# TESTING

## TESTING

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, Testing presents an interesting anomaly for the software engineer.

**Testing Objectives**

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a probability of finding an as yet
- UN discovered error.
- A successful test is one that uncovers an undiscovered error.
- These above objectives imply a dramatic change in view port.
- Testing cannot show the absence of defects, it can only show that software errors are present.

**Test Case Design**

Any engineering product can be tested in one of two ways:

## 4.1. Testing Methodologies

**White Box Testing**

This testing is also called as glass box testing. In this testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

**Basis Path Testing**

- Flow graph notation
- Cyclomatic Complexity
- Deriving test cases Control Structure Testing
- Condition testing
- Data flow testing

- Loop testing

**Black -Box Testing**

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing
- Graph matrices

**Software Testing Strategies**

A Strategy for software testing integrates software test cases into a series of well planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers he set of activities that ensure that the software that has been built is traceable to customer’s requirements.

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

**Integration Testing**

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

**Top-Down Integration**

Top down integrations is an incremental approach for construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control program. Modules subordinate to the main program are incorporated in the structure either in the breath-first or depth-first manner.

**Bottom-up Integration**

This method as the name suggests, begins construction and testing with atomic modules i.e., modules at the lowest level. Because the modules are integrated in the bottom up manner the processing required for the modules subordinate to a given level is always available and the need for stubs is eliminated.

**Regression Testing**

In this contest of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagate unintended side effects.

**Validation Testing**

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in the manner reasonably expected by the customer.Reasonable expectations are those defined in the software requirements specifications. Information contained in those sections form a basis for validation testing approach.

Reasonable expectation is defined in the software requirement specification a document that describes all user-visible attributes of the software. The specification contains a section titled “Validation Criteria”. Information contained in that section forms the basis for a validation testing approach.

**Validation Test Criteria**

Software validation is achieved through a series of black-box tests that demonstrate conformity with requirement. A test plan outlines the classes of tests to be conducted, and a test procedure defines specific test cases that will be used in an attempt to uncover errors in conformity with requirements. Both the plan and procedure are designed to ensure that all functional requirements are satisfied, all performance requirements are achieved, documentation is correct and human-engineered; and other requirements are met.

After each validation test case has been conducted, one of two possible conditions exists: (1) The function or performance characteristics conform to specification and are accepted, or (2) a deviation from specification is uncovered and a deficiency list is created. Deviation or error discovered at this stage in a project can rarely be corrected prior to scheduled completion. It is often necessary to negotiate with the customer to establish a method for resolving deficiencies.

**Configuration Review**

An important element of the validation process is a configuration review. The intent of the review is to ensure that all elements of the software configuration have been properly developed, are catalogued, and have the necessary detail to support the maintenance phase of the software life cycle. The configuration review sometimes called an audit.

**Alpha and Beta Testing**

It is virtually impossible for a software developer to foresee how the customer will really use a program. Instructions for use may be misinterpreted. Strange combination of data may be regularly used; and output that seemed clear to the tester may be unintelligible to a user in the field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements. Conducted by the end user rather than the system developer, an acceptance test can range from an informal "test drive" to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time.

The beta test is conducted at one or more customer sites by the end user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a “live” application of the software in an environment that cannot be controlled by the developer. The customer records all problems that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, the software developer makes modification and then prepares for release of the software product to the entire customer base.

**System Testing**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system.  Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocated functions.

**Security Testing**

Attempts to verify the protection mechanisms built into the system.

**Performance Testing**

This method is designed to test runtime performance of software within the context of an integrated system.

## 4.2. Test Case

Test cases are vital components in software development, providing systematic procedures to verify that the software behaves as expected under various conditions. They typically consist of input data, expected outcomes, and execution steps. Efficient test cases cover a wide range of scenarios, including normal operation, boundary conditions, error handling, and performance benchmarks. Each test case should be well-documented, specifying the purpose, preconditions, and postconditions. Additionally, it's crucial to prioritize test cases based on their criticality and impact on the system. Regular updates and maintenance of test cases ensure the software's reliability and robustness throughout its lifecycle.

| **S.No.** | **Test Cases** | **Input** | **Expected Result** | **Actual Result** | **Status** |
|---|---|---|---|---|---|
| **1** | User Registration | Enterall fields | Usergets registered | Registrationis successful | pass |
| **2** | User Registration | If user miss any field | User not registered | Registration isun successful | fail |
| **3** | Admin Login | Give the user name and password | Admin home page should be opened | Admin home Page has been opened | Pass |

**Table 4.2.1:** Test cases