What is Testing?

* TC reporting/Soaking
* TC Naming conversation
* SP
* TC movement
* Test driven development

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

**What is Test case?**

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

**Typical Test Case Parameters:**

* Test Case ID
* Test Scenario
* Test Case Description
* Test Steps
* Prerequisite
* Test Data
* Expected Result
* Test Parameters
* Actual Result
* Environment Information
* Comments

## Applications of Software Testing

* **Cost Effective Development** - Early testing saves both time and cost in many aspects, however reducing the cost without testing may result in improper design of a software application rendering the product useless.
* **Product Improvement** - During the SDLC phases, testing is never a time-consuming process. However diagnosing and fixing the errors identified during proper testing is a time-consuming but productive activity.
* **Test Automation** - Test Automation reduces the testing time, but it is not possible to start test automation at any time during software development. Test automaton should be started when the software has been manually tested and is stable to some extent. Moreover, test automation can never be used if requirements keep changing.
* **Quality Check** - Software testing helps in determining following set of properties of any software such as
  + Functionality
  + Reliability
  + Usability
  + Efficiency
  + Maintainability
  + Portability

**What is Testing?**

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

**Who does Testing?**

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called **Unit Testing**. In most cases, the following professionals are involved in testing a system within their respective capacities −

* Software Tester
* Software Developer
* Project Lead/Manager
* End User

Different companies have different designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, QA Analyst, etc.

It is not possible to test the software at any time during its cycle. The next two sections state when testing should be started and when to end it during the SDLC.

**When to Start Testing?**

An early start to testing reduces the cost and time to rework and produce error-free software that is delivered to the client. However, in Software Development Life Cycle (SDLC), testing can be started from the Requirements Gathering phase and continued till the deployment of the software.

It also depends on the development model that is being used. For example, in the Waterfall model, formal testing is conducted in the testing phase; but in the incremental model, testing is performed at the end of every increment/iteration and the whole application is tested at the end.

Testing is done in different forms at every phase of SDLC −

* During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.
* Reviewing the design in the design phase with the intent to improve the design is also considered as testing.
* Testing performed by a developer on completion of the code is also categorized as testing.

**When to Stop Testing?**

It is difficult to determine when to stop testing, as testing is a never-ending process and no one can claim that a software is 100% tested. The following aspects are to be considered for stopping the testing process −

* Testing Deadlines
* Completion of test case execution
* Completion of functional and code coverage to a certain point
* Bug rate falls below a certain level and no high-priority bugs are identified
* Management decision

**Verification & Validation**

These two terms are very confusing for most people, who use them interchangeably. The following table highlights the differences between verification and validation.

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Verification** | **Validation** |
| 1 | Verification addresses the concern: "Are you building it right?" | Validation addresses the concern: "Are you building the right thing?" |
| 2 | Ensures that the software system meets all the functionality. | Ensures that the functionalities meet the intended behavior. |
| 3 | Verification takes place first and includes the checking for documentation, code, etc. | Validation occurs after verification and mainly involves the checking of the overall product. |
| 4 | Done by developers. | Done by testers. |
| 5 | It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software. | It has dynamic activities, as it includes executing the software against the requirements. |
| 6 | It is an objective process and no subjective decision should be needed to verify a software. | It is a subjective process and involves subjective decisions on how well a software works. |

**Software Testing Tools**

The following tools can be used for automation testing −

* HP Quick Test Professional
* Selenium
* IBM Rational Functional Tester
* SilkTest
* TestComplete
* Testing Anywhere
* WinRunner
* LoadRunner
* Visual Studio Test Professional
* WATIR

## Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

The following table lists the advantages and disadvantages of black-box testing.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Well suited and efficient for large code segments. | Limited coverage, since only a selected number of test scenarios is actually performed. |
| Code access is not required. | Inefficient testing, due to the fact that the tester only has limited knowledge about an application. |
| Clearly separates user's perspective from the developer's perspective through visibly defined roles. | Blind coverage, since the tester cannot target specific code segments or errorprone areas. |
| Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems. | The test cases are difficult to design. |

## White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

The following table lists the advantages and disadvantages of white-box testing.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively. | Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased. |
| It helps in optimizing the code. | Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested. |
| Extra lines of code can be removed which can bring in hidden defects. | It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools. |
| Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing. |  |

## Grey-Box Testing

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Offers combined benefits of black-box and white-box testing wherever possible. | Since the access to source code is not available, the ability to go over the code and test coverage is limited. |
| Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications. | The tests can be redundant if the software designer has already run a test case. |
| Based on the limited information available, a grey-box tester can design excellent test scenarios especially around communication protocols and data type handling. | Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested. |
| The test is done from the point of view of the user and not the designer. |  |

## A Comparison of Testing Methods

The following table lists the points that differentiate black-box testing, grey-box testing, and white-box testing.

|  |  |  |
| --- | --- | --- |
| **Black-Box Testing** | **Grey-Box Testing** | **White-Box Testing** |
| The internal workings of an application need not be known. | The tester has limited knowledge of the internal workings of the application. | Tester has full knowledge of the internal workings of the application. |
| Also known as closed-box testing, data-driven testing, or functional testing. | Also known as translucent testing, as the tester has limited knowledge of the insides of the application. | Also known as clear-box testing, structural testing, or code-based testing. |
| Performed by end-users and also by testers and developers. | Performed by end-users and also by testers and developers. | Normally done by testers and developers. |
| Testing is based on external expectations - Internal behavior of the application is unknown. | Testing is done on the basis of high-level database diagrams and data flow diagrams. | Internal workings are fully known and the tester can design test data accordingly. |
| It is exhaustive and the least time-consuming. | Partly time-consuming and exhaustive. | The most exhaustive and time-consuming type of testing. |
| Not suited for algorithm testing. | Not suited for algorithm testing. | Suited for algorithm testing. |
| This can only be done by trial-and-error method. | Data domains and internal boundaries can be tested, if known. | Data domains and internal boundaries can be better tested. |

## Key Terms:

## Functional testing

## Non-functional testing

## Performance testing

## Capacity planning testing

## User acceptance testing