# Unit 4 – Chapter 10

# Testing

## Prerequisites:

Software development life cycle, software requirements, design, and development

## Unit Outcomes:

The objective of this chapter is to introduce the concept of software testing. At the end of this chapter, students will be able to:

- Understand the software testing process
- Recognize the development, release, and user testing
- Recommend a performance testing method
- Analyze structural and functional testing

## 10.1 Introduction

Software testing is the execution of a program to find its faults. The purpose of software testing is to prove that a program does the required function without errors. During testing, the program is executed using a set of test inputs known as test cases. Different errors, irregularities, defects, etc. are checked during the test run. Also, whether the software fulfills the functional and non-functional requirements or not is tested. The different terminologies used in testing are **bugs, defects, errors, faults, and failures.**

The defect in the software is informally known as a **bug**. This term is frequently used by software test engineers to indicate that the software is not functioning as per the given requirements. The bug may be due to an algorithm, resources, or logic.

When the system is not meeting the requirements then it is said to have a **defect**. The defect is present if the actual output deviates from the expected output. Defects can be major, critical, or minor.

When there is a mistake in a code then it is called an **error**. People make mistakes due to a lack of understanding of the requirements, programming language features, or less field experience. Software with an error cannot be compiled or executed. Errors may be the syntax, semantics, hardware, testing errors, etc.

Whenever there is an error, the system fails to work correctly. This state of incorrect working of a system is known as a **fault.** There may be a fault in the interface, or front end, security, the performance of the software, etc.

The system leads to **failure** if there are multiple defects. If the defect is causing incorrect functioning of the system, then the system is in failure condition.

### 10.1.1 Testing Objectives

The testing objectives can be given as:

1. To show to the software developer and client that the developed software meets the specified requirements. If the software is a customized product, then for each requirement in the requirement document there should be at least a single test case. If the software is a generic software product, then there should testing for all the features and common testing during its launch. This is known as **validation testing**, where the performance of the software is checked using suitable test cases. Test cases are the set of data related to the problem. Thus, validation testing is a process of finding errors by executing the program in a real environment.
2. To find those situations where the software behavior is erroneous, not desirable, and not meeting its specification. If there are defects in the developed software, then it may lead to accidents such as security violations, system crashes, wrong computations, and loss of data. This testing is known as **defect testing**. The test cases used in this type may not be exactly as used in the application but are designed to find the errors.

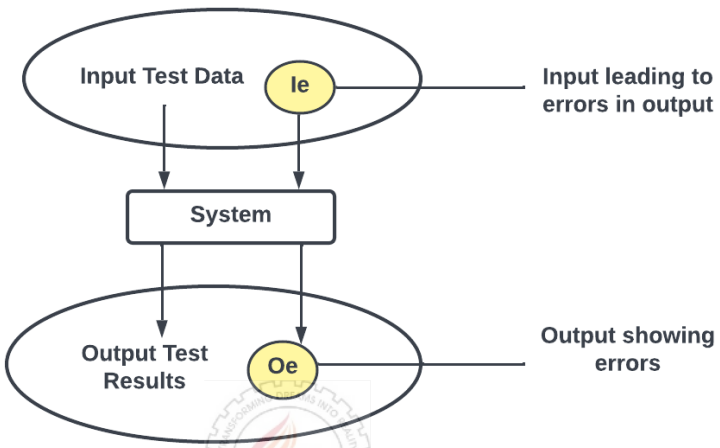

Figure 10.1 Program testing using an input-output model

There is no specific boundary in both approaches. Consider Figure 10.1, the system receives data from input set **I** and generates output as **O**. Some of the output is incorrect and is given by **Oe**. This defective output is generated by some incorrect input values given **Ie**. The defect testing aims at finding those input values in the set **Ie** which are causing errors. Validation testing concentrates on the inputs outside **Ie**, finding whether all the system requirements are met or not.

Barry Boehm, a well-known scientist in software engineering, has shown this difference in Validation and Verification, where 'Validation: Are we building the right product?' and 'Verification: Are we building the product right?'

### 10.1.2 Verification vs. Validation

In **verification**, the software requirements specification, design, and code are checked. It may not include program execution. Code inspections, reviews, or code walkthroughs which are discussed in Unit 3 are performed in this stage. The verification is performed by the system analysts and designers to check whether the SRS is matching with software architecture. High-level design and database design are tested in verification. Thus, it comes in the early stages of software development.

**Validation** is a generic process that is used to ensure that the software meets the client's requirements. It involves the execution of the code. The target used in the validation is always the code. Validation is performed after the verification of the code. The people involved in validation include the software development team.

The objective of verification and validation is to develop the confidence in the software system that it is **fit for its purpose**. The level of confidence in the system depends on the purpose of the system. Consider developing a safety alarm system and a word processing typesetting system. Obliviously the level of confidence is more important in the safety alarm system than in the typesetting project. The expectation from the users is yet another issue in building the system's confidence. In the initial stages, the users will be tolerable, hence less testing is required. More detailed testing is required as the software is matured and users have become reliable in using it. The market status also determines the confidence in the software. The quality of the software depends on the cost and market status. Cheap software may be preferred by clients, and they will tolerate the failure of it if any.

The verification and validation process may involve software inspections and reviews along with software testing. The requirement specifications, design models, code, and the proposed tests are checked in the inspection. This is known as **V & V technique.**

### 10.1.3 Software Inspection

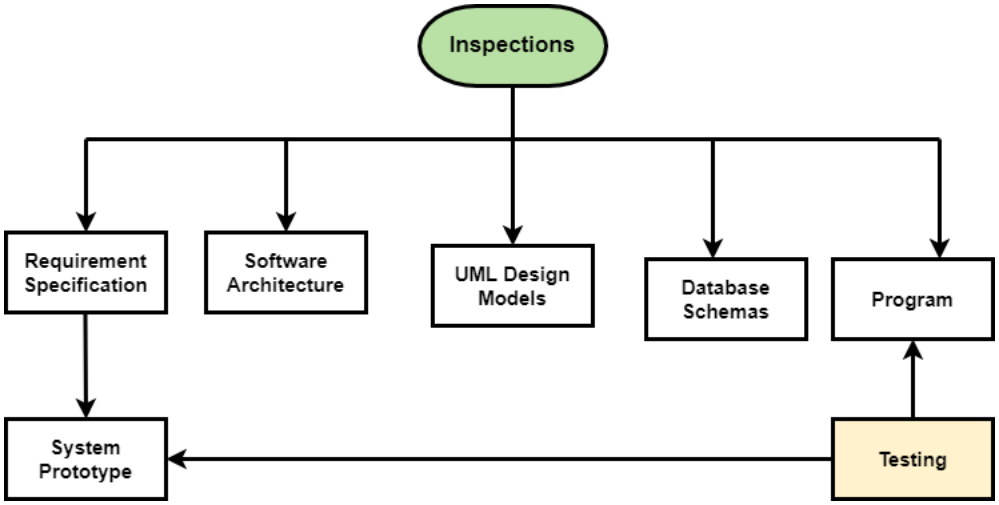


Figure 10.2 Inspection and Testing

The software inspection and testing in different software stages of the software process are shown in Figure 10.2. Software inspections are a static verification performed during the development stage as we discussed in Unit 3-Chapter 9. In contrast software testing is related to the dynamic verification or software during run time. The testing involves the use of test data and operations.

As the inspection is a static process, there cannot be the communication of errors. Whereas in testing the errors may get hidden or masked. Code inspection can be done without additional cost on an incomplete program. Along with the functionality of a system, an inspection can be used for checking coding standards, portability, and maintainability. However, an inspection cannot check the system's performance and usability. The requirements and coding standards can be verified using inspection but the system functioning with the client's real data can be tested. Thus, both testing and inspection are necessary for V & V process.

## 10.2 Testing Process

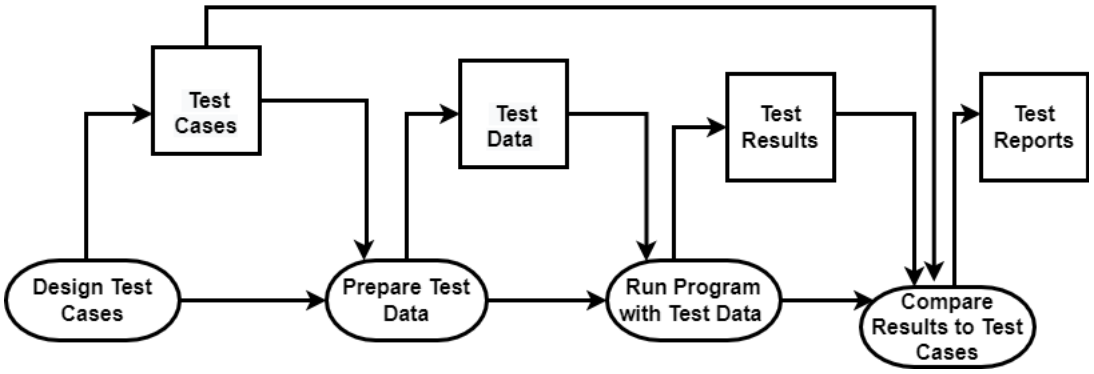


Figure 10.3 A model of the testing process

A traditional testing process is shown in Figure 10.3. The test cases specify the input and output in the system with documentation about it. Test data is the input from the test cases. Test data can be generated automatically from test cases. But the test cases cannot be generated automatically. Test cases are manually prepared by a team of people who are aware of the requirements, design, and coding of software. Test results and reports can be generated using automatic test execution. Thus, testing cases can be manual or automatic. In recent years, automatic testing becomes popular where the tests are encoded in the program to be tested. Automatic testing checks whether the program is working or not and gives correct results. But looks of the output, menu options, graphical interface, etc. must be tested using the manual method. Manual testing also involves the preparation of test cases, obtaining test data, and comparing. Checking of side effects can be carefully observed in manual methods.

The commercial software testing process goes through the following stages.

1. Development testing
2. Release testing
3. User testing

Let us discuss the different testing methods involved in the above types.

## 10.3 Development Testing

System designers and software developers perform development testing to check for errors and defects during system development. This type includes unit, component, and system/integration testing methods. Unit testing and integration testing methods are discussed below.

### 10.3.1 Unit Testing

Unit testing refers to the testing of different modules in a system. Each module is tested separately. The different procedures with different input/output. The global data structures with module access are verified. Unit testing is path testing, and it helps to correct the errors at the early stage. The simplest unit testing is executing every statement of a program. Examples: checking loops, functions working or not in a program; Checking initialization values; verifying incorrect precedence of operators, etc.

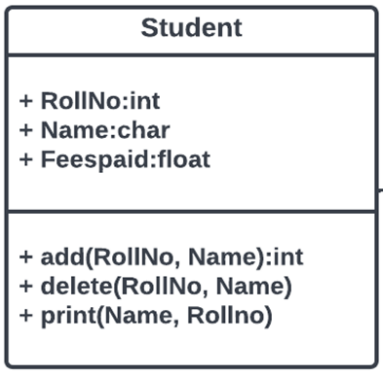


Figure 10.4 The Student Object Interface

Consider, for example, the student object. It has three attributes where RollNo is autogenerated as a student record is created. The Name and Fees paid are the identifiers used for storing variables. The different methods of adding a record, deleting, and printing a record must be verified with the database. If this class is inherited or contained in another class, then the testing must be done with the subclass or derived class. This can be performed in detail using integration testing.

Unit testing can be carried out using automated tests without manual intervention.

A test automation framework can be used to run program tests. In the framework, generic test classes are provided which run the tests and produce test reports using some GUI.

**Selection of test cases:**

It is not possible to perform complete testing of a domain as practically the systems are very huge and infinite. Therefore, the test case design should be of reasonable size and must identify as many errors as possible. The random selection of test cases does not perform effective testing. The test cases should be suitable to check whether the component is doing the job correctly or not. The test cases should reveal the defects if present in the component. The test cases can be chosen based on the following strategies.

**Partition testing**: The inputs with common characteristics are grouped are processed similarly. From each group, the test cases are selected. Input data and output results can be always categorized in different classes where all members of a class are related. Each class can be called an equivalence partition or domain where each program behaves equivalently for each class member. Test cases should be chosen from each partition.
For example, consider a program specification where the inputs are integer values, and the inputs are partitioned based on several digits as given in table 10.1. This information can be used to partition the input and possible test values.

| More than 99 | More than 999 | More than 9999 | More than 99999 |
|---|---|---|---|
| 1 to 2 input values | 3 to 6 input values | Between 7 and 9 input values | 10 to 11 input values |

Table 10.1 Equivalence Partitions

**Guideline-based testing:** In this method, testing guidelines are used based on the previous experience of developers. The testing guidelines provide a set of input values. These inputs may be those which force the system to end up in an error situation. The inputs that cause a buffer overflow or invalid outputs are determined. The computations that take a long time or very less time are validated.

### 10.3.2 The Integration Testing

Different modules are integrated using a software plan. The integration testing checks the steps and sequence by which the different modules form a single system. The integration plan is a stepwise procedure. After every partially integrated system is tested after every step. The structure chart can be used to prepare a plan and it denotes the order by which different modules call each other. If there is a change in the requirements by the client, then it cannot be included in the unit testing. Integration testing covers this checking. The different interfaces of modules with hardware, database, cloud, etc. are verified using integration testing.
Example: Consider online purchases under the category of clothes and electronics on Amazon's online shopping website. The different components that are tested in unit testing are:

- Login page
- Display category
- Order from clothes

- Order from electronics
- Order summary,
- Payment Information
- Branch to bank login, UPI, or credit/debit card page
- The order confirmation

These units must be combined in one system and checked using integration testing.
After completion of unit testing, the units are combined to create components. Component testing performs testing on a set of units and interfaces connecting it.
The integration testing approaches are used to make an integration plan. The commonly used integration testing approaches are top-down and bottom-up integration testing.
In top-down integration testing, the main module with a few subroutines is tested first. They are combined and then tested with program stubs for the components at the bottom. Program stubs are small programs that work as a substitute for long programs. As the lower modules are not available in this approach, the top-level routine testing cannot be efficient. In bottom-up testing, each subsystem is tested separately and then integrated and tested as a whole. There may be many components in each subsystem communication with interfaces. The flow control and data exchange between the different modules is tested while integrating.

Integration testing can be a mixture of top-down and bottom-up approaches and it is known as sandwich/mixed testing. It performs the testing as and when the modules are available.

The simplest form of integration testing is a big-bang technique, where all the modules are combined, and the system is tested. This approach is suitable for low-size programs. Identification and fixing of errors are complicated and time-consuming in this approach.

## 10.4 Release Testing or Functional Testing

In a release, testing is carried out by the development team, and it is aimed to convince the system supplier that it is good enough to use. The non-functional requirements such as performance, dependability, and functionality are checked, and it works normally without failure. Here the tests are based on the system specification. It is also known as functional testing as only the functionality is checked and not the software implementation.
The different forms of release testing are:

### 10.4.1 Requirement-based Testing

This is a validation technique used to demonstrate that the system has met its requirements. Consider in a project Hospital Management System (HMS), a patient has been given a prescription. If the patient is allergic to any of the contents in the medicines, then it should contain information about it. If there are any exceptions such that the allergic condition can be ignored then, the relevant details must be mentioned.

### 10.4.2 Scenario Testing

Different scenarios of the real system working are collected during requirement collection. These test cases are used to check the functionality of the system. The scenarios can be reused multiple times.

### 10.4.3 Performance Testing

The performance testing shows the developed system's profile. This method is based on the non-functional requirements, given in the requirement specification document. The emergent properties such as reliability and performance are tested. Performance testing can be conducted using several types. Some of the performance testing methods are as given below. These methods can be performed based on how the functions are performed in the system.

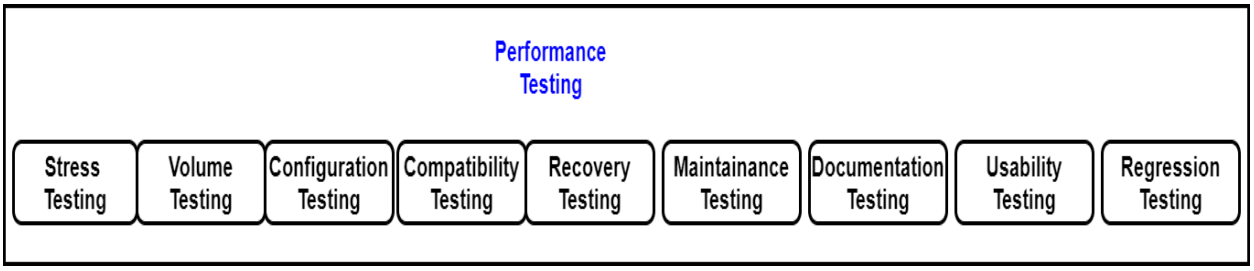


Figure 10.5 Types of Performance Testing

1. **Stress testing**: It begins with a light load of test cases and as the test progresses, the load is increased slowly. The system is also checked for extreme load or over-loaded. This type of performance testing is known as stress testing where the load is beyond the limits on purpose to check the failure behavior. Stress testing can help identify whether there is any data corruption or loss of services due to

overload. During testing, some of the defects which are not identified during normal situations can be identified.

2. **Volume testing**: The data structures in the program are tested for handling large volumes of data. For example, arrays, stacks, and queues are tested for overflow or underflow conditions.
3. **Configuration testing**: The combination of hardware and software configuration is verified in this testing method. The suitability of different devices such as printers, scanners, servers, etc. is tested. Whether the developed software performs well in the given environment is validated. For example, configuration testing can be performed on software working on several clients and servers located at different locations.
4. **Compatibility testing**: Compatibility testing is used to test the software compatibility with various hardware, browsers, mobile devices, databases, networks, operating systems, etc. This testing aims at checking the interfaces used in communication with different systems. The time and volume of data retrieval are determined in this type of testing. Compatibility can be also verified in older and new versions of the software.
5. **Recovery testing**: This testing is performed to test how the system responds to the faults, loss of services such as power failure, interrupted internet connection, data corruption, etc. Whether the system can recover from such accidents is tested in recovery testing. Disconnecting the Wi-Fi router, printer, or power shutdown may be done to check the data loss and error in the software.
6. **Maintenance testing:** This testing is performed on the already installed system. For example, in bank software, the system must work 24 X 7 for all days of the year. Here maintenance of the software is crucial, and it should be done frequently. The diagnostic tools and software programs are used to maintain the software. The addition of new features, versions, etc requires this type of testing.
7. **Documentation testing**: This testing is used to check the necessary user guide and manual and technical details of the system. The accuracy and consistency of the manual and the type of users for whom the manual is designed are verified here.
8. **Usability testing**: In this testing method, the user interface is tested. Whether all the user requirements are fulfilled is verified. For example, if the software is

developed for a supermarket, then the front end, ability of the user to order the items, processing of a bill, etc. is tested

9. **Regression Testing**: Regression testing refers to testing when an existing system is changed or upgraded. This testing method checks if there are any new bugs after the addition of new features or are there any performance issues. In this testing method, it may not be necessary to run the testing for the entire code, but only those test cases that can affect the functions with change can be run. For example, consider a bookshop management system, where there are features such as add (), delete (), and print () operations for book records up to 1000. Suppose this system is enhanced with several records up to 5000 and operations append () and update () are inserted. Then the test cases required only for testing these two features are designed and regression testing is performed. The test cases in regression testing are based on the following points.

    - Test cases which have defects more often
    - Important features of the software product
    - Modified features in recent time
    - Test cases which include many connecting modules
    - Test data checking boundary cases
    - Success and failure test cases for the system

## 10.5 System Testing

System testing, also known as user testing, is done by the users by entering input and suggestions on system testing. User testing is performed after the performance and release testing. The user environment can help test the usability and reliability of the developed system. For example, in a hospital management system, a developer can't prepare clinical environments such as patients, nurses, doctors, emergencies, etc. In such situations where the live data is necessary, user testing is done. User testing can be performed informally or from a formal external supplier. There are three types of user testing.

### 10.5.1 Alpha Testing

In this testing, the software development team and users work together to test the developed software at the developer's site. Alpha testing is a form of acceptance testing

where the defects and issues are identified before the release of the final product. This testing is known as alpha as it is done in the early stage of software development.

Example: Google released the newly developed software component to their high-level developers to test the software.

### 10.5.2 Beta Testing

In beta testing, users can experiment with the software and come up with issues (if any) for the developer. Here the developed system is tested with real data and a real environment. A Beta version of the software is given to limited users of the system and it helps in improving the quality of the system. With the help of customer validation, software failures can be reduced. where a release of the software is made available to users. Users can experiment and raise problems that they discover with the system developers.

Example: Techno-Center is a company that develops an information management system for colleges and universities to manage all the academic activities. The activities include details of a student related to admission, exam, library, hostels, laboratory, attendance, etc. The company first releases the software to a few small colleges and gets feedback. Based on the feedback, some minor defects are corrected and then released in the customer market.

### 10.5.3 Acceptance Testing

In acceptance testing, the end-users test the system and decide whether the system is ready for deployment. Customers give the decision to the system developers. After acceptance testing, the payment can be done to the developers. Acceptance testing consists of six steps as given below.

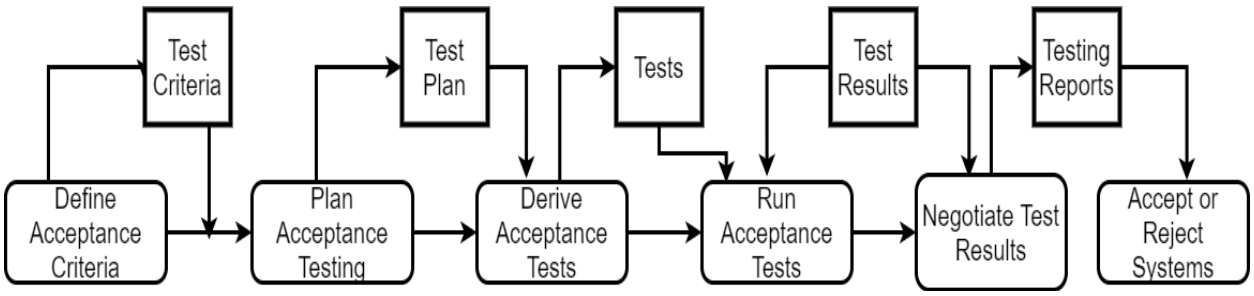


Figure 10.5 The process of acceptance testing

a) Define acceptance criteria: A system contract between clients and developers is signed in this stage. Here the requirements are in the preliminary stage and may get revised in future stages.
b) Plan acceptance testing: In this stage, an acceptance plan with budget, resources, and time is prepared. The requirements coverage and identifying various risks in the software are identified in this stage.
c) Derive acceptance tests: The tests are derived after completing the acceptance plan. In this stage, both the functional and non-functional characteristics are identified.
d) Run acceptance tests: The approved acceptance plan is executed in the actual client environment. The agreed acceptance tests are executed in the user environment. Here end-users are given some training. The interaction between users and the system is established.
e) Negotiate test results: In this stage, the client and the developer negotiate on acceptance of the developed software
f) Reject/accept system: In this stage, the developer and customer decide whether the system should be accepted or rejected. If the system is not developed as per expectation, then further development is necessary to fix the defects. Once it is done, the system is accepted.

**Levels of Testing:**

In summary testing methods can be classified based on where they are added in the steps of the software development life cycle. The four general levels of testing are:

1. Unit testing: Happens in the coding and development phase. Tests the smallest element of the software. E.g.: module
2. Integration testing: This happens in the testing phase. Works by combining units in a whole system. E.g.: Checking whether the modules are connected correctly.
3. System testing: This testing is classified based on who is performing the testing. In this testing, one category is alpha and beta testing performed by experts and the second category is acceptance testing done by the end-user. System testing also includes testing techniques based on non-functional characteristics such as volume stress, compatibility, configuration, etc. e.g. A website getting tested for the number of users or number of downloads etc.
4. Regression testing: Testing is performed in the maintenance stage. E.g., upgrade available on mobile android versions or updates on what's app or adobe, etc.

There are different levels of testing in the requirement, design, development, testing, and deployment of a system. A comprehensive testing list is given in Figure 10.6.

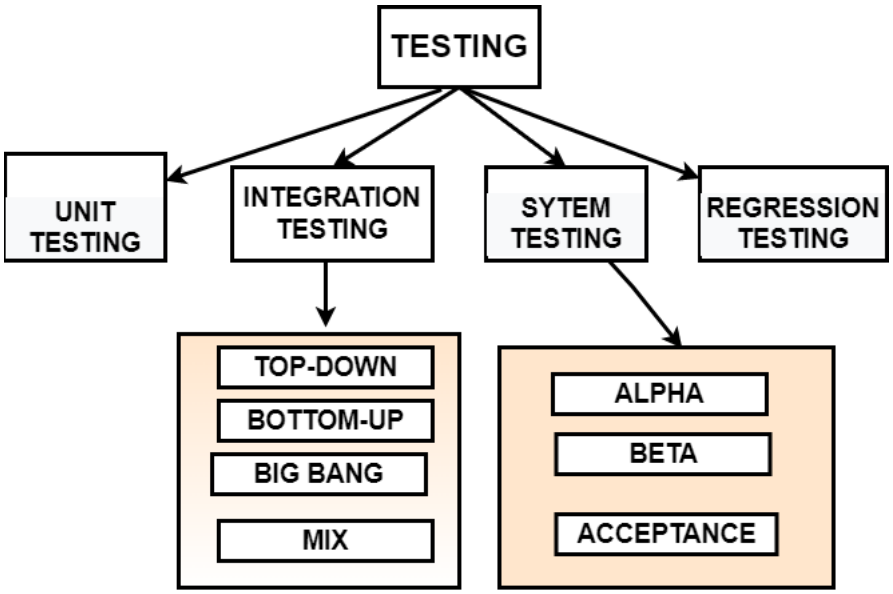


Figure 10.6 The levels of Testing

**Structural and functional testing**

Testing methods are also classified based on the design of software and the functionality of the software. Structural testing refers to the testing of the code and design by the software developers. It is based on the specifications of the code, and input/output constraints. The different data structures, hardware, and features of the software such as front end, menus, and low-level components are tested in structural testing. The logical errors and syntax errors are identified in structural testing.

Functional Testing refers to testing whether the system meets the predetermined requirements. The processing of information in different functions is verified using test cases in functional testing. Along with the functional requirements, the functional testing methods address quality attributes such as reliability, maintainability, security, etc.

Functional testing concentrates on user acceptance and the performance of the system. Structural testing is known as white box testing or clear box testing. The functionalities of the system are tested using a popular testing method known as black-box testing. These methods are based on the design of the test cases. The test cases are designed either for the program structure or for the functions of the system to be validated. These testing methods are given in detail in chapter 11.

## 10.7 Multiple Choice Questions (MCQs)

1. The functional requirement of a system can be given by:
    a. Availability
    b. Efficiency
    c. Portability
    d. None of the above
2. The incorrect testing method out of the following is:
    a. Combination Testing
    b. System testing
    c. User testing
    d. Functional Testing
3. Alpha testing is performed at
    a. Client's end
    b. Developer's end
    c. Both a and b
    d. None of the above
4. Integration testing is used for
    a. Detecting errors in modules
    b. Detecting syntax errors
    c. Detecting interface errors
    d. None of the above
5. System testing deals with only functional testing
    a. True
    b. False
6. Static testing can be used to detect
    a. Defect
    b. Failure
7. When all the testing is done in one step of Integration testing, then it is known as
    a. Top-down integration testing
    b. Bottom-up integration testing
    c. Big bang integration testing
    d. Mixed integration testing

8. Integration testing plan can be presented using
    a. Structure chart
    b. Flow chart
    c. Data flow diagram
    d. None of the above
9. When the system is fixed for bugs during its upgraded version, then the testing is known as
    a. Integration testing
    b. Functional testing
    c. Non-functional testing
    d. Regression testing
10. A testing performed before handling the software to the client is known as:
    a. Beta testing
    b. Alpha testing
    c. Gamma testing
    d. Delivery testing

## 10.8 Review Questions

1) Differentiate between verification and validation with a suitable example
2) Explain integration testing and its different types.
3) Explain the need of performing regression testing.
4) Explain different levels of testing with a simple example
5) What is the difference between acceptance and system testing? Justify your answer with a suitable example.
6) What are alpha and beta testing? Where is it used?
7) Differentiate between top-down and bottom-up testing methods. Discuss the test cases where these two methods are used.
8) Compare unit testing and functional testing.
9) Discuss the advantages and drawbacks of program inspection over-testing.

10) Discuss the importance of unit testing by taking a specific scenario in web-based application software.

## 10.9 MCQ solution keys

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1: d | 2: a | 3: b | 4: c | 5: b | 6:a | 7:c | 8:a | 9:d | 10:b |