What is Software Testing

"Testing is the process of executing a program or system with the intent of finding errors."by Myers 1979

"Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results." by Hetzel 1983

- A software process based on well-defined software quality control and testing standards, testing methods, strategy, test criteria, and tools.

How to define Software Testing Principles

Testing

The execution of a program to find its faults

Verification

The process of proving the programs correctness.

Validation

The process of finding errors by executing the program in a real environment

Debugging

Diagnosing the error and correct it

Software Testing Principles

- To remove as many defects as possible before test since the quality improvement potential of testing is limited

Testing Process Goals

- Validation testing
  - To demonstrate to the developer and the system customer that the software meets its requirements;
  - A successful test shows that the system operates as intended.
  - Verification – Defect testing
  - To discover faults or defects in the software where its behavior is incorrect or not in conformance with its specification;
  - A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.
  - **Fundamentals of Software Testing**

  - Software testing needs planning, tests need specifying, once executed they need results recording, and post completion should be easily audit
  - **VERIFICATION & VALIDATION**

- **Verification** - typically involves reviews and meeting to evaluate documents, plans, code, requirements, and specifications. This can be done with checklists, issues lists, walkthroughs, and inspection meeting.

Objectives	To test the function of a program or unit of code such as a program or module To test internal logic To verify internal design To test path & conditions coverage To test exception conditions & error handling
When	After modules are coded
Input	Internal Application Design Master Test Plan Unit Test Plan
Output	Unit Test Report
Who	Developer
Methods	White Box testing techniques Test Coverage techniques
Tools	Debug Re-structure Code Analyzers Path/statement coverage tools
Education	Testing Methodology Effective use of tools

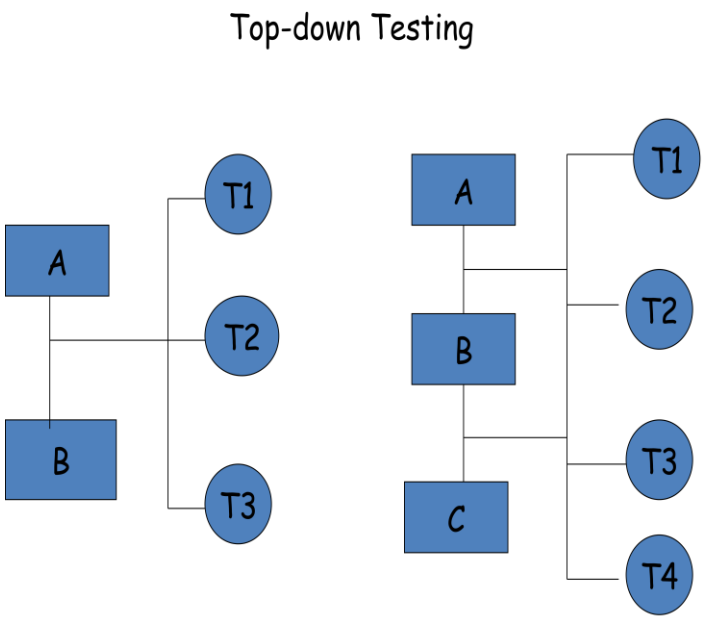
Validation - typically involves actual testing and takes place after verifications are completed.Validation and Verification process continue in a cycle till the software becomes defects free.

1.Unit Testing

- Individual components are tested.
- It is a path test.
- To focus on a relatively small segment of code and aim to exercise a high percentage of the internal path
- Disadvantage: the tester may be biased by previous experience. And the test value may not cover all possible values.

2.Integration Testing

- Test assembled components
  - These must be tested and accepted previously
- Focus on interfaces
  - Might be interface problem although components work when tested in isolation
  - Might be possible to perform new tests
- Strategies
  - Bottom-up, start from bottom and add one at a time
  - Top-down, start from top and add one at a time
  - Big-bang, everything at once
  - Functional, order based on execution
  - **Top-down Integration Test**
  - The control program is tested first. Modules are integrated one at a time. Emphasize on interface testing
  - Advantages: No test drivers needed
  - Interface errors are discovered early
  - Modular features aid debugging
  - Disadvantages: Test stubs are needed
  - Errors in critical modules at low levels are found late.



– **Bottom-up Integration Test**

Allow early testing aimed at proving feasibility

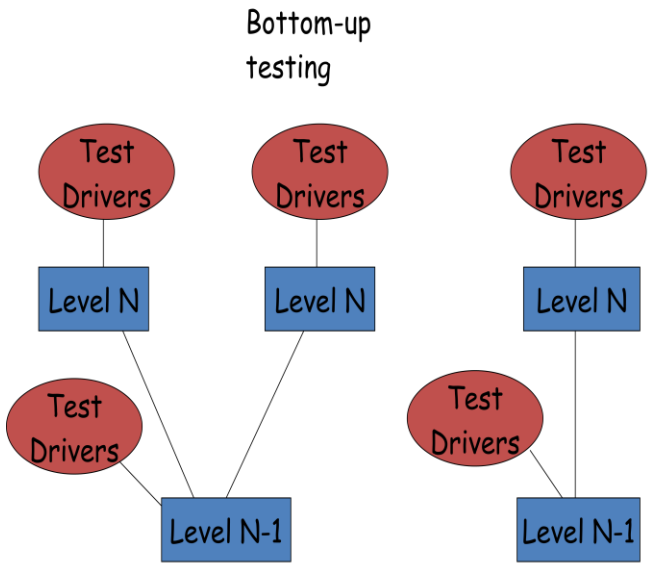
Emphasize on module functionality and performance

– Advantages: No test stubs are needed

Errors in critical modules are found early

Disadvantages: Test drivers are needed

Interface errors are discovered late



- Simulation of other components

- Stubs receive output from test objects
- Drivers generate input to test objects
- Note that these are also SW, i.e., need testing etc.

Objectives	To technically verify proper interfacing between modules, and within sub-system
When	After modules are unit tested
Input	Internal & External Application Design Master Test Plan Integration Test Plan
Output	Integration Test report
Who	Developers
Methods	White and Black Box techniques Problem / Configuration Management
Tools	Debug Re-structure Code Analyzers
Education	Testing Methodology Effective use of tools

- **3.Function Testing**
- Designed to exercise the to its external specifications
- Testers not biased by knowledge of the program's design.
- Disadvantages:
- The need for explicitly stated requirements
- Only cover a small portion of the possible test conditions.

4.System Testing

- Testing of groups of components integrated to create a system or sub-system;
- The responsibility of an independent testing team;
- Tests are based on a system specification.

- Functional testing
    - Test end to end functionality
    - Requirement focus
        - Test cases derived from specification
    - Use-case focus
        - Test selection based on user profile
- Non-functional testing
- Quality attributes
    - Performance, can the system handle required throughput?
    - Reliability, obtain confidence that system is reliable
    - Timeliness, testing whether the individual tasks meet their specified deadlines

Objectives	• To verify that the system components perform control functions • To perform inter-system test • To demonstrate that the system performs both functionally and operati • To perform appropriate types of tests relating to Transaction Flow, Insta Regression etc.
When	• After Integration Testing
Input	• Detailed Requirements & External Application Design • Master Test Plan • System Test Plan
Output	• System Test Report
Who	• Development Team and Users
Methods	• Problem / Configuration Management
Tools	• Recommended set of tools

5.Acceptance Testing

- User (or customer) involved
- Environment as close to field use as possible
- Focus on:
  - Building confidence
  - Compliance with defined acceptance criteria in the contract

– **Objectives**	– To verify that the system meets the user requirements
– **When**	– After System Testing
– **Input**	– Business Needs & Detailed Requirements – Master Test Plan – User Acceptance Test Plan
– **Output**	– User Acceptance Test report
Who	Users / End Users
Methods	• Black Box techniques • Problem / Configuration Management
Tools	Compare, keystroke capture & playback, regression testing

6.Re-Test and Regression Testing

- Conducted after a change
- Re-test aims to verify whether a fault is removed

- Re-run the test that revealed the fault

- Regression test aims to verify whether new faults are introduced
  - How can we test modified or newly inserted programs?
    - Ignore old test suites and make new ones from the scratch or
    - Reuse old test suites and reduce the number of new test suites as many as possible

Should preferably be automated

- Test the effects of the newly introduced changes on all the previously integrated code.
- The common strategy is to accumulate a comprehensive regression bucket but also to define a subset.
- The full bucket is run only occasionally, but the subset is run against every spin.
- Disadvantages:

To decide how much of a subset to use and which tests to select

7. Alpha and Beta Testing

- ☐ It's best to provide customers with an outline of the things that you would like them to focus on and specific test scenarios for them to execute.
- ☐ Provide with customers who are actively involved with a commitment to fix defects that they discover.

8. **Black box testing**

- ☐ No knowledge of internal design or code required.
- ☐ Tests are based on requirements and functionality

Black Box - testing technique

- ☐ Incorrect or missing functions
- ☐ Interface errors
- ☐ Errors in data structures or external database access
- ☐ Performance errors
- ☐ Initialization and termination errors
- ☐ Based on requirements and functionality
- ☐ Not based on any knowledge of internal

- ☐ design or code
- ☐ Covers all combined parts of a system
- ☐ Tests are data driven

9.White box testing

Knowledge of the internal program design and code required.

Tests are based on coverage of code statements, branches, paths, conditions Based on knowledge of internal logic of an application's codeTests are logic driven

10.Load testing

Testing an application under heavy loads. Eg. Testing of a web site under a range of loads to determine, when the system response time degraded or fails.

11.Alpha testing

Testing done when development is nearing completion; minor design changes may still be made as a result of such testing.

12.Beta-testing

Testing when development and testing are essentially completed and final bugs and problems need to be found before release

13.Mutation testing

To determining if a set of test data or test cases is useful, by deliberately introducing various bugs.Re-testing with the original test data/cases to determine if the bugs are detected.

14.Incremental Testing

A disciplined method of testing the interfaces between unit-tested programs as well as between system components. Involves adding unit-testing program module or component one by one, and testing each result and combination.

15.Stress testing

Evaluates a system or component at or beyond the limits of its specified requirements. Determines the load under which it fails and how.

16.Performance Test

Evaluate the compliance of a system or component with specified performance requirements. Often performed using an automated test tool to simulate large number of users.