7 Common Types Of Software Testing

You may have come across business case studies about software products being launched with much fanfare, with millions of dollars in advertising and marketing spend, only to become colossal failures in the field or the marketplace. These can include anything from a mobile app that doesn't get as many eyeballs or downloads as expected to an automated aircraft system that fails mid-flight, causing tragic loss of human life. All these failures stem from a common source — **inadequacy in software testing.**

Software testing is one of the most critical components of the **Software Development Life Cycle (SDLC)** and yet so little is known about it. For instance, did you know that there are more than 150 different types of software tests that are being conducted today? And many more are being added regularly!
Though there are 150+ types of testing namely,

- Functional testing
- Load testing
- Exploratory testing
- Non-functional testing
- Performance testing
- Regression testing
- Sanity testing
- Security testing
- Smoke testing
- Stress testing
- Unit testing
- White-box testing
- Accessibility testing
- Acceptance testing
- Black box testing
- End to end testing
- And more to follow …

Some of these testing can be done manually or automated.
In this blog, we'll see the 7 most **common types of software testing**.

- ## Acceptance testing

Acceptance Testing is a form of **software testing** in which systems are tested for compliance with business and technological requirements to evaluate its suitability for final delivery to end customers. Simply put, Acceptance Testing assesses whether the given software system fulfills its purpose.
Generally, Acceptance Testing is conducted by teams of functional testers and developers against a set of functional specifications. It is a critical form of testing as it validates if the software meets the end-criteria for which it was developed. Often live information and real use cases are considered in Acceptance Testing, which makes it an integral part of the SDLC. Inadequate Acceptance Testing has historically caused major losses to several organizations, as the price of fixing errors far exceeds the cost of comprehensive testing.
Apart from the testing teams, Acceptance Testing can even be conducted by end-users, which is termed User Acceptance Testing (UAT) or Beta Testing. Often end-users provide the most valuable feedback that goes into developing a great product. Hence UAT is a vital part of Acceptance Testing and not only ensures the absence of bugs but also helps developers to proactively fill in functionality gaps before the product enters the market.

- ## Integration testing

Today, most software is developed in modules, which are then integrated to build a larger system. Often, the lack of compatibility among different modules is the chief source of software defects that affect its viability. Therefore Integration Testing is conducted in which individual software modules are integrated and tested as a whole. It evaluates the compliance of the ‘full’ system as opposed to its individual components.
Different forms of Integration Testing include:

- **String Testing**, which evaluates a collection of logically related modules after they have integrated and before production
- **Thread Testing**, which evaluates the capability of a system to carry out specific processes or threads, early on the integration phase

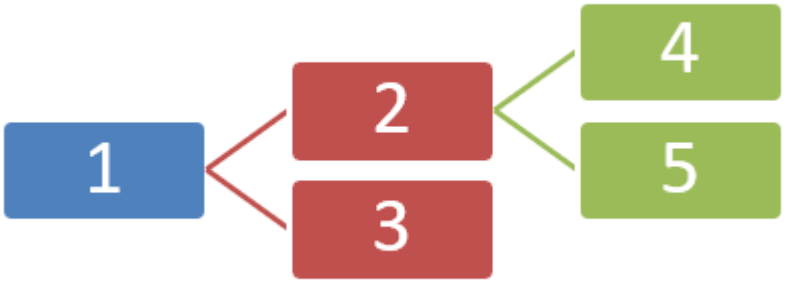
Integration Testing is implemented using Stubs and Drivers which are dummy programs that simulate data communication between modules, without implementing the complete system logic. The three common types of Integration Testing are based on their strategic approach. They include:
1. Big Bang Approach

This involves completing the entire integration process and conducting the testing of all its modules in a single phase.

2. Incremental Approach

The Incremental Approach, on the other hand, conducts testing in a multi-phased manner. Based on the order in which tests are conducted, this can be further sub-divided:

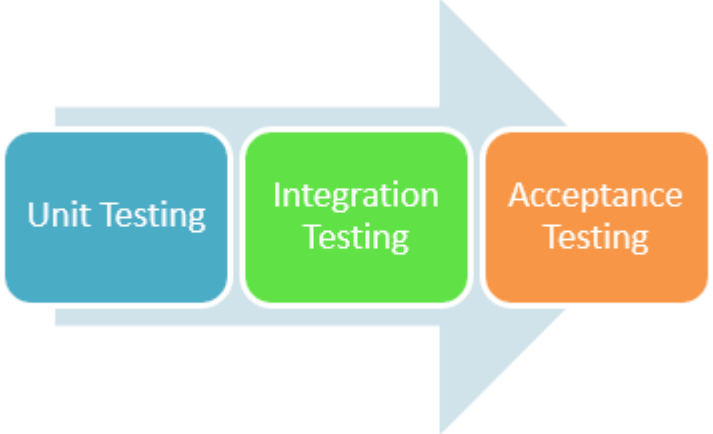
a. In the Top-Down approach, all the top integrated modules are tested first, then the branch of the module systematically, until finally the last related module is tested.



b. Bottom-Up

c.The Sandwich approach, as the name suggests, adopts a combination of the Top-Down and Bottom-Up approaches.

- **Unit testing**



Unit Testing is the fundamental building block of the entire testing family and involves testing the individual components that go into the complete system. It evaluates the smallest test-able part of the system and generally involves only a limited number of inputs/outputs. Under Unit Testing, the “units” that are tested include blocks of source code, program modules, usage/operating procedures. As may be expected, they are tested for fitness of purpose, i.e., whether they accomplish the tasks that they are supposed to execute and behave as expected.

- ## Functional testing

Functional Testing, as the name suggests, checks whether the software system meets all its functional specifications. As part of functional tests, various inputs are provided to the system in accordance with its functionality and the output is used to verify whether or not it meets the requirements.
Functional Testing plays a significant role in testing as it ensures whether an application is ready for release. Functional Testing can be manual or automated, and it generally falls in the category of Black Box testing, with the functional testers examining individual components vis-à-vis the entire system.

Some of the common types of Functional Testing include:

- Component Testing

Involves testing individual modules or components to evaluate its functionality, and includes code chunks, web pages, mobile screens, and so on.

- Smoke Testing

The origin of the term is believed to be in plumbing, where smoke was used to detect cracks in pipes. Similarly in Smoke Testing, the critical issues are the main focus, and the objective is to fix them first, rather than performing a comprehensive system testing.

- Sanity Testing

In this form of testing, new builds with minor updates are tested to check whether defects have been fixed in the new version and whether any new defects have been introduced. It is not a comprehensive set of tests but only a subset of the entire suite designed to examine the effect of software modifications.

- ## Performance Testing

While Functional Tests only check whether the system meets functional requirements, Performance Testing examines other equally critical factors such as the speed, stability, scalability, reliability, and responsiveness under specified workloads. The aim of Performance Testing is not only to find defects but to eliminate performance bottlenecks.
Performance Testing is among the most important form of testing, as it evaluates issues that are most likely to affect usage and directly impact the end-user, such as page load times, network response times, server request processing times, concurrent user volumes, and memory or CPU usage. It enables developers to pinpoint the issues that need to be addressed before the product can be deemed market-ready.
Some forms of Performance Testing are:

- Load Testing

Conducted by consistently increasing the load on a system to determine threshold values, it includes reading/writing large volumes of data, executing multiple applications, and so on. Volume Testing and Scalability Testing operate on a similar principle, by increasing the volume of data and the user load respectively.

- Stress Testing

Stress Testing checks whether systems operate in a graceful manner under stress, e.g., under low CPU, memory, or bandwidth conditions.

- Spike Testing

As the name suggests, Spike Testing creates periodic spikes in demands on the system to examine whether it continues to perform within acceptable limits.

- Soak/Endurance Testing

This involves testing the system under a constant load for a long duration and checking for memory leaks, system failure, overheating, and other such performance issues.

- **Regression Testing**

Regression Testing is one of the most common forms of testing and involves re-execution of previous test cases. It repeats all previous functional and non-functional tests to ensure that the system continues to perform satisfactorily even after changes, updates, or modifications. In case the system fails to perform, it would be a regression, hence the name Regression Testing. This form of testing can be termed Selective with only a subset of existing test cases

to analyze the impact of new code or it can be Progressive to ensure that new modifications do not adversely impact already existing functionality.

Based on the level of testing, the types of testing include:

- Unit Regression Testing, which narrowly focuses on code units while blocking complex interactions and dependencies.
- Partial Regression Testing, in which new code is tested against existing code to ensure acceptable performance of the system.
- Complete Regression Testing, which is carried out after major overhauls and multiple modifications to existing code.

- ## Usability Testing

Usability Testing deals with the way end-users interact with a given software system. Typically it involves observation of subjects by researchers to understand the user experience in the real world. It plays a key role in User-Centric Interaction Design or User Experience (UX) Design, where it is used for various purposes:

- Discovering usability problems
- Benchmarking performance
- Usage pattern mapping
- Ease of use

Summary
Software of all sizes and types trust us with all **types of software testing** especially – test automation. We love being your extended testing buddy and helping you launch quality products with confidence.
Don’t let production defects hinder your customer experience. Try testing with Zuci today!

Orignal Blog: https://www.zucisystems.com/blog/7-common-types-of-software-testing/