Chapter 2

What are the levels of testing?

A level of software testing is a process where every unit or component of a software/system is tested. The main goal of system testing is to evaluate the system's compliance with the specified needs.

There are many different testing levels which help to check behaviour and performance for software testing. These testing levels are designed to recognize missing areas and reconciliation between the development lifecycle states. In SDLC models there are characterized phases such as requirement gathering, analysis, design, coding or execution, testing, and deployment.

All these phases go through the process of software testing levels. There are mainly four testing levels are:

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

Each of these testing levels has a specific purpose. These testing level provide value to the software development lifecycle.

1) **Unit testing:**

A Unit is a smallest testable portion of system or application which can be compiled, liked, loaded, and executed. This kind of testing helps to test each module separately.

The aim is to test each part of the software by separating it. It checks that component are fulfilling functionalities or not. This kind of testing is performed by developers.

2) **Integration testing:**

Integration means combining. For Example, In this testing phase, different software modules are combined and tested as a group to make sure that integrated system is ready for system testing.

Integrating testing checks the data flow from one module to other modules. This kind of testing is performed by testers.

3) **System testing:**

System testing is performed on a complete, integrated system. It allows checking system's compliance as per the requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing.

System testing is most often, the final test to verify that the system meets the specification. It evaluates both functional and non-functional need for the testing.

4) **Acceptance testing:**

Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing is basically done by the user or customer. However, other stockholders can be involved in this process.

Other Types of Testing:

- Regression Testing
- Buddy Testing
- Alpha Testing
- Beta Testing

Testing Strategies

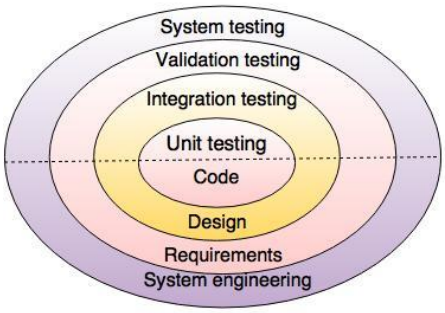


Fig. - Testing Strategy

- A **testing strategy** is a general approach to the testing process rather than a method of devising particular system or component tests. Different testing strategies may be adopted depending on the type of system to be tested and the development process used. There are two different strategies available: **Top-Down Testing** and **Bottom-Up Testing.**
- In **Top-Down Testing**, high levels of a system are tested before testing the detailed components. The application is represented as a single abstract component with sub-components represented by **stubs**. Stubs have the same interface as the component but very limited functionality.
- After the top-level component has been tested, its **sub-components** are implemented and tested in the same way. This process continues recursively until the bottom - level components are implemented. The whole system may then be completely tested. Top-down testing should be used with **top-down program development** so that a system component is tested as soon as it is coded. Coding and testing are a single activity with no separate component or module testing phase.
- If top-down testing is used, unnoticed **design errors** may be detected at an early stage in the testing process. As these errors are usually structural errors, early detection means that extensive re-design re-implementation may be avoided. Top-down testing has the further advantage that we could have a **prototype system** available at a very early stage, which itself is a psychological boost. Validation can begin early in the testing process as a demonstrable system can be made available to the users.
- **Bottom-Up Testing** is the opposite of Top-Down. It involves testing the modules at the lower levels in the hierarchy, and then working up the hierarchy of modules until the final module is tested. This type of testing is appropriate for **object-oriented systems** in that individual objects may be tested using their own test drivers. They are then integrated and the object collection is tested.

Unit Testing:

Unit Testing is a level of software testing where individual units/ components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software.

- It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc.
- In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a

unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

- **Unit tests are basically written and executed by software developers** to make sure that code meets its design and requirements and behaves as expected.
- The goal of unit testing is to segregate each part of the program and test that the individual parts are working correctly.
- This means that for any function or procedure when a set of inputs are given then it should return the proper values. It should handle the failures gracefully during the course of execution when any invalid input is given.

Unit Testing Method

It is performed by using the White Box Testing method.

When is it performed?

It is the first level of software testing and is performed prior to Integration Testing.

Who performs it?

It is normally performed by software developers themselves or their peers. In rare cases, it may also be performed by independent software testers.

Unit Testing Tasks

- Unit Test Plan
  - Prepare
  - Review
  - Rework
  - Baseline
- Unit Test Cases/Scripts
  - Prepare
  - Review
  - Rework
  - Baseline
- Unit Test
  - Perform

Unit Testing Benefits

- Unit testing increases confidence in changing/ maintaining code. If good unit tests are written and if they are run every time any code is changed, we will be able to promptly catch any defects introduced due to the change. Also, if codes are already made less interdependent to make unit testing possible, the unintended impact of changes to any code is less.
- Codes are more reusable. In order to make unit testing possible, codes need to be modular. This means that codes are easier to reuse.
- Development is faster. How? If you do not have unit testing in place, you write your code and perform that fuzzy 'developer test' (You set some breakpoints, fire up the GUI, provide a few inputs that hopefully hit your code and hope that you are all set.) But, if you have unit testing in place, you write the test, write the code and run the test. Writing tests takes time but the time is compensated by the less amount of time it takes to run the tests; You need not fire up the GUI and provide all those inputs. And, of course, unit tests are more reliable than 'developer tests'. Development is faster in the long run too. How? The effort required to

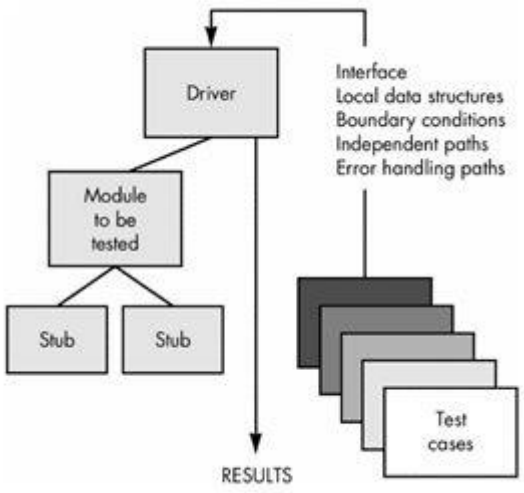
find and fix defects found during unit testing is very less in comparison to the effort required to fix defects found during system testing or acceptance testing.

- The cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels. Compare the cost (time, effort, destruction, humiliation) of a defect detected during acceptance testing or when the software is live.
- Debugging is easy. When a test fails, only the latest changes need to be debugged. With testing at higher levels, changes made over the span of several days/weeks/months need to be scanned.
- Codes are more reliable. Why? I think there is no need to explain this to a sane person.

Advantages of Unit testing:

- 1. Issues are found at early stage. Since unit testing are carried out by developers where they test their individual code before the integration. Hence the issues can be found very early and can be resolved then and there without impacting the other piece of codes.
- 2. Unit testing helps in maintaining and changing the code. This is possible by making the codes less interdependent so that unit testing can be executed. Hence chances of impact of changes to any other code get reduced.
- 3. Since the bugs are found early in unit testing hence it also helps in reducing the cost of bug fixes. Just imagine the cost of bug found during the later stages of development like during system testing or during acceptance testing.
- 4. Unit testing helps in simplifying the debugging process. If suppose a test fails then only latest changes made in code needs to be debugged.

What is meant by Stubs and Drivers?



Defining Stubs:

Stubs are used to test modules and are created by the team of testers during the process of **Top-Down Integration Testing**. With the assistance of these test stubs testers are capable of stimulating the behaviour of the lower level modules that are not yet integrated with the software. Moreover, it helps stimulates the activity of the missing components.

Types of Stubs:

There are basically four types of stubs used in top-down approach of integration testing, which are mentioned below:

- Displays the trace message.

- Values of parameter are displayed.
- Returns the values that are used by the modules.
- Returns the values selected by the parameters that were used by modules being tested.

Defining Drivers:

Drivers, like stubs, are used by software testers to fulfil the requirements of missing or incomplete components and modules. These are usually complex than stubs and are developed during **Bottom-Up approach of Integration Testing**. Drivers can be utilized to test the lower levels of the code, when the upper level of codes or modules is not developed. Drivers act as pseudo codes that are mainly used when the stub modules are ready, but the primary modules are not ready.

Stubs and Drivers: Example

Consider an example of a web application, which consists of 4 modules i.e., **Module-A**, **Module-B**, **Module-C** and **Module-D**. Each of the following modules is responsible for some specific activity or functionality, as under:

Module-A→ Login page of the web application.
Module-B → Home page of the web application.
Module-C → Print Setup.
Module-D → Log out page.
Modules A, B, C & D involves the interdependencies of each module over other.

It is always preferable, to perform testing, in parallel, to the development process. Thus, it implies that subsequent testing must be carried out, immediately after the development of the each module.

Module-A will be tested, as soon as, it develops. However, to carry out and validate the testing procedures in respect of module-A, there urges the need of **Module-B**, which is not yet developed. The expected functionality of the login page **(module-A)** could be validated; only if it is directed to the home page **(Module-B)**, based on the valid and correct inputs.

But, on the non-availability of the **Module-B**, it will not be possible to test **module-A**. These types of circumstances, introduces the stubs & drivers in the **process of software testing**. A dummy module, representing the basic functionality or feature of the module-B, is being developed, and thereafter, it is being integrated with the module-A, to perform testing, efficiently.

Similarly, **stubs and drivers**, are used to fulfil the requirements of other modules, such as

Log out page (Module-D), needs to be directed to the login page **(Module-A)**, after successfully logging out from the application. In the event of unavailability of **Module-A**, stubs and drivers will work as a substitute for it, in order to carry out the testing of module-D.

Why are Stubs & Drivers Required?

Though, the important of **stubs and drivers** is immense during software testing, but its relevance as well as the requirement ends once the equivalent module is created. As they act as a temporary replacement of modules, these roles in testing culminates once final output is generated. Therefore, these pseudo modules or stubs and drivers are mainly used for following reasons:

- Commonly used in porting, distributed computing, as well as general software development.
- Used for meeting necessary requirements unavailable modules.
- They are required when the system needs to interact with external systems.
- Stubs and drivers are used to test modules.
- Stubs are used to **test the functionality** of modules, whereas drivers are used when the main module is not ready.

Are Stubs and Drivers, Same?

Yes, basically the features and purpose of stubs and drivers are same. Both of them work as a substitute for the missing or unavailable module. However, the difference between them can be visualized during the **integration testing**.

- Stubs are commonly referred to as "called programs" and are being used in top bottom approach of the integration testing, whereas drivers are "calling program" and they are used in bottom-up integration testing.
- Stubs are similar to the components, which are under test, in a very simple and basic form, whereas driver is used to invoke the component that needs to be tested.
- Stubs, are usually, considered for low level modules, whereas drivers represents the high level modules.

Key Points:

- Stubs and Drivers works as a substitute for the missing or unavailable module.
- They are specifically developed, for each module, having different functionalities.
- Generally, developers and unit testers are involved in the development of stubs and drivers.
- Although, it provides ease to carry out **of individual components**, without concerning the availability of other modules, but it is a time-consuming process, as it requires developing dummy for each missing module.
- Their most common use may be seen in the integration **incremental testing**, where stubs are used in top bottom approach and drivers in a bottom up approach.

Comparison of Stubs & Drivers:

Even though both stubs and drivers are dummy/pseudo codes, there are various features of these two components that are different from each other. Therefore, to signify their differences, here is a comparison of stubs and drivers.

	Stub	**Driver**
Type	Dummy codes	Dummy codes
Description	Routines that don't actually do anything except declare themselves and the parameters they accept. The rest of the code can then take these parameters and use them as inputs	Routines that don't actually do anything except declare themselves and the parameters they accept. The rest of the code can then take these parameters and use them as inputs
Used in	Top Down Integration	Bottom Up Integration
Purpose	To allow testing of the upper levels of the code, when the lower levels of the code are not yet developed.	To allow testing of the lower levels of the code, when the upper levels of the code are not yet developed.

What is Integration Testing?

Integration Testing is the phase in software testing, wherein individual software modules are combined and tested as a group. It occurs after **unit testing** and before **validation testing**.

- When software application is tested through integration testing, it exposes the problems and issues with the interfaces among program components before trouble occurs in real world execution.
- This type of testing takes its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.
- Moreover, integration testing is a component of **extreme programming**, which is a pragmatic method of software development that takes meticulous approach to build a product by means of continual testing and revision.

Different Types of Integration Testing:

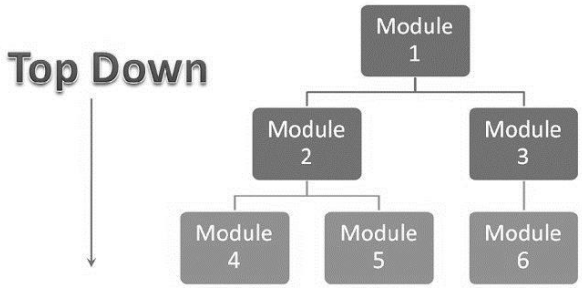
It is a type of software testing that not only ensures the quality of the software, but also makes sure that the software is tested in the production like environment before it is released for the use of the end users and clients.

Integration testing, which is also known as **Integration and Testing (I&T)**, is performed through various types of approaches that are either executed by the integration tester or by a test team.

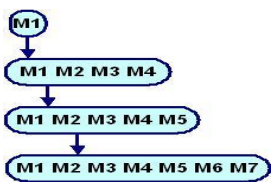
1. **Incremental Integration Testing:**

   In **Incremental Integration Testing**, the developers integrate the modules one by one using stubs and drivers to uncover defects in the software program. Here, each module has a definitive role to play in the project or product structure and has clearly defined dependencies, which can be known only at the runtime. The most important quality of incremental integration testing is that the defects are found early in a smaller assembly, when it is relatively easy to detect the root cause of the same. Incremental Integration Testing is performed on software through the following approaches:

   a. **Top-Down Integration Approach:**



- In **top-down approach** the testing takes place from top to bottom, following the control flow or architectural structure. Additionally, components or systems are substituted by stubs.
- This approach uses a Program called "Stub". While integrating the modules with this Top Down Approach, if at any stage it is found that some mandatory module is missing, then in such an event that particular module is replaced with a temporary program known as "Stub".



- The testing starts with M1. To test M1 in isolation, communications to modules M2, M3 and M4 needs to be simulated by the tester by some means, since these modules may not be

ready yet. To simulate responses of M2, M3 and M4 whenever they are to be invoked from M1, "stubs" are created.

- Simple applications may require stubs, which would simply return the control to their superior modules. More complex situation demand stubs to simulate a full range of responses, including parameter passing.
- Stubs may be individually created by the tester or they may be provided by a software testing harness, which is a sort of software specifically designed to provide a testing environment.
- In the above illustration, M1 would require stubs to simulate the activities of M2, M3 and M4. The integration of M3 would require a stub or stubs for M5 and M4 would require stubs for M6 and M7. Elementary modules i.e. the modules which do not call subordinates would not require any stubs.

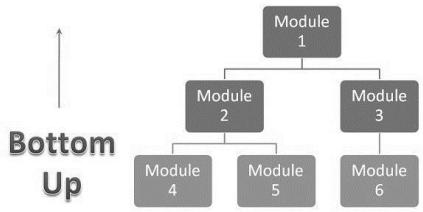
Advantages:

- The tested product is extremely consistent in this approach as the integration testing is performed in an **environment** that is similar to the real world environment.
- The stubs in top-down approach can be written in lesser time compared to drivers as they are simpler to author.
- Here, fault **localization** is easier.
- In top-down approach the critical modules are tested on priority, which further helps testers in finding major design flaws as early as possible.

Disadvantages:

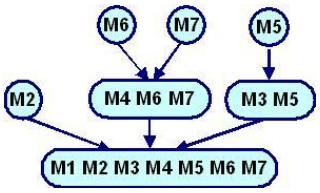
- It requires several stubs.
- The modules at the lower level of the software are tested inadequately.
- The basic functionality of the software is tested at the end of the cycle.

b. **Bottom-Up Approach Integration Approach:**



- In this approach of incremental integration testing, all the modules at the lower level of the software are tested with higher modules until all the modules are tested. Moreover, the **bottom up testing** takes place from the bottom of the control flow to upwards. Unlike top-down approach, here the components or systems are substituted with driver.
- This approach uses a Program called "Driver". While integrating the modules with this Bottom Up Approach, if at any stage it is found that some mandatory module is missing, then in such an event that particular module is replaced with a temporary program known as "Driver".
- If M5 is ready, we need to simulate the activities of its superior, M3. Such a "driver" for M5 would simulate the invocation activities of M3. As with the stub, the complexity of a driver would depend upon the application under test. The driver would be responsible for invoking the module under test, it could be responsible for passing test data and it might be

responsible for receiving output data. Here as well, the driving function can be provided through a testing harness or may be created by the tester as a program.

- 

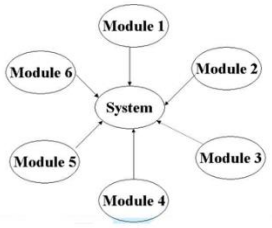
- For the above-mentioned Bottom-Up example, drivers must be provided for modules M2, M5, M6, M7, M3 and M4. However there is no need for a driver for the topmost Module, M1.

  Advantages:
  - No time is wasted waiting for all modules to be developed.
  - In this approach development and testing can be done together so that the product or application will be efficient and as per the customer specifications.

  Disadvantages:
  - The key interface defects are caught at the end of the cycle.
  - Test drivers are required to be created for modules at all levels except the top control.
  - The critical modules, which control the flow of the application are tested last and may be prone to defects.

2. **Non-Incremental Integration Testing:**

   Whenever the relationship between the modules is not clear, non-incremental integration testing or big bang integration is executed. In this case the data is created in one module and is combined with all the other modules to check as well as test the flow of data between them. It is because of this that non-incremental integration testing is also known as Big Bang Integration.
   - **Big Bang Integration Testing:**

   - In this type of integration testing approach, most of the developed modules are coupled together to form a complete software system or a major part of the system, which is then used for integration testing. This method is very effective for saving time in the **integration testing process**. However, if the test cases and their results are not properly recorded, the entire integration process will be more complicated and may prevent the testing team form getting their desired goals or results of integration testing.

     Advantages:
     - Here all components of the software are integrated at once.
     - Convenient for small systems.
     - Helps testers in saving integration testing time.

Disadvantages:

- Fault localisation is difficult in this approach of integration testing.
- Given the number of interfaces that are required to be tested in this approach, some of the interface links can be missed easily.
- As all modules are tested at once, high risk critical modules are not isolated and tested in priority.

3. **Sandwich Testing:**

Sandwich testing is a culmination of both incremental as well as non incremental integration testing, wherein Bottom-Up approach is focused on middle to top layer, Top-Down approach is concerned about layers from middle to downwards and the Big Bang approach is followed for the middle layer. This type of testing combines the advantages of all the three approaches and is mainly used to test large projects.

Advantages:

- Sandwich approach is very useful for large enterprises and huge projects that further have several subprojects.
- When development follows a spiral model and the module itself is as large as a system, then one can use sandwich testing.
- Top-Down and Bottom-Up approach both start as per development schedule.
- Units are tested and brought together to make a system.
- Integration is done downwards.
- The resources that are required are immense and big team perform both top-down and bottom-up method of testing at a time or one after the other.

Disadvantages:

- As both Top-Down and Bottom-Up approaches are executed on the software, the cost of testing is very high.
- It cannot be used for smaller systems with huge interdependence between the modules.
- It only makes sense when the individual subsystem is as good as completed system.
- Different skill sets are required for testers at different levels.

Integration Testing Example

Let us understand Integration Testing with example. Let us assume that you work for an IT organization which has been asked to develop an online shopping website for Camp World, a company that sells camping gear.

After requirements gathering, analysis and design was complete, one developer was assigned to develop each of the modules below.

1. User registration and Authentication/Login
2. Product Catalogue
3. Shopping Cart
4. Billing
5. Payment gateway integration
6. Shipping and Package Tracking

After each module was assigned to a developer, the developers began coding the functionality on their individual machines. They deployed their respective modules on their own machines to see what worked and what didn't, as they went about developing the module.

After they completed the development, the developers tested their individual functionalities as part of their **unit testing** and found some defects. They fixed these **defects**. At this point they felt their modules were complete.

The QA Manager suggested that integration testing should be performed to confirm that all the modules work together.

When they deployed all of their code in a common machine, they found that the application did not work as expected since the individual modules did not work well together. There were bugs like – after logging in, the user's shopping cart did not show items they had added earlier, the bill amount did not include shipping cost etc.

In this way, Integration Testing helps us identify, fix issues and ensure that the application as a whole, works as expected.

Steps – How to do Integration Testing

- Choose the module or component to be tested based on the strategy or approach
- Unit testing of the component is to be completed for all the features
- Deploy the chosen modules or components together and make initial fixes that are required to get the integration testing running
- Perform **functional testing** and test all the use cases for the components chosen
- Perform **structural testing** and test the components chosen
- Record the results of the above testing activities
- Repeat the above steps until the complete system is fully tested

Difference Between – Integration Testing and/vs Unit Testing

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

It is carried out with the help of reusable test cases	It is carried out with the help of stubs and drivers
It comes under White Box Testing	It comes under both Black Box and White Box Testing
It is performed by developers	It is performed by either testers or developers
Unit testing is done to confirm if the unit of code works as expected	Integration testing is done to confirm if the different modules work as expected, when integrated together
In unit testing, the unit is not dependent on anything outside the unit being tested	In Integration testing, the components may have inter-dependency on each other or external systems
Unit testing is done by developer	Integration testing is done by the testing team
In the Software Testing Life Cycle (STLC), Unit testing is the first test to be executed	Integration testing is usually done before system testing and it comes after unit testing.

GUI Application Integration Test

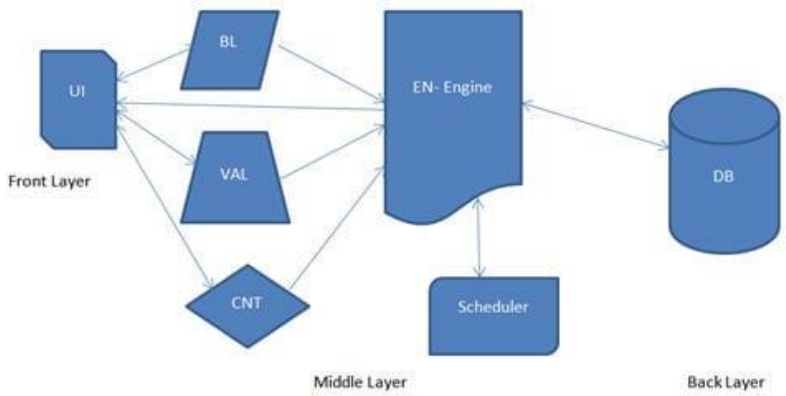
Now let's talk about how we can imply integration testing in Black box technique.

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.

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:



Now, having seen the architecture of **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, semi-annually & annually)

DB – Is the Database.

the entire web application, as a single unit, Integration testing, in this case, will focus on the flow of data between the modules.

The questions here are:

1. How the BL, VAL and the CNT module will read and interpret ate the data entered in the UI module?
2. Is BL, VAL and CNT module receiving the correct data from UI?
3. In which format the data from BL, VAL and CNT is transferred to the EQ module?
4. How will the EQ read the data and extract the query?
5. Is the query extracted correctly?
6. Is the Scheduler getting the correct data for reports?
7. Is the result set received by the EN, from the database is correct and as expected?
8. Is EN able to send the response back to the BL, VAL and CNT module?
9. Is UI module able to read the data and display it appropriately to the interface?

In the real world, the communication of data is done in an XML format. So whatever data the user enters in the UI, it gets converted into an XML format.

In our scenario, the data entered in the UI module gets converted into XML file which is interpreted by the 3 modules BL, VAL and CNT. The EN module reads the resultant XML file generated by the 3 modules and extracts the SQL from it and queries into the database. The EN module also receives the result set and converts it into an XML file and returns it back to the UI module which converts the results in user readable form and displays it.

In the middle we have the scheduler module which receives the result set from the EN module, creates and schedules the reports.

So where Integration testing does comes into the picture?

Well, testing whether the information/data is flowing correctly or not will be your integration testing, which in this case would be validating the XML files. Are the XML files generated correctly? Do they have the correct data? Are the data is being transferred correctly from one module to another? All these things will be tested as part of Integration testing.

Try to generate or get the XML files and update the tags and check the behaviour. This is something very different from the usual testing which testers normally do, but this will add value to the tester's knowledge and understanding of the application.

Few other sample test conditions can be as follows:

- Are the menu options generating the correct window?
- Are the windows able to invoke the window under test?
- For every window, identify the function calls for the window that the application should allow.
- Identify all calls from the window to other features that the application should allow
- Identify reversible calls: closing a called window should return to the calling window.
- Identify irreversible calls: calling windows closes before called window appears.
- Test the different ways of executing calls to another window e.g. – menus, buttons, keywords.

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.

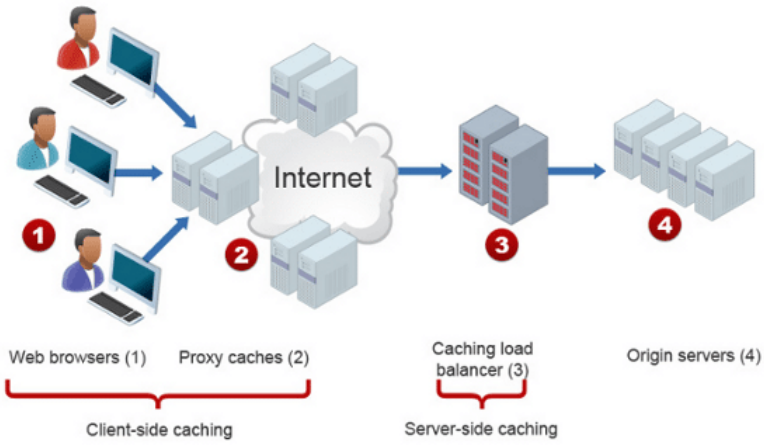
Many integration test cases can be written for this specific site. The above four points are just an example to understand what Integration test cases are included in testing.

The difference between Integration Testing and System Testing?

INTEGRATION TESTING	SYSTEM TESTING
It is a low level testing	It is a high level testing

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

Web Application Testing



In today's ever-changing and competitive web-based business scenario, organizations always need to **test their web based applications** before the launch of their website.

By web application testing, any organization can be sure that the web application will work perfectly and will be easily accepted by the end-users. The web application testing techniques also check the web application's browser compatibility; load testing, scalability testing, stress testing and resolution testing.

Testing Methods for Web Application Testing

Here are few of the basic testing techniques for web application testing:

1. Functionality Testing:

In Functional testing we need check the each components are functioning as expected or not, so it is also called as "**Component Testing**". Functional testing is to testing the functionality of the software application under test. Basically, it is to check the basic functionality mentioned in the functional specification document. Also check whether software application is meeting the user expectations. We can also say that checking the behaviour of the software application against test specification.

In this testing activities should include:

Link Testing:

Check for all **broken links** on your web pages & all links are working correctly. Along with you can check the different links on you web pages:

- Internal links
- Outgoing links
- MailTo Links
- Anchor Links

Web form testing:

In the web application testing the "Forms Testing" is the essential part of testing any web site. The main purpose of Form is to get the information from user & store into the database. And keep on interact with them. Below are the test cases which should be considered while doing form testing:

- First thing to test in the form is the Validations on each form fields. Here two types of Validation need to be consider – "Client side" & "Server side" validations.
- Check default values are being populated
- Check for all Mandatory fields. Check if a user not entered a required field showing a mandatory error message.
- Add information using form & update information using form.
- Tab orders.
- Check for the default values of fields.
- Forms are optimally formatted for better readability
- In Negative testing enter invalid inputs to the forms fields.

Cookies testing:

Cookies are the small text file that gets saved the user's system. These files are saved in the desired location and used by the browsers to use the cookies. Basically used to maintain the session mainly

login sessions the Cookies are used. Instructive information is recorded in the cookie (like Sessions) and that can be retrieved for the web pages. User can able to enable or disable the Cookies in browser options. The basic test to check if cookie is stored in the user's machine in the encrypted format. (I am writing separate article on Cookie Testing, post it soon)

So let's see what should be tested in Cookies testing:

- Test the application by disabling the Cookies
- Test the application after corrupting the cookies.
- Check the behaviour of application after removing the all the cookies for the website you are testing.
- Check website writing cookies are working or not on different browser.
- Check if cookies for authenticated login are working or not.
- Check if behaviour of application after deleting the cookies (sessions) by clearing cache or after cookies expired
- Check if login to application after deleting the cookies (sessions)

Test HTML and CSS:

To optimize web site in the search engine then testing of HTML and CSS make important role. In this testing check that different search engines can crawl your site without any error. You should check for all Syntax Errors, Colour Schemas in the readable, check for Standard Compliance and the standards are followed like W3C, ISO, ECMA, IETF, WS-I, OASIS

Test business workflow:

This would include:

- End to end testing of business scenarios/workflow which ensure that the completeness of website testing.
- Testing of Positive as well as negative scenarios.

2. Usability testing:

Now a day's the Usability testing is playing an important role of any web application. This testing is to be carried out by testers like you to ensure that cover all possible test cases which targeted audience of the web application are doing regularly.

This would include –

Navigation testing of the web site:

- All possible options like Menus, Links or buttons on web pages should be visible & accessible from all the web pages.
- Web pages navigation should be Easy to use
- Help instruction content should be clear & should satisfy the purpose.
- All options on header, footer & left/right navigation should be consistent throughout the pages.

Content testing of the web site:

- No spelling or grammatical errors mistake in content throughout the page.
- Alt text should be present on Images
- No broken images
- Your task is to validate all for UI testing
- Follow some standard on content building on web page
- All content should be legible & easy to understand.
- Dark colour infuriates the users, so avoid using dark colours in the theme.
- Proper size images should be placed on web page
- All the anchor text links should be working properly.

3. Compatibility testing.

In software application testing the Compatibility testing is the non-functional part of testing. It is ensuring that how application's working in the supported environments. Customers are uses web application on different Operating systems, Browser compatibility, computing capacity of Hardware Platform, Databases and Bandwidth handling capacity of networking hardware. The Compatibility testing is to make sure that "Is web application show correctly across different devices?"

This would include-

Browser Compatibility Test:

Web applications are rendering differently on different browsers. The objective of browser compatibility testing is to ensure that no any errors on the different web browsers while rendering the sites. In Browser Compatibility Testing you need to ensure that your web application is being displayed properly on different browsers. Also check AJAX, JavaScript and authentication are functioning correctly.

OS compatibility:

In new technology newer graphics designs are used & different APIs are used which may not work on different Operating systems. Also on rendering of different objects like text fields, buttons may display different on different Operating System. So testing of web application should be carried out on different OS like Windows, MAC, Solaris, Unix, Linux with different OS flavors.

Mobile browsing:

In latest Mobi technology you also test out Mobile Browser Compatibility too. It may be possible of Compatibility issues on Mobile browsers. So in the new Mobi technology age you testing of web pages on mobile browsers should be carried out.

4. Database Testing:

Data reliability is key part in the Database testing. So for web application should be thoroughly tested.

Testing activities would include-

- Check if queries are executed without any errors.
- Creating, updating or deleting data in database should maintain the data integrity.
- More time should not take to execute the queries, if required tune the queries for better performance.
- Check load on database while executing heavier queries & check the result.
- Collect data from database & represent on the web pages correctly.

5. Crowd Testing:

Crowd testing is when a large group of perfect strangers try your product then give you phenomenally helpful feedback on usability, bugs and features.

To test the software application Crowd testing can be used. It not limited to web applications, but for all kinds of applications including mobile application testing. Crowd testing is dependent on the quality of the crowd. Also it depends on a crowd that is composed out of a large group of diver's people. *It used do system tests for performance and usability testing. Simply this* is complementary to 'normal' testing. The mainly complicated job of crowd testing is determining a good enough crowd.

6. Interface Testing:

In the Interface testing mainly three areas should be covered: Web Server, Application Server and Database Server. Ensure that all the communications between these all servers should be carried out correctly. Verify that if connection between any servers is reset or lost then what is happing. Check if any request interrupts in-between then how application is responding. On returns of any error

from web server or database server to application server then error should be Errors are handled properly & display such errors to the user.

- Web Server: Check if all web requests are accepting and not any requests are denied or leakages.
- Application Server: Check if request is sending correctly to the any server & displayed correctly. Check if errors are catch properly & displayed to admin user.
- Database Server: Check if database server is returns correct result on query request.

Check if all three servers are connected to each & test request is processing correctly. And any error in between then error should be displayed to user.

7. Security testing:

The security testing is carried out to ensure that is any information leakage in terms of encrypting the data. In e-commerce website the Security testing is play an important role. In this testing check if secure information is to check whether how to store sensitive information such as credit cards etc.
Testing Activities will include-

- Check if unauthorized access to secure pages, if user changes from "https" to "http" (secure to non-secure) in secure pages then proper message should be display and vice versa.
- Check if accessing internal pages directly entering URLs in browser. If login is required then user should redirected to login page or appropriate message should be displayed.
- Most of the information related to transactions, error messages, login attempts should be logged in log file.
- Check if restricted files are able to access for download.
- Check if internal Web directories or files are not accessible unless & until not configured for download.
- Check if CAPTCHA is added & working properly for logins to prevents automates logins attempts.
- Check if try to access others information by changing parameter in query string. For example if you are editing the information & in URL you are seeing UserID = 123, try to change this parameter values & check if application is not providing the other users information. It should display Access denied for this user to view others users information.
- Check if sessions are got expired after pre-defined amount of time if user not using session.
- Check if user not able to pass login page for invalid username/password combination.
- Check if user is navigated to encrypted SSL pages for secure website.

8. Performance Testing:

Performance testing is to check the web application is working under the heavy load. This tests the server response time and throughput under various load conditions.
Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation.
Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.

The above image clearly explains to us that ***performance testing is the superset for both load & stress testing.*** Other types of testing included in performance testing are spike testing, volume testing, endurance testing, and scalability testing. Thus, performance testing is basically a very wide term.

Performance Testing Goal:

The primary goal of performance testing includes establishing the benchmark behavior of the system. There are a number of industry-defined benchmarks which should be met during performance testing.

Performance testing does not aim to find defects in the application. It also does not pass or fail the test. Rather, it addresses the critical task of setting the benchmark and standard for an application. Performance testing should be done very accurately. Close monitoring of the application/system performance is the primary characteristic of performance testing.

The benchmark and standard of the application should be set in terms of attributes like speed, response time, throughput, resource usage, and stability. All these attributes are tested in a performance test.

Example:

For instance, you can test the application network performance through 'Connection Speed vs. Latency' chart. Latency is the time difference between the data to reach from the source to destination.

A 70kb page would not take more than 15 seconds to load for the worst connection of 28.8kbps modem (latency=1000 milliseconds), while the page of the same size would appear within 5 seconds for the average connection of 256kbps DSL (latency=100 milliseconds).A 1.5mbps T1 connection (latency=50 milliseconds) would have the performance benchmark set as 1 second to achieve this target.

Example would be that of a Request-response model. We can set a benchmark that the time difference between the generation of request and acknowledgment of response should be in the range of x ms (milliseconds) and y ms, where x and y are the standard digits.

A successful performance test should project most of the performance issues, which could be related to database, network, software, hardware, etc.

9. Load Testing –

It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc. are also monitored.

Load testing is meant to test the system by constantly and steadily increasing the load on the system until it reaches the threshold limit. It is a subset of performance testing.

Load testing can be easily done by employing any of the suitable automation tools available in the market. WAPT and Load Runner are two such famous tools that aid in load testing. Load testing is also famous by names like **volume testing** and **endurance testing**.

However, volume testing mainly focuses on databases. Endurance testing tests the system by keeping it under a significant load for a sustained time period.

The sole purpose of load testing is to assign the system the largest job it can possibly handle to test the endurance of the system and monitor the results. An interesting fact here is that sometimes the system is fed with an empty task to determine the behaviour of the system in the zero-load situation.

The attributes which are monitored in a load test include peak performance, server throughput, response time under various load levels (below the threshold of break), and adequacy of H/W environment, how many user applications it can handle without affecting the performance.

Goals of Load Testing:

Loading testing identifies the following problems before moving the application to market or Production:

- Response time for each transaction
- Performance of System components under various loads
- Performance of Database components under different loads
- Network delay between the client and the server
- Software design issues
- Server configuration issues like a Web server, application server, database server etc.
- Hardware limitation issues like CPU maximization, memory limitations, network bottleneck, etc.

Load testing will determine whether the system needs to be fine-tuned or modification of hardware and software is required to improve performance.

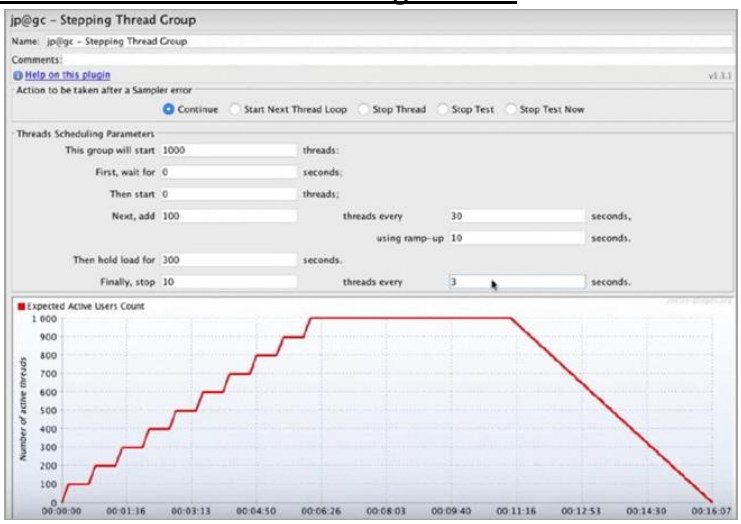
Examples of load testing include

- Downloading a series of large files from the internet
- Running multiple applications on a computer or server simultaneously
- Assigning many jobs to a printer in a queue
- Subjecting a server to a large amount of traffic
- Writing and reading data to and from a hard disk continuously

Let's consider to check the email functionality of an application that could be flooded with 1000 users at a time. Now, 1000 users can fire the email transactions (read, send, delete, forward, reply) in many different ways.

If we take one transaction per user per hour, then it would be 1000 transactions per hour. By simulating 10 transactions/user, we could load test the email server by occupying it with 10000 transactions/hour.

Another Example of a load test is shown in the image below:



The above image depicts a load test done in the tool called JMeter. This test is done to identify how many users a system can handle. In this test, 100 users are added after every 30 seconds until the load reaches 1000 users. Each step takes 30 seconds to complete and JMeter waits for 30 seconds before kicking off the next step.

Once the load reaches 1000 threads, all of them will continue running for 300 seconds (5 mins) together and then finally stops 10 thread every 3 seconds.

Advantages and disadvantages of Load testing:

Following are the advantages of Load testing:

- Performance bottlenecks identification before production
- Improves the scalability of the system
- Minimize risk related to system downtime
- Reduced costs of failure
- Increase customer satisfaction

Disadvantages of Load testing:

- Need programming knowledge to use load testing tools.
- Tools can be expensive as pricing depends on the number of virtual users supported.

10) Stress Testing

Under stress testing, various activities to overload the existing resources with excess jobs are carried out in an attempt to break the system down. **Negative testing**, which includes removal of the components from the system is also done as a part of stress testing.

Also known as **fatigue testing**, this testing should capture the stability of an application by testing it beyond its bandwidth capacity.

Thus, basically, stress testing evaluates the behaviour of an application beyond peak load and normal conditions.

The purpose of stress testing is to ascertain the failure of the system and to monitor how the system recovers back gracefully. The challenge here is to set up a controlled environment before launching the test so that you can precisely capture the behaviour of the system repeatedly under the most unpredictable scenarios.

The issues that would eventually come out as a result of stress testing may include synchronization issues, memory leaks, race conditions, etc. If the stress test is checking how the system behaves in the situation of a sudden ramp up in the number of users, then it is called a spike test.

If the stress test is to check the system's sustainability over a period of time through slow ramp up in the number of users, it is called as a soak test.

Stress Testing Goal:

The goal of stress testing is to analyse post-crash reports to define the behaviour of the application after failure.

The biggest challenge is to ensure that the system does not compromise the security of sensitive data after the failure. In a successful stress testing, the system will come back to normality along with all its components even after the most terrible breakdown.

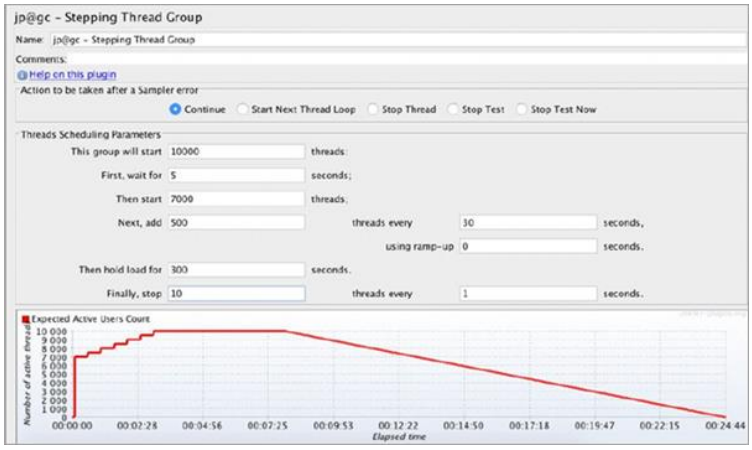
Example:

As an example, a word processor like Writer1.1.0 by OpenOffice.org is utilized in the development of letters, presentations, spread sheets etc. The purpose of our stress testing is to load it with excess characters.

To do this, we will repeatedly paste a line of data, until it reaches its threshold limit of handling a large volume of text. As soon as the character size reaches 65,535 characters, it would simply refuse to accept more data.

The result of stress testing on Writer 1.1.0 produces the result that it does not crash under the stress and it handles the situation gracefully which makes sure that the application is working correctly even under rigorous stress conditions.

Another Example of a load test which depicts a spike test through sudden ramp-up of 7000 users is shown below:



Difference between Load And Stress Testing

To summarize, let's observe the major differences between load testing, stress testing as well as performance testing in the below table:

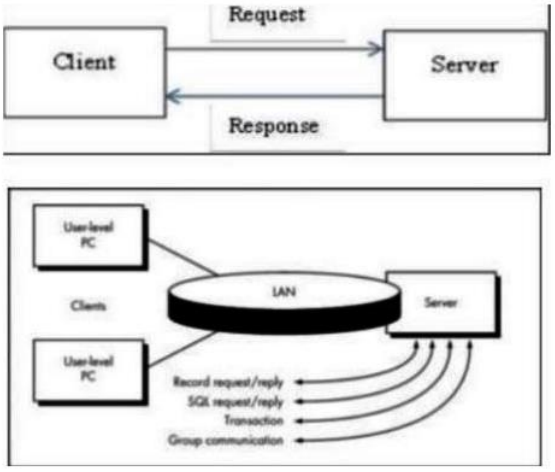
	Performance Testing	Load testing	Stress testing
Domain	Superset of load and stress testing	A subset of performance testing.	A subset of performance testing.
Scope	Very wide scope. Includes - Load Testing, Stress Testing, capacity testing, volume testing, endurance testing, spike testing, scalability testing and reliability testing, etc.	Narrower scope as compared to performance testing. Includes volume testing and endurance testing.	Narrower scope as compared to performance testing. Includes soak testing and spike testing.
Major goal	To set the benchmark and standards for the application.	To identify the upper limit of the system, set SLA of the app and see how the system handles heavy load volumes.	To identify how the system behaves under intense loads and how it recovers from failure. Basically, to prepare your app for the unexpected traffic spike.
Load Limit	Both – below and above the threshold of a break.	Till the threshold of break	Above the threshold of break
Attributes studied	Resource usage, reliability, scalability, resource usage,	peak performance, server throughput, response time under	Stability beyond bandwidth capacity,

	Performance Testing	Load testing	Stress testing
	response time, throughput, speed, etc.	various load levels **(below the threshold of break),** adequacy of H/W environment, the number of user app can handle, load balancing requirements, etc.	response time **(above the threshold of break),** etc.
Issues identified through this testing type	All performance bugs including runtime bloat, the scope for optimization, issues related to speed, latency, throughput, etc. Basically – anything related to performance!	Load balancing problems, bandwidth issues, system capacity issues, poor response time, throughput issues, etc.	Security loopholes with overload, data corruption issues at overload situation, slowness, memory leaks, etc.

Difference between Functional and Load Testing:

Functional Testing	Load Testing
Results of functional tests are easily predictable as we have proper steps and preconditions defined	Results of load tests are unpredictable
Results of functional tests vary slightly	Load test results vary drastically
Frequency of executing Functional Testing will be high	A frequency of executing load testing will be low
Results of functional tests are dependent on the test data	Load testing depends on the number of users.

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

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

In Client-server testing there are several clients communicating with the server.

1. Multiple users can access the system at a time and they can communicate with the server.
2. Configuration of client is known to the server with certainty.
3. Client and server are connected by real connection.
4. Testing approaches of client server system:

- Component Testing: One need to define the approach and test plan for testing client and server individually. When server is tested there is need of a client simulator, whereas testing client a server simulator, and to test network both simulators are used at a time.
- Integration testing: After successful testing of server, client and network, they are brought together to form system testing.
- Performance testing: System performance is tested when number of clients is communicating with server at a time. Volume testing and stress testing may be used for testing, to test under maximum load as well as normal load expected. Various interactions may be used for stress testing.
- Concurrency Testing: It is very important testing for client-server architecture. It may be possible that multiple users may be accessing same record at a time, and concurrency testing is required to understand the behaviour of a system in this situation.
- Disaster Recovery Business continuity testing: When the client server are communicating with each other , there exit a possibility of breaking of the communication due to various reasons or failure of either client or server or link connecting them. The requirement specifications must describe the possible expectations in case of any failure.
- Testing for extended periods: In case of client server applications generally server is never shutdown unless there is some agreed Service Level Agreement (SLA) where server may be shut down for maintenance. It may be expected that server is running 24X7 for extended period. One needs to conduct testing over an extended period to understand if service level of network and server deteriorates over time due to some reasons like memory leakage.
- Compatibility Testing: Client server may be put in different environments when the users are using them in production. Servers may be in different hardware, software, or operating system environment than the recommended. Other testing such as security testing and compliance testing may be involved if needed, as per testing and type of system.

  Following types of testing tools can be useful for client server testing:

  1) **Load/Stress Testing Tools**: Example Astra Site Test by Mercury Interactive, Silk Performer by Seague Software to evaluate web based systems when subjected to large volumes of data of transactions.

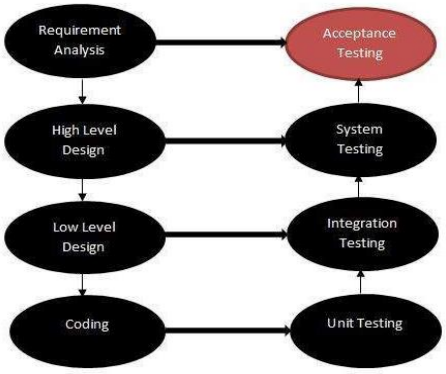
  2) **Performance Testing**: Example Load Runner to see the performance of the client server system.

  3) **UI testing**: Example Win Runner by Mercury Interactive to perform UI testing.

Acceptance Testing

Acceptance testing is performed after system testing is done and all or most of the major defects have been fixed. The goal of acceptance testing is to establish confidence in the delivered software/system that it meets the end user/customers' requirements and is fit for use Acceptance testing is done by user/customer and some of the project stakeholders.

This is a type of testing done by users, customers, or other authorised entities to determine application/software needs and business processes.



Acceptance testing is the most important phase of testing as this decides whether the client approves the application/software or not. It may involve functionality, usability, performance, and U.I of the application. It is also known as user acceptance testing (UAT), operational acceptance testing (OAT), and end-user testing.

It is one of the final stages of the software's testing cycle and often occurs before a client or customer accepts the new application. Acceptance tests are black-box system tests. Users of the system perform tests in line with what would occur in real-time scenarios and verify whether or not the software/application meets all specifications.

Types of user acceptance testing are:

1. Alpha & Beta Testing
2. Contract Acceptance Testing.
3. Regulation Acceptance Testing .
4. Operational Acceptance testing .

The requirement for acceptance testing should be cleared at the early stages of product development. By the time a design is completed, an acceptance test plan draft has to be prepared and given to the client for approval. After that, the acceptance test plan has to be approved by both parties.

When is it performed?

Acceptance Testing is the fourth and last level of software testing performed after System Testing and before making the system available for actual use.

Who performs it?

- *Internal Acceptance Testing* (Also known as Alpha Testing) is performed by members of the organization that developed the software but who are not directly involved in the project (Development or Testing). Usually, it is the members of Product Management, Sales and/or Customer Support.
- *External Acceptance Testing* is performed by people who are not employees of the organization that developed the software.

- *Customer Acceptance Testing* is performed by the customers of the organization that developed the software. They are the ones who asked the organization to develop the software. [This is in the case of the software not being owned by the organization that developed it.]
- *User Acceptance Testing* (Also known as Beta Testing) is performed by the end users of the software. They can be the customers themselves or the customers' customers.

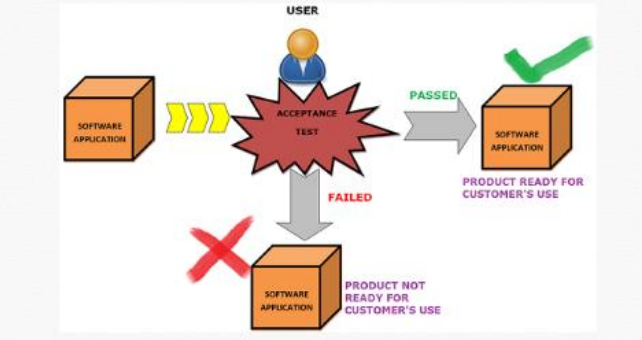
Why Acceptance Tests?

Though System testing has been completed successfully, the Acceptance test is demanded by the customer. Tests conducted here are repetitive, as they would have been covered in System testing.

This is because:

- To gain confidence in the product that is getting released to the market.
- To ensure that the product is working in the way it has to.
- To ensure that the product matches current market standards and is competitive enough with the other similar products in the market.

Acceptance Criteria



Acceptance criteria are defined on the basis of the following attributes

- Functional Correctness and Completeness
- Data Integrity
- Data Conversion
- Usability
- Performance
- Timeliness
- Confidentiality and Availability
- Installability and Upgradability
- Scalability
- Documentation

Benefits

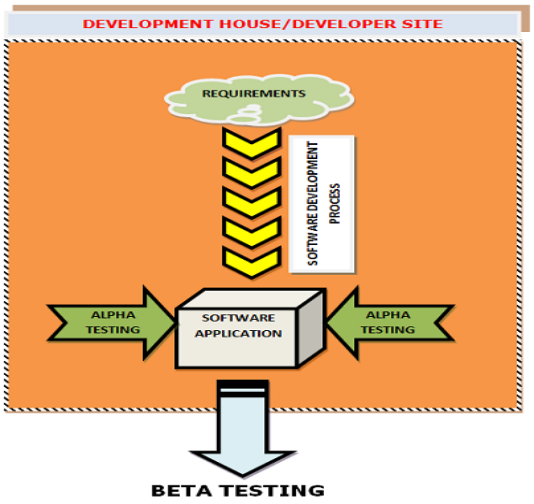
Acceptance testing has the following benefits, complementing those which can be obtained from unit tests:

- encouraging closer collaboration between developers on the one hand and customers, users or domain experts on the other, as they entail that business requirements should be expressed
- providing a clear and unambiguous "contract" between customers and developers; a product which passes acceptance tests will be considered adequate (though customers and developers might refine existing tests or suggest new ones as necessary)
- decreasing the chance and severity both of new defects and regressions (defects impairing functionality previously reviewed and declared acceptable)

Types of Acceptance Testing:

Broadly, there are two main **types of Acceptance Testing**

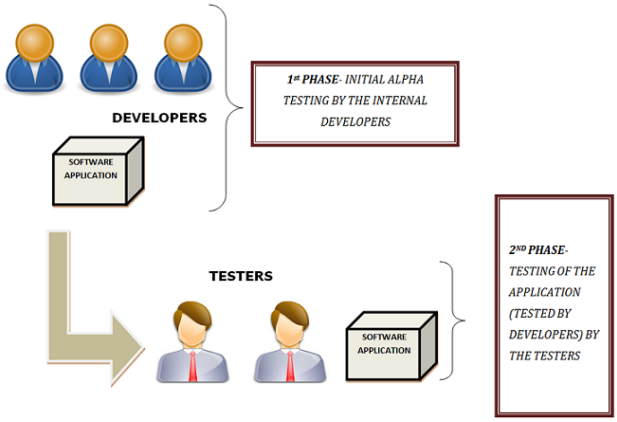
1. **Alpha Testing:**
   - **Alpha testing** is an **on-site user acceptance testing** of the software product that is performed prior to release of the product to clients or users.
   - It is carried out at developer's site by the group of skilled testers as well as developers, once the **software development process** reaches its culmination.
   - The team responsible for executing **alpha testing** is independent of designing team. The main intent behind **alpha testing** is to emulate the real-user environment, through black-box and white-box testing approach, as well as to ensure that the product performs and functions as per its intended functionality.
   - Ensuring the product readiness from user's perspective not only delivers confidence in the system but also helps in assessing user's needs and expectations, which may be used to improve or upgrade the quality of the product before its market release.



This group is independent of designing team. The main intent behind **alpha testing** is to emulate the real-user environment, through **black-box testing** and **white-box testing** approach.

Alpha Testing Phases

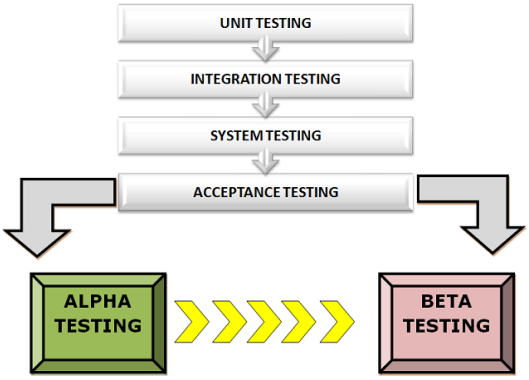
To ensure that the software performs accurately and does not impact the reputation of the organization, final testing is implemented by the company in the form of alpha testing. This testing is executed in two phases, which are:



- **Initial Testing by Internal Developers:** First phase of alpha testing consists of internal developers of the organization who are responsible for carrying out the initial alpha testing, with help of debugging software so as to locate bugs, quickly.
- **Testing of Application by Testers:** In second phase, developer's tested software is being handed or passed over to testing/QA team for additional testing of the application against user's needs and requirements in the real-environment.

When to Perform Alpha Testing?

Being a type of user acceptance testing, alpha testing is performed once the product has goes through stages of testing and is prepared for its release. Moreover, it is executed before beta testing, which is another important acceptance testing technique and can be termed as a **field testing**. During this testing, minor design changes can be made in the software to improve its quality as well as functionality. Additionally, software product is subjected to alpha testing from the developer's site, where independent developers monitor and record user experience and make necessary changes to enhance them.



Reasons for Alpha Testing:

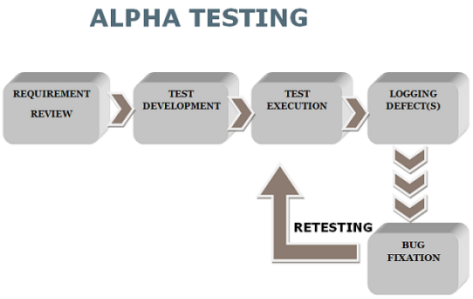
As the final stage of testing, **alpha testing** is an extremely important and popular **testing technique** that helps team to deliver a quality and effective software. Performed before the release of the product, this testing can be termed as the first round of independent testing that makes sure that the software, systems, and projects run as per their intended plan. Other reasons for alpha testing are:

- Ensures software system does not freeze part-way through the user experience.
- Refines the software product by finding and rectifying bugs that were not discovered through previous tests.
- Allows the team to test the software in a real world like environment.
- Validates the quality, effectiveness, and functionality of the software before it is released in the real world.
- To ensure the success of the software product.

Alpha Testing Process:

Alpha Testing of the software encompasses following activities in a sequential manner:

1. **Requirement Review:** Studying and analysing the specified and available requirements and specifications.
2. **Test Development:** Based on the outcomes of step 1, test plan along with the set of extensive test cases is defined and developed.



3. **Test Execution:** Execution of the test plan and the test cases.
4. **Logging Defects:** Logging the identified and detected bug found in the application.
5. **Bug Fixation:** Once all the defects and bugs are identified and logged, steps are taken to fix them.
6. **Retesting:** Re-testing of the application post bug-fixation process.

Alpha Testing Features:

- It is a type of **acceptance testing**.
- Alpha Testing is done near to completion stage of software development.
- It is, usually carried out in labs, thereby provides specific and controlled environment.
- Alpha Testing generally involves testing on the software prototype.
- It may be seen an **in-house testing** as this testing is performed by internal developers and testers within the organization.
- No public involvement and early detection of bugs.
- As Testing is performed at developer's site, it enables developer to record the error with the ease to resolve found bugs, quickly.
- Alpha Testing helps in gaining confidence in the **user-acceptance of software product**.
- It is preceded by unit testing, integration testing & system testing and followed by the beta testing.
- Black-box and White-box testing are two techniques to achieve alpha testing.
- Alpha Testing ensures the maximum possible quality of the software, before passing it to clients for executing **beta testing**.
- It is generally done for **testing software applications**, products and projects.

Advantage of Alpha Testing:

- Alpha Testing simulates the targeted environment for the software product along with the real-time users actions.
- **Alpha tests** are useful in detecting defects, especially in design and functions at an early stage of testing.
- Defects may be fixed or resolved, subsequently and quickly after their identification.
- Helps in gaining confidence for the user acceptance of the developed software product.

Alpha Testing Disadvantages:

- The difference between the tester's **test data** for testing the software, and the client or customer's data from their own perspectives may result into deviation in the software functioning.
- Although lab environment is used to simulate the real environment, still lab cannot furnish all the requirements of the real environment such as multiple variants of conditions, factors and circumstances.
- Alpha testing does not involves in-depth testing of the software product.

Beta Testing

- Though performed at the end of **software testing life cycle (STLC)**, beta testing is a salient testing type, executed after the culmination of **alpha testing**.
- This type of testing can be considered a form of external user acceptance testing, during which a version of the software, known as **beta version**, is released to a limited audience to be tested for **accessibility**, **usability**, **reliability**, **functionality**, and more.
- Carried out at client's or customer's site by the end-users or stakeholders, this is the last testing phase, before software's entry in the market.
- Beta testing can be seen as the second phase of testing after the completion of first phase comprising of **unit testing**, **integration testing**, and **system testing**.

Features of Beta Testing:

The main goal behind beta testing is to place the developed software application in the hands of the intended users under real-world environment & conditions, in order to evaluate and assess product from users-perspective, which may prove to be useful in exploring **defects** ,that are not desirable in final release of the software product. Other features of beta testing are:

- Beta testing is carried out in real-environment at user's site, thereby helps in providing a factual position of the quality.
- It is generally done for testing software products like utilities', applications, operating systems, etc.
- Beta testing is also called as **field testing** and **pre-release testing**.
- Testing is performed by the client, the stakeholder and the end-users.
- It is done after **alpha testing** and before the actual market release of the software product.
- Unlike alpha testing, which is a type of **white box testing**, beta testing is always **black box testing**.
- Beta Testing is preceded by Alpha testing and followed by the release of the final product.
- Performed in the absence of testers and presence of users, just opposite to alpha.
- Unlike alpha testers who are highly skilled, beta testers may involve the participation of naïve personnel like end-users.
- Generally, reliability, robustness and security of the software application are being evaluated by the intended users.
- Most of the suggestion and improvement gets due for getting implemented in the next future version of the software.

What is a Beta Version of Software?

You all must be aware of beta versions of the software that are released to restricted numbers of user to accept their feedback and suggestion on the quality improvement. Therefore, given below are the two types of beta version releases:

1. **Closed Beta Version:** Closed beta version, which is also known as **private beta**, is released to a group of selected and invited people, who test the software and evaluate its numerous features and specifications. This beta version represents a software that is capable to deliver value, but is not ready to be used by everyone because of issues like lack of documentation or missing vital features.
2. **Open Beta Version:**
   Open beta or public beta is opened to public. Any user as a tester can evaluate and assess the beta version to provide relevant feedback and reviews, in addition to detection of flaws(if any) to improve the quality of the final release. Additionally, this version serves the dual

purpose of demonstrating a product to potential customers and testing the software with the assistance of wide user based, which can help the team find various undetected errors and issues.

This releasing of beta version is oriented by the beta testing process.

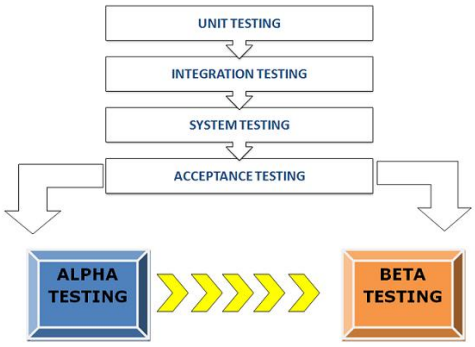
How is Beta Testing Performed?

As stated earlier, the process of beta testing is performed by a group of end-users. However, this process is not executed without any specific test strategy or plan. Before the testers- in this case the end users -execute the process of testing, a set of steps are followed by the test team to ensure the accuracy of as well as the effectiveness of the testing. Hence, the process of beta testing involves:

1. **Planning:** Like any other testing process, beta testing also begins with proper planning. During this stage, the team prepares a testing strategy and defines the goal of testing. Moreover, the team defines the users required for testing, the duration, as well as other necessary details related to the process.
2. **Participant Recruitment:** In the second stage of the process, the team recruits participants- a group of selected end users -for the purpose of testing. This group can vary as per the requirement of the organization and the product.
3. **Product Launch:** Once a team of testers (users) is recruited, a beta version of the product is launched or installed at the client or user side, which is then tested by the team for quality assurance.
4. **Collect & Evaluate Feedback:** After the culmination of the testing phase, the developers collect the feedback provided by the testers and evaluate it. Finally, based on the feedback, issues and bugs are fixed and resolved by the responsible individual.
5. **Closure:** Once all the issues are rectified and the exit criteria is met by the team, the beta testing is concluded and the promised incentives or rewards are offered to the participants.

When to Perform Beta Testing?

Acceptance Testing is the last **level of testing** that comprises of alpha testing and beta testing to ensure the market release of flawless product. Alpha testing is done at production site by the professional tester itself, whereas beta testing is performed at user's site (in Real world environment) by the group of intended users (Beta Testers).



After performing the in-house testing of the application in the form of alpha testing, system is handed over to the users to carry out the beta testing of the system, i.e. as soon as alpha testing of the application gets over, beta testing phase starts.

Advantages of Beta Testing:

Though performed at the end of software development and software testing life cycle, beta testing offers numerous benefits to software developers, testers, as well as the users. It is with the

assistance of this type of testing, that the team of developers and testers are able to test the software product before its final release in the market. Therefore, the advantages offered by beta testing are:

- With beta testing, the software developers can get their product/application tested prior to its release.
- It enables the user to install, test, and send feedback regarding the developed software, which helps the team improve its quality.
- Helps discover defects and issues in the system that were overlooked or left undetected by the team of software testers.
- With the assistance of beta testing, the effectiveness and impact of a product, on the users, can be examined.
- Enhances the quality, performance, and functionality of the software.
- Helps get direct feedback from the users.
- Reduces risk of product failure via user validations.

Disadvantages of Beta Testing:

Being informed about the disadvantages offered by a testing technique is always a sensible thing to do. It helps the team take necessary precautions and enables them to increase its effectiveness. Some of the disadvantages of beta testing are:

- During this type of testing, the software engineers have no control over the process of testing, as it is performed by the users in the real world environment.
- Another disadvantage of beta testing is that, it can be difficult to find end users willing to test the software.
- It can be a time consuming process and can delay the final release of the product.

Alpha Testing Vs Beta Testing

Alpha Testing	**Beta Testing**
1. This is the first stage of user acceptance testing.	1. This is the second stage of user acceptance testing.
2. Alpha testing is performed at developer's site.	2. Beta Testing is performed in real world environment.
3. Here, testing activities can be controlled.	3. Here, testing activities cannot be controlled.
4. In this type of testing, black box and white box testing techniques are involved.	4. In this type of testing only black box testing technique is involved.
5. It is performed after unit, integration, and system testing.	5. It is performed after alpha testing.

Regression Testing

Regression testing is a type of software testing which ensures that previously developed and tested software application working in the same way as it was working before recent code/configuration changes done.

In this testing, test cases are re-executed in order to check whether the earlier functionality of the software is working as projected and the new configuration/code changes have not introduced any new issues/bugs.

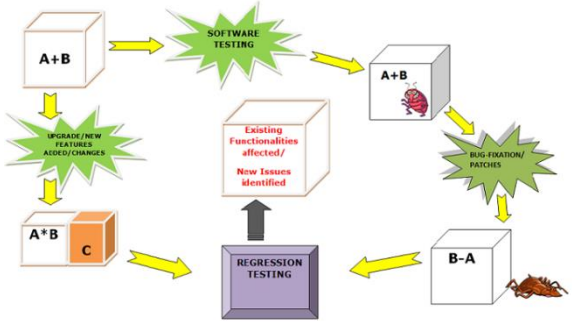
Need of Regression Testing:

This testing is required when there is any:

- Change in requirements and code is modified as per the changed requirements
- Added new features in product
- Bug fixing
- Fixing of performance related issues

Regression Testing Example

Below, given figure clearly defines the necessity and working of the regression testing.



We have a software application with functionality: **A+B**. On testing the application, a bug gets identified and detected. Patches & bug-fixation process including changes in the code is being carried out to remove or resolve the identified bug. However, in the process of bug-removal, the existing functionality of the application also gets affected: **B-A**. Thus, regression testing is carried out to evaluate whether existing functionalities has been impaired or not along with the occurrence of any new bug or issue in the event of changes and bug-fixation.

Further, we can also see that during upgrade or addition of new feature; **C**, the existing functionalities of the application; A+B has been impacted with the change in the functionality; **A*B**. Here, also regression tests are executed over software application to find out whether existing functionalities has been affected or not.

Types of Regression Testing

Often, regression testing is done through several phases of testing. It is for this reason, that there are several types of regression testing. A few have been mentioned below:

- **Unit regression** – Unit regression testing, executed during the unit testing phase, tests the code as a single unit. It has a narrow and focused approach, where complex interactions and dependencies outside the unit of code in question are temporarily blocked.
- **Partial regression** – Partial regression is performed after impact analysis. In this testing process, the new addition of the code as a unit is made to interact with other parts of older existing code, in order to determine that even with code modification, the system functions in silos as desired.
- **Complete regression** – Complete regression testing is often carried out when the code changes for modification or the software updates seep way back into the roots. It is also

carried out in case there are multiple changes to the existing code. It gives a comprehensive view of the system as a whole and weeds out any unforeseen problems. A sort of a "final" regression testing is implemented to certify that the build (new lines of code) has not been altered for a period of time. This final version is then deployed to end-users.

Types of Regression testing techniques:

We have four types of regression testing techniques. They are as follows:

1) Corrective Regression Testing: Corrective regression testing can be used when there is no change in the specifications and test cases can be reused.

2) Progressive Regression Testing: Progressive regression testing is used when the modifications are done in the specifications and new test cases are designed.

3) Retest-All Strategy: The **retest**-all strategy is very tedious and time consuming because here we reuse all tests which results in the execution of unnecessary test cases. When any small modification or change is done to the application then this strategy is not useful.

4) Selective Strategy: In selective strategy we use a subset of the existing test cases to cut down the retesting effort and cost. If any changes are done to the program entities, e.g. functions, variables etc., then a test unit must be rerun. Here the difficult part is to find out the dependencies between a test case and the program entities it covers.

When to use it:

Regression testing is used when:

- Any new feature is added
- Any enhancement is done
- Any bug is fixed
- Any performance related issue is fixed

Advantages of Regression testing:

- It helps us to make sure that any changes like bug fixes or any enhancements to the module or application have not impacted the existing tested code.
- It ensures that the bugs found earlier are NOT creatable.
- Regression testing can be done by using the automation tools
- It helps in improving the quality of the product.

Disadvantages of Regression testing:

- If regression testing is done without using automated tools then it can be very tedious and time consuming because here we execute the same set of test cases again and again.
- Regression test is required even when a very small change is done in the code because this small modification can bring unexpected issues in the existing functionality.

Benefits of Regression Testing

Conducting regression tests benefits companies in a number of ways such as:

- It increases the chance of detecting bugs caused by changes to software and application
- It can help catch defects early and thus reduce the cost to resolve them
- Helps in researching unwanted side effects that might have been occurred due to a new operating environment
- Ensures better performing software due to early identification of bugs and errors
- Most importantly, it verifies that code changes do not re-introduce old defects

GUI Testing

- GUI testing is a testing technique in which the application's user interface is tested whether the application performs as expected with respect to user interface behaviour.
- GUI Testing includes the application behaviour towards keyboard and mouse movements and how different GUI objects such as toolbars, buttons, menubars, dialog boxes, edit fields, lists, behavior to the user input.
- Graphical User Interface GUI testing is a process of testing an application's visual elements, such as images, texts, buttons, etc., to validate their expected performance as well as their functional accuracy. By testing the graphical user interface, the team can validate various graphical icons and visual indicators, like the **radio button, check box, text box, list box, menu, dialog box, bars**, among other things.
- Performed manually or with the assistance of automation tools, the GUI tests are focused on testing the functionality as well as elements of the graphical user interface, which are visible to the user to ensure that it meets the requested specifications. This goal is achieved through the use of a variety of test cases and test scripts.

GUI Testing: Features

Clarity about GUI testing can further be achieved by understanding its various features. Hence, following are some critical features of GUI testing and other aspects related to it.

1. The process of GUI testing is more difficult than command in line interface testing.
2. Majority of testing tools used for GUI testing are primarily focused on regression testing.
3. It also confirms the appearance elements, such as the font and images, conforms to their design specifications.
4. Manual tests performed for GUI testing are time consuming and resource intensive.
5. Automated testing can be more challenging for GUI as the user interface can change often.
6. It is performed from the perspective of the user rather than the developer or tester.
7. Assists the team in gathering necessary information, which allows them to decide whether an application is ready for deployment or not.

Different Ways of GUI Testing:

There are mainly three important GUI testing techniques, which are used by software testers around the world to validate the accuracy as well as the quality of graphical user interface elements. These three techniques are:

I. **Manual Based Testing:** Testers checked all the graphics with the prerequisites in business document manually. For example, a test can check the multiplication (33X5) manually.

II. **Record & Replay:** This is an automated GUI testing tool; all the tasks are recorded during the test. The recorded steps or tasks are executed with expected behavior. Now, this can be repeated several times with various data sets.

III. **Model Based Testing:** This type of testing acts as a graphic description (just like science models). Such testing predicts the system's behavior and this technique generates the test cases efficiently. Charts and decision tables are some of the model-based techniques.

GUI Testing Checklist:

- The Elements of GUI like size, font, Length, width and so on must check.
- Check the correct display of error messages.
- Font size and readability of fonts.
- Alignment of text.

- Quality and clarity of images.
- Alignment of images should be proper.
- Positioning of all the GUI elements for different resolution.

Advantages of GUI Testing:

By implementing the process of GUI testing during the early stages of software development life cycle (SDLC), the team can enjoy several advantages, such as an increase in the speed of development, improvement in quality, and reduction of the risks at the end of the cycle. Apart from these, there are other advantages of GUI testing, like:

- Tests the user interface from the users perspective.
- Efficiently reduces the number of risks towards the end of development life cycle.
- Offers developers and testers ease of use and learning.
- Helps validate the compliance of various icons and elements with their design specifications.
- Increases the reliability and improves quality of the product.

Disadvantages of GUI Testing:

- Though the advantages offered by GUI testing are numerous, there are still few drawbacks of this testing, which require acknowledgment. Hence, here are the disadvantages/drawbacks of GUI testing.
- It requires more memory resources, which leads the system to perform slowly.
- The process of testing is time consuming and may require extra software for running GUIs.
- Since the interface of an application changes frequently, the team might have to refactor recorded test script to improve its accuracy.
- Limited access or no access to the source code makes the process of testing difficult.

Some Test Cases to Consider in UI Testing

When you are testing the GUI, there are many test cases to take into consideration. These are not the only ones of course but it's a good checklist!

- **Font type and size.** Make sure that the font is the same on all screens (or at least is in the same family). Keep it professional. Same for the font sizes (headers, body text, etc.)
- **Colors.** Don't be inconsistent. Stay with the same colors and follow a style guide. You can't be using four different variations of orange (unless it's really part of the design). Look at hyperlinks, background, buttons, body text, etc.
- **Icon styles.** You shouldn't go for five different styles of icons. If you choose "flat" icons, stay with flat icons.
- **Visual inconsistencies.** Consistency is key, always. The look and feel should be the same throughout the app, and other than the look and feel, abbreviations also have to be consistent.
- **Dialog box consistency:** Again with consistency. If your user "exits" in some dialog boxes, you shouldn't use "leave" on others.
- **Required fields.** It is always better to specify that a field is required by adding an asterisk and by providing the user with a sort of warning if the data is not given.
- **Data type errors.** Always ensure that the right type of data is given (dates, ages, weight, etc.).

- **Same document, multiple opens.** When a document is opened or downloaded more than once, instead of overwriting another one, you can rename it by adding a number to the file name.
- **Field widths.** Obviously, if there is a certain amount of characters permitted and that the data entered shouldn't exceed a specific number, you should make this clear.
- **Onscreen instructions.** Screens that are not clear should contain some sort of onscreen instructions that will help and guide the user. Keep it brief and straight to the point!
- **Progress bars.** When waiting for results, progress bars are great so that users understand that they have to wait for something and that the process is still on.
- **Confirm savings.** If you can do changes to the app without needing to save, it's still always nice to make sure that the user doesn't want to save before moving to another screen.
- **Confirm delete.** As we confirm savings, it's always good to confirm that the user wants to delete an item. I am sure that many of you (as I) have deleted something on a page without really wanting to do so!
- **Drop-down list type ahead.** When you have hundreds of options to choose from, it's much nicer to have the option to type ahead than to have to go through the whole list.
- **Invalid options.** Sometimes, for some specific options, you need to validate other characteristics before being able to use that option. This option should only be shown available when all characteristics are fulfilled. An example would be not showing that you can go to the next screen before fulfilling all the required information.
- **Menu items.** Only show menu items that are available at the moment instead of showing all the items even though they are not available.
- **Error messages.** Error messages should be informative.
- **Working shortcuts.** If you've got shortcuts on your application, make sure they all work, no matter which browsers are being used.