Outcomes & Introduction

Unit 4 Outcomes

- Explain the software testing process.
- Compare verification and validation.
- Understand the importance of software inspection.
- Analyze testing methods: unit, system, integration, performance, acceptance, regression, alpha, and beta testing.

Unit 4 Outcomes

- Differentiate between development, release, and user testing.
- Compare white box and black box testing.
- Explain test stubs and drivers, test cases, and test suites.
- Illustrate static testing strategies: code review, code walkthrough, and automated testing methods.

Testing

- Software testing is a technique to check whether the developed software product does the required function without any defects.
- During testing, the program is executed using sample or artificial data known as test cases.
- Different errors, irregularities, defects, etc. are identified.
- Testing can be manual or automated.

Testing

- Testing is useful in checking whether software fulfills the functional and non-functional requirements or not.
- A common picture of testing is that all untested code contains some defects.
- The different terms in testing are bugs, defects, errors, faults, and failures.

Need of Testing

- Nissan cars had software failure which led to accidents and 1 million cars were recalled from the market.
- Due to software defects some of the retailers on Amazon observed that their prices is reduced, and they were in heavy loss.
- Windows 10 was weak in security due to a software bug.
- China Airlines' fighter plane F-35 was with a software bug, and it crashed leading to the loss of more than 200 people.

Testing Objectives

Bugs, Defects, Errors

- The defect in the software is informally known as a bug. 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. Defects can be major, critical, or minor.
- When there is a mistake in a code then it is called an error. Errors may be the syntax, semantics, hardware, testing errors, etc.

Faults, and Failures

- The 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.

Testing Objective - 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 there should be at least a single test case.
- This is **validation testing:** a process of finding errors by executing the program in a real environment.

Testing Objective - 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.
- This is also known as verification: The process of checking that the software meets specifications.

Verification vs. Validation

Verification

- In verification, the software requirements specification, design, and code are checked. It may not include program execution.
- Code inspections, reviews, or code walkthroughs are performed in this stage. The verification is performed by the system analysts and designers.
- High-level design and database design are tested in verification in the early stages of software development.

Validation

- 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.

Verification and Validation

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

Verification and Validation

- 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. The level of confidence is more important in the safety alarm system than in the typesetting project.

Verification and Validation

- The expectation from the users is yet another issue in building the system's confidence.
- Along with software testing, the verification and validation process may involve software inspections and reviews.
- The requirement specifications, design models, code, and the proposed tests are checked in the inspection.
- This is known as V & V technique

Software Inspection

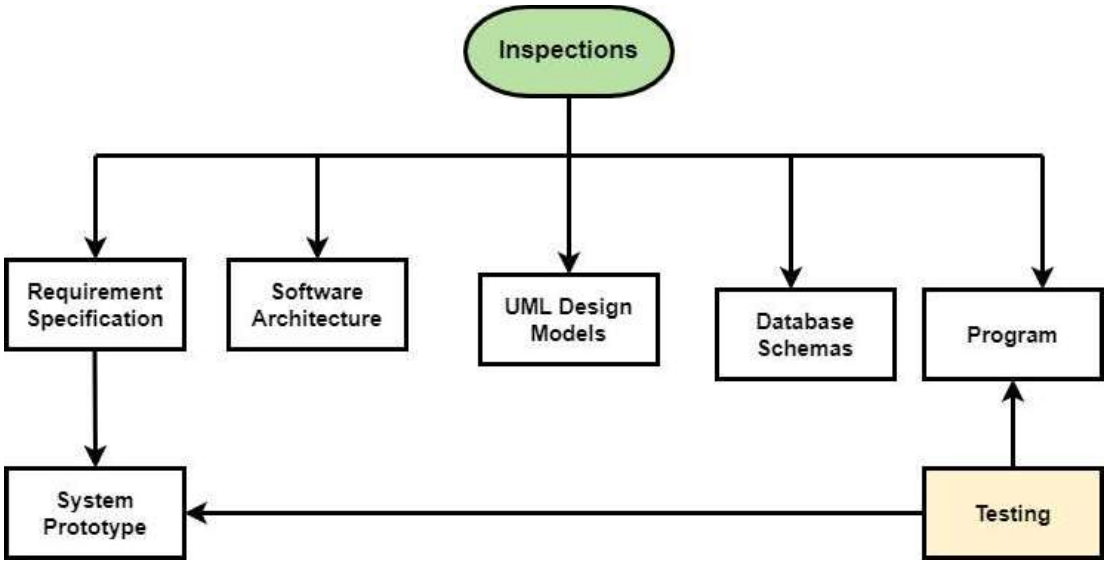
Software Inspection

- As the inspection is a process, to discover problems in the static system representation. (Verification)
- 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.

Software Inspection

- The requirements and coding standards can be verified using the inspection process.
- Testing is used for the system functioning with the client's real data.
- Thus, both testing and inspection are necessary for V & V process.

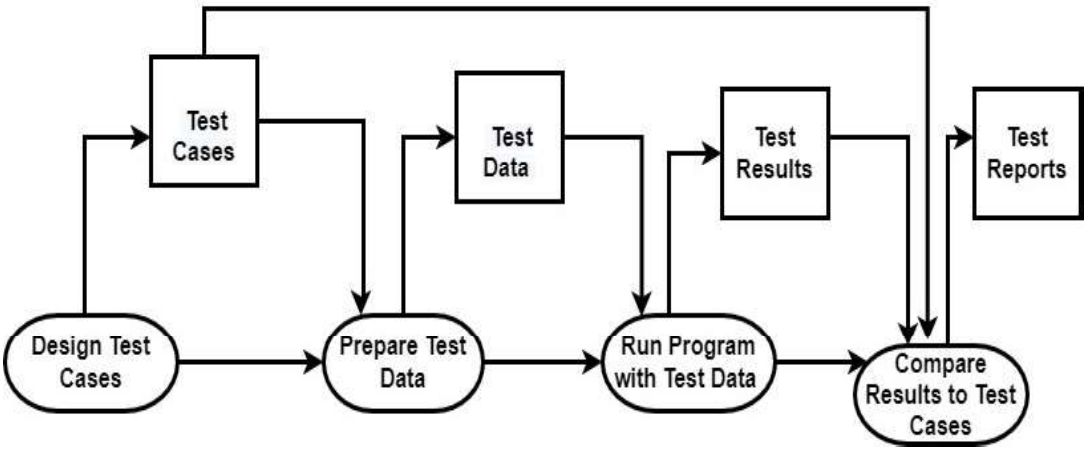
Software Inspection



Unit 4 – Testing

Software Testing Process

Software Testing Process



Software Testing Process

- 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.

Software Testing Process

- Test results and reports can be generated by automatic method and manually.
- The tests are encoded in the program to be tested.
- 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 and test data. Checking of side effects can be carefully observed in manual methods.

Software Testing Process

- The commercial software testing process goes through the following stages.

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

Unit Testing

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 integration testing methods.

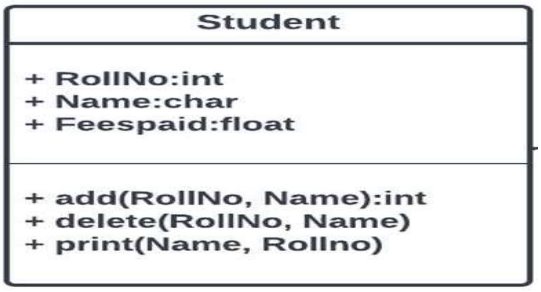
Unit Testing

- Unit testing refers to the testing of smaller pieces of larger programs. For e.g., module.
- Each module is tested separately before combining.
- The different procedures with different input/output are tested.
- The global data structures with module access are verified.

Unit Testing

- Unit testing 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.

Unit Testing



- Data structures: RollNo, Name, and Fees paid
- Methods: of adding a record, deleting, and printing a record must be verified with the database.

Equivalence Partitions in Unit Testing

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

- Partition testing: The inputs with common characteristics are grouped are processed similarly.
- 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.

Guideline-based Unit Testing

- In this method, testing guidelines are used based on the previous experience of developers.
- The testing guidelines provide a set of input values.
- The inputs which may 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.

Integration Testing

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.
- A partially integrated system is tested after every step.

Integration Testing

- The structure chart can be used to prepare a plan and it denotes the order by which different modules call each other.
- Integration testing verifies for any errors if the unit testing phase adds certain new requirements.
- The different interfaces of modules with the front end, hardware, database, cloud, etc. are verified using integration testing.

Integration Testing

- Integration testing is classified as follows:
- Top-down
- Bottom-up
- Mixed
- Big-Bang integration testing.

Top-down Integration Testing

- In top-down integration testing, the main module or higher modules are tested first.
- Lower modules are tested later. If they are not ready, then program stubs are used as dummy components.
- Program stubs are small programs that work as a substitute for long programs.

Bottom-up Integration Testing

- In bottom-up testing, lower modules are tested first.
- Lower modules are then integrated and tested as a whole.
- Lower modules are submodules and higher modules are main modules.
- If higher modules are not ready, then drivers are used as dummy modules.

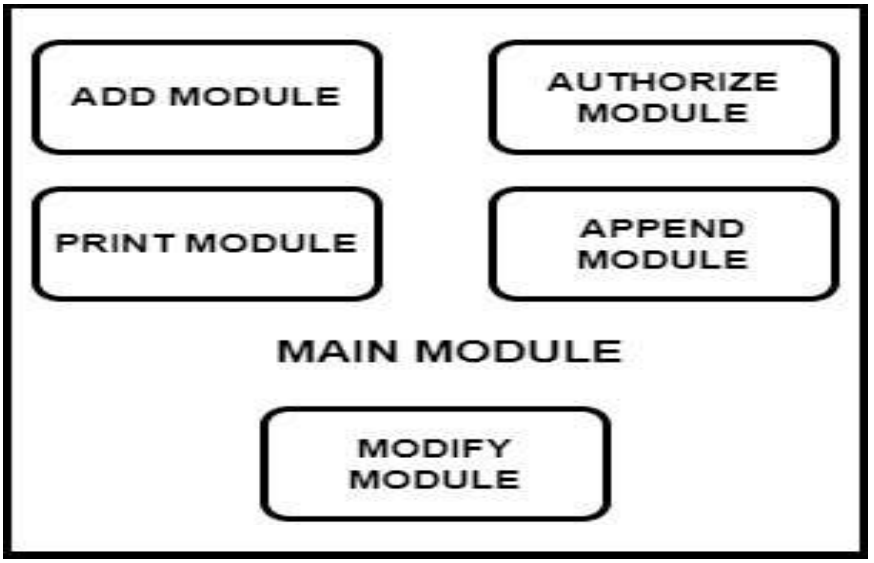
Mixed Integration Testing

- Integration testing can be a mixture of top-down and bottom-up approaches and it is known as a sandwich/mixed testing.
- It performs the testing as and when the modules are available.
- It is used in large systems with several modules.
- It is expensive because of the combination of top-down and bottom-up.
- It is not suitable if there is high coupling.

Big-bang Integration Testing

- The simplest form of integration testing is a big-bang technique, where all the modules are simply put together, and the system is tested.
- Big-bang is useful in small systems.
- Error handling is very difficult in the Big-bang method.
- High-risk modules are not given more priority.
- Due many modules getting tested, it takes more time.

Integration Testing



Release Testing

Release Testing

- Release testing is also known as functional testing.
- Here, 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 tests are based on the system specification.
- Goal: To verify the release on production/LIVE environments.

Release Testing

- Only the functionality is checked and not the software implementation.
- The different forms of release testing are:
    1. Requirement-based testing
    2. Scenario-based testing
    3. Performance-based testing
    4. Regression testing

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.

Scenario Testing

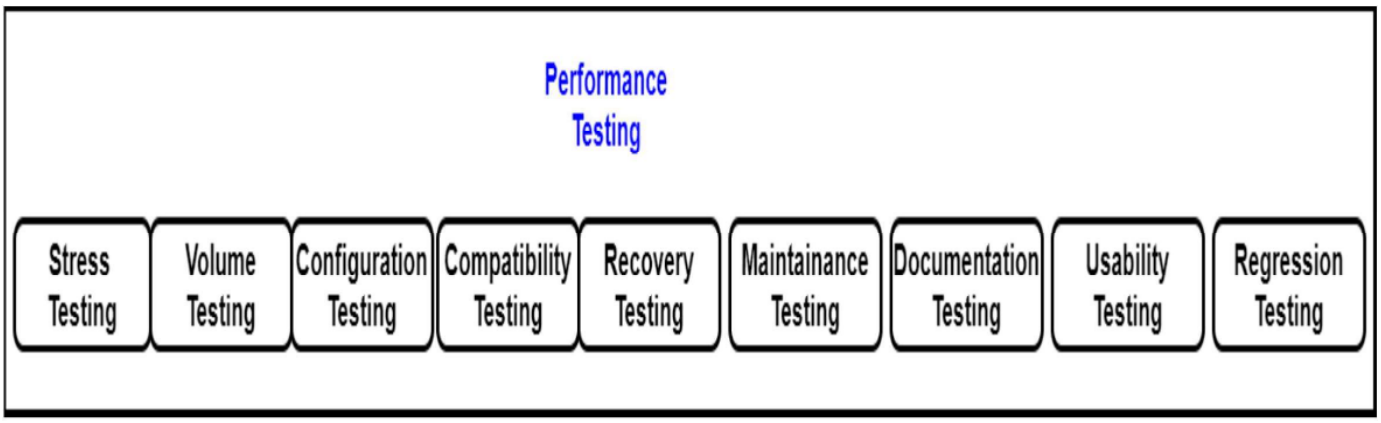
- Different scenarios of the real system working are collected during requirement collection.
- These test cases are used to check the functionality of the system.
- Test scenarios are a copy of the end-user usage.
- The scenarios can be reused multiple times.

Performance Testing – Part 1

Performance Testing

- The performance testing shows how the developed system performs under a given workload.
- The reliability of a developed software is tested.
- This method mainly concentrates on speed, stability, and scalability.
  - Speed – How is the application response, is it quick enough?
  - Scalability – How much maximum load a system can handle?
  - Stability – Is the system stable under dynamic load?

Performance Testing



Stress testing

- Stress testing 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 overloaded.
- The load is beyond the limits on purpose to check the failure behavior.
- It can help identify whether there is any data corruption or loss of services due to overload.
- Some of the defects which are not identified during normal situations can be identified.

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
- If the database is used for storing data and the software is given data access. The performance of the software is checked by varying the data.

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 on software working on several clients and servers located at different locations.

Unit 4 – Testing

Performance Testing – Part 2

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.

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

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.

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.

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

Regression Testing

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.

Regression Testing

- 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 upgraded.
- Then the test cases required only for testing these two features are designed and regression testing is performed.

Regression Testing: Test Cases

- 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

System Testing

System Testing

- System testing is also known as user testing
- It is done by the users by entering input and suggestions on system testing.
- User testing is performed after the performance and release testing.

System 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, system testing is done. System testing can be performed informally or from a formal external supplier.

Alpha Testing

- 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 releases the newly developed software component to their high-level developers to test the software.

Beta Testing

- In beta testing, users can experiment with the software and produce issues (if any) for the developer.
- 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.
- Users can experiment and raise problems that they discover with the system developers.

Beta Testing

- Example: Techno-Center is a company that develops an information management system for colleges and universities to manage all 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.

Acceptance Testing

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.
- It is the last phase of software testing.

Acceptance Testing

- Define acceptance criteria: A system contract between clients and developers is signed. Here the requirements are in the preliminary stage and may get revised in future stages.
- Plan acceptance testing: An acceptance plan with budget, resources, and time is prepared. The requirements coverage and identifying various risks in the software are identified.

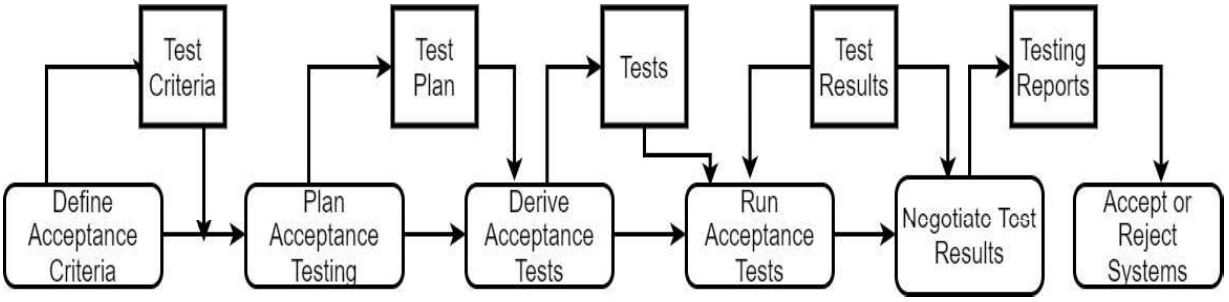
Acceptance Testing

- **Derive acceptance tests:** The tests are derived after completing the acceptance plan. In this stage, both the functional and non-functional characteristics are identified.
- **Run acceptance tests:** The approved acceptance plan is executed in the actual client environment. The agreed acceptance tests are executed in the user's environment. Here end-users are given some training. The interaction between users and the system is established.

Acceptance Testing

- **Negotiate test results:** In this stage, the client and the developer negotiate the acceptance of the developed software
- **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 expectations, then further development is necessary to fix the defects. Once it is done, the system is accepted.

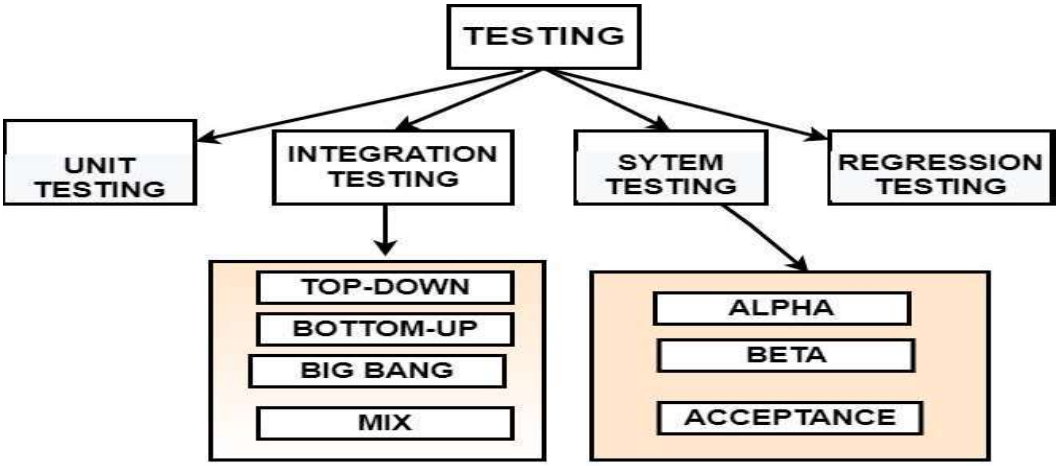
Acceptance Testing Process



Unit 4 – Testing

Levels of Testing

Levels of Testing



Levels of Testing

- **Unit testing:** Happens in the coding and development phase. Tests the smallest element of the software. E.g., module
- **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.

Levels of Testing

- **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.

Levels of Testing

- **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.

Structural and Functional Testing

- Testing methods are also classified based on the design of software and the functionality of the software.
- 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 for the program structure and for the functions of the system to be validated.

Structural Testing

- Structural testing refers to the testing of the code and design by the software developers. Specifications of the code, input/output constraints.
- E.g., Data structures, hardware, front end, menus, low-level components, etc.
- The logical and syntax errors are identified in structural testing.

Functional 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.**