Quality of a Software

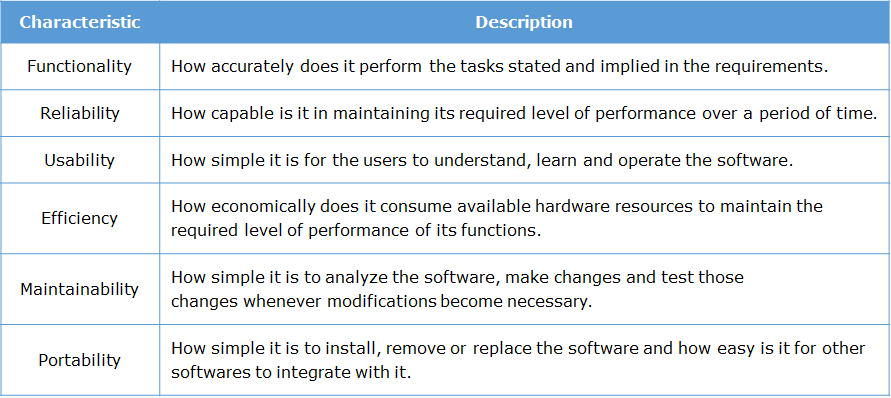
In all the scenarios, mentioned earlier, the software malfunctioned in various ways. Therefore, Software Quality had to be checked.

**What is Software Quality?**

Software Quality is the point to which a system, component or process conforms to the specified requirements as per user expectations, contractual agreements, and regulatory conditions.

**Characteristics of software quality**

As per ISO standard model (ISO/IEC 9126),  quality of a software can be evaluated based on the following characteristics.



What is Software Testing?

To improve and determine Software Quality, tests are conducted based on the:

* software
* processes that are used to build it

**Software testing** is a set of processes and tasks that takes place throughout software development life cycle. It helps to reduce the risk of failures that may occur during operational use and, thus, ensure the software system quality.

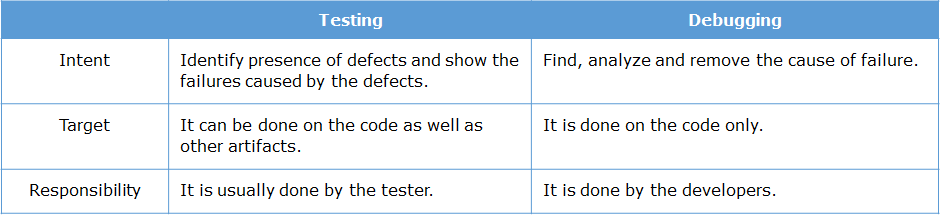
# Objectives of software testing

### Following are the objectives of the software testing:

* Preventing defects from entering the system
* Finding defects existing in the system
* Measuring the system quality

What is not Software Testing?

Debugging is a term which is confused with testing. Let us understand the difference between debugging and testing.



Errors, Defects and Failures

Some of the terms like errors, defects and failures are used interchangeably in day-to-day conversations. However, these terms vary in terms of testing.

Before we proceed any further, let's understand the difference between these terminologies.

# ****Error Vs Mistake****

Error, which also means mistake, is something that is  missed, overlooked or implemented incorrectly during designing or developing the software.

**Situation**: In the "Olympic hammer throw" case there must have been some logical error committed by the designer or the developer which was missed by the testing team that led to the particular situation to happen in real time environment.

# ****Defect Vs Fault, or Bug****

Defect is a flaw in the system, which arises due to an error committed during design or development activity.

**Situation**: In the "Olympic hammer throw" case, because of the error, the system was not able to accept the successive throws when it was operational.

# ****Failure****

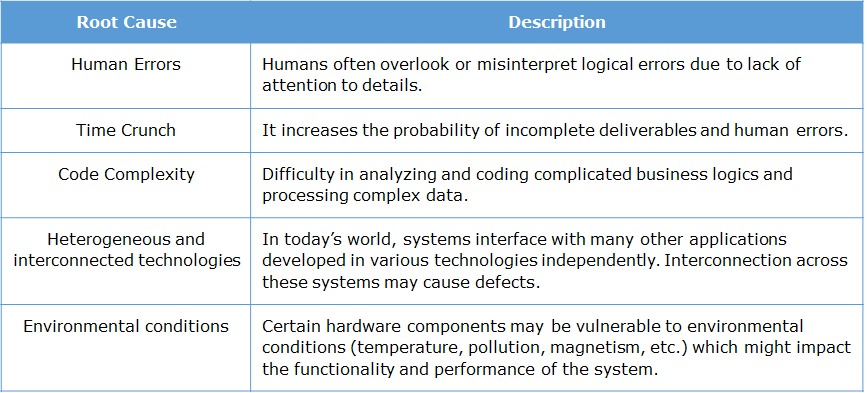
The manifestation of any existing defect in the system, when it is executed in real time, may lead to a failure. Failures can be avoided by detecting and correcting defects and preventing errors.

**Situation**: In the "Olympic hammer throw" case, the defect led to the mismatch in the result, which leads to the failure of the system.

**Note:** Not all defects lead to failure. Some stay dormant in the code and may never get noticed at all.

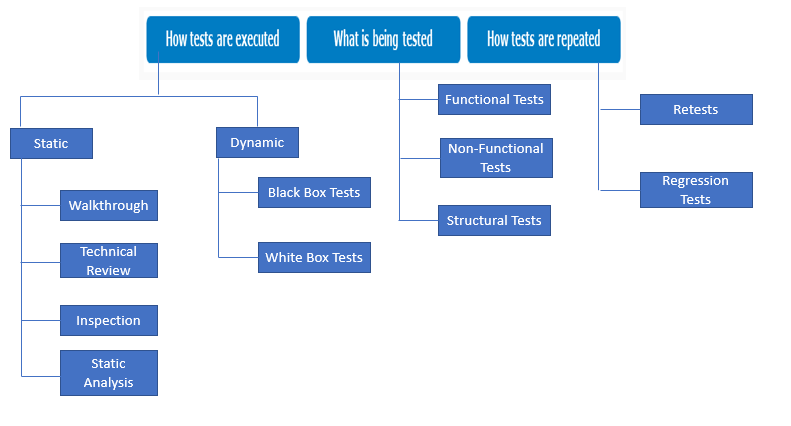
# ****Defect root causes****

To avoid new defects and detect existing defects, it is necessary to understand how they enter into the system. Below table is with some of the root causes.

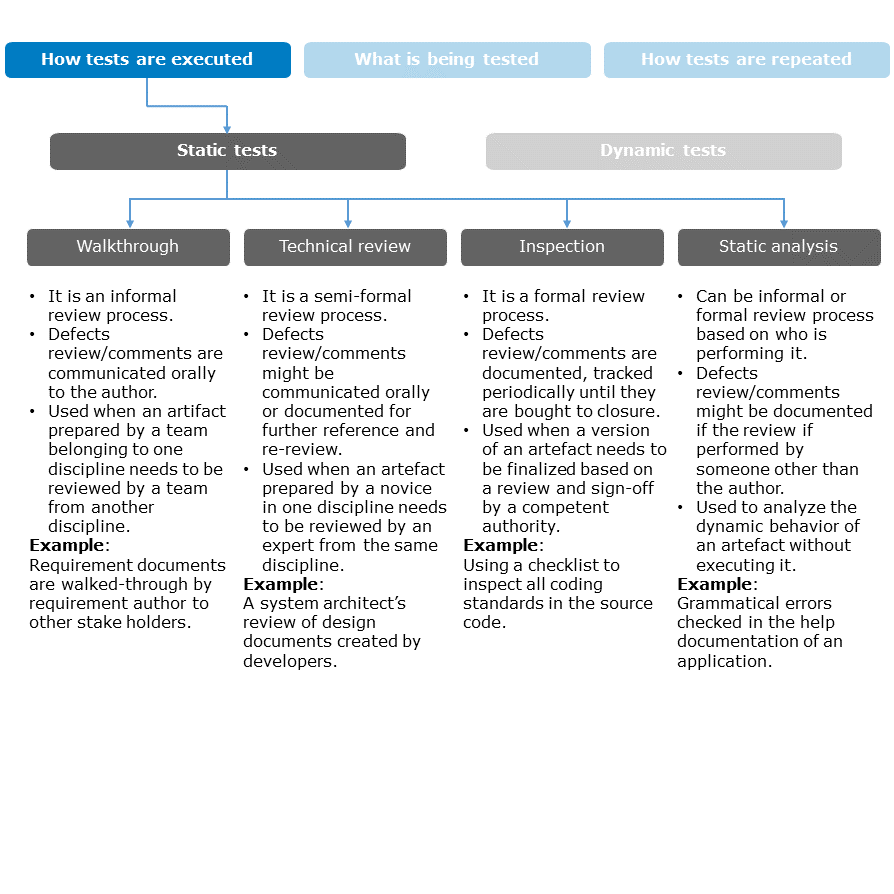


How to categorize tests

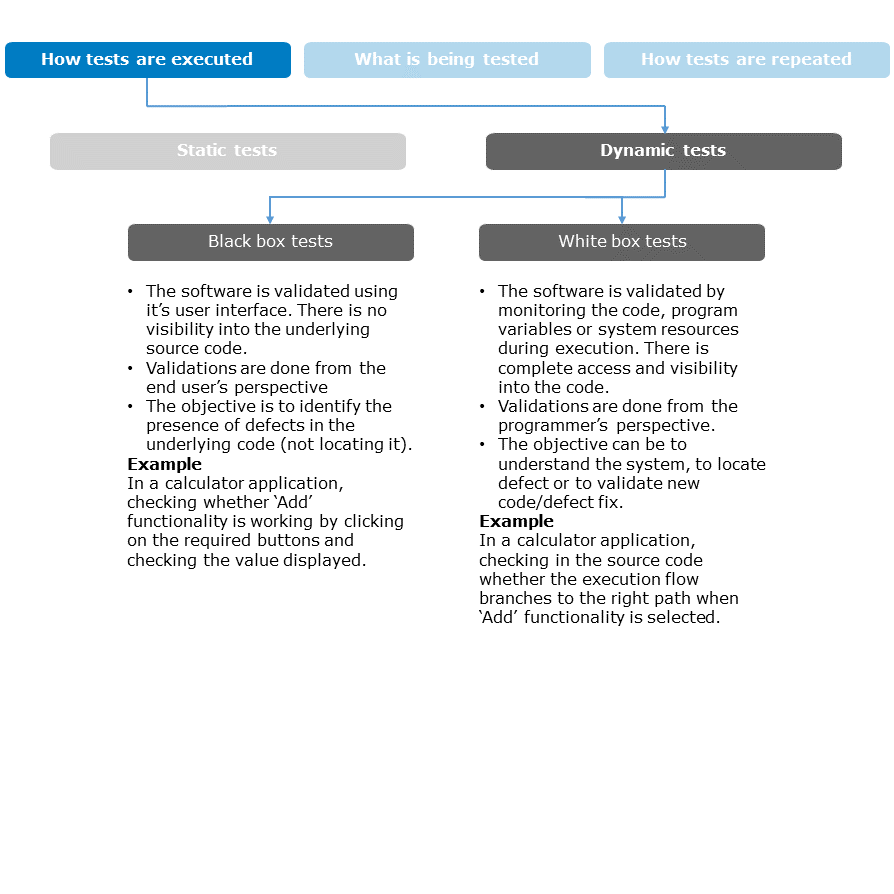
There are different types of tests such as:



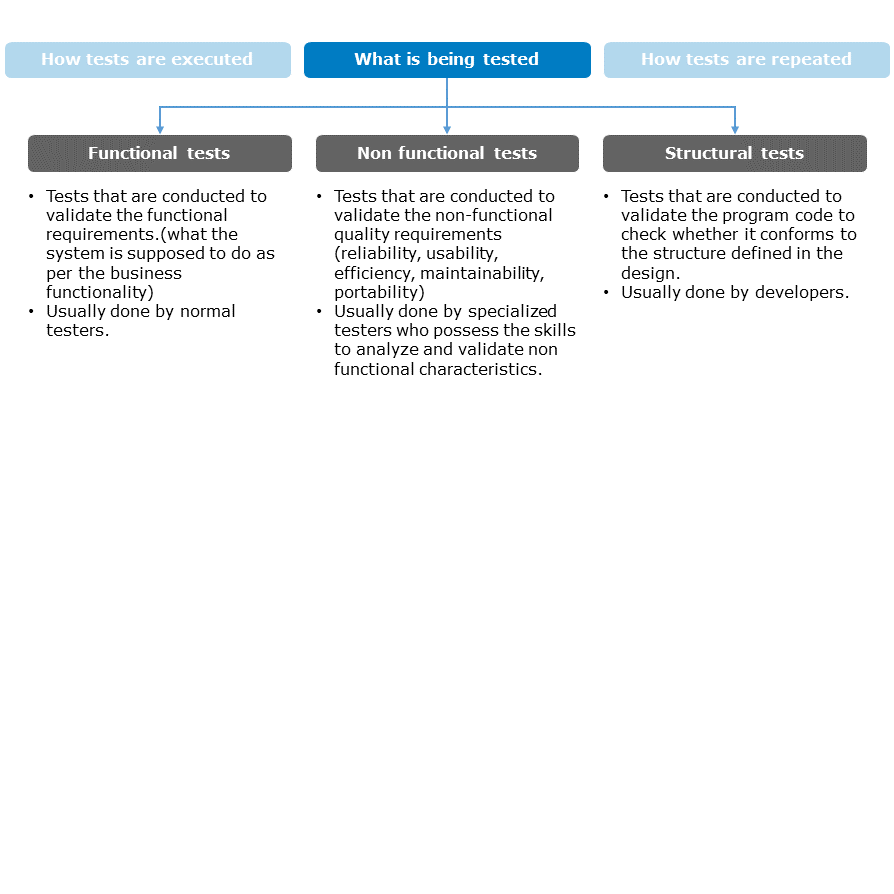
Categorization Based on How Tests are Executed - Static Tests



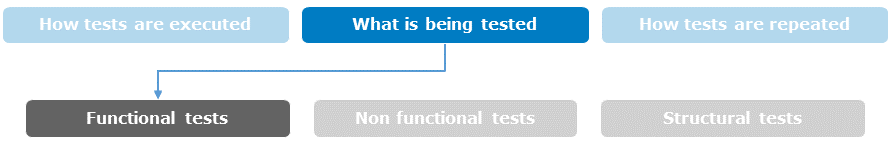
Categorization Based on How Tests are Executed - Dynamic Tests



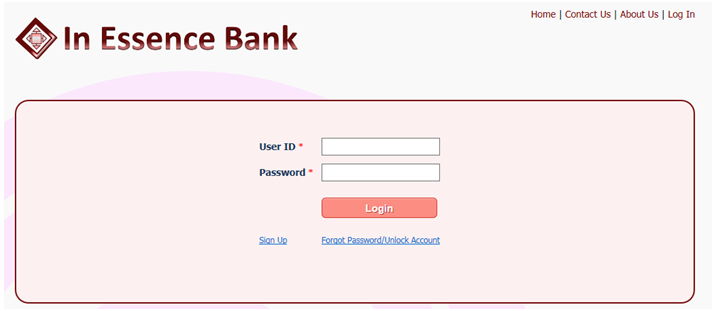
Categorization Based on What is Being Tested



Categorization Based on What is Being Tested - Functional Tests



Consider the login page of a banking application -



# ****Smoke Tests****

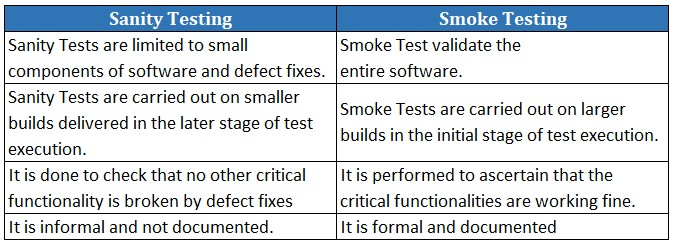
* Smoke tests are the first tests that is performed during the test execution phase immediately after the first code builds are delivered to the testing team.
* Their objective is to ascertain the software's test readiness - i.e.is check whether the software is ready for the full-blown testing that is about to follow.
* Smoke tests are performed on basic and critical components of the software. They are selected based on how quickly the tests can be carried out. Some examples of smoke tests are to check whether:
  + the application is launching without any issues
  + all the GUI components are available
* Smoke tests can be informal (without documenting the tests and their results) or formal.

**Example** **-**

* To check whether the objects are appearing on the login page.
* To check if user can put values inside the "User Id" and "Password" objects.
* To check if all the links on the page are working.

# ****Sanity Tests****

* Sanity tests are like smoke tests in terms of their objectives (validate test readiness) and their test selection (how quickly they can be executed).
* They differ from smoke tests in terms of their scope and time of execution.



**Example -**

* If a defect for a broken "Sign up" link has been fixed, before conducting a full blown testing of 'Sign up' functionality, a tester usually checks if that link and the near by ones (forgot password) are all working.

# 

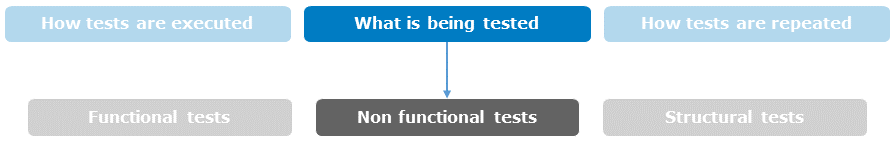
# ****System Tests****

* These are the actual tests that have been designed, from the functional requirements, to validate the software.
* These tests form the bulk of the functional tests during the testing phase.
* They are executed once the smoke/sanity tests confirms test readiness.
* They are mandated to be formal (tests and their results must be documented).

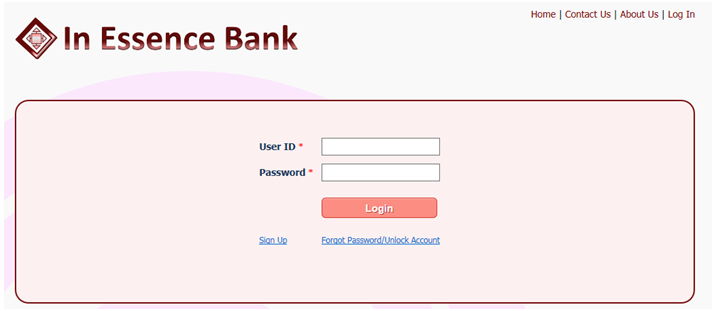
**Example -**

* To check if user can login on entering valid "user id" and "password" details.
* To check if user is getting appropriate error message as per the requirement on entering invalid "user id" and "password" details.

Categorization Based on What is Being Tested - Non-Functional tests



Consider the login page of a banking application -



Some of the most prominent types of Non-Functional tests are:

**Performance Tests**

These tests:

* the response times of each application component are validated to check if they are under acceptable limits, in different possible situations.
* are conducted in an environment (hardware configuration) resembling the real-time environment as close as possible.
* there is specialized skill sets required to work with performance testing tools, configuring environments, etc.

**Example -**

* Checking that on entering correct login credentials the user can login within time<=3 sec.
* Checking whether 1000 users simultaneously logging in from their terminals with their own valid credentials can login within time<=3 sec.

**Security Tests**

These tests:

* are conducted to check whether the application allows access to only authorized users and withstands unauthorized attacks.
* requires specialized skill sets in terms of hacking methods and knowledge of authentication methodologies, encryption standards, network protocols, etc.

**Example -**

* Checking whether the user is taken back to their account page on clicking the back button of the browser after logging out.

**Accessibility Tests**

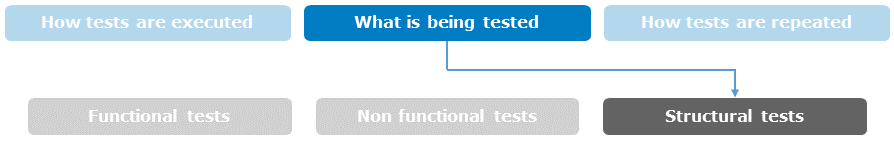
These tests:

* are to establish whether individuals with disabilities will be able to operate the system in question.
* requires specialized skill sets in terms of knowledge of international accessibility standards for specific disability types and human behavior sciences.

**Example -**

* Checking that the login page of the application supports disabled users (color blindness, motor deficiency, etc.), in accordance with the W3C Accessibility Standards.

Categorization Based on What is Being Tested - Structural Tests



Let us consider the following pseudo-code to understand structural testing:

1. 1: Read Q
3. 2: If p+q > 100 then
5. 3: Print “Large”
7. 4: End if
9. 5: If p > 50 then
11. 6: Print “pLarge”
13. 7: End if

**Statement Coverage Tests**

These tests are aimed at exercising all programming statements with minimal tests.

**Example**: For the above piece of code, if P=51, Q=50 is considered, then statements 1,2,3,4,5,6,7 will execute. It means all the statements are executed and 100% Statement Coverage is attained with only one pair of values.

**Decision Coverage Tests**

These tests are aimed at ensuring that all decision outcomes, in the program, are tested at least once.

**Example**: With the statement coverage tests only "TRUE" behaviour of the Decision Statements gets executed. If P=50, Q=50 is considered, statements 1, 2,4,5,7 will be executed. It only "FALSE" part of the program that is executed so 100% Decision Coverage is covered with two test cases.

**Path Coverage Tests**

These tests are aimed at checking all possible program execution flow paths through the program.

**Example**: Consider all the possible combinatorial outcomes of the two IF statements - "True-True", "True-False", "False-True", "False-False".

Depending on the project requirement and strategy the developer might choose the best technique out of these.

Categorization Based on How Tests are Repeated

