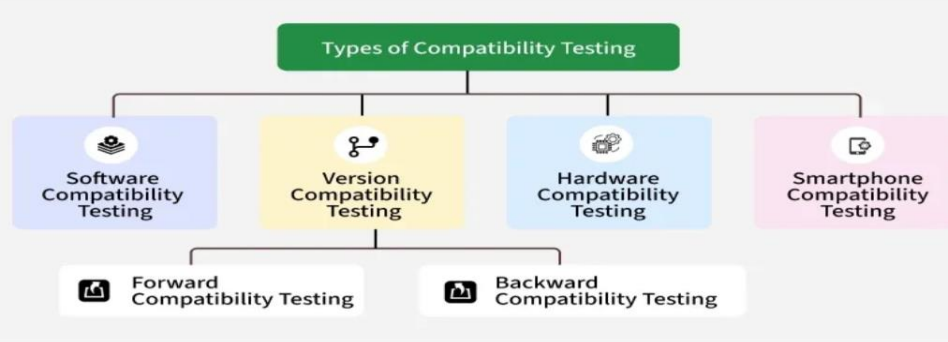
UNIT II – LEVELS OF TESTING

Black Box Testing: Requirements based Testing – Positive and Negative Testing – Boundary Value Analysis – Decision Tables – State Based Testing – Compatibility Testing – User Documentation Testing – Domain Testing. Levels of Testing: System and Acceptance Testing.

Compatibility Testing

Compatibility testing is a type of non-functional testing. It checks whether a software application works properly on different operating systems, browsers, hardware devices, networks, and platforms. This testing is usually done after the application becomes stable.

The main purpose of compatibility testing is to ensure that the software runs correctly in different environments. It helps find compatibility problems before the software is released, improving the user experience and reducing future issues.



Types of Compatibility Testing

Software Compatibility Testing

Software compatibility testing checks whether an application works properly on different software and hardware environments.

Examples:

- Checking the application on different operating systems such as Windows, Linux, and macOS.
- Checking compatibility with different databases such as Oracle, SQL Server, and MongoDB.
- Checking whether the application works on different devices such as mobile phones, tablets, and computers.

Types of Compatibility Testing Based on Version

1. Forward Compatibility Testing

- Checks whether the software works correctly with newer versions of software or hardware.
- Example: Testing an application developed for Windows 10 on Windows 11.

2. Backward Compatibility Testing

- Checks whether the software works correctly with older versions of software or hardware.
- Example: Testing a new version of an application on Windows 8 or an older browser.

Hardware

Checking compatibility with a particular size of

- RAM
- ROM
- Hard Disk
- Memory Cards
- Processor
- Graphics Card

Smartphones

It is used to verify the compatibility of the software against the multiple mobile platforms namely, blackberry, iOS, Android etc.

Network

It is used to verify the compatibility of the software against the different bandwidths, speed, and capacity.

Browser Compatibility

It is used to verify the compatibility of the software against the multiple browsers, and its various versions.

Why is Compatibility Testing Important?

- It helps ensure that users are satisfied by making the application work properly in different environments.
- It allows the software to run smoothly on various platforms, devices, and operating systems.
- It helps detect compatibility issues and bugs early during the software development process.

Compatibility Testing Defects

Some common defects found during compatibility testing are:

- Differences in the **user interface** on different platforms or devices.
- Changes in **font size** that affect readability.
- **Alignment issues** where text or images are not properly positioned.
- **Broken frames or layouts** that do not display correctly.

- **Overlapping content**, where text or images cover each other.

Examples:

- Changes in the **graphical user interface (GUI)** on different devices or browsers.
- Differences in **font size**, making the text difficult to read.
- **Text alignment problems** on web pages.
- **Overlapping text or content** on the screen.
- **Broken frames or page layouts** that do not display correctly.

2.7 User Documentation Testing

Testing Documentation

Testing documentation is an important part of the **Software Testing Life Cycle (STLC)**. It helps in planning, performing, and recording all testing activities during software development.

- **Defines the Testing Approach:** It explains **what to test, how to test, and when to test** for organized testing.
- **Supports Release Decisions:** It provides details about **test results, defects, and test coverage** to help decide whether the software is ready for release.

Types of Test Documentation

1. Test Scenario

A **test scenario** gives a high-level overview of what needs to be tested. It describes different possible user actions or application flows but does not include detailed test steps or test data.

2. Test Case

A **test case** is a detailed document that includes test inputs, execution steps, expected results, and actual results. It is created based on a test scenario.

3. Test Plan

A **test plan** describes the overall testing process. It includes the project scope, testing objectives, resources, schedule, testing strategy, and responsibilities.

4. Requirement Traceability Matrix (RTM)

A **Requirement Traceability Matrix (RTM)** is a document that links each requirement to one or more test cases. It ensures that all requirements are covered during testing.

5. Test Strategy

A **test strategy** defines the overall testing approach for the project. It includes testing types, levels, scope, and methods. Once approved, it is generally not changed.

6. Bug Report

A **bug report** records the details of defects found during testing. It includes the defect description, status, severity, and resolution, helping testers and developers track and fix bugs.

7. Test Execution Report

A **test execution report** is prepared after testing is completed. It summarizes the testing results, including the number of test cases executed, passed, failed, skipped, and the total defects identified.

Advantages of Test Documentation

- **Removes Confusion:** It clearly explains the testing process and responsibilities, reducing doubts and confusion.
- **Provides a Standard Process:** It establishes a consistent approach for testing and serves as a reference for new team members.
- **Supports Product Demonstration:** It helps the marketing team show that the software has been thoroughly tested.
- **Ensures Quality Delivery:** It helps deliver a high-quality software product within the planned schedule.
- **Provides User Support:** It includes manuals and user guides to help users install, configure, and use the software.
- **Improves Customer Communication:** It maintains clear and transparent communication between the development team and customers.

Disadvantages of Test Documentation

- **Time-Consuming:** Updating test documents takes extra time whenever the software requirements change.
- **Needs Skilled Effort:** Test documentation is often assigned to less experienced team members, which may affect its quality.
- **Increases Project Cost:** Preparing and maintaining test documents requires time and resources, increasing the overall cost.
- **Poor Documentation Causes Problems:** Incomplete or incorrect documentation can lead to misunderstandings between the customer and the project team, affecting product quality.

2.8 Domain Testing

Domain testing is a **black-box testing technique** in which a minimum number of carefully selected input values are used to verify that the software correctly accepts **valid inputs** and rejects **invalid inputs**. It focuses on checking the boundaries and valid ranges (domains) of input values.

Key Points:

- Uses the **least number of test inputs** to achieve effective testing.

- Verifies that the software accepts only **valid input values**.
- Ensures that **invalid or improper inputs are rejected**.
- Checks whether the software produces the **correct output** for valid inputs.
- Helps identify errors related to **input validation** and **boundary conditions**.

Example:
If a system accepts marks between **0 and 100**:

- Valid inputs: **0, 50, 100** → Accepted.
- Invalid inputs: **-1, 101** → Rejected with an appropriate error message.

Domain testing ensures that the software behaves correctly within the specified input domain and does not process invalid data.

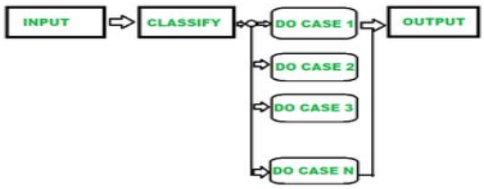
Importance of Domain Testing

- **Checks the complete input range** to ensure the software handles all valid and invalid inputs correctly.
- **Finds errors in specific input ranges** that other testing methods may miss.
- **Helps developers understand input limits**, reducing defects during development.
- **Improves testing quality** by focusing on important input values and finding more bugs effectively.

Structure (Steps) of Domain Testing – Simple

1. **Determine the Domain:** Identify the valid input range and requirements of the software.
2. **Split the Domain:** Divide the input range into smaller groups (partitions) of similar values.
3. **Choose Test Cases:** Select representative test cases from each partition.
4. **Design Test Data:** Prepare valid, invalid, and boundary input values for testing.
5. **Run Test Cases:** Execute the test cases and compare the actual results with the expected results.
6. **Boundary Value Analysis:** Test the minimum, maximum, and boundary values of each input range.
7. **Check Error Handling:** Verify that the software displays proper error messages and handles invalid inputs without crashing.

8. **Automate Testing (Optional):** Use automation tools for repetitive or complex test cases to save time and improve coverage.



Domain Testing

Domain Knowledge

Domain knowledge is a good understanding of a particular business area or field. It includes knowing the rules, processes, requirements, and terminology of that domain. Domain knowledge helps in developing high-quality software, reducing development time, improving efficiency, and providing better customer service.

Domain knowledge is essential for domain testing because it helps testers understand the application, create accurate test cases, identify defects, and ensure that the software meets business requirements.

Examples:

- **Online Banking:** A tester should understand banking activities such as login, fund transfer, and bill payment.
- **Retail:** A tester should know retail processes like warehouse management, inventory, and billing.
- **Healthcare:** A tester should understand healthcare workflows to ensure the system works correctly, as errors can affect patient safety

Skills Required for Domain Testing

- **Domain Knowledge:** Understand the business area of the software.
- **Analytical Skills:** Analyze requirements and identify test scenarios.
- **Testing Knowledge:** Know techniques like Boundary Value Analysis and Equivalence Partitioning.
- **Communication Skills:** Communicate clearly with developers and team members.
- **Test Design:** Create and execute effective test cases.
- **Attention to Detail:** Identify small errors and defects.
- **Teamwork:** Work well with developers, business analysts, and other team members.
- **Problem-Solving Skills:** Find and resolve testing issues efficiently.

Domain testing requires selecting the correct input values to verify that the software produces the expected output. It mainly focuses on testing input ranges using techniques such as **Boundary Value Analysis (BVA)** and **Equivalence Partitioning (EP)**.

Advantages of Domain Testing

- **Makes efficient use of testing resources** by focusing on important input values.
- **Provides better test coverage** with fewer test cases.
- **Detects errors in boundary and critical input ranges**, improving software quality.
- **Reduces testing cost** by identifying defects early and minimizing unnecessary testing.

Disadvantages of Domain Testing

- **Finding the correct input boundaries** can be difficult.
- **Not suitable for testing the entire functionality** of complex software.
- **Cannot cover all possible test scenarios**, so it should be used with other testing techniques.