OXFORD
HIGHER EDUCATION

Software Testing, 2nd edition

NARESH CHAUHAN

OXFORD
HIGHER EDUCATION

Chapter 2
Software Testing Terminology and Methodology

Objectives

- Difference between error, fault and failure
- Life cycle of a bug
- How bug affects economics of software testing
- How bugs are classified
- Testing principles
- Software testing life cycle (STLC) and its models
- Difference between verification and validation
- Development of software testing methodology

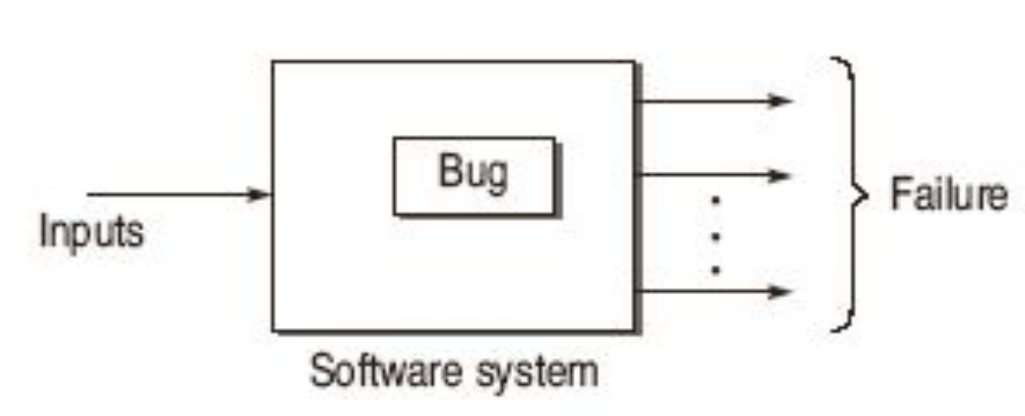
Software Testing Terminology

- **Failure**

  The inability of a system or a component to perform a required function according to its specification.

- **Fault / Defect / Bug**

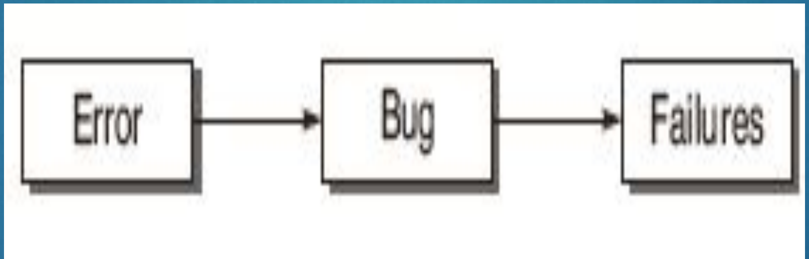
  Fault is a condition that causes a system to produce failure. It can be said that failures are manifestation of bugs.



Software Testing Terminology

- **Error**

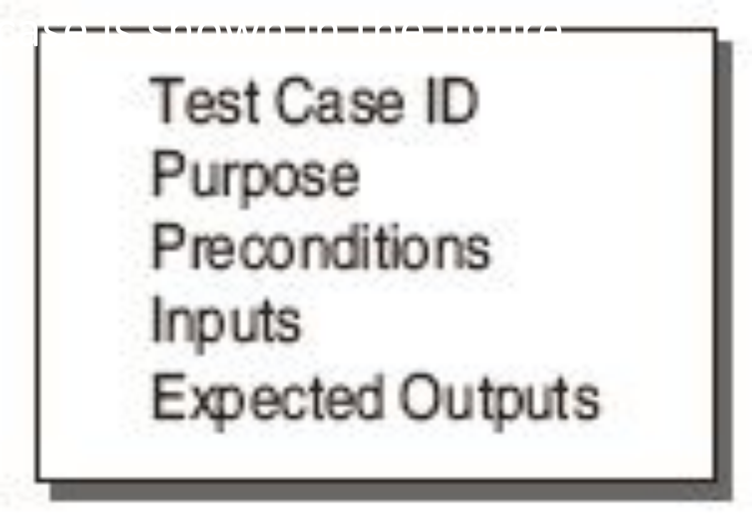
  Whenever a member of development team makes any mistake in any phase of SDLC, errors are produced. It might be a typographical error, a misleading specification, a misunderstanding of what a subroutine does, and so on. Thus, error is a very general term used for human mistakes.



Software Testing Terminology

- **Test Case**

It is a well-documented procedure designed to test the functionality of a feature in the system. A test case has an identity and is associated with a program behaviour. The primary purpose of designing a test case is to find errors in the system. A test case needs to specify a set of inputs and the corresponding expected outputs. The sample for a test case is shown in the figure.



Test Case ID
Purpose
Preconditions
Inputs
Expected Outputs

Software Testing Terminology

- **Testware**

  The documents created during the testing activities are known as Testware.

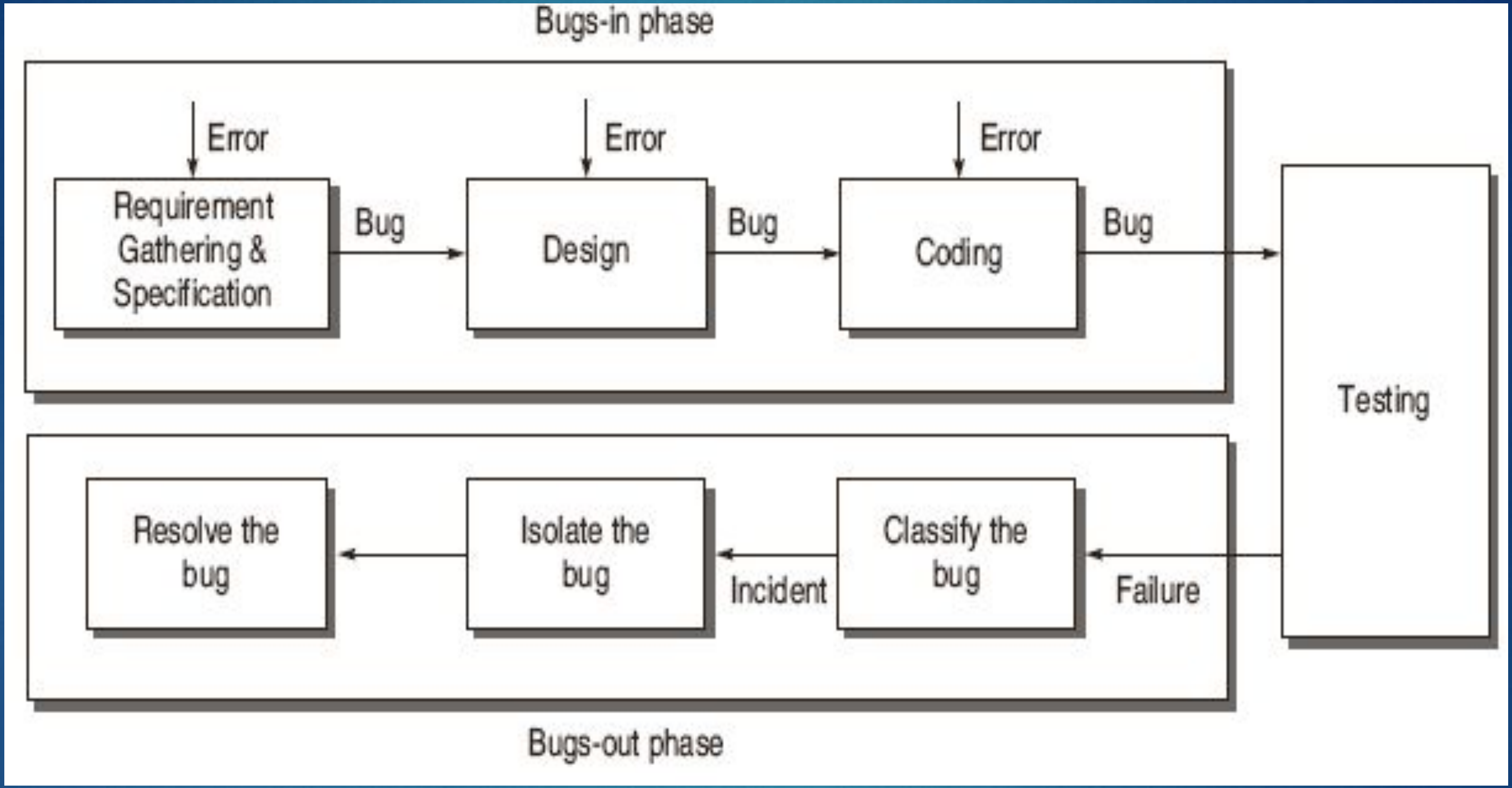
- **Incident**

  The symptom(s) associated with a failure that alerts the user to the occurrence of a failure.

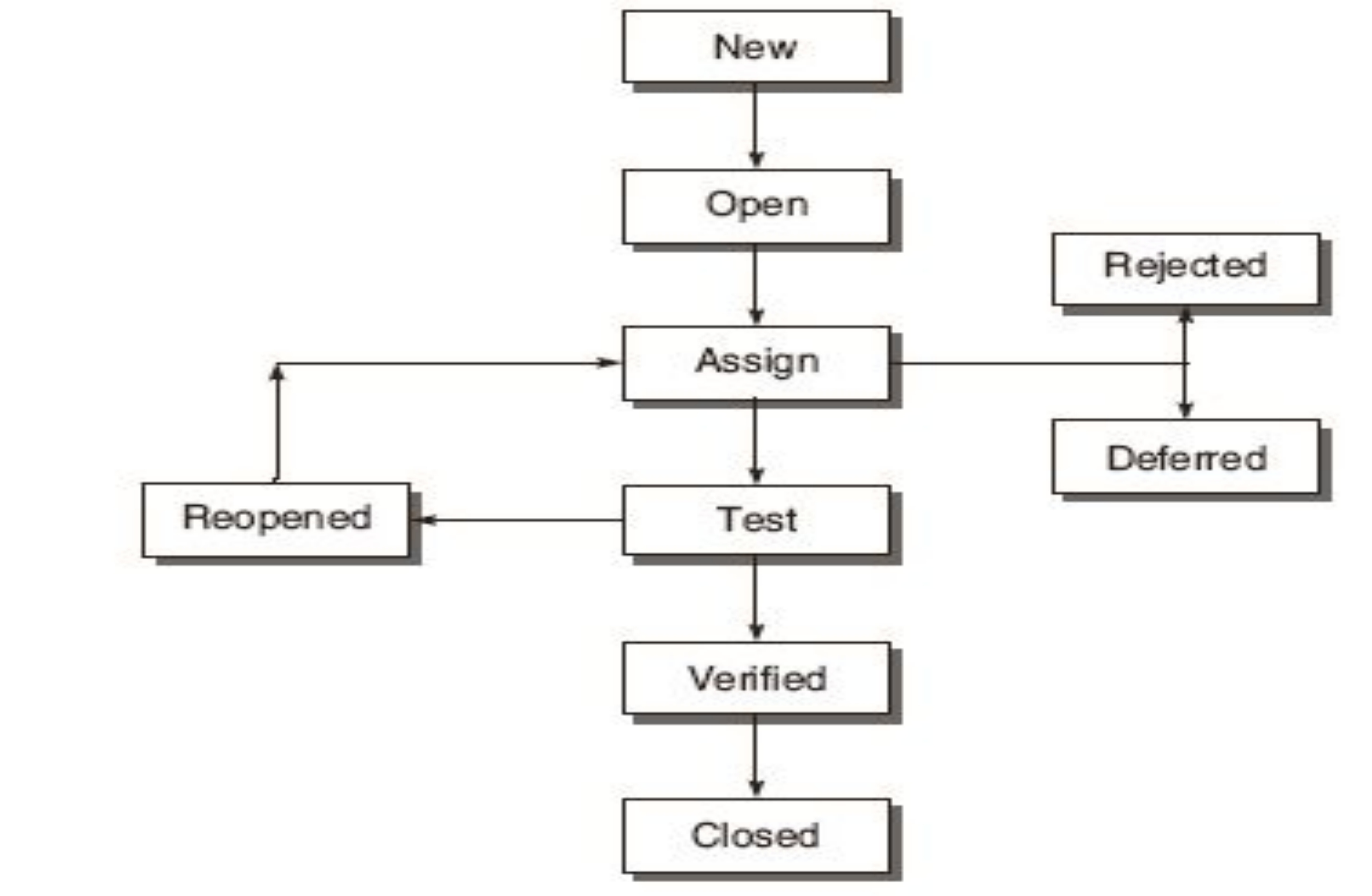
- **Test Oracle**

  It is the means to judge the success or failure of a test.

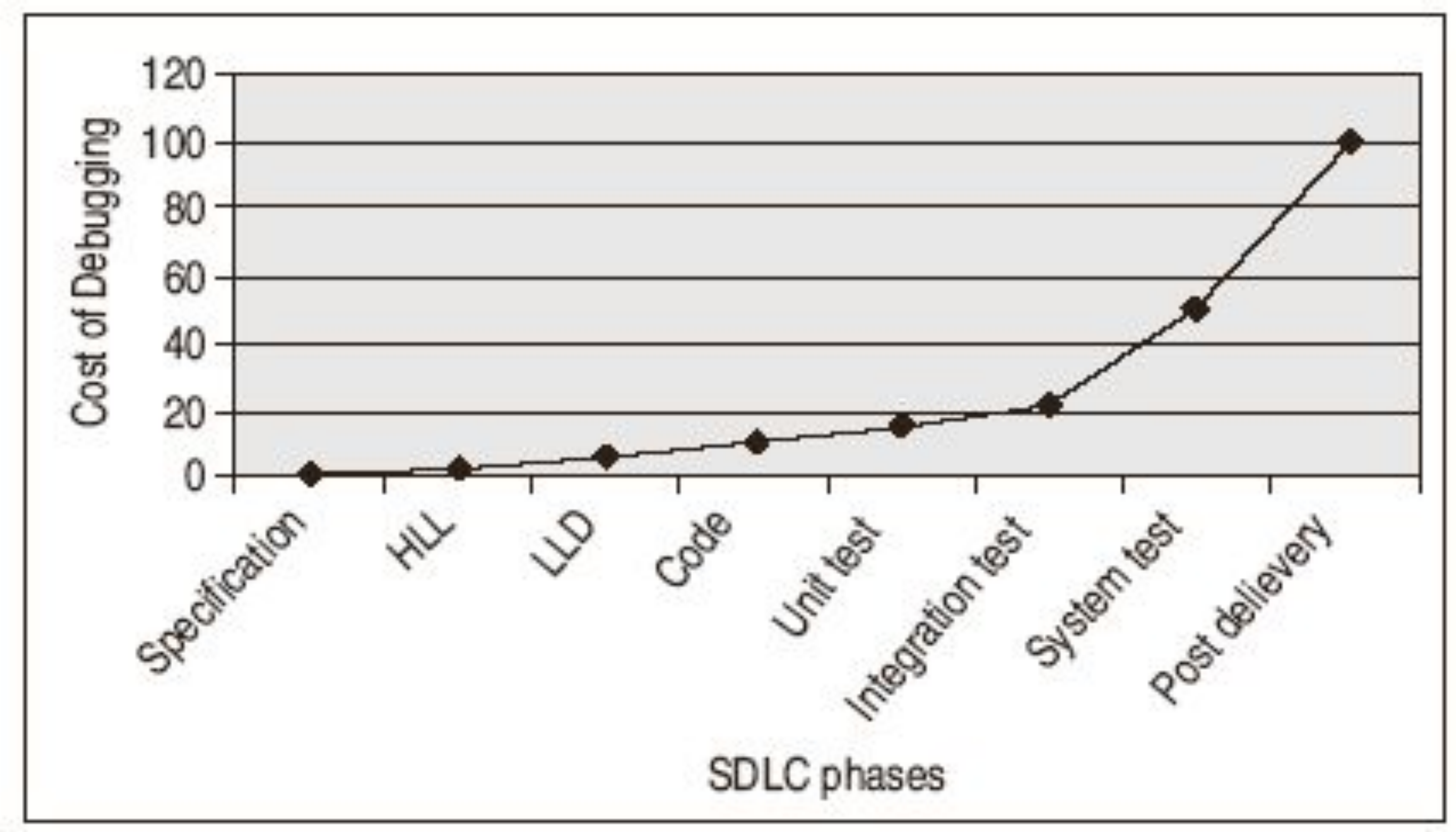
Life Cycle of a Bug



States of a Bug



Bugs Affect Economics of Software Testing



Bug Classification based on Criticality

- **Critical Bugs**

  This type of bug has the worst effect as it stops or hangs the normal functioning of the software.

- **Major Bug**

  This type of bug does not stop the functioning of the software but it causes a functionality to fail to meet its requirements as expected.

- **Medium Bugs**

  These are less critical in nature as compared to critical and major bugs.

- **Minor Bugs**

  These are just mild bugs, which occur without any effect on the expected behavior or continuity of the software.

Bug Classification based on SDLC

- Requirements and Specifications Bugs
- Design Bugs
  - Control Flow Bugs
  - Logic Bugs
  - Processing Bugs
  - Data Flow Bugs
  - Error Handling Bugs
  - Race Condition Bugs
  - Boundary Related Bugs
  - User Interface Bugs
- Coding Bugs
- Interface and Integration Bugs
- System Bugs
- Testing Bugs

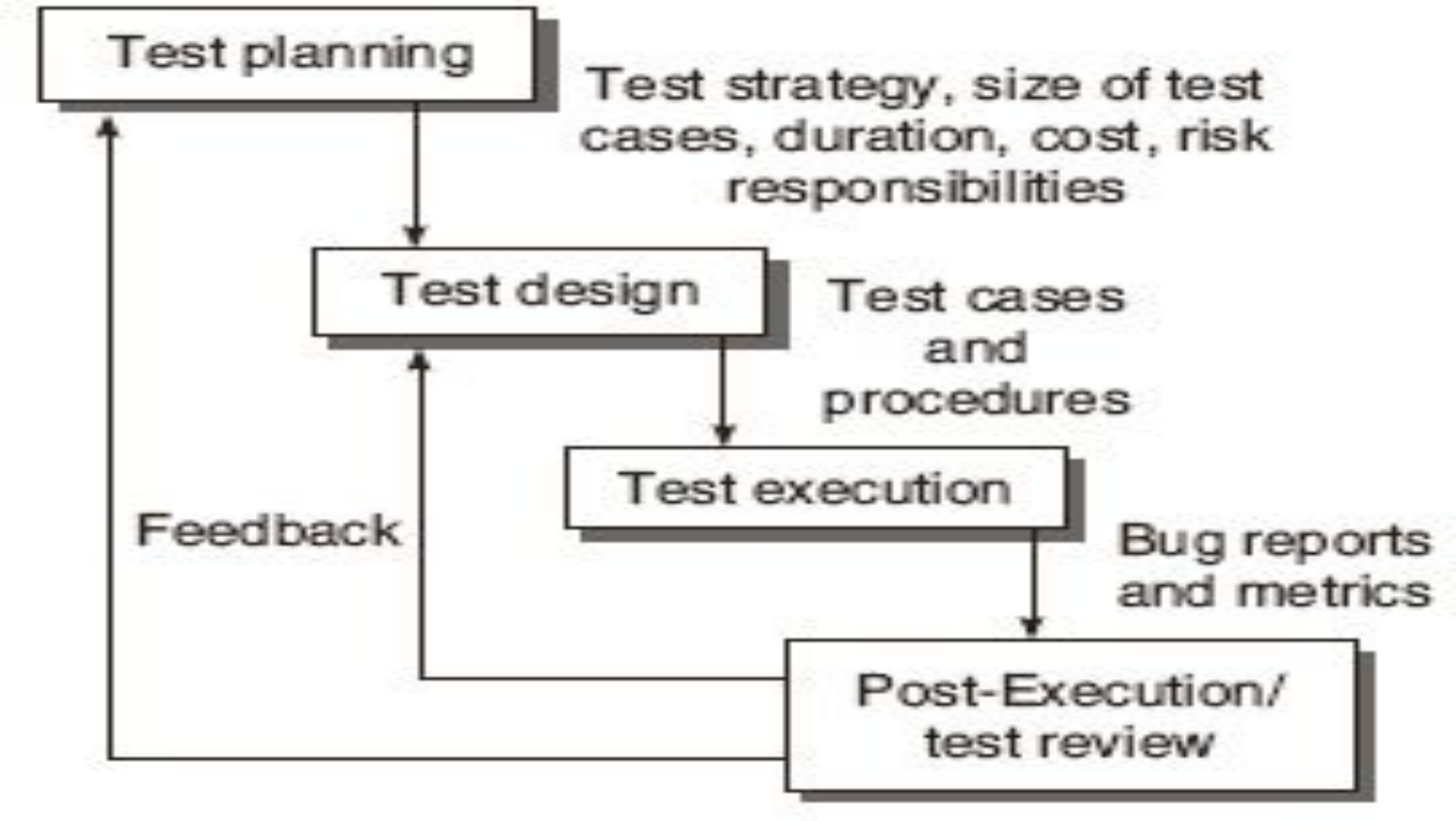
Testing Principles

- Effective testing, not exhaustive testing.
- Testing is not a single phase performed in SDLC.
- Destructive approach for constructive testing.
- Early testing is the best policy.
- Probability of existence of an error in a section of a program is proportional to the number of errors already found in that section.
- Testing strategy should start at the smallest module level and expand towards the whole program.

Testing Principles

- Testing should also be performed by an independent team. Everything must be recorded in software testing.
- Invalid inputs and unexpected behavior have a high probability of finding an error.
- Testers must participate in specification and design reviews.

Software Testing Life Cycle (STLC)

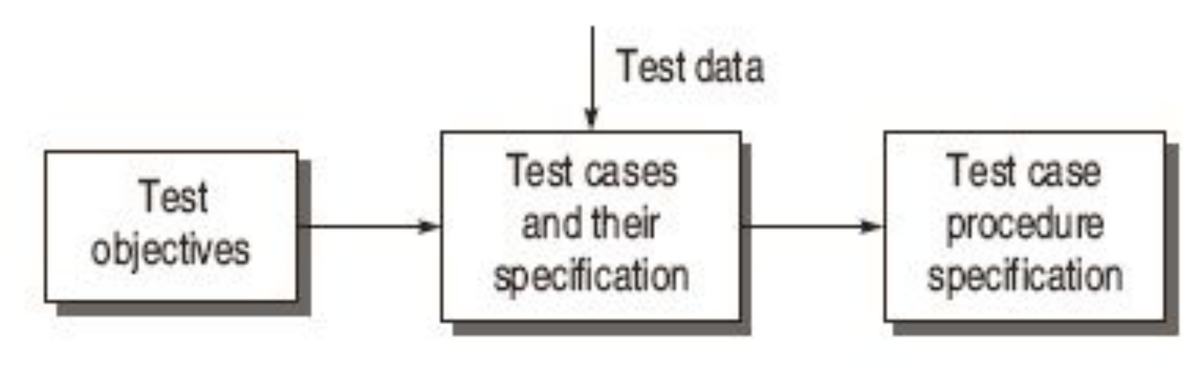

Test planning
Test strategy, size of test cases, duration, cost, risk responsibilities
Test design
Test cases and procedures
Test execution
Feedback
Bug reports and metrics
Post-Execution/ test review

Test Planning

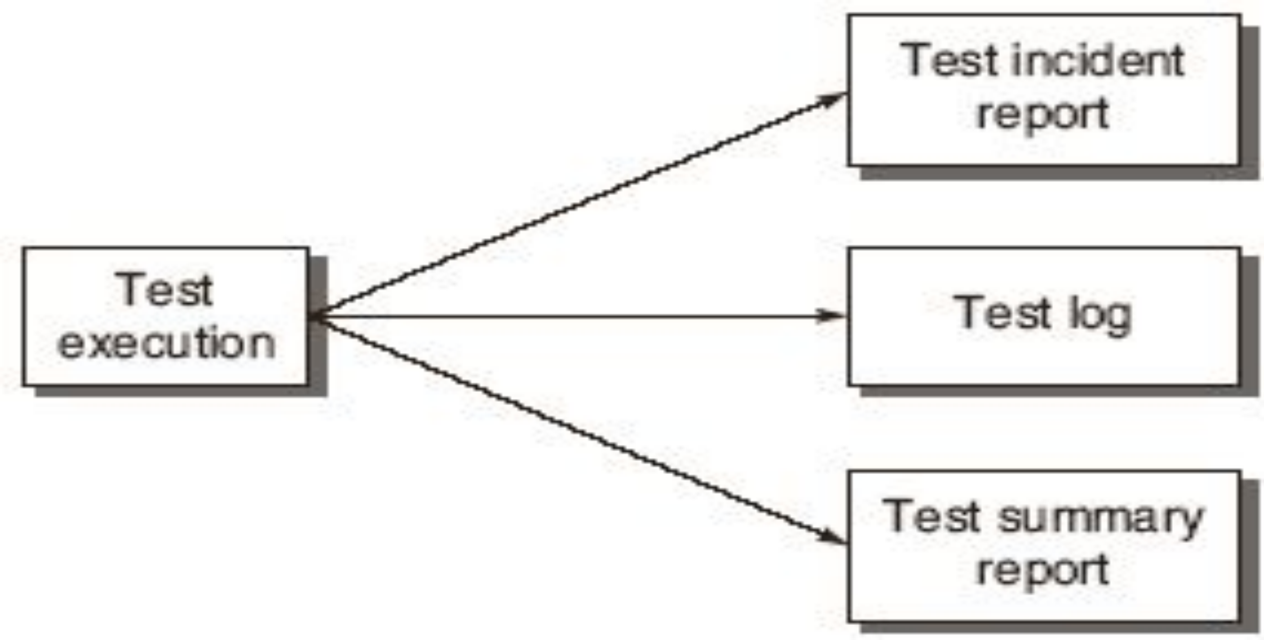
- Define the test strategy.
- Estimate the number of test cases, their duration, and cost.
- Plan the resources such as the manpower, tools required, documents required.
- Identify areas of risks.
- Define the test completion criteria.
- Identify methodologies, techniques and tools for various test cases.
- Identify reporting procedures, bug classification, databases for testing, bug severity levels, project metrics.

Test Design

- Determine the test objectives and their prioritization.
- Prepare the list of items to be tested.
- Map items to test cases.
- Select test case design techniques.
- Create test cases and test data.
- Set up the test environment and supporting tools.
- Create test procedure specification.


Test data
Test objectives
Test cases and their specification
Test case procedure specification

Test Execution



Test execution

Test incident report

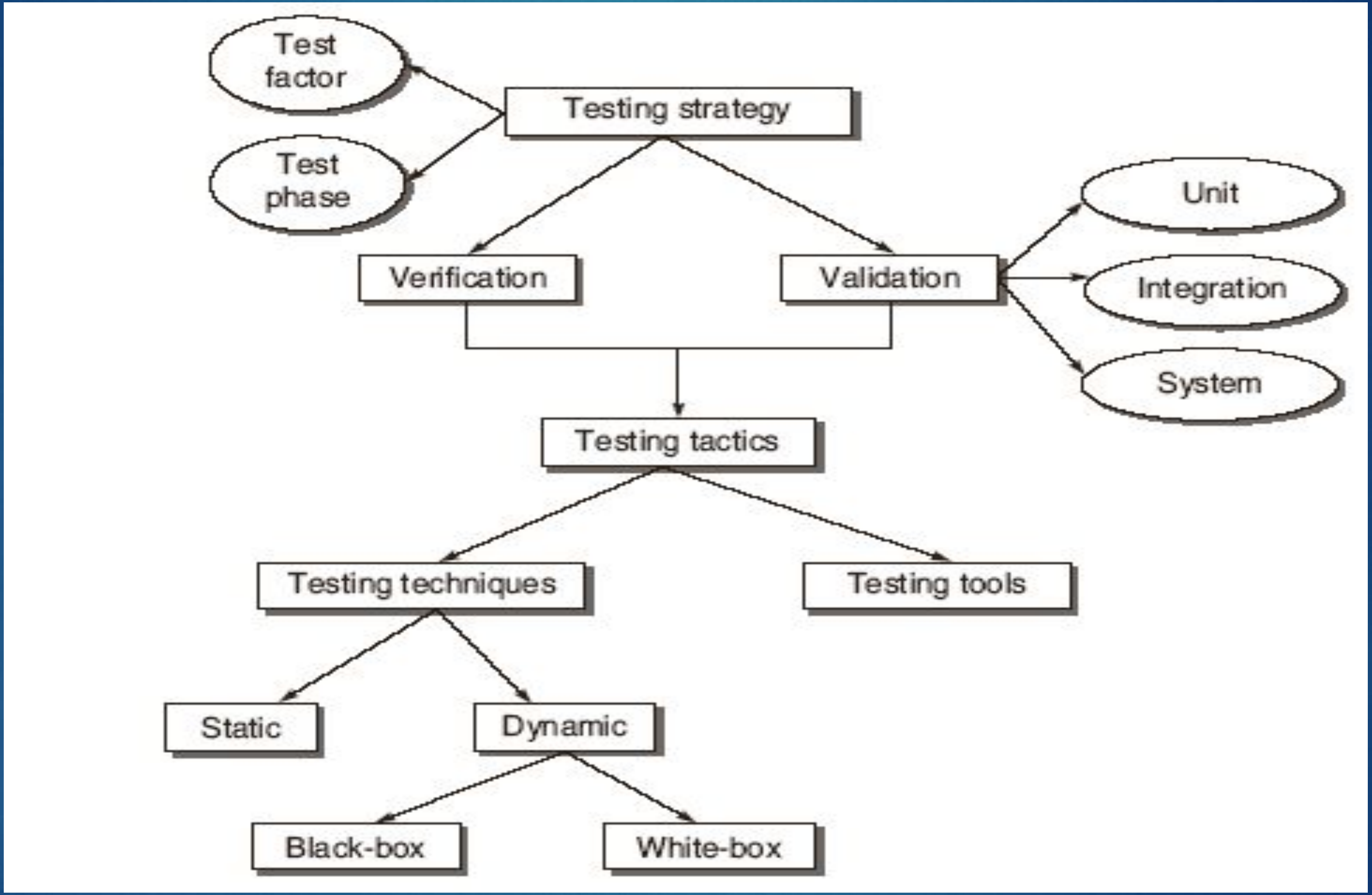
Test log

Test summary report

Post-Execution / Test Review

- Understanding the bug
- Reproducing the bug
- Analysing the nature and cause of the bug
- Reliability analysis
- Coverage analysis
- Overall defect analysis

Software Testing Methodology

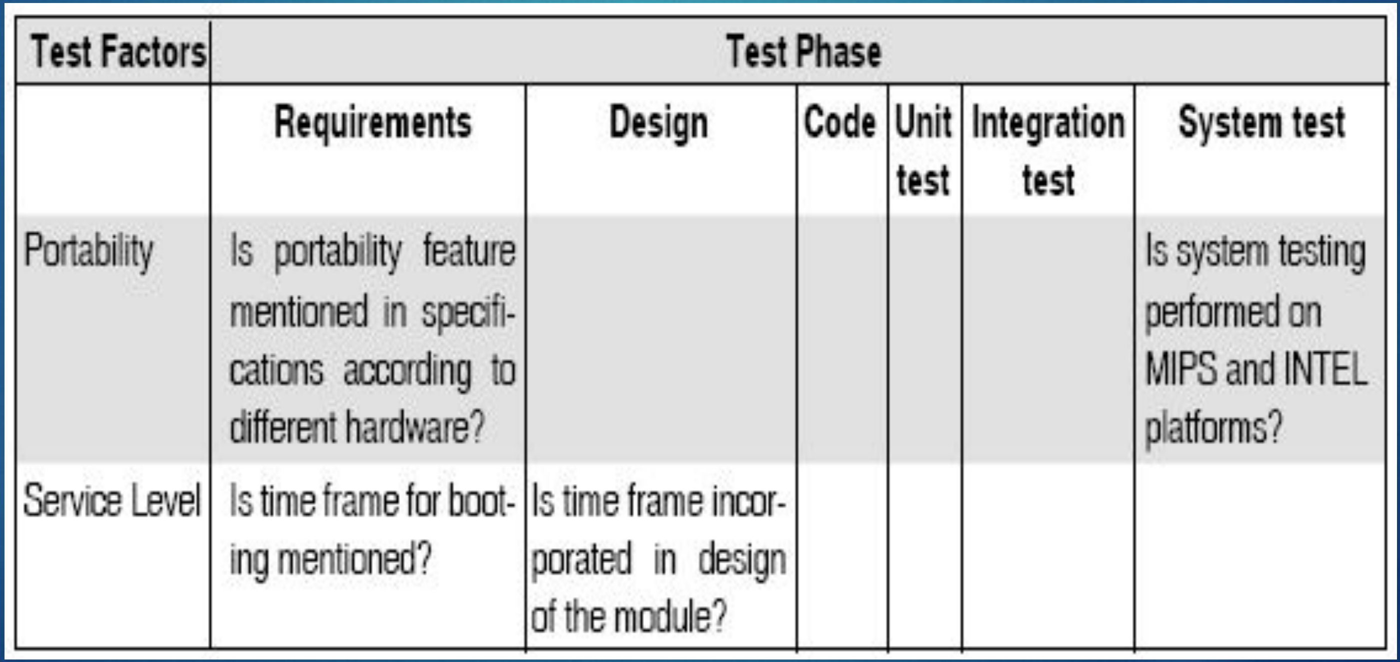


Test Strategy Matrix

- Select and rank test factors

- Identify the system development phases

- Identify the risks associated with system under development



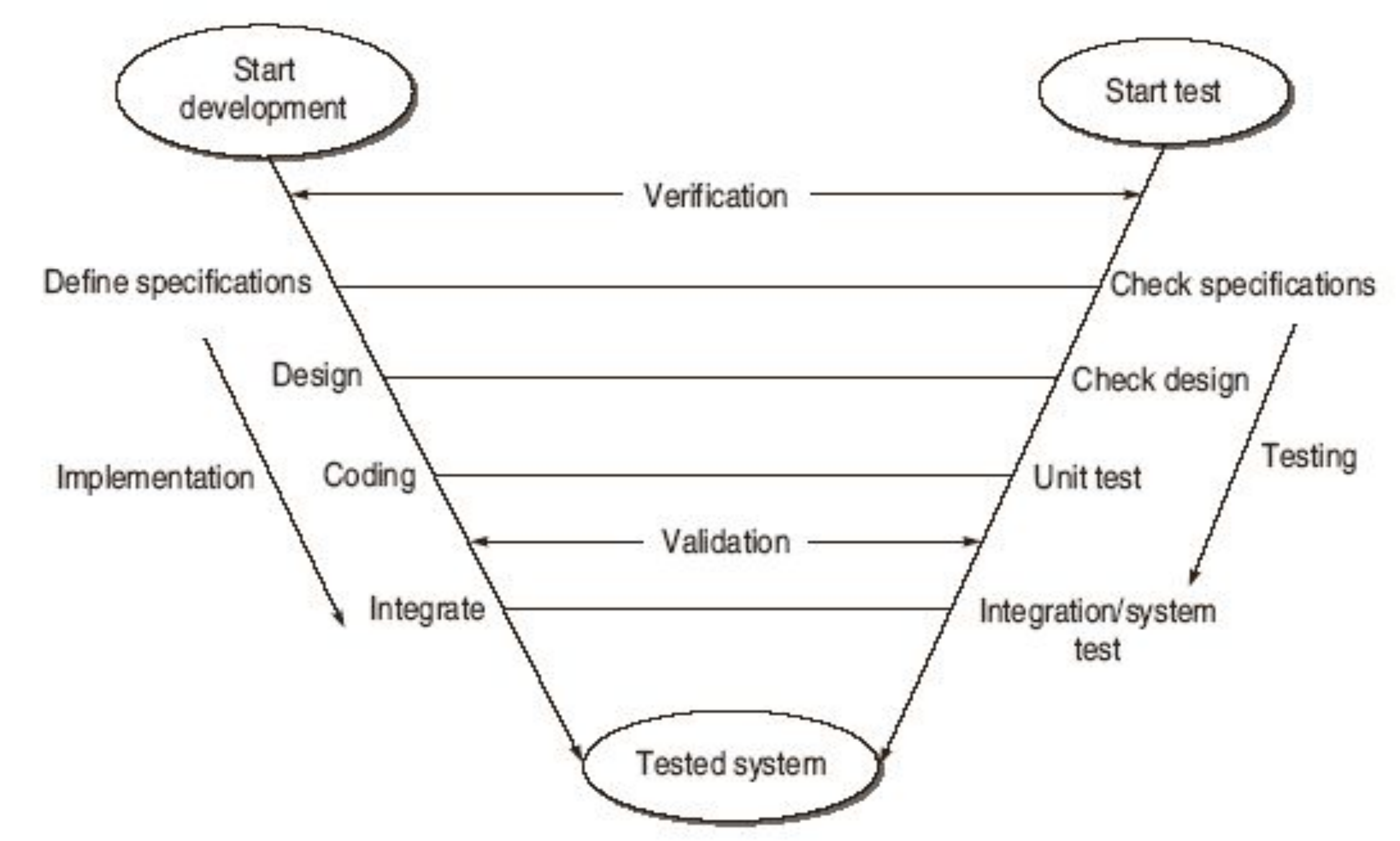
Test Factors	Test Phase					
	Requirements	Design	Code	Unit test	Integration test	System test
Portability	Is portability feature mentioned in specifications according to different hardware?					Is system testing performed on MIPS and INTEL platforms?
Service Level	Is time frame for booting mentioned?	Is time frame incorporated in design of the module?				

Development of Test Strategy

Verification: “Are we building the product right?”

Validation: “Are we building the right product?”

V Testing Life Cycle Model



Validation Activities

- Unit testing
- Integration testing
- Function testing
- System testing
- Acceptance testing

Testing Tactics

Testing tactics are the ways to perform various types of testing. This can be done in two ways: *manual testing* and *automated testing*.

- Manual testing is the type of testing performed when the technique is applied manually, that is, no automation tool is used.

- On the contrary, automated testing is performed with the help of testing to save time and effort. To automate the test cases, sometimes, the tester may need to write a set of routines.

Testing Tactics

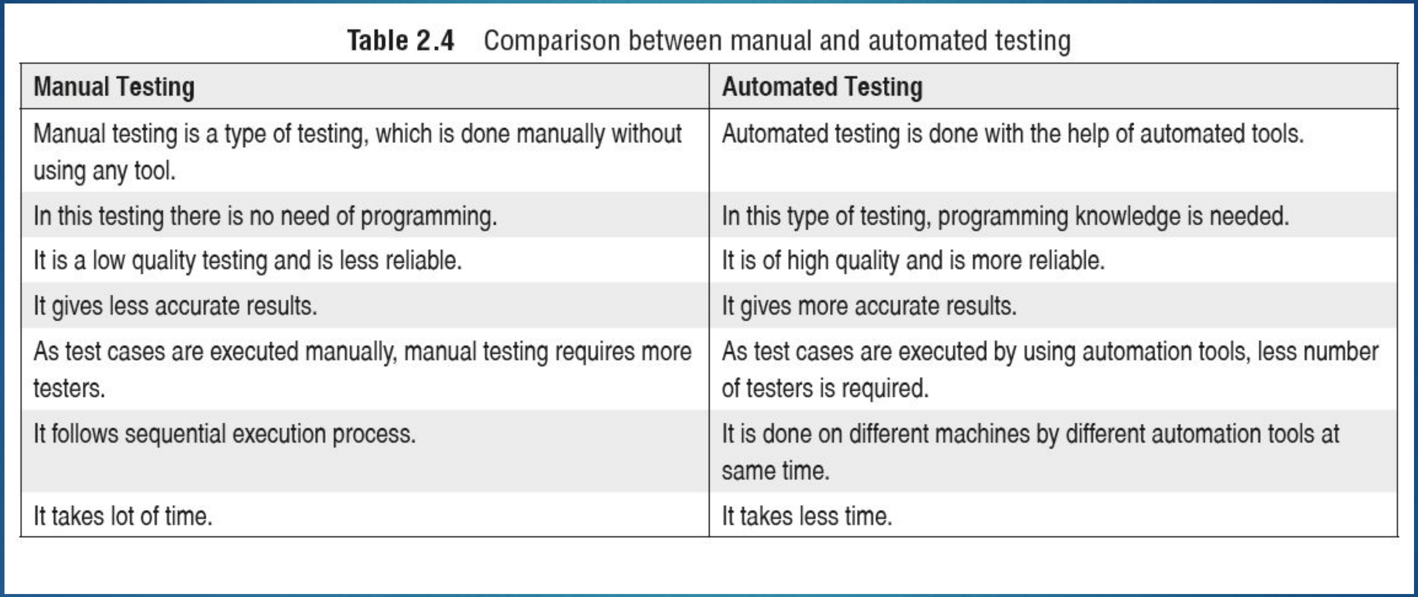


Table 2.4 Comparison between manual and automated testing

Manual Testing	Automated Testing
Manual testing is a type of testing, which is done manually without using any tool.	Automated testing is done with the help of automated tools.
In this testing there is no need of programming.	In this type of testing, programming knowledge is needed.
It is a low quality testing and is less reliable.	It is of high quality and is more reliable.
It gives less accurate results.	It gives more accurate results.
As test cases are executed manually, manual testing requires more testers.	As test cases are executed by using automation tools, less number of testers is required.
It follows sequential execution process.	It is done on different machines by different automation tools at same time.
It takes lot of time.	It takes less time.

Testing Tactics

Actual methods for designing test cases, that is, software testing techniques, implement the test cases on the software. These techniques can be categorized into:

- *Static testing*: It is a technique for assessing the structural characteristics of source code, design specifications or any notational representation that conforms to well-defined syntactic rules. It is called as static because we never execute the code in this technique.
- *Dynamic testing:* In this technique, the code is run on a number of inputs provided by the user and the corresponding results are checked. This type of testing is further divided into two parts: (a) black-box testing and (b) white-box testing.

Testing Tactics

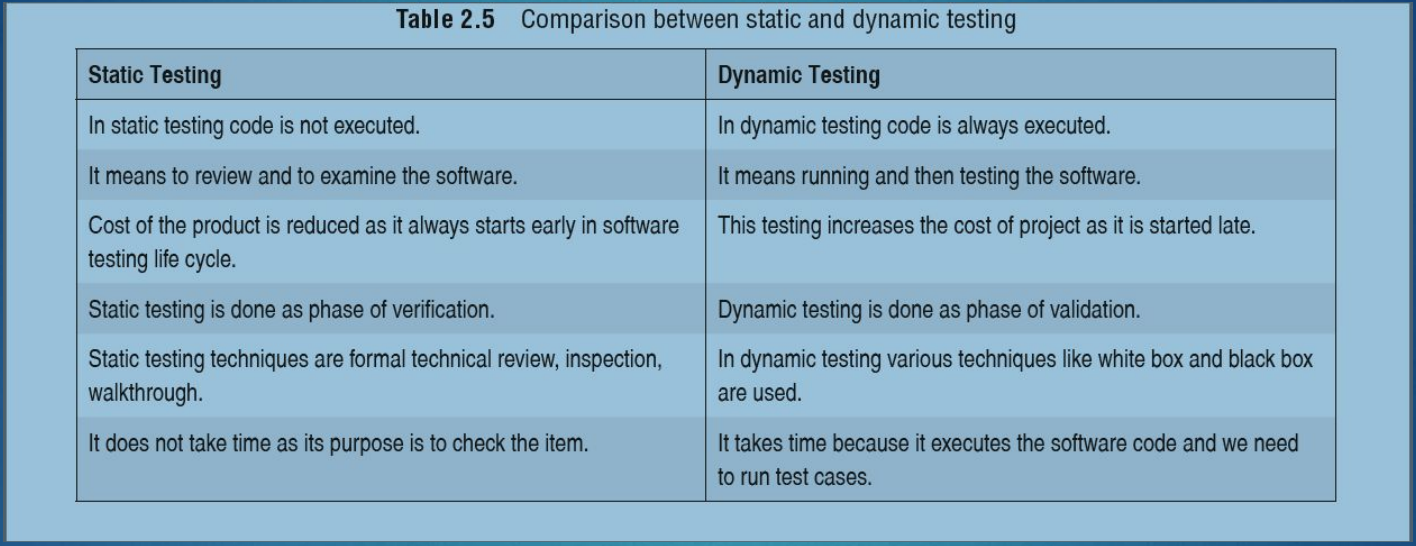


Table 2.5 Comparison between static and dynamic testing

Static Testing	Dynamic Testing
In static testing code is not executed.	In dynamic testing code is always executed.
It means to review and to examine the software.	It means running and then testing the software.
Cost of the product is reduced as it always starts early in software testing life cycle.	This testing increases the cost of project as it is started late.
Static testing is done as phase of verification.	Dynamic testing is done as phase of validation.
Static testing techniques are formal technical review, inspection, walkthrough.	In dynamic testing various techniques like white box and black box are used.
It does not take time as its purpose is to check the item.	It takes time because it executes the software code and we need to run test cases.

Testing Tactics

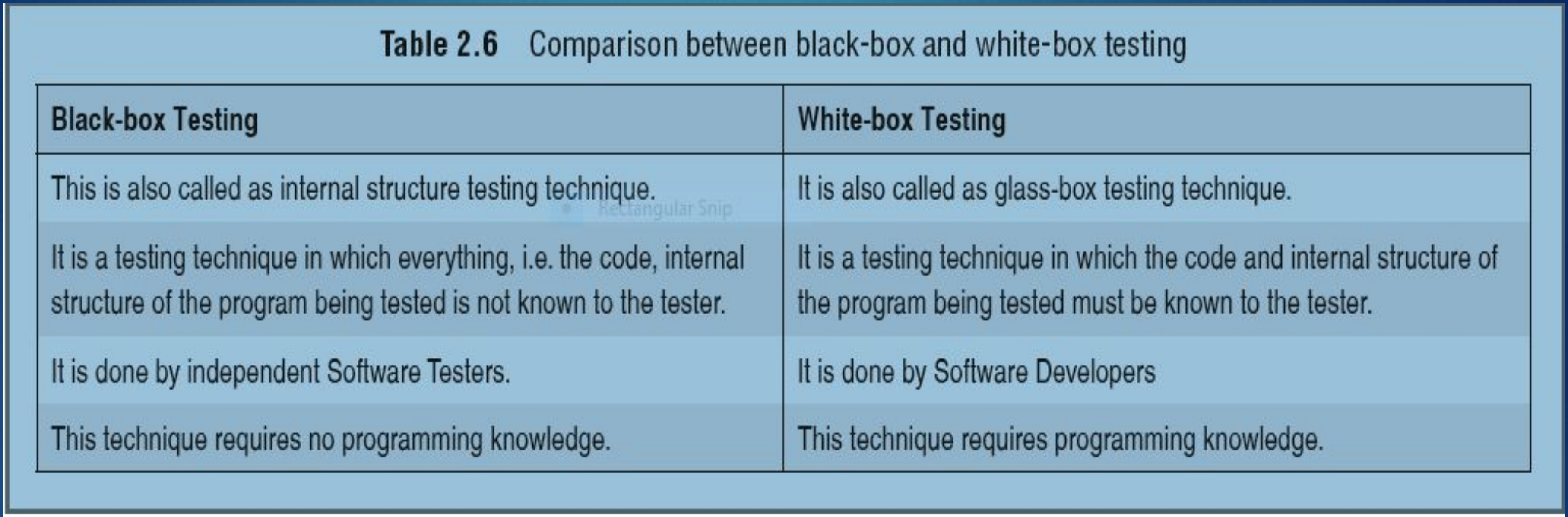


Table 2.6 Comparison between black-box and white-box testing

Black-box Testing	**White-box Testing**
This is also called as internal structure testing technique.	It is also called as glass-box testing technique.
It is a testing technique in which everything, i.e. the code, internal structure of the program being tested is not known to the tester.	It is a testing technique in which the code and internal structure of the program being tested must be known to the tester.
It is done by independent Software Testers.	It is done by Software Developers
This technique requires no programming knowledge.	This technique requires programming knowledge.