Software Testing

Software testing is the process of evaluation a software item to detect differences between given input and expected output. It also assesses the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Basics of software testing

There are two basics of software testing: blackbox testing and whitebox testing.

Blackbox Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

Whitebox Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

Black box testing is often used for validation and white box testing is often used for verification.

Types of testing

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

Unit Testing

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input.

Integration Testing

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

Functional Testing

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

System Testing

System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.

Stress Testing

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

Performance Testing

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

Usability Testing

Usability testing is performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

Regression Testing

Regression testing is the testing after modification of a system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

Beta Testing

Beta testing is the testing which is done by end users, a team outside development, or publicly releasing full pre-version of the product which is known as beta version. The aim of beta testing is to cover unexpected errors. It falls under the class of black box testing.

CASE tools:

Computer-Aided Software Engineering (CASE) technologies are tools that provide automated assistance for software development **.** The goal of introducing CASE tools is the reduction of the time and cost of software development and the enhancement of the quality of the systems developed. The interest in CASE tools and environments is based on expectations about increasing productivity, improving product quality, facilitating maintenance, and making software engineers' task less odious and more enjoyable.

Most classifications of CASE tools start by considering whether the tool is upper CASE, lower CASE, or integrated CASE . An upper CASE tool (front end CASE) provides support for the early stages in the systems development life cycle such as requirements analysis and design. A lower CASE tool (back end CASE) provides support for the later stages in the life cycle such as code generation and testing. Integrated CASE tools support both the early and later stages.

CASE Tools offer an excellent array of features that support the development and business community through its Automated Diagram Support feature. The various popular features that aid the development community are listed below:

- Checks for syntactic correctness
- Data dictionary support
- Checks for consistency and completeness
- Navigation to linked diagrams
- Layering
- Requirements traceability
- Automatic report generation
- System simulation
- Performance analysis

CASE Tools and its scope

CASE technology is the automation of step by step methodologies for software and system development. CASE tools are characterized by the stage or stages of software development life cycle on which they focus. Since different tools covering different stages share common information, it is required that they integrate through some central repository system (data dictionary) to have a consistent view of such information. In phases of software development life cycle integrated through a central data dictionary. Case Tools are used in many ways in our organizations. Case tools can be broadly classed into these broader areas:

- Requirement Analysis Tool
- Structure Analysis Tool
- Software Design Tool
- Code Generation Tool
- Test Case Generation Tool
- Document Production Tool
- Reverse Engineering Tool

Making a case for and against and CASE Tools

For	Against
Helps standardization of notations and diagrams	Limitations in the flexibility of documentation
Help communication between development team members	May lead to restriction to the tool's capabilities
Automatically check the quality of the models	Major danger: completeness and syntactic correctness does NOT mean compliance with requirements
Reduction of time and effort	Costs associated with the use of the tool: purchase + training
Enhance reuse of models or models' components	Staff resistance to CASE tools

Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable **modules**, such that each contains everything necessary to execute only one aspect of the desired functionality.