# Software Testing

CH:1 INTRODUCTION

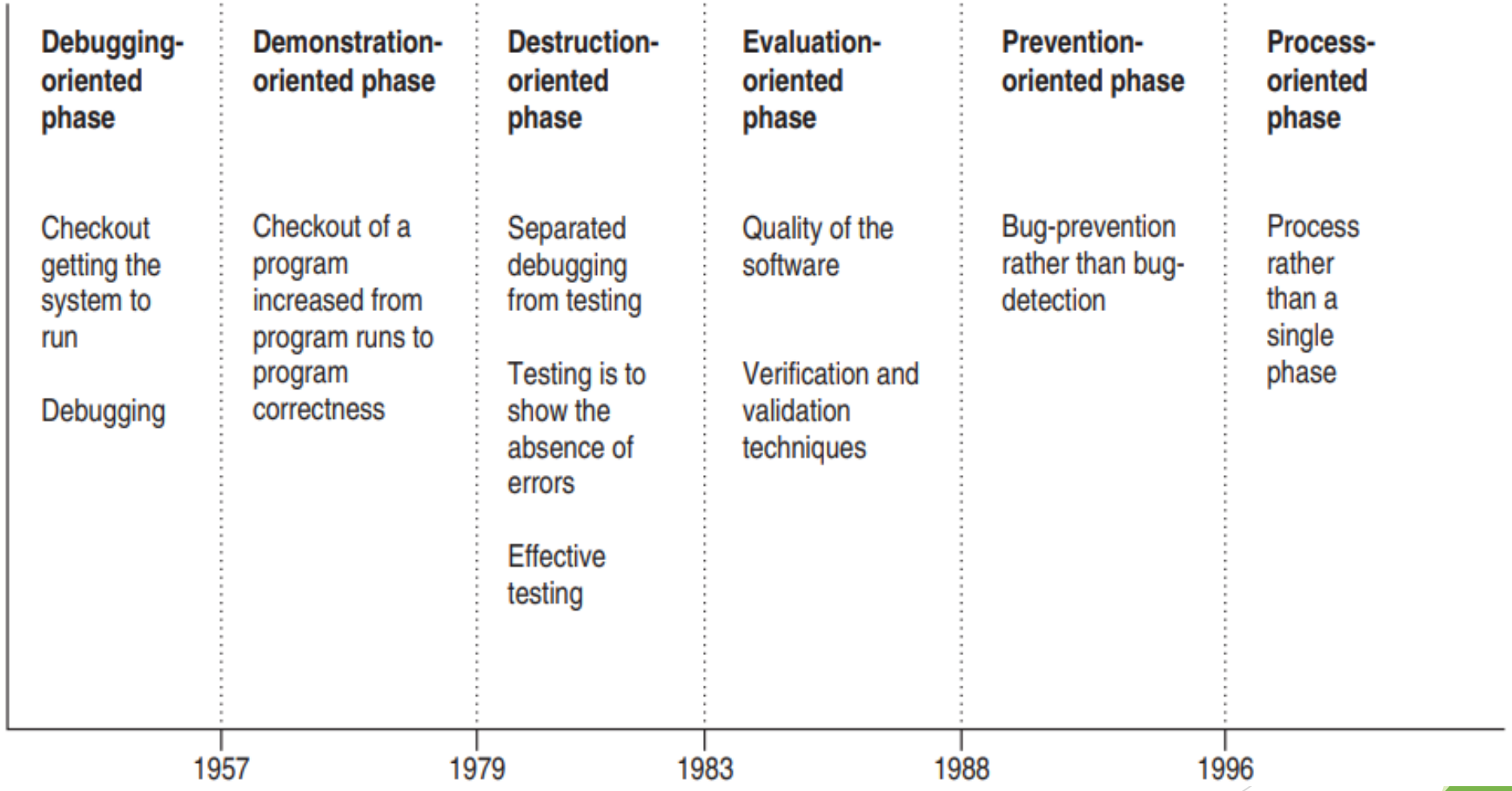
# INTRODUCTION:

- Software has pervaded our society, from modern households to spacecrafts. It has become an essential component of any electronic device or system.
- This is why software development has turned out to be an exciting career for computer engineers in the last 10–15 years.
- However, software development faces many challenges. Software is becoming complex, but the demand for quality in software products has increased. This rise in customer awareness for quality increases the workload and responsibility of the software development team. That is why software testing has gained so much popularity in the last decade.
- Organizations have separate testing groups with proper hierarchy. Software development is driven with testing outputs. If the testing team claims the presence of bugs in the software, then the development team cannot release the product.

# INTRODUCTION:

- There still is a gap between academia and the demand of industries.
- In the early days of software development, software testing was considered only a debugging process for removing errors after the development of software.
- By 1970, the term ‘software engineering’ was in common use. But software testing was just a beginning at that time. In 1978, G. J. Myers realized the need to discuss the techniques of software testing in a separate subject. He wrote the book The Art of Software Testing which is a classic work on software testing.
- Myers discussed the psychology of testing and emphasized that testing should be done with a mindset of finding errors and not to demonstrate that errors are not present.

# EVOLUTION OF SOFTWARE TESTING



# Definition of Software Testing:

- Testing is the process of demonstrating that there are no errors.
- Testing is the process of executing a program with the intent of finding errors. [Myers]
- Testing can show the presence of bugs but never their absence. [W. Dijkstra]
- Testing is a support function that helps developers look good by finding their mistakes before anyone else does. [James Bach]
- Testing is concurrent lifecycle process of engineering, using and maintaining test-ware (testing artifacts) in order to measure and improve the quality of the software being tested. [Craig]

# Goals of Software Testing:

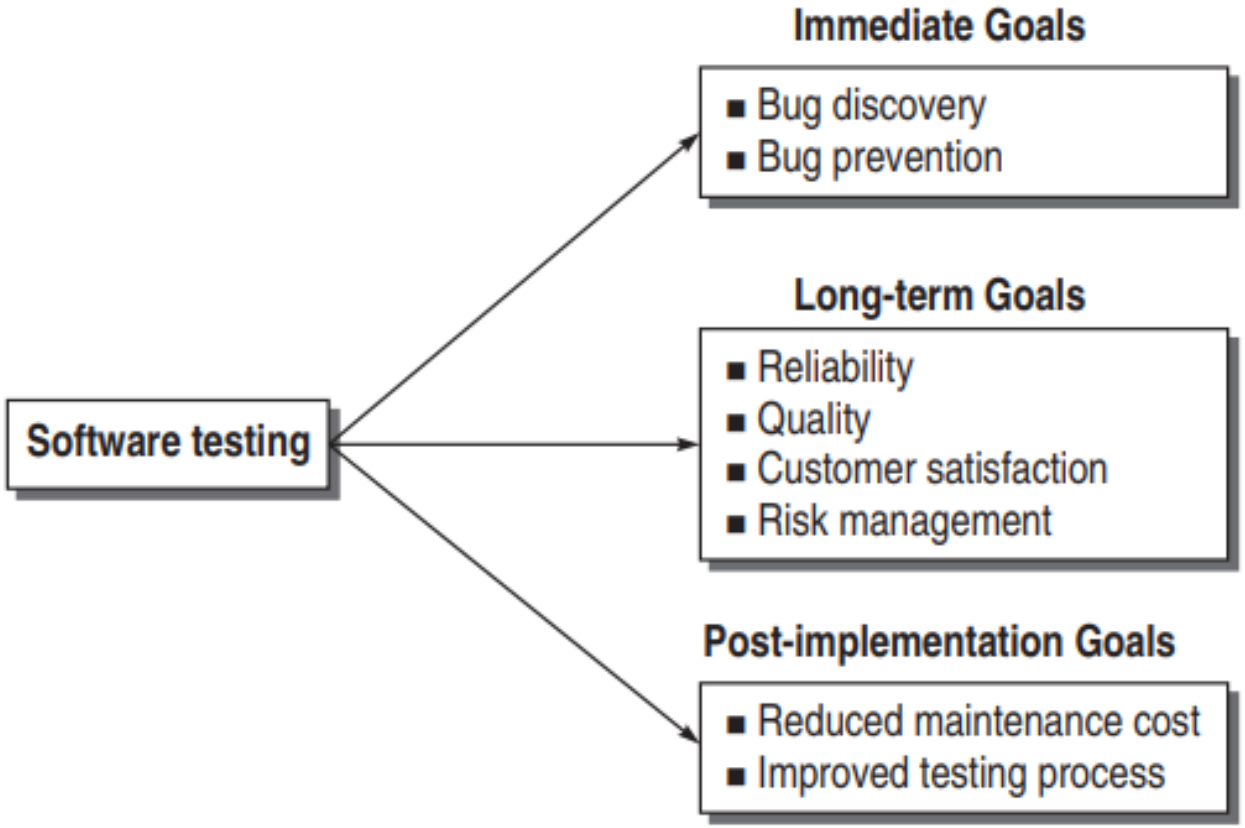



**Figure 1.2** Software testing goals

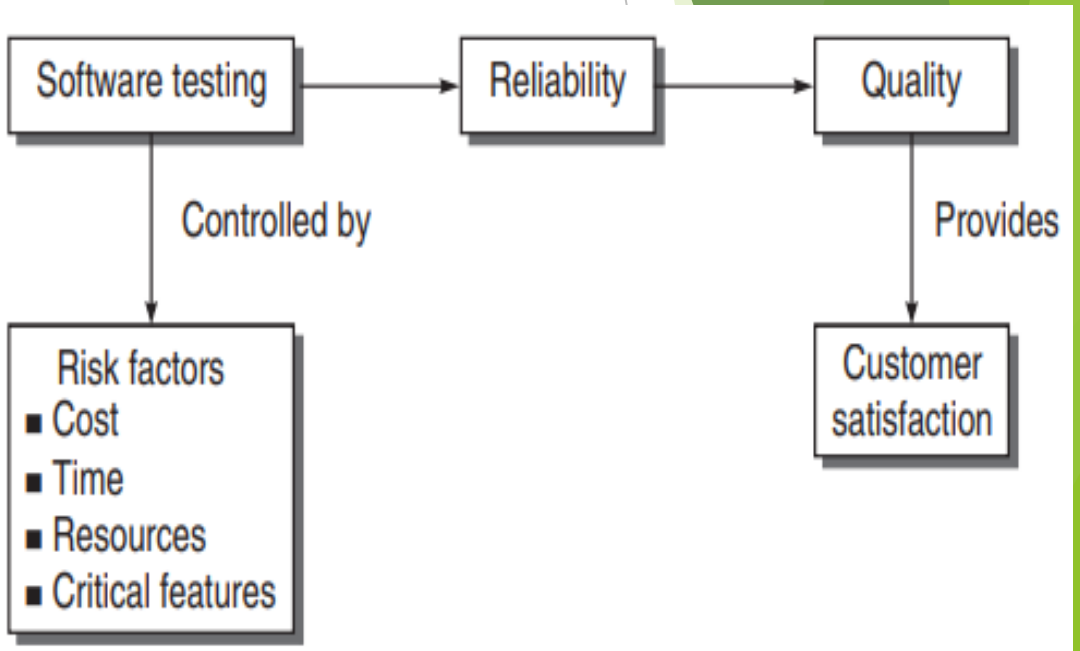



**Figure 1.5** Testing controlled by risk factors

# Failure, Fault, Defect and Bug:

- **Failure:** When the software is tested, *failure is the first term being used. It* means the inability of a system or component to perform a required function according to its specification. In other words, when results or behavior of the system under test are different as compared to specified expectations, then failure exists.
- **Fault/Defect/Bug:** Failure is the term which is used to describe the problems in a system on the output side, as shown in Fig. 2.1. *Fault is a condition that in* actual causes a system to produce failure. Fault is synonymous with the words *defect or bug. Therefore, fault is the reason embedded in any phase of SDLC* and results in failures. It can be said that failures are manifestation of bugs. One failure may be due to one or more bugs and one bug may cause one or more failures. Thus, when a bug is executed, then failures are generated. But this is not always true. Some bugs are hidden in the sense that these are not executed, as they do not get the required conditions in the system. So, hidden bugs may not always produce failures. They may execute only in certain rare conditions.

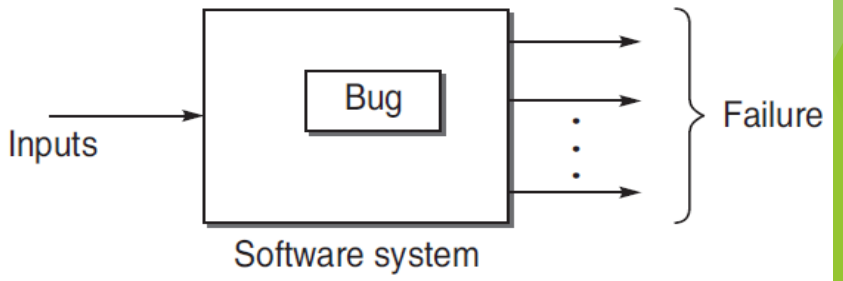



**Figure 2.1** Testing terminology

# Error:

- **Error:** Whenever a development team member makes a mistake in any phase of SDLC, errors are produced. It might be a typographical error, a misleading of a specification, a misunderstanding of what a subroutine does, and so on. Error is a very general term used for human mistakes. Thus, an error causes a bug and the bug in turn causes failures, as shown in Fig. 2.2.


**Figure 2.2** Flow of faults

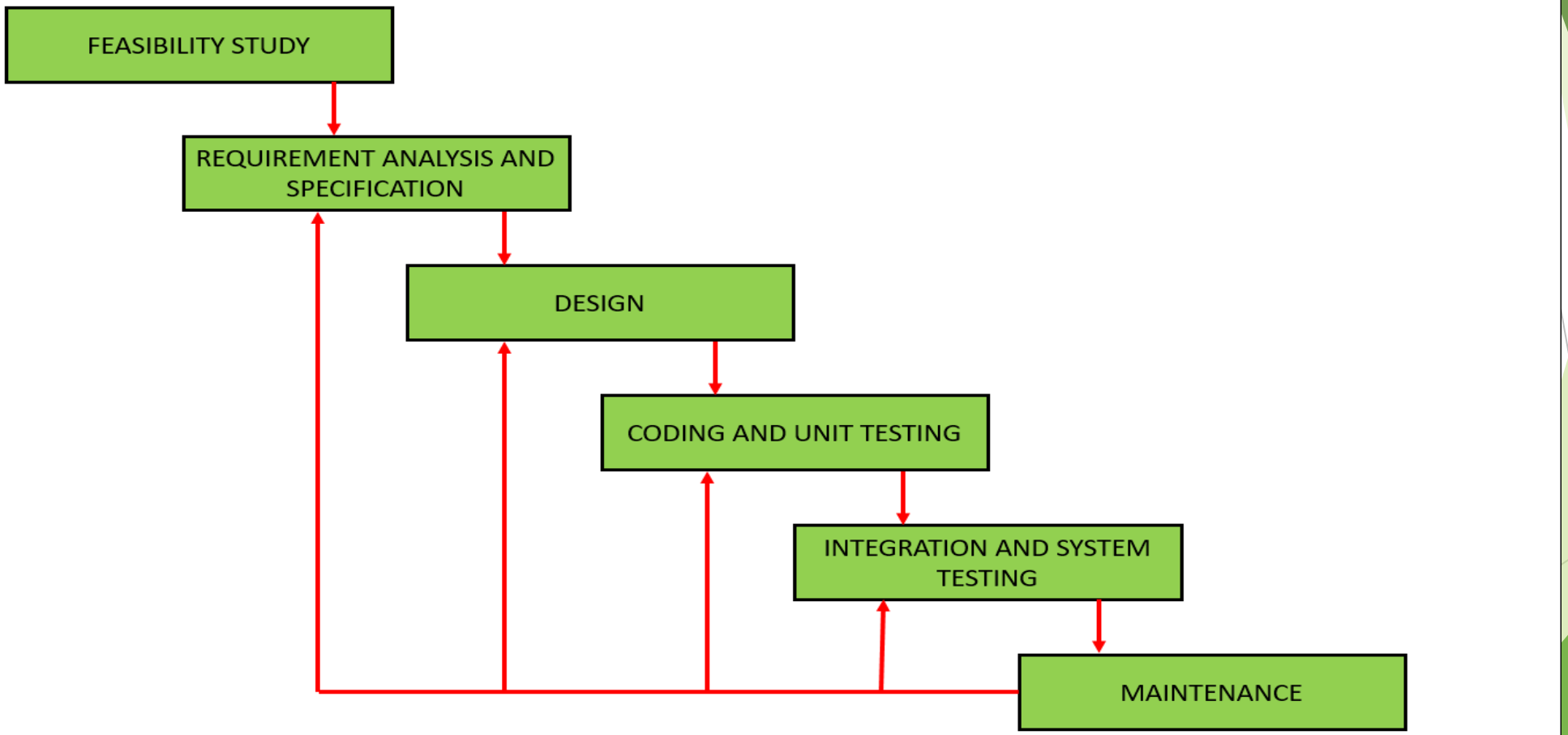
# Testing Principles:

- 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 should also be performed by an independent team.

# Testing Principles:

- 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.

# SDLC (Iterative Waterfall Model)



# Software Testing Life Cycle (STLC):

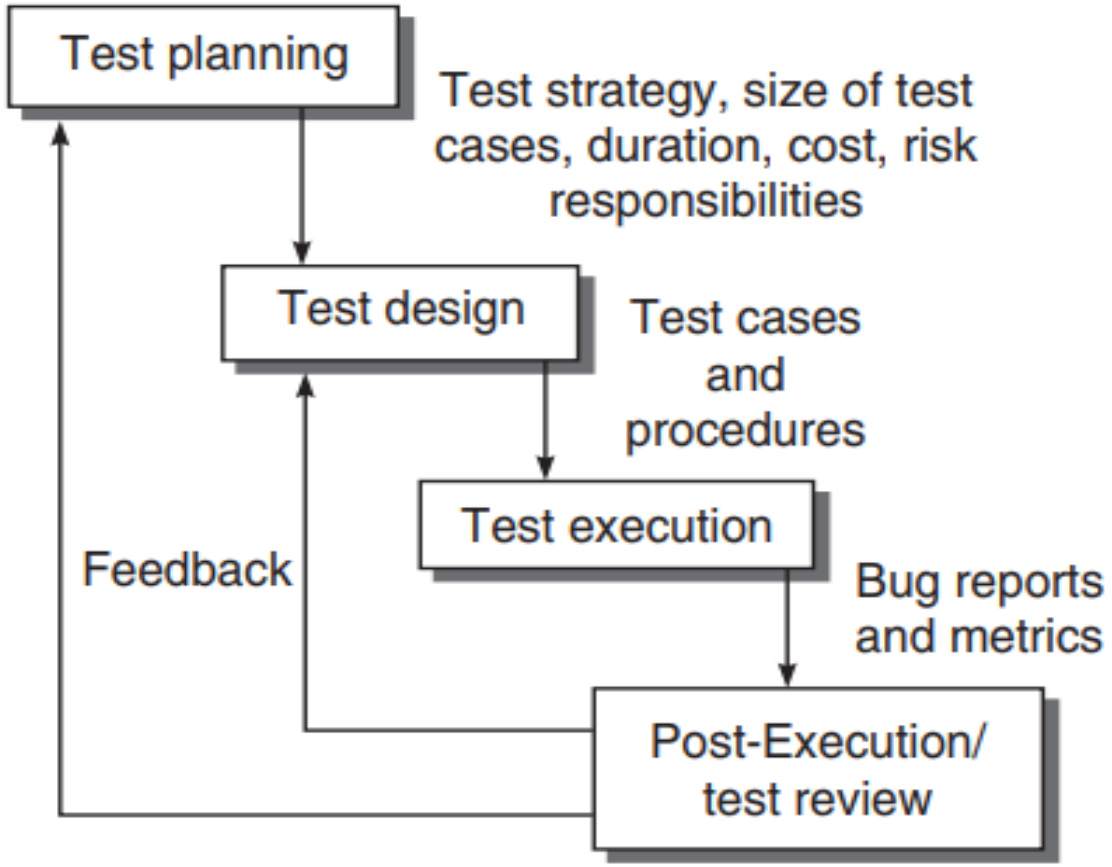



**Figure 2.8** Software testing life cycle

# 1. Test Planning

The goal of test planning is to take into account the important issues of testing strategy, viz. resources, schedules, responsibilities, risks, and priorities, as a roadmap. Test planning issues are in tune with the overall project planning. Broadly, following are the activities during test planning:

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

# Test Planning:

Based on the planning issues as discussed above, analysis is done for various testing activities. The major output of test planning is the test plan document. Test plans are developed for each level of testing. After analyzing the issues, the following activities are performed:

- Develop a test case format.
- Develop test case plans according to every phase of SDLC.
- Identify test cases to be automated (if applicable).
- Prioritize the test cases according to their importance and criticality.
- Define areas of stress and performance testing.
- Plan the test cycles required for regression testing.

# Test Design:

One of the major activity in testing is the design of test cases. The test design is an important phase after test planning. It includes following activity:

- Determine test objectives and their prioritization.
- Preparing list of items to be tested.
- Mapping items to test cases.

*(There is only one rule in designing test cases: Cover all features, but do not make too many test cases)*

- Selection of test case design techniques. [Blackbox / Whitebox testing]
- Creating test cases and test data. [Input required and Output Expected]
- Setting up test environment and supporting tools. [H/W, testers, interfaces, OS etc.]
- Creating test procedure specification. [Description of how the test cases will run. It is a sequence of steps]

# Attributes of a good test cases:

1. A good test case is one that has been designed keeping in view the criticality and high-risk requirements in order to place a greater priority upon, and provide added depth for testing the most important function.
2. A good case should be designed such that there is a high probability of finding an error.
3. Test cases should not be overlapped or redundant.
4. It is possible to combine a series of tests into one test case.
5. A successful test case is one that has the highest probability of detecting an as – yet –undiscovered error.

# Test Execution

- In this phase, all test cases are executed including verification and validation.
- Verification test cases start at the end of each face of SDLC. Validation test cases start after the completion of a module.
- It is the decision of the test team to opt for automation or manual execution.
- Test results are documented in the incident reports, test logs, testing status, and test summary reports.
- ***Test incident*** is the report about any unexpected event during testing, which needs further investigation to resolve the bug.
- ***Test log*** is a record of the testing events that take place during the test.
- ***Test summary report*** is an evaluation report describing summary of all the tests and is prepared when the testing is over.

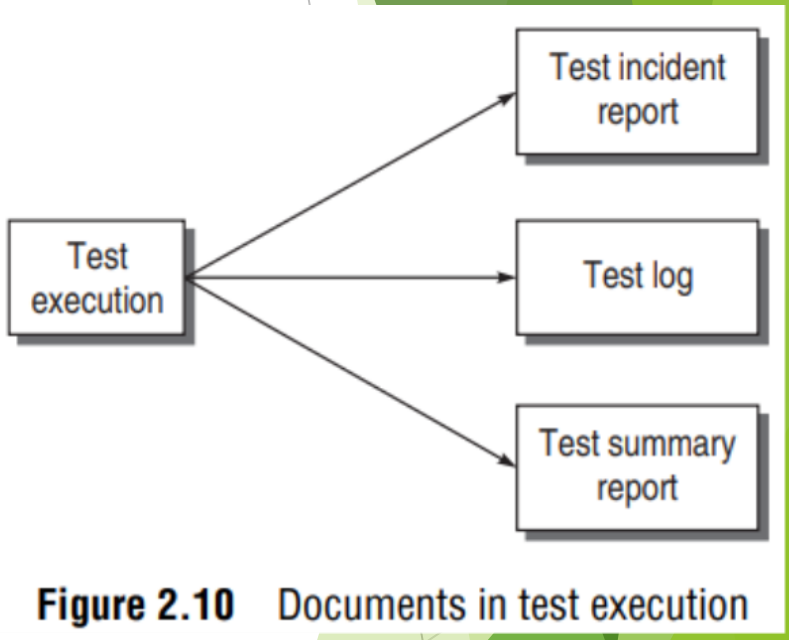
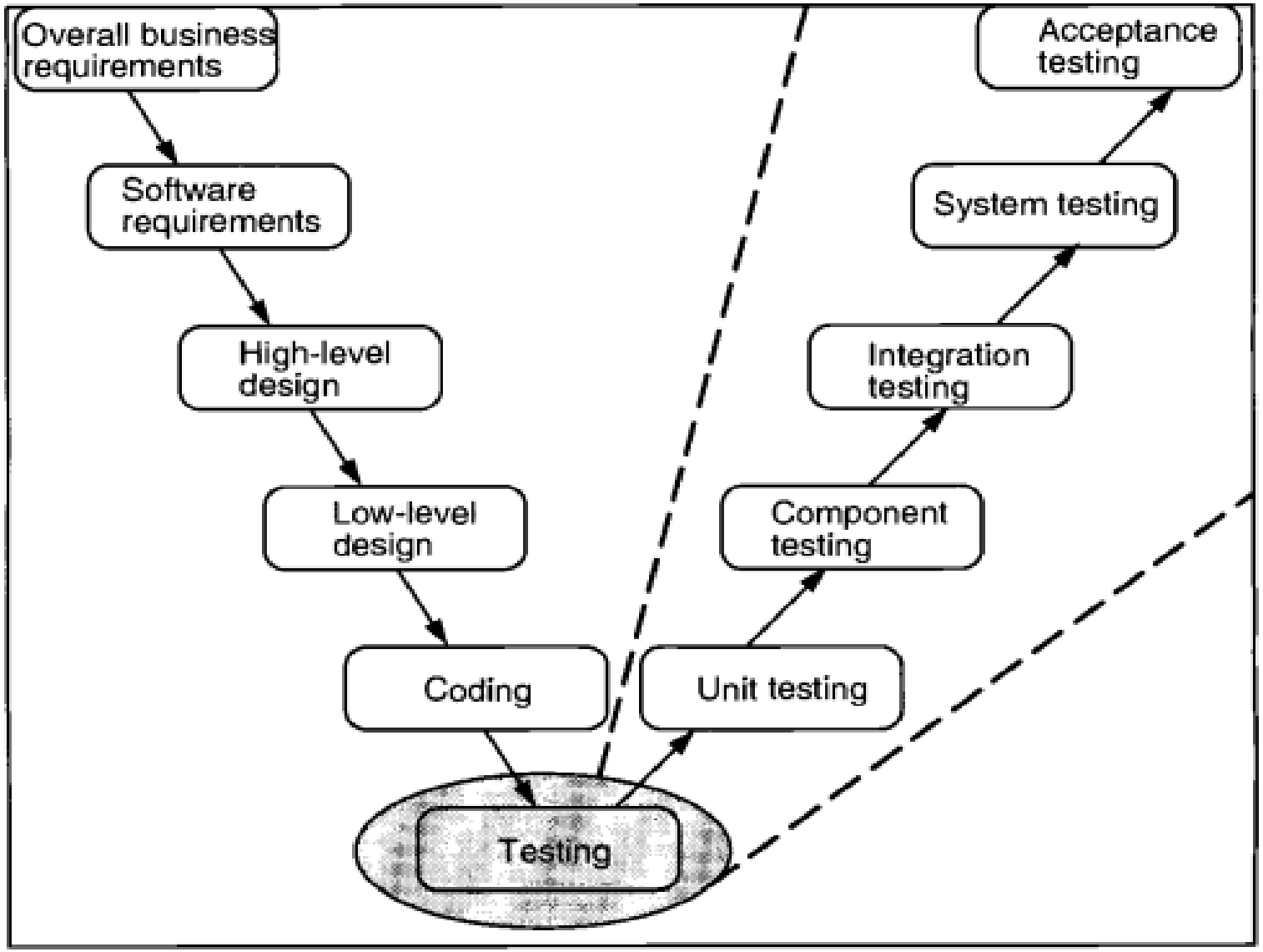
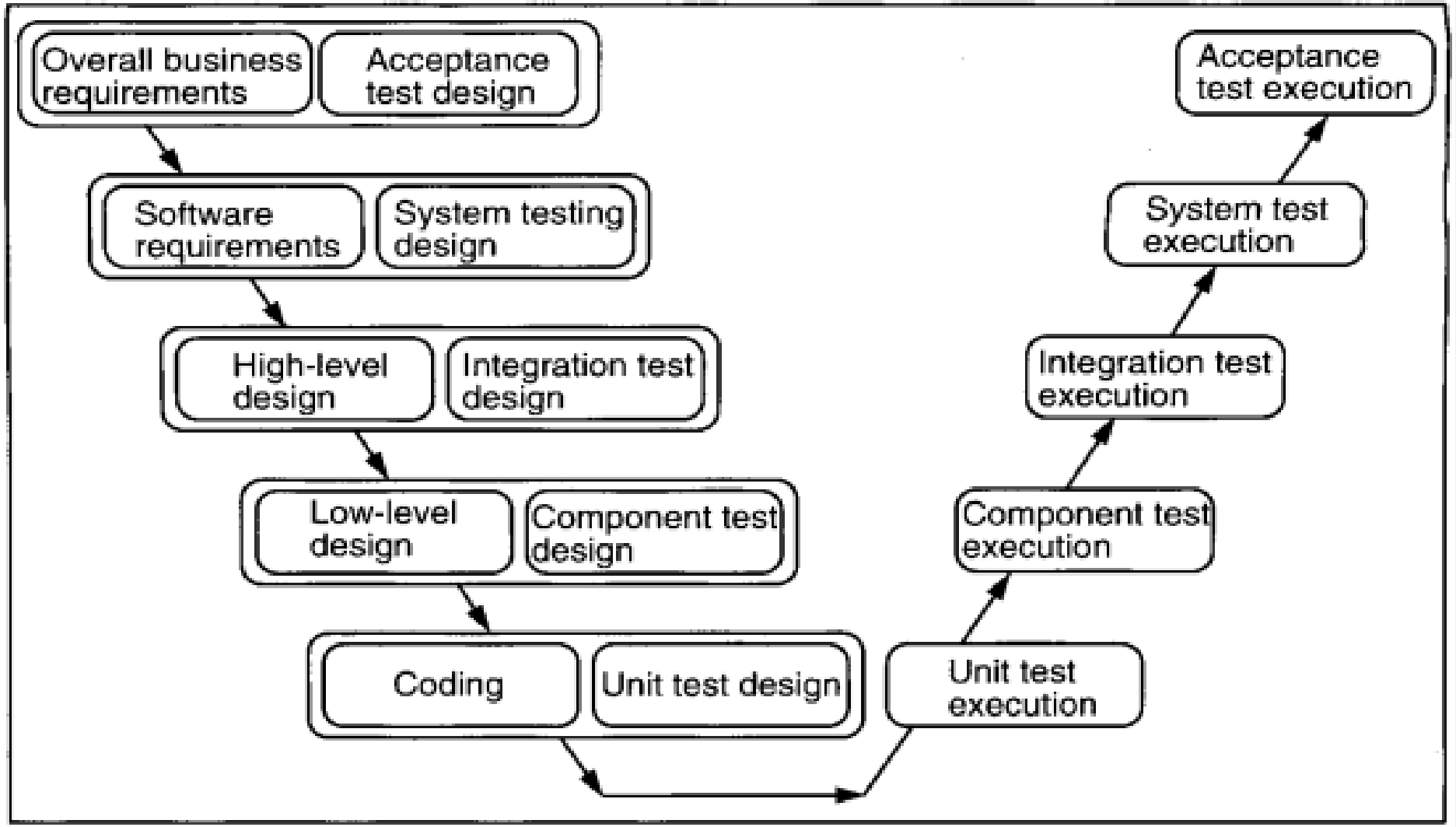



**Figure 2.10** Documents in test execution

# V-testing:


Overall business requirements
Software requirements
High-level design
Low-level design
Coding
Testing
Unit testing
Component testing
Integration testing
System testing
Acceptance testing

# V-testing:


Overall business requirements
Acceptance test design
Software requirements
System testing design
High-level design
Integration test design
Low-level design
Component test design
Coding
Unit test design
Unit test execution
Component test execution
Integration test execution
System test execution
Acceptance test execution

# Testing Life Cycle Model:

- Verification and Validation V&V( are the building blocks of a testing process. Life cycle involves continues testing of the system during the development process.
- In V-testing concept, as the development team attempts to implement the software, the testing team concurrently starts checking the software.
- When the project starts, both the system development and the system test process begin.
- The team that is developing the system begins the system development process and the team that is conducting the system test begins planning the system test process.

# V-Testing Life Cycle Model:

- In V-Testing concept, as the development team attempts to implement the software, the testing team concurrently starts checking the software.
- When the project starts, both the system development and the system test process begin.
- The team that is developing the system begins the system development process and the team that is conducting the system test begins planning the system test process as shown in the figure given below.
- The V&V process involves:

1. Verification of every step of SDLC and,
2. Validation of the verified system at the end.

# V-Testing Life Cycle Model:

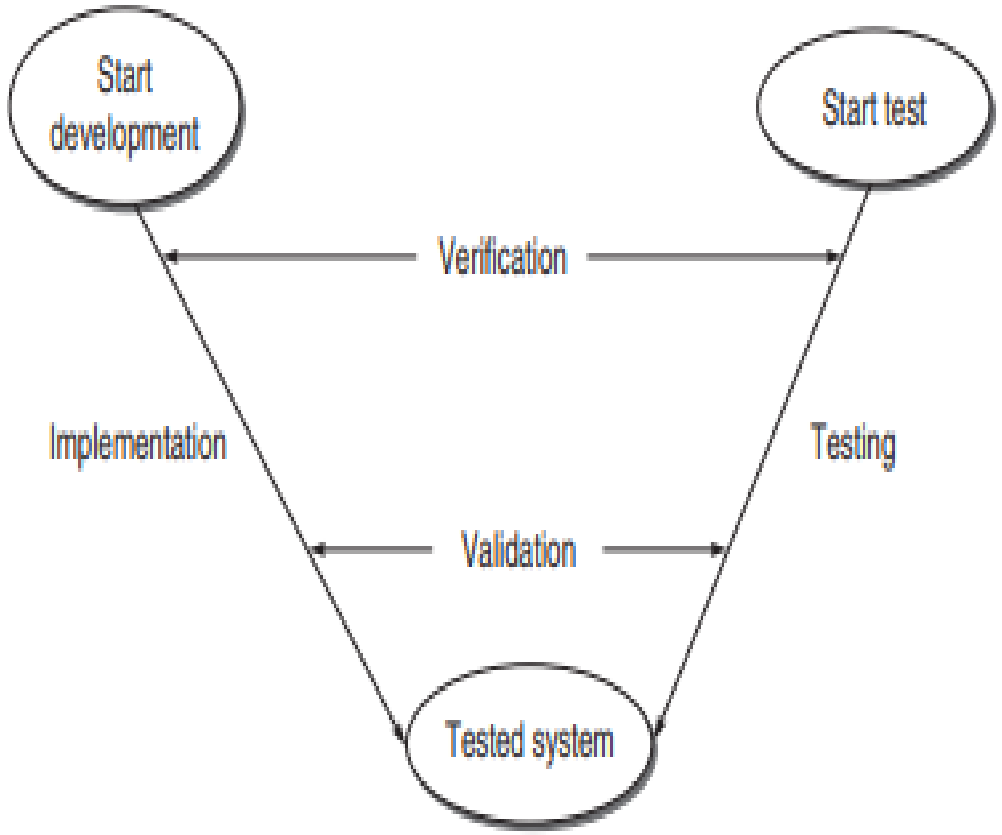



**Figure 2.12** V-testing model

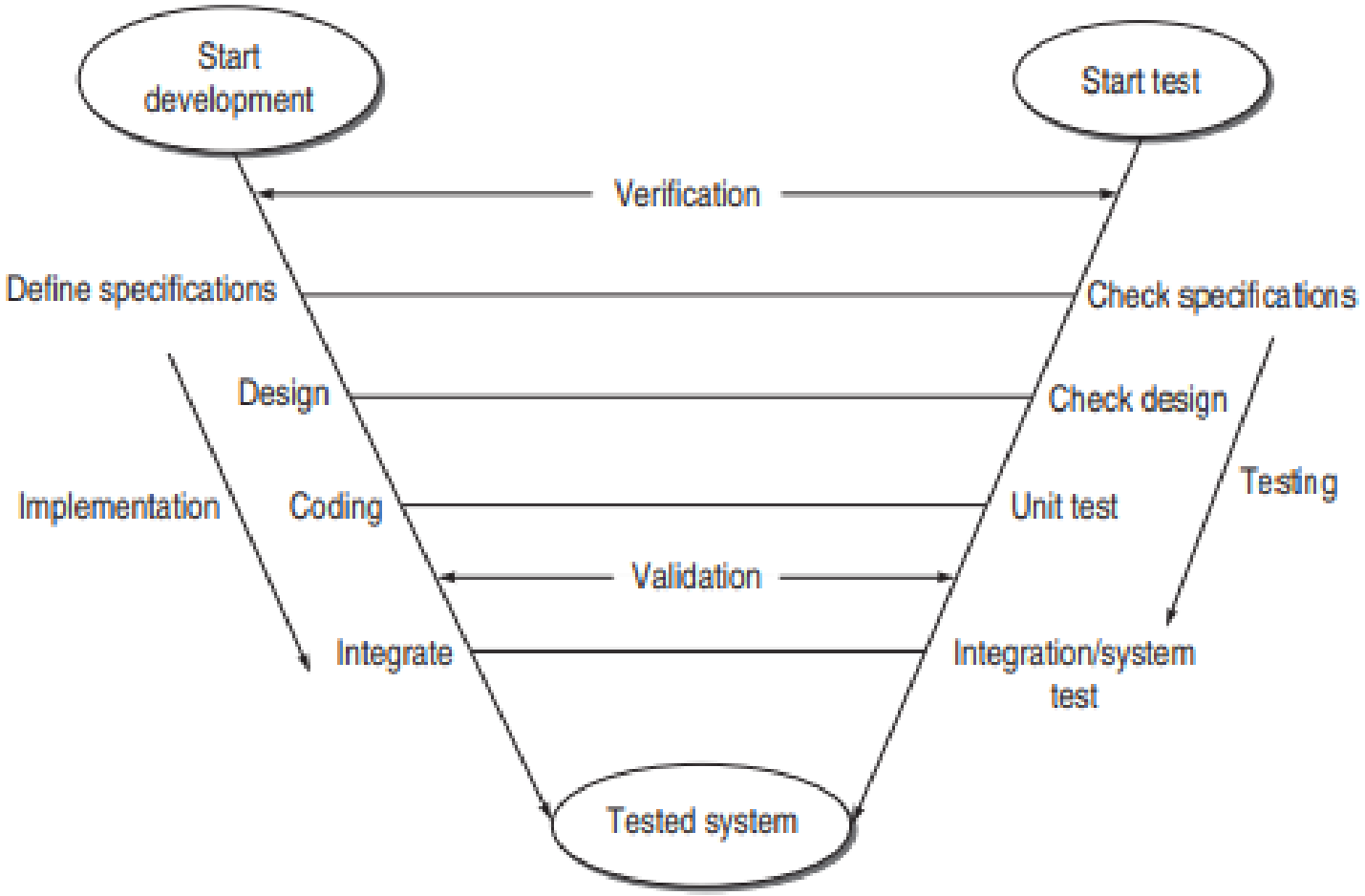



**Figure 2.13** Expanded V-testing model

# V-Testing Life Cycle Model:

VERIFICATION:

- Verification is a set of activities that ensures correct implementation of specific functions in a software.
- What is the need of verification?

1. If the verification is not performed at early stages, there is always a chance of mismatch between the required product and the delivered product.
2. Verification exposes more errors.
3. Early verification decreases the cost of fixing bugs.
4. Early verification enhances the quality of software.

- What is the goal of verification?

1. Everything must be verified
2. Results of verification may not be binary
3. Even implicite quantities must be verified.

# V-Testing Life Cycle Model:

VALIDATION ACTIVITIES:

- Validation involves the following three activities, which are also known as the three levels of validation testing.

1) Unit Testing: It is major validation efforts performed on the smallest module of the system. If avoided, many bugs becomes latent bugs and are released in the final product. Unit testing is a basic level of testing which can not be overlooked and confirms the behavior of a single module according to its functional specification.
2) Integration Testing: It is a validation technique which combines all unit-tested modules and performs a test on their aggregation. Integration testing is needed because of interfacing. Unit modules are not independent, and are related to each other by interfacing between unit modules.
3) System Testing: This testing level focuses on testing the entire integrated system. It incorporates many types of testing, as the full system can have various users in different environments.

# Software Testing Techniques:

**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 [16]. It is called as static because we never execute the code in this technique. For example, the structure of code is examined by the teams but the code is not executed.

# Software Testing Techniques:

**DYNAMIC TESTING:**

All the methods that execute the code to test a software are known as dynamic testing techniques. 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.

**[A] Black-box testing:** This technique takes care of the inputs given to a system and the output is received after processing in the system. What is being processed in the system? How does the system perform these operations? Black-box testing is not concerned with these questions. It checks the functionality of the system only. That is why the term black-box is used. It is also known as functional testing. It is used for system testing under validation.

**[B] White-box testing:** This technique complements black-box testing. Here, the system is not a black box. Every design feature and its corresponding code is checked logically with every possible path execution. So, it takes care of the structural paths instead of just outputs. It is also known as structural testing and is used for unit testing under verification.

# Difference between Static & 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 verification | Dynamic testing is done as phase validation |
| Static testing techniques are formal technique 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 the test cases. |

# Difference 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 the independent Software Testers | It is done by the Software Developers |
| The technique requires no programming knowledge | This technique requires programming knowledge |

# ASSIGNMENT-1:

**ANSWER THE FOLLOWING QUESTIONS IN BREIF:**

1. What is software testing? List and explain goals of software testing.
2. What are the principles of software testing? Explain them.
3. Explain Software Testing Life Cycle (STLC).
4. What is V-testing life cycle model? How is it used? Explain in detail.
5. State the difference between static and dynamic testing?

**MCQs:**

| Chapter | No. of objectives |
|---|---|
| 1 | 1.1 to 1.6 |
| 2 | 2.1 to 2.14 (All) |