UNIT 4
SOFTWARE ANALYSIS AND TESTING

1. There are two types of approaches for identifying defects in the software are -
   - **Static analysis**: Code is not executed but is evaluated through some heuristics or some tools for locating defects.
   - **Dynamic analysis**: Code is executed, and the execution is used for determining defects.

2. **Code inspection**
   It is often applied at the unit level. It can be viewed as "static testing" in which defects are detected in the code, not by executing the code but through a manual process. Three phases are:
   Planning → Self Review → Group review meeting

3. **Software Testing fundamentals**
   - **Error**: Difference between actual output and the correct output.
   - **Fault**: Condition that causes a system to fail.
   - **Failure**: Inability of a system to perform required function.
   - **Test case**: Set of test inputs and execution conditions.
   - **Test suite**: Related test cases are executed to test some specific behaviors.
   - **Test Harness**: Testing framework.

4. **Software Test Process**
   
   ![Test Process Diagram]

   **Test Activities**:
   - (1) Test planning
   - (2) Test case design
   - (3) Test execution
   - (4) Data collection
   - (5) Effective evaluation
5. Testing Levels:
   - Client Demands → Acceptance Testing
   - Requirements Gathering → System Testing
   - Design → Integration Testing
   - Coding → Unit Testing

6. Test Criteria - Reliability and Validity:
   - A test criterion is reliable if all the test cases detect some error.
   - A test criterion is valid if for any error in the program there is some test which causes error in the program.

7. Test Case Design:
   - Create a set of tests that are effective in testing. Two guidelines:
     1. Maximize the number of errors detected.
     2. Minimize the number of test cases.

8. Test Oracle:
   - It is a mechanism which is used to check the correctness of the output of the program for the given test case.

9. Test Techniques:
   - Test Types:
     1. Black Box Testing - demonstrates how functions are operational.
     2. White Box Testing - Procedural details are closely examined.
Black Box Testing—(Behavioural Testing)

Focus on functional requirements of the software.

- Incorrect or missing functions
  
- Interface errors
  
- Errors in test data
  
- Performance errors
  
- Initialization of termination errors

Black box technique one:

1. Equivalence Partitioning
   
   Divides the input domain into classes of data and from this data test cases can be derived.
   
   If input condition specific:
   
   (i) range: one valid and two invalid equivalence classes are defined
   
   (ii) specific value: one valid and two invalid equivalence classes are defined
   
   (iii) set: one valid and one invalid equivalence classes are defined
   
   (iv) Boolean: one valid and one invalid equivalence classes are defined

2. Boundary Value Analysis (BVA)

   Born by checking boundary conditions.
   
   If input condition specific:
   
   (i) range: test cases for value x and y and value above x and below y are
   
   (ii) values: test cases of min. and max. values and also just above and just below them
   
   (iii) output condition specific:
   
   (i) range: same as above
   
   (ii) values: same as above

White Box Testing—(Glass box testing)

It is based on close examination of procedural details.

Test cases are devised for:

- Examining all independent paths
- Exercising all logical paths
- Executing all loops
- Exercising internal data structures
White Box Techniques are:
(1) Structural Testing -
   Test cases are according to program structure that is exercising all program statements.
(2) Conditional Testing -
   To test the logical conditions in the program module.
(3) Loop Testing -
   To test the loop constructs for any particular:
   - Simple loops: n = m means m times through the loop (n > m).
   - Nested loops: Start with innermost loops.
   - Concatenated loops: Same as simple loop but...
   - Unstructured loops: Cannot design effectively, needs to redesign.
(4) Path Testing -
   This is a structural testing strategy. Steps are:
   1. Design the flow graph for the program or a component.
   2. Calculate the cyclomatic complexity.
   3. Select a basis set of paths.
   4. Generate test cases for those paths.

Cyclomatic complexity = Total no. of regions in the flow graph
                     = E - N + 2 (Edges E, Nodes N)
                     = P + 1 (Predicate decision making).

Unit Testing -
Individual components are tested independently to ensure their quality.

Module

Interface
- Areal Data Structures
- Boundary Conditions
- Independent Paths
- Error Handling Paths

Test Cases
13. **Unit Testing Frameworks**

- Third-party products that are not distributed as part of the compiled unit 
  e.g. - JUnit, TestNG, Isolation.NET, Isolator++, etc.

14. **Integration Testing**

- Group of dependent components or tested together. Unknown errors
  - (1) Design and construction of software architecture
  - (2) Integrated function or operations at subsystem level
  - (3) Interfaces and interaction between them
  - (4) Resource integration and/or environmental integration
  
  Two approaches are -
  
  (1) **Non-incremental Integration**
  - Big Bang (Approach is simple)
  
  (2) **Incremental Integration**
  
  (a) Top down integration testing
  
  (b) Bottom up integration testing
  
  (3) Regression Testing - Used to check if defects propagated to other modules by changes made to existing program
  
  (4) Smoke Testing - Used for time critical projects wherein the project needs to accessed on frequent basis

15. **Validation Testing**

- Testing is based on requirements

  (1) Interface input/output
  
  (2) System functional and information data
  
  (3) System interfaces with external inputs
  
  (4) User interface
  
  (5) System behavior on Performance

**Acceptance Testing**

- To test whether the software works correctly in the work environment

  Types are:
  
  - Alpha Test - developer or custome under supervision of developer
  
  - Beta Test - Customers without the developer being present
1. **System Testing**
   - Main focus is on system functions, performance, reliability, recoverability, installation, interface, usage, functions, etc.
   - Various types are -
     1. Recovery Testing
     2. Security Testing
     3. Stress Testing
     4. Performance Testing

2. **Test Plan**
   - Getting generated during the development. Four types of test plan -
     1. Testing Process
     2. Requirements Traceability
     3. Tested Items
     4. Testing Schedule
     5. Test Recording Procedure
     6. Hardware and Software Requirements
     7. Constraints

3. **Testing Tools**
   - Used to automate the testing activity. Two types -
     1. **Static Testing tools** -
        - Static analyzers → analyze the code systematically
        - Code inspectors → code analysis can be incorporated in program
     2. **Dynamic Testing tools** -
        - Activities are input validation, fault finding, displaying read
        and test planning.
        - Various tools are output comparator, coverage analyzers and test data generators.

4. **Testing Matrices**
   - Cyclomatic complexity and Halstead's metrics
   - Halstead's matrix - Estimating the testing efforts (e)
     \[ e = \frac{V}{PL} \]
     \[ PL = \frac{1}{\sum_{i=1}^{n_1} \left( \frac{n_1}{2} \right) \times (N_2/n_2)} \]
   - \( V \rightarrow \) program volume, \( PL \rightarrow \) program level
   - \( n_1 = \) Total distinct operators, \( N_2 = \) all operands
   - \( n_2 = \) Total distinct operands in program
Percentage of overall testing effort = \frac{Testing effort of specific module}{Testing effort of all the modules}