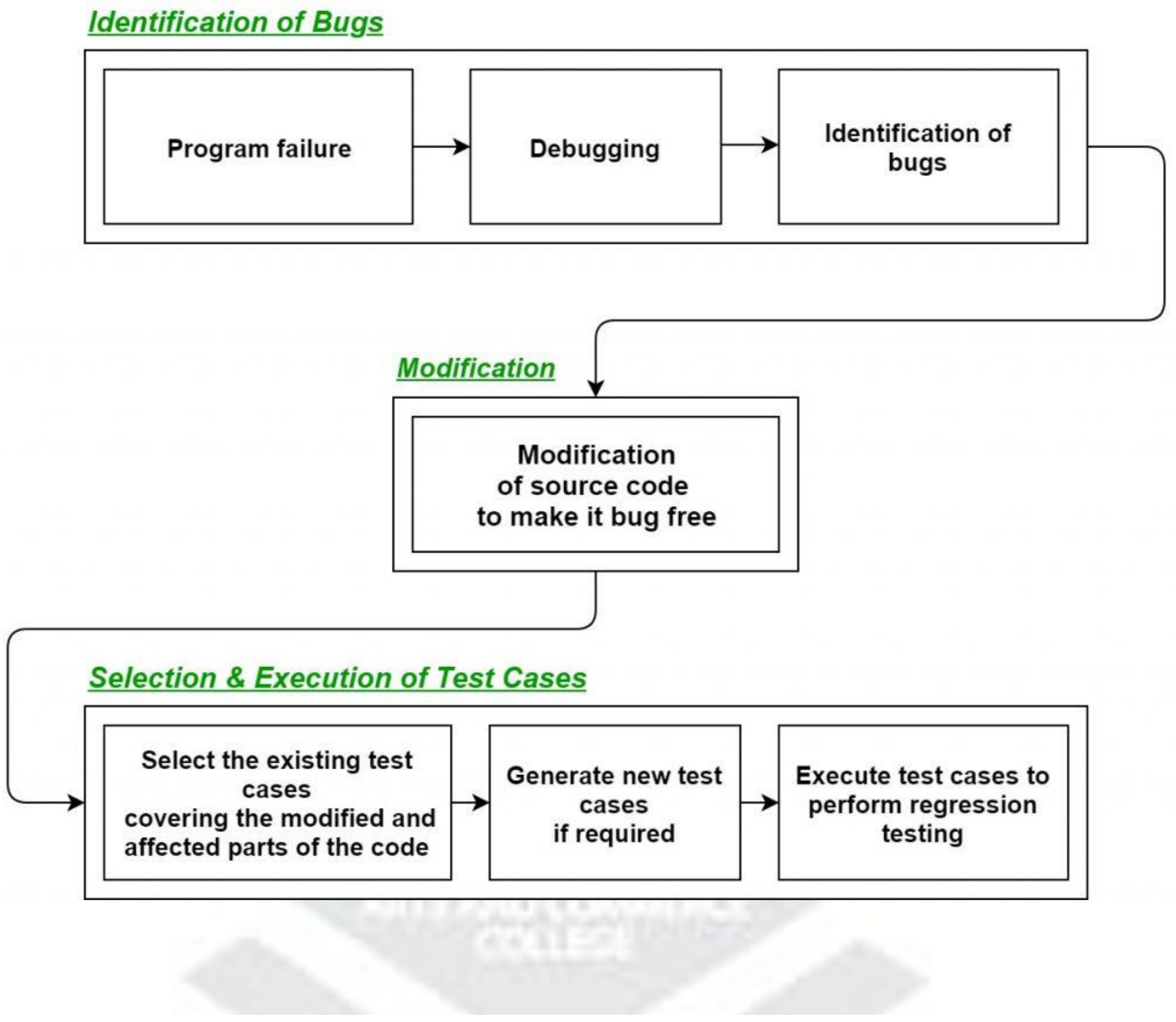
Regression Testing is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made. Regression means return of something and in the software field, it refers to the return of a bug.

When to do regression testing?

- When a new functionality is added to the system and the code has been modified to absorb and integrate that functionality with the existing code.
- When some defect has been identified in the software and the code is debugged to fix it.
- When the code is modified to optimize its working.

Process of Regression testing:
Firstly, whenever we make some changes to the source code for any reasons like adding new functionality, optimization, etc. then our program when executed fails in the previously designed test suite for obvious reasons. After the failure, the source code is debugged in order to identify the bugs in the program. After identification of the bugs in the source code, appropriate modifications are made. Then appropriate test cases are selected from the already existing test suite which covers all the modified and affected parts of the source code. We can add new test

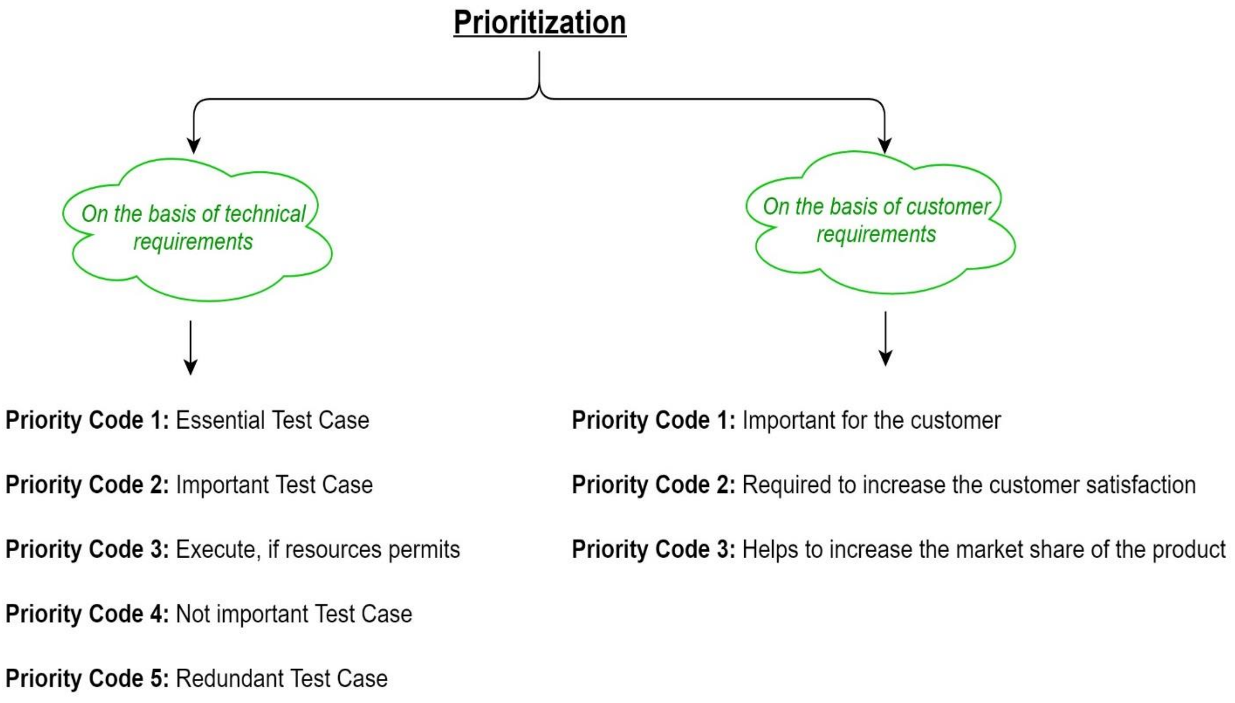
cases if required. In the end regression testing is performed using the selected test cases.


Identification of Bugs
Program failure
Debugging
Identification of bugs
Modification
Modification of source code to make it bug free
Selection & Execution of Test Cases
Select the existing test cases covering the modified and affected parts of the code
Generate new test cases if required
Execute test cases to perform regression testing

Techniques for the selection of Test cases for Regression Testing:

- **Select all test cases:** In this technique, all the test cases are selected from the already existing test suite. It is the most simple and safest technique but not much efficient.
- **Select test cases randomly:** In this technique, test cases are selected randomly from the existing test-suite but it is only useful if all the test cases are equally good in their fault detection capability which is very rare. Hence, it is not used in most of the cases.
- **Select modification traversing test cases:** In this technique, only those test cases are selected which covers and tests the modified portions of the source code the parts which are affected by these modifications.

- **Select higher priority test cases:** In this technique, priority codes are assigned to each test case of the test suite based upon their bug detection capability, customer requirements, etc. After assigning the priority codes, test cases with highest priorities are selected for the process of regression testing. Test case with highest priority has highest rank. For example, test case with priority code 2 is less important than test case with priority code 1.


Prioritization
On the basis of technical requirements
On the basis of customer requirements

Priority Code 1: Essential Test Case

Priority Code 2: Important Test Case

Priority Code 3: Execute, if resources permits

Priority Code 4: Not important Test Case

Priority Code 5: Redundant Test Case

Priority Code 1: Important for the customer

Priority Code 2: Required to increase the customer satisfaction

Priority Code 3: Helps to increase the market share of the product

Tools for regression testing: In regression testing, we generally select the test cases form the existing test suite itself and hence, we need not to compute their expected output and it can be easily automated due to this reason. Automating the process of regression testing will be very much effective and time saving. Most commonly used tools for regression testing are:

- Selenium
- WATIR (Web Application Testing In Ruby)
- QTP (Quick Test Professional)
- RFT (Rational Functional Tester)
- Winrunner
- Silktest

Advantages of Regression Testing:

- It ensures that no new bugs has been introduced after adding new functionalities to the system.

- As most of the test cases used in Regression Testing are selected from the existing test suite and we already know their expected outputs. Hence, it can be easily automated by the automated tools.
- It helps to maintain the quality of the source code.

Disadvantages of Regression Testing:

- It can be time and resource consuming if automated tools are not used.
- It is required even after very small changes in the code.

Unit 5 : Specialized Testing and Testing Tools

Testing Tools:

Tools from a software testing context can be defined as a product that supports one or more test activities right from planning, requirements, creating a build, test execution, defect logging and test analysis.

Classification of Tools

Tools can be classified based on several parameters. They include:

- The purpose of the tool
- The Activities that are supported within the tool
- The Type/level of testing it supports
- The Kind of licensing (open source, freeware, commercial)
- The technology used

Types of Tools:

S.No.	Tool Type	Used for	Used by
1.	Test Management Tool	Test Managing, scheduling, defect logging, tracking and analysis.	testers
2.	Configuration management tool	For Implementation, execution, tracking changes	All Team members

3.	Static Analysis Tools	Static Testing	Developers
4.	Test data Preparation Tools	Analysis and Design, Test data generation	Testers
5.	Test Execution Tools	Implementation, Execution	Testers
6.	Test Comparators	Comparing expected and actual results	All Team members
7.	Coverage measurement tools	Provides structural coverage	Developers
8.	Performance Testing tools	Monitoring the performance, response time	Testers
9.	Project planning and Tracking Tools	For Planning	Project Managers
10.	Incident Management Tools	For managing the tests	Testers

Tools Implementation - process

- Analyse the problem carefully to identify strengths, weaknesses and opportunities
- The Constraints such as budgets, time and other requirements are noted.
- Evaluating the options and Shortlisting the ones that are meets the requirement
- Developing the Proof of Concept which captures the pros and cons
- Create a Pilot Project using the selected tool within a specified team

- Rolling out the tool phase wise across the organization

Test case Design Technique

Following are the typical design techniques in software engineering:

1. Deriving test cases directly from a requirement specification or black box test design technique. The Techniques include:

- **Boundary Value Analysis (BVA)**
- **Equivalence Partitioning (EP)**
- **Decision Table Testing**
- **State Transition Diagrams**
- **Use Case Testing**

2. Deriving test cases directly from the structure of a component or system:

- **Statement Coverage**
- **Branch Coverage**
- **Path Coverage**
- **LCSAJ Testing**

3. Deriving test cases based on tester's experience on similar systems or testers intuition:

- **Error Guessing**
- **Exploratory Testing**

Apache Jmeter

Apache JMeter may be used to test performance both on static and dynamic resources, Web dynamic applications.
It can be used to simulate a heavy load on a server, group of servers, network or object to test its strength or to analyze overall performance under different load types.

Apache JMeter features include:

- Ability to load and performance test many different applications/server/protocol types:
    - Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …)
    - SOAP / REST Webservices
    - FTP
    - Database via JDBC
    - LDAP
    - Message-oriented middleware (MOM) via JMS
    - Mail - SMTP(S), POP3(S) and IMAP(S)
    - Native commands or shell scripts
    - TCP
    - Java Objects
- Full featured Test IDE that allows fast Test Plan **recording (from Browsers or native applications), building and debugging**.
- **CLI mode (Command-line mode (previously called Non GUI) / headless mode)** to load test from any Java compatible OS (Linux, Windows, Mac OSX, …)
- A complete and **ready to present dynamic HTML report**
- Easy correlation through ability to extract data from most popular response formats, **HTML, JSON , XML or any textual format**
- Complete portability and **100% Java purity**.
- Full **multi-threading** framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.
- Caching and offline analysis/replaying of test results.
- **Highly Extensible core:**
    - Pluggable Samplers allow unlimited testing capabilities.
    - **Scriptable Samplers** (JSR223-compatible languages like Groovy and BeanShell)
    - Several load statistics may be chosen with **pluggable timers**.
    - Data analysis and **visualization plugins** allow great extensibility as well as personalization.
    - Functions can be used to provide dynamic input to a test or provide data manipulation.
    - Easy Continuous Integration through 3rd party Open Source libraries for Maven, Gradle and Jenkins.

WinRunner

HP WinRunner Automation Tool was owned by Mercury Interactive. When HP acquired Mercury Interactive, they stopped Winrunner support and suggested to use the HP Functional Testing software.

LoadRunner

LoadRunner is a software testing tool from Micro Focus. It is used to test applications, measuring system behaviour and performance under load. LoadRunner can simulate thousands of users concurrently using application software, recording and later analyzing the performance of key components of the application.

Learning Resources:		
1	Reference Books	1. Software Engineering – A Practitioners Approach, Roger S. Pressman, Tata McGraw Hill 2. Software Engineering for Students- A Programming Approach, Douglas Bell, Pearson
2	Websites	• https://www.softwaretestingmaterial.com/ • https://www.guru99.com/ • https://www.softwaretestinghelp.com/ • https://www.tutorialspoint.com/