## SYSTEM TESTING

System tests are designed to validate a **fully developed system to assure that it meets its requirements.** The test cases are therefore designed solely based on the SRS document.

There are essentially three main kinds of system testing depending on who carries out testing:

1. **Alpha Testing:** Alpha testing **refers to the system testing carried out by the test team within the developing organisation.**

2. **Beta Testing**: Beta testing is the system **testing performed by a select group of friendly customers.**

3. **Acceptance Testing:** Acceptance testing is **the system testing performed by the customer to determine whether to accept the delivery of the system**.

### 1. Smoke Testing

For smoke testing, **a few test cases are designed to check whether the basic functionalities are working.** For example, for a library automation system, the smoke tests may check whether books can be created and deleted, whether member records can be created and deleted, and whether books can be loaned and returned.

### 2. Performance Testing

Performance testing is carried **out to check whether the system meets the nonfunctional requirements identified in the SRS document.**

### 3. Error Seeding

Error seeding, as the name implies, it involves seeding the code with some known errors.

## Regression testing

Regression testing is **the practice of running an old test suite after each change to the system or after each bug fix to ensure that no new bug has been introduced due to the change or the bug fix.**

## SOFTWARE RELIABILITY

The reliability of a software product can also be defined as **the probability of the product working "correctly" over a given period of time.** Reliability of a product depends **not only on the number of latent errors** but also on **the exact location of the errors**. Apart from this, reliability also depends upon how the product is used, or on its execution profile.

**The main reasons that make software reliability more difficult to measure than hardware reliability:**

**The reliability improvement due to fixing a single bug depends on where the bug is located in the code.**

**The perceived reliability of a software product is observer-dependent.**

**The reliability of a product keeps changing as errors are detected and fixed**.

Hardware components fail due to very different reasons as compared to software components. **Hardware components fail mostly due to wear and tear, whereas software components fail due to bugs.**

## SOFTWARE QUALITY

A good quality product does exactly **what the users want it to do, since for almost every product**. To give an example of why this is so, **consider a software product that is functionally correct. That is, it correctly performs all the functions that have been specified in its SRS document. Even though it may be functionally correct, we cannot consider it to be a quality product, if it has an almost unusable user interface.**

**Another example** is that of a **product which does everything that the users wanted but has an almost incomprehensible and un maintainable code**. Therefore, the traditional concept of quality as "fitness of purpose" for software products is not wholly satisfactory.

The modern view of a quality associates with a software product **several quality factors** (or attributes) such as the following:

**Portability:** A software product is said to be portable, if it can be easily made to work in different **hardware and operating system environments, and easily interface with external hardware devices and software products.**

**Usability:** A software product has good usability, if different categories of users (i.e., both expert and novice users) **can easily invoke the functions of the product.**

**Reusability:** A software product has good reusability, if different modules of the product can easily **be reused to develop new products**.

**Correctness:** A software product is correct, **if different requirements as specified in the SRS document have been correctly implemented.**

**Maintainability:** A software product is maintainable, **if errors can be easily corrected as and when they show up, new functions can be easily added to the product, and the functionalities of the product can be easily modified, etc.**