Unit-2: Methods of Testing

2.1. Software Verification and Validation:

The first part may be termed 'verification of the process used for developing a product while the second part may be called 'validation of a product which is created by testers and given to customers for acceptance testing/using it in production.

Quality planning includes planning for the processes used to build and test software to create a right product.

It may include software certification which may be termed 'validation', as well as finding conformance to the requirements defined by the processes and standards called 'verification'.

Quality is not an accident but an outcome of planned activities carried out in various phases of Software Development Life Cycle (SDLC).

Verification:

- Verification is a disciplined approach to evaluate whether a software product fulfils the requirements or conditions imposed on them by the standards or processes.
- It is done to ensure that the processes and procedures defined by the customer and/or organisation for development and testing are followed correctly.
- **Validation:**
- Validation is a disciplined approach to evaluate whether the final built software product fulfils its specific intended use.
- It is meant to validate the requirements as defined in requirement specification, ensure that the application as developed matches with the requirements defined, and is fit for the intended use.
- Real-life scenarios are achieved in test lab to perform validation of the application under testing.
- Validation steps may involve test bed preparation, test scenario definitions, test case definitions, test data definitions, test case execution, defect identification, retesting, and regression testing.

2.2. Simple difference between Black box testing and White box testing:

	Black Box Testing	White Box Testing
Access to information	Technical documentation only; no access to the code or information on the architecture of a tested product	Full access to the code and architecture of a tested product
Tested object	Software behavior and functionality (how a product under test works)	Software behavior and backend logic (how a product under test is built)
Test performers	QA engineers	Developers (mainly)
Required information	It is not necessary to know backend logic	Access to software code is necessary for running some tests
Testing type	External, functional	Internal, structural
Test basis	Business and functional requirements	Functional and technical requirements
Skills required	Testing practices	Knowledge of programming languages and coding
Testing level	Higher levels (system and acceptance testing)	Lower levels (unit and integration testing)
Testing reports	Defects in features and performance	Both product and code defects

2.3. Static and Dynamic testing:

Two other terms used to describe how software is tested are static testing and dynamic testing.

Static testing refers to testing something that's not running—examining and reviewing it.

Dynamic testing is what you would normally think of as testing— running and using the software.

Static Testing	Dynamic Testing
1. Static Testing is white box testing which is done at early stage of development life cycle. It is more cost effective than dynamic testing	1. Dynamic Testing on the other hand is done at the later stage of development lifecycle.
2. Static Testing Methods include Walkthroughs, code review.	2. Dynamic testing involves functional and nonfunctional testing
3. It is done before code deployment and without execution of code.	3. It is done after code deployment with the execution of codes.
4. This type of testing comes under Verification.	4. This type of testing comes under Validation.
5. Static testing achieves 100% statement coverage in a relatively short time.	5. Dynamic testing may involve running several test cases, each of which may take longer time.
6. Static testing is about prevention.	6. Dynamic testing is about cure.
7. In Static Testing techniques a checklist is prepared for testing process.	7. In Dynamic Testing technique the test cases are executed.

2.4. Black-Box Testing Techniques-Equivalence Partitioning:

Selecting test cases is the single most important task that software testers do and equivalence partitioning, sometimes called equivalence classing, is the means by which they do it.

Equivalence partitioning is the process of methodically reducing the huge (infinite) set of possible test cases into a much smaller, but still equally effective, set.

For example, without knowing anything more about equivalence partitioning, would you think that if you tested 1+1, 1+2, 1+3, and 1+4 that you'd need to test 1+5 and 1+6? Do you think you could safely assume that they'd work?

How about 1+99999999999999999999999999999999 (the maximum number you can type in)?

Simple definition:

An equivalence class or equivalence partition is a set of test cases that tests the same thing or reveals the same bug.

2.5. Data Testing:

The simplest view of software is to divide its world into two parts: the data (or its domain) and the program.

The data is the keyboard input, mouse clicks, disk files, printouts, and so on. The program is the executable flow, transitions, logic, and computations.

A common approach to software testing is to divide up the test work along the same lines.

When you perform software testing on the data, you're checking that information the user inputs, results that he receives, and any interim results internal to the software are handled correctly.

Examples of data would be

• The words you type into a word processor

• The numbers entered into a spreadsheet

2.6. State Testing:

So far what you've been testing is the data—the numbers, words, inputs, and outputs of the software.
The other side of software testing is to verify the program's logic flow through its various states.
A software state is a condition or mode that the software is currently in.

2.7. Other Black Box Test Techniques:

- The remaining categories of black-box test techniques aren't standalone methods as much as they are variations of the data testing and state testing that has already been described.
- If you've done thorough equivalence partitioning of your program's data, created a detailed state map, and developed test cases from these, you'll find most software bugs that a user would find.
- What's left are techniques for finding the stragglers(remaining bugs), the ones that, if they were real living bugs, might appear to have a mind of their own, going their own way.
- Finding them might appear a bit subjective and not necessarily based on reason, but if you want to flush out every last bug, you'll have to be a bit creative.

2.8. White-Box Testing Techniques-Data Coverage:

Data Coverage means At least one test case for each data item / variable / field in the program
Data includes all the variables, constants, arrays, data structures,
keyboard and mouse input, files and screen input and output, and I/O to other
devices such as modems, networks, and so on.

Data Flow

Data flow coverage involves tracking a piece of data completely through the software.
At the unit test level this would just be through an individual module or function.
If you test a function at this low level, you would use a debugger and watch variables
to view the data as the program runs

2.9. Code Coverage:

- You must attempt to enter and exit every module, execute every line of code, and follow every logic and decision path through the software.
- This type of testing is known as ***code coverage*** testing.
- Code coverage is **dynamic white-box testing** because it requires you to have full access to the code to view what parts of the software you pass through when you run your test cases.
- The simplest form of code coverage testing is using your **compiler's debugger to view the lines of code** you visit as you single-step through the program.

2.10. Other White Box Test Techniques :

- Unit Testing
- Static Analysis
- Dynamic Analysis
- Statement Coverage
- Branch testing Coverage
- Security Testing
- Mutation Testing

- **Unit Testing:**

Unit Testing is one of the basic steps, which is performed in the early stages.
Most of the testers prefer performing to check if a specific unit of code is functional or not.
Unit Testing is one of the common steps performed for every activity because it helps in removing basic and simple errors

- **Static Analysis:**

The static analysis is an important step because it helps in filtering simple errors in the initial stage of the process.

- **Dynamic Analysis:**

Dynamic Analysis is the further step of static analysis in general path testing. Most of the people prefer performing both static and dynamic at the same time.

- **Statement Coverage:**

It means checking the codes **Statement** by **Statement**
You could run your tests and add test cases until every statement in the program is touched.
Unfortunately, statement coverage is misleading. It can tell you if every statement is executed, but it can't tell you if you've taken all the paths through the software.

- **Branch testing Coverage:**

The modern-day software and web applications are not coded in a continuous mode because of various reasons.
It is necessary to branch out at some point in time because it helps in segregating effectively.
Branch coverage testing gives a wide room for testers to find quick results.
It helps in verifying all the possible branches in terms of lines of code.

- **Security Testing**

It is a known fact that security is one of the primary protocol, which needs to be in place all the time.
Most of the companies prefer having a regular security testing activity because of obvious reasons.
It is essential to have a process in place to protect the application or software automatically.
Security testing is more like a process because it comes with a lot of internal steps to complete.
It verifies and rectifies any kind of unauthorized access to the system.
The process helps in avoiding any kind of breach because of hacking or cracking practices.

- **Mutation Testing:**

The last step in the process and requires a lot of time to complete effectively.
Mutation testing is generally conducted to re-check any kind of bugs in the system.
The step is carried out to ensure using the right strategy because of various reasons.
It gives enough information about the strategy or a code to enhance the system from time to time.

White Box testing

The term 'white box' is used because of the internal perspective of the system. The **clear box or white box, or transparent box** name denotes the ability to see through the software's outer shell into its inner workings.

It is performed by Developers, and then the software will be sent to the testing team, where they perform black-box testing. The main objective of white-box testing is to test the application's infrastructure. It is done at lower levels, as it includes unit testing and integration testing. It requires programming knowledge, as it majorly focuses on code structure, paths, conditions, and branches of a program or software. The primary goal of white-box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

It is also known as structural testing, clear box testing, code-based testing, and transparent testing. It is well suitable and recommended for algorithm testing.

Black Box testing

The primary source of black-box testing is a specification of requirements that are stated by the customer. It is another type of manual testing. It is a software testing technique that examines the functionality of the software without knowing its internal structure or coding. It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function. In this testing, the test engineer analyzes the software against requirements, identifies the defects or bugs, and sends it back to the development team.

In this method, the tester selects a function and gives input value to examine its functionality, and checks whether the function is giving the expected output or not. If the function produces the correct output, then it is passed in testing, otherwise failed.

Black box testing is less exhaustive than White Box and Grey Box testing methods. It is the least time-consuming process among all the testing processes. The main objective of implementing black box testing is to specify the business needs or the customer's requirements.

In other words, we can say that black box testing is a process of checking the functionality of an application as per the customer's requirement. Mainly, there are three types of black-box testing**: functional testing, Non-Functional testing,** and **Regression testing**. Its main objective is to specify the business needs or the customer's requirements.