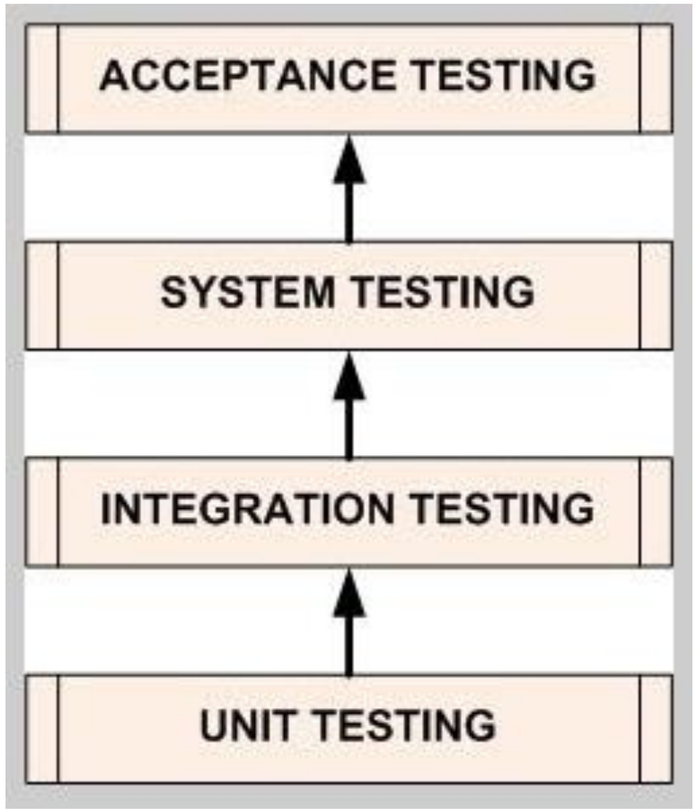
TestLeaf

Always Ahead

Software Testing Levels

TestLeaf
Always Ahead

Software Testing Levels



TestLeaf
Always Ahead

TestLeaf
Always Ahead

Unit Testing

- A unit is the smallest testable part of any software.
- 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.
- It is a level of software testing where individual units/ components of a software are tested.
- Isolate a section of code and verify its correctness
- Done during the development (coding) of an application.
- Unit testing is a Whitebox testing technique that is usually performed by the developer. Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing.

Unit Testing Tools

There are several automated tools available to assist with unit testing. We will provide a few examples below:

- **Junit**: Junit is a free to use testing tool used for Java programming language. It provides assertions to identify test method. This tool test data first and then inserted in the peace of code.

- **NUnit**: NUnit is widely used unit-testing framework use for all .net languages. It is open source tool which allows writing scripts manually. It supports data-driven tests which can run in parallel.

- **JMockit**: JMockit is open source Unit testing tool. It is code coverage tool with line and path metrics. It allows mocking API with recording and verification syntax. This tool offers Line coverage, Path Coverage, and Data Coverage.

- **EMMA**: EMMA is an open-source toolkit for analyzing and reporting code written in Java language. Emma support coverage types like method, line, basic block. It is Java-based so it is without external library dependencies and can access to the source code.

- **PHPUnit**: PHPUnit is a unit testing tool for PHP programmer. It takes small portions of code which is called units and test each of them separately. The tool also allows developers to use pre-define assertion methods to assert that system behave in a certain manner.

Integration testing

- **INTEGRATION TESTING** is a level of software testing where individual units are combined and tested as a group.
- Integration Testing focuses on checking data communication amongst the modules.

Approaches/Methodologies/Strategies of Integration Testing:
The Software Industry uses variety of strategies to execute Integration testing

Big Bang Approach :
Incremental Approach: which is further divided into following
Top Down Approach
Bottom Up Approach
Sandwich Approach - Combination of Top Down and Bottom Up

Big Bang Approach

- Here all component are integrated together at once, and then tested.
- Involves integrating the modules to build a complete software system.
- This is considered a high-risk approach because it requires proper documentation to prevent failure.

Advantages:

Convenient for small systems.

Disadvantages:

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

Incremental Approach

- In this approach, testing is done by joining two or more modules that are ***logically related***. Then the other related modules are added and tested for the proper functioning.
- Process continues until all of the modules are joined and tested successfully.
- This process is carried out by using dummy programs called **Stubs and Drivers**. Stubs and Drivers do not implement the entire programming logic of the software module but just simulate data communication with the calling module.
- **Stub**: Is called by the Module under Test.
- **Driver**: Calls the Module to be tested.

- Incremental Approach in turn is carried out by two different Methods:
  - **Bottom Up**
  - **Top Down**

Bottom up Integration

In the bottom up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing

Advantages:

- Fault localization is easier.
- No time is wasted waiting for all modules to be developed unlike Big-bang approach

Disadvantages:

- Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
- Early prototype is not possible

Top down Integration

- In Top to down approach, testing takes place from top to down following the control flow of the software system.
- Takes help of stubs for testing.

Advantages:

- Fault Localization is easier.
- Possibility to obtain an early prototype.
- Critical Modules are tested on priority; major design flaws could be found and fixed first.

Disadvantages:

- Needs many Stubs.
- Modules at lower level are tested inadequately.

TestLeaf
Always Ahead

System Testing

- **SYSTEM TESTING** is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.
- System test falls under the **black box testing** category of software testing.
- System Testing is the third level of software testing performed after Integration Testing and before Acceptance Testing.

TestLeaf
Always Ahead

Acceptance Testing

ACCEPTANCE TESTING is a level of software testing where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

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.

Regression Testing
After enhancements in the application, testing is done to ensure that the existing functionality of the application is working fine

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

Smoke Testing Build verification test done to ensure that major functionalities of the application are working fine

Sanity Testing is done to ensure that the addressed scope of the release is tested and same is working fine

Exploratory Testing is done with the experience of the tester to explore the application to understand the functionality and find defects. This is done without formal documentation

Exhaustive testing is a test approach in which all possible data combinations are used for testing. Exploratory testing includes implicit data combinations present in the state of the software/data at the start of testing.

Security Testing is done to check how the application is secure from internal and external threats. This testing includes how much software is secure from malicious programs, viruses and how to secure & strong the authorization and authentication processes are.

Load Testing is to check how much load or maximum workload a system can handle without any performance degradation.

Performance Testing
Testing performed to check how application or software performs under workload in terms of responsiveness and stability

Stress Testing
Stress Testing is done to check the breakpoint at which application crashes under maximum workload

UI testing involves the testing of Graphical User Interface of the Software. This testing ensures that the GUI should be according to requirements in terms of color, alignment, size and other properties.

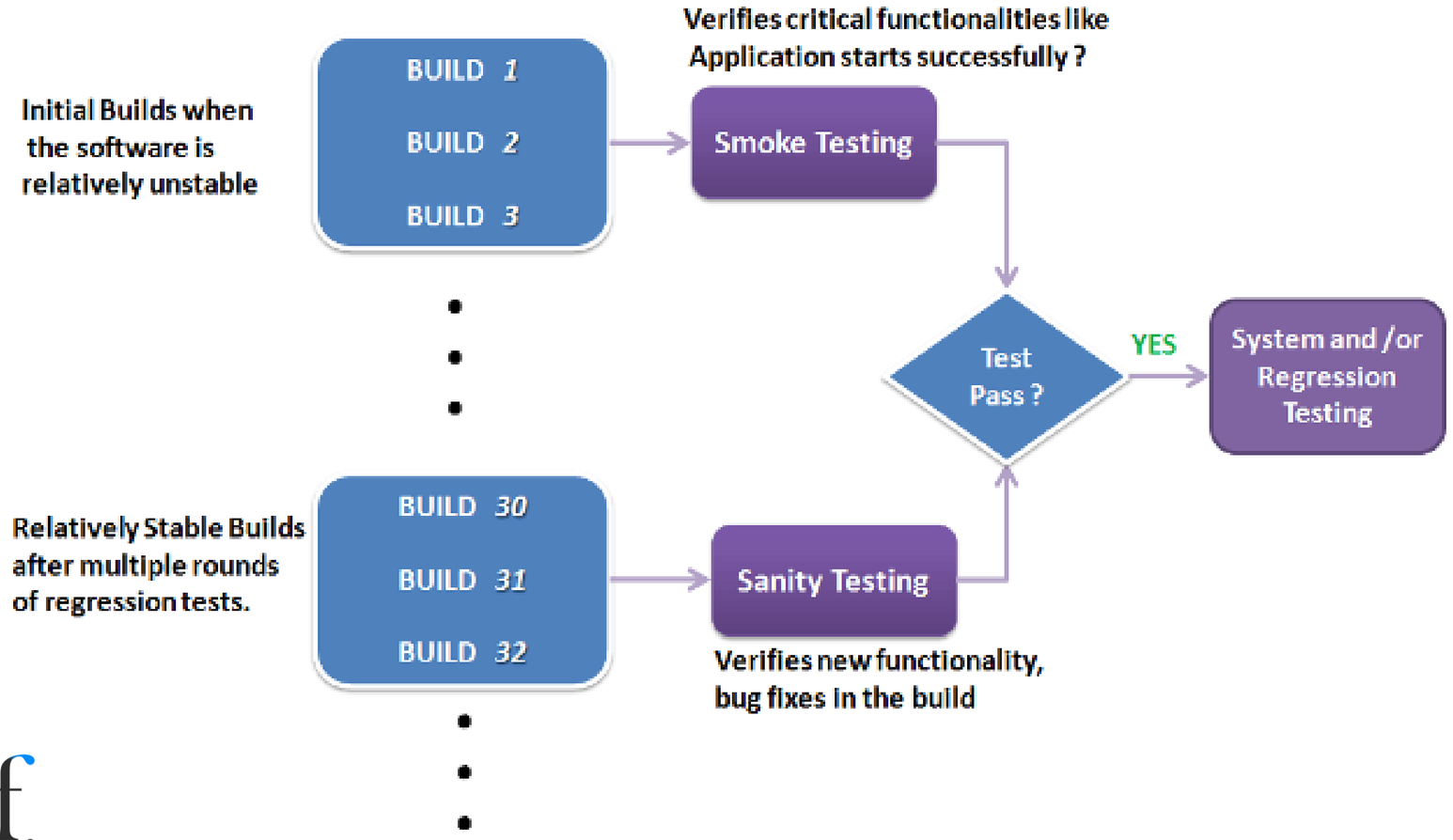
Usability testing ensures that a good and user friendly GUI is designed and is easy to use for the end user. UI testing can be considered as a sub part of Usability testing.

Compatibility Testing is a type of Software testing to check whether your software is capable of running on different hardware, operating systems, applications, network environments or Mobile devices.

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

Smoke and Sanity Testing

Smoke and Sanity testing are the most misunderstood topics in Software Testing. There is enormous amount of literature on the subject, but most of them are confusing



TestLeaf
Always Ahead

Software Build

- If you are developing a simple computer program which consists of only one source code file, you merely need to compile and link this one file, to produce an executable file. This process is very simple.
- Usually this is not the case. A typical Software Project consists of hundreds or even thousands of source code files. Creating an executable program from these source files is a complicated and time-consuming task.
- You need to use "build" software to create an executable program and the process is called "**Software Build**"

Smoke Testing

- Smoke Testing is performed after software build to **ascertain that the critical functionalities of the program is working fine**.
- It is executed **"before"** any detailed functional or regression tests are executed on the software build
- The **purpose is to reject a badly broken application**, so that the QA team does not waste time installing and testing the software application.
- In Smoke Testing, the **test cases chosen cover the most important functionality** or component of the system.
- The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system is working fine.
- For Example a typical smoke test would be - Verify that the application launches successfully, Check that the GUI is responsive ... etc.

Sanity Testing

- After receiving a **software build, with minor changes in code, or functionality, Sanity testing is performed to ascertain that the bugs have been fixed and no further issues are introduced due to these changes.**

- The goal is to determine that the proposed functionality works roughly as expected. **If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.**

- The **objective is "not" to verify thoroughly the new functionality**, but to determine that the developer has applied some rationality (sanity) while producing the software.

- For instance, if your scientific calculator gives the result of 2 + 2 =5! Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

TestLeaf
Always Ahead

Smoke And Sanity Testing - Diff

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

TestLeaf
Always Ahead