**Name:Nishanthini V**

**Batch No:2022-9615**

**Enroll number: EBEON0323765163**

**(26/06/23) SOFTWARE TESTING:**

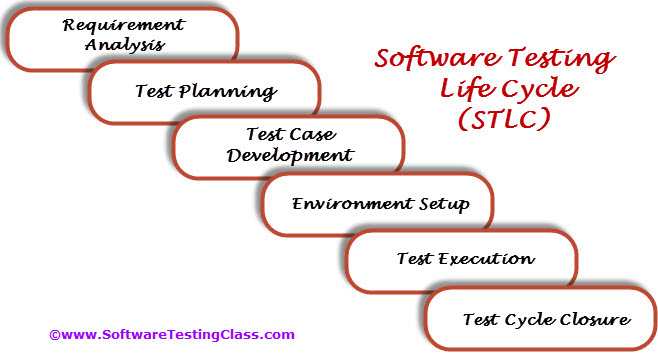
The procedure of software testing is also known as STLC (Software Testing Life Cycle) which includes phases of the testing process.The testing process is executed in a well-planned and systematic manner. All activities are done to improve the quality of the software product. Customer requirement will reach or not.

STLC stands for Software Testing Life Cycle. STLC is a sequence of different activities performed by the testing team to ensure the quality of the software or the product.

**SDLC**

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

**SOFTWARE TESTING LIFE CYCLE:**



**REQUIREMENT ANALYSIS**

In this phase, the testing team analyzes the software requirements to gain a clear understanding of the expected behavior and functionality. They identify testable requirements and potential test scenarios based on the information provided.

**TEST PLANNING**

The test planning phase involves creating a detailed test plan that outlines the testing approach, test objectives, scope, test environment setup, resource allocation, and timelines. Test deliverable, such as test cases, test data, and test scripts, are also prepared during this phase..

**TEST CASE DEVEOPMENET**

In this phase, test cases and test scripts are developed based on the requirements and test objectives defined in the previous phases. Test cases are designed to cover different scenarios, including positive and negative test cases, boundary conditions, and exception handling.

**SENVIRONMENT SETUP**

The testing team sets up the required hardware and software environments necessary for executing the tests. This includes configuring the test tools, preparing test data, and establishing any necessary test infrastructure or test beds.

**TEST EXECUTION**

During this phase, the test cases are executed using the prepared test data and scripts. Testers perform functional and non-functional tests, record the outcomes, and compare the actual results with the expected results. Any defects or issues encountered are reported and tracked for resolution.

**TEST CYCLE CLOSURE**

The test closure phase involves evaluating the overall testing process and collecting lessons learned. The testing team reviews the test artifacts, assesses the test coverage, and identifies areas for improvement. A final test report is prepared, and the testing activities are formally closed.

**Two types of Testing**



* **Manual Testing -Manual Clicking**

A kind of software testing in which a software tester develops and executes the test cases without using any automated testing tools

* **Automation Testing - code**

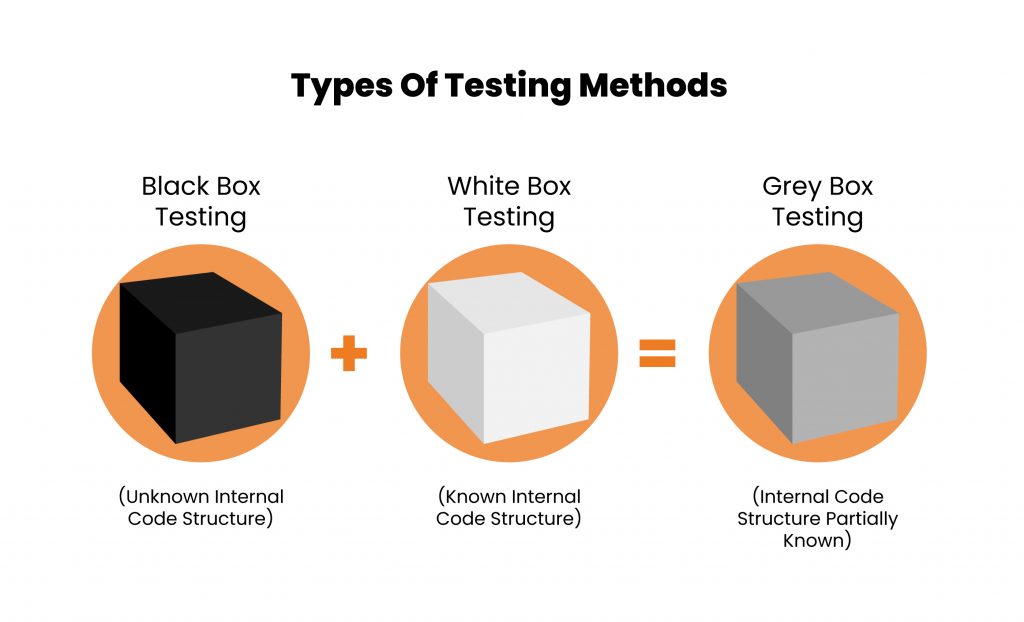
A software testing technique that automates the process of validating the functionality of software and ensures it meets requirements before being released into production

**Categorized into 2 behaviour of the software**

Functional testing -checking the particular functionality working properly

Non-functional Testing -speed,

**Method of testing**

****

**White Box Testing** - internal behaviour (internal storage & logic & Internal Programming Code)

**Black Box Testing**- Checking the external behaviour of software

**Grey Box Testing** - combination of both

**LEVELS OF TESTING(CISA)**

**COMPONENT TESTING**- individual component

**INTEGERATION TESTING** connection b/w 2 modules.eg.ATM connection between all the component

**top down approach eg-main page to cart module** driver is the first module and stub is last module and it also called dummy module and **bottom up approaches** cart item to main page

**SYSTEM TESING** -checking the whole product. Complete checking black box, white box, automation check all type of testing it will stop at if 90% of customer requirement

In this there are so many types of testing called positive testing, negative testing,gorilla testing etc

**ACCEPTANCE TESTING**

alpha - testing at developer side

beta - testing at customer side

**(27/06/23) BUG LIFE CYCLE?**

Bug Life Cycle in software testing is the specific set of states that defect or bug goes through in its entire life. The purpose of Defect life cycle is to easily coordinate and communicate current status of defect which changes to various assigners and make the defect fixing process systematic and efficient.

## **DEFECT STATES WORKFLOW**

* ****New:**** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
* ****Assigned:**** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
* ****Open****: The developer starts analyzing and works on the defect fix
* ****Fixed****: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
* ****Pending retest****: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is “pending retest.”
* ****Retest****: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”
* ****Verified****: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”
* ****Reopen****: If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.
* ****Closed****: If the bug is no longer exists then tester assigns the status “Closed.”
* ****Duplicate****: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”
* **Not Accepted**: If the error is not accepting by Developer means then it goes to the Application Manager and if Application Manager can accept the error means then asked to the developer to fix it.Otherwise Application Manger will give the proper reason for negelating this error.
* ****Rejected****: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”
* ****Deferred****: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs
* ****Not a bug****: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.

**BUG LIFE CYCLE**



**NO**

1. Tester finds the defect
2. Status assigned to defect- New
3. A defect is forwarded to Project Manager for analyze
4. Project Manager decides whether a defect is valid
5. Here the defect is not valid- a status is given “Rejected.”
6. So, project manager assigns a status **rejected**. If the defect is not rejected then the next step is to check whether it is in scope. Suppose we have another function- email functionality for the same application, and you find a problem with that. But it is not a part of the current release when such defects are assigned as a **postponed or deferred**status.
7. Next, the manager verifies whether a similar defect was raised earlier. If yes defect is assigned a status **duplicate**.
8. If no the defect is assigned to the developer who starts fixing the code. During this stage, the defect is assigned a status **in- progress.**
9. Once the code is fixed. A defect is assigned a status **fixed**
10. Next, the tester will re-test the code. In case, the[Test Case](https://www.guru99.com/test-case.html)passes the defect is **closed.** If the test cases fail again, the defect is **re-opened** and assigned to the developer.
11. Consider a situation where during the 1st release of Flight Reservation a defect was found in Fax order that was fixed and assigned a status closed. During the second upgrade release the same defect again re-surfaced. In such cases, a closed defect will be **re-opened.**

## **7 PRINCIPLES OF SOFTWARE TESTING**

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy

**1)Testing Shows the Presence of Defects:**

The primary objective of testing is to uncover defects or discrepancies between expected and actual system behavior. Testing helps identify problems that need to be addressed and ensures the software meets the specified requirements.

**2)Exhaustive Testing is Impossible:**

It is practically impossible to test every possible input, scenario, or condition within a software system. Instead, testing efforts should focus on prioritizing test cases based on risks, requirements, and other factors to maximize the likelihood of detecting defects.

**3)Early Testing:**

Testing activities should be initiated as early as possible in the software development lifestyle. Starting testing early helps identify and rectify defects at the earliest stages, reducing the cost and effort required for fixing issues later in the development process.

**4)Defect Clustering:**

It has been observed that defects tend to cluster in specific modules, functionalities, or areas of the software system. By identifying and addressing the high-risk areas, testing efforts can be optimized to maximize defect detection.

**5)Pesticide Paradox:**

Repeating the same tests repeatedly may lead to a decrease in the effectiveness of those tests. Test cases should be periodically reviewed and updated to ensure they remain relevant and continue to provide value in detecting defects.

**6)Testing is Context-Dependent:**

Testing should be customized and adapted to the specific needs of the project, including factors such as the software's complexity, criticality, target audience, and regulatory requirements. Different projects may require different testing approaches, techniques, and tools. Mobile app VS Website.

**7)Absence-of-Errors Fallacy:**

The absence of detected defects in testing does not guarantee the absence of defects in the software. Testing can increase confidence in the quality of the software, but it cannot prove that the software is entirely defect-free. Other factors, such as the test coverage, test environment, and the effectiveness of testing, should also be considered.99.99% error free

**DIFFERENCE BETWEEN BUG,ERROR,FAULT.MISTAKE,FAILURE**

**Bug:**

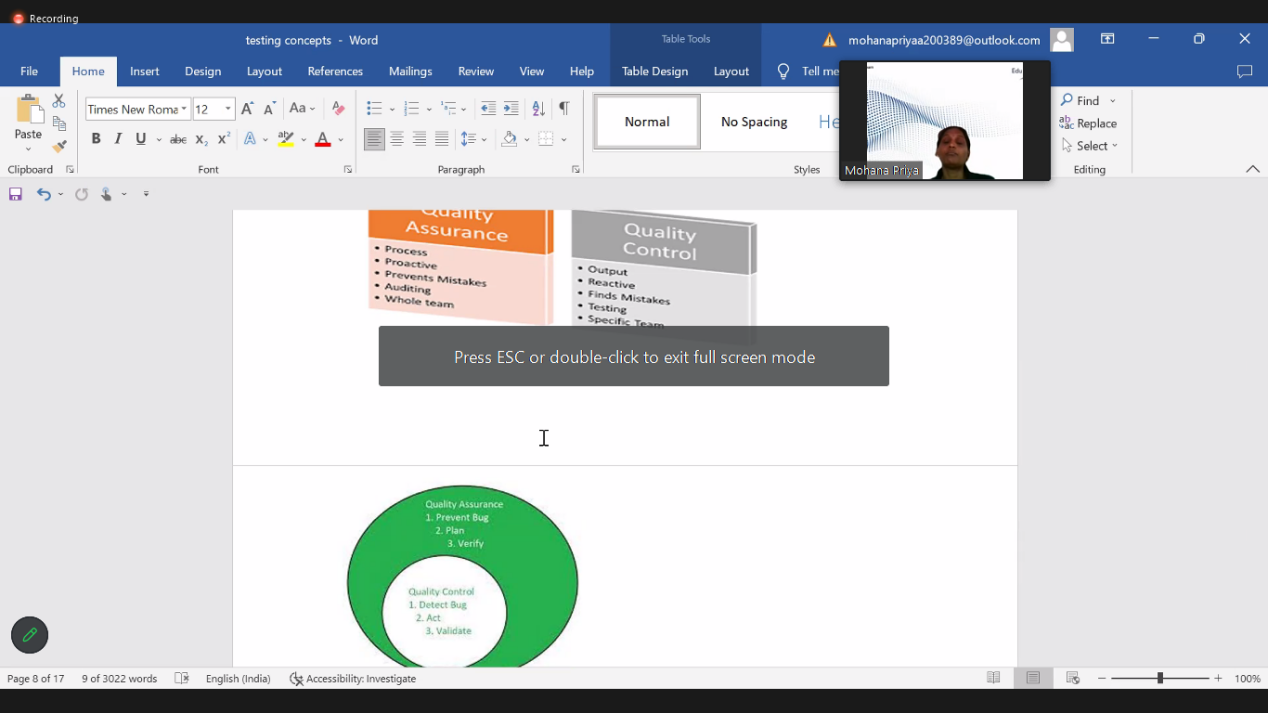
A bug refers to a flaw or defect in a software program that causes it to behave in an unintended or erroneous way. Bugs can occur due to coding errors, logic flaws, or problems with system configuration. Bugs can lead to unexpected behavior, crashes, or incorrect output.

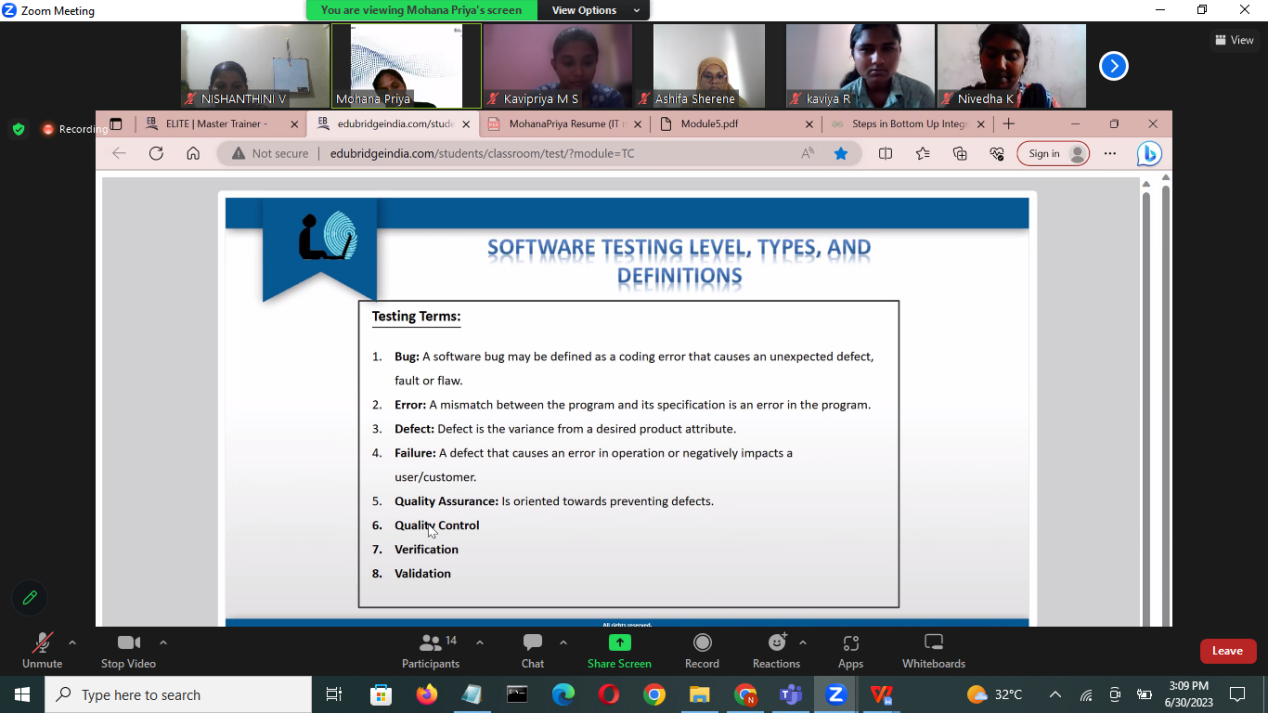
**Error**: In the context of software development, an error refers to a human action or oversight that produces an incorrect or unintended result. It can occur during any phase of the software development process, such as requirements gathering, design, coding, or testing. Errors can introduce bugs or other problems into the software.

**Fault:** The term "fault" is often used interchangeably with "bug." It represents a defect or malfunction in a software system that results in incorrect or unexpected behavior. A fault can be caused by errors in code, hardware failures, or external factors.

**Defect :**Gap b/w actual output and expected output

**Mistake:** A mistake is an error made by a person during the software development process. It can include errors in understanding requirements, making incorrect design decisions, or implementing code that does not fulfill the intended functionality. Mistakes can lead to bugs or faults in the software.





**Software Requirements Specification**

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform.

**Test Planning Document**

A Test Plan is a detailed document that catalogs the test strategies, objectives, schedule, estimations, deadlines, and resources required to complete that project. Think of it as a blueprint for running the tests needed to ensure the software is working correctly – controlled by test managers.

Retest and regression - tester

Again testing some testing new changes that cause new bug/defect

Functional testing -checking a particular functionality eg cal app where Addition is working submit button is working etc.. unit ,smoke ,integration first ta test

Non - functional - checking the capability of performance ,how much loads, reliability -long time,support-ability volume,recovery.how much speed,performance etc.

Smoke testing - basic component

Monkey Testing - no flow what ever tester want

Gorilla Testing/frustration Testing - repeating we are testing same

Positive Testing = purposely

Negative

Static Testing - without implementation of code checking the document

Dynamic Testing- Code behaviour with implementation of code checking the document

Exploratory-Proper flow

Load - maximum quality ,

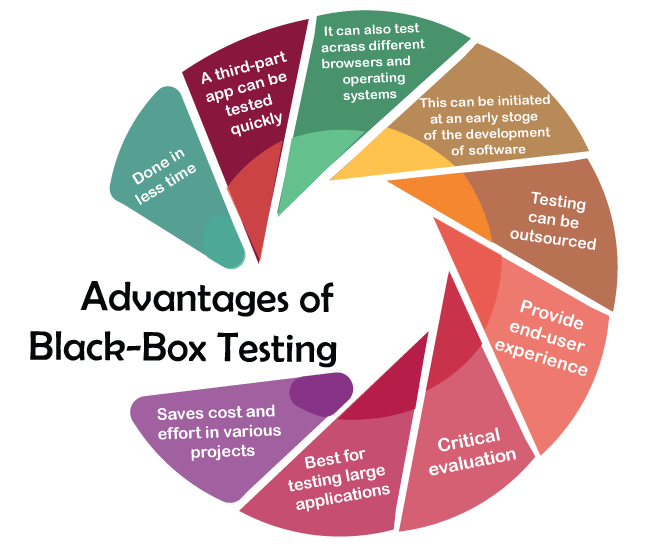
**Mechanism of Black Box testing**

Equivalence Partitioning-- checking the range eg above 18 people can vote and it deals with number and categories as valid and invalid for the specific ranger

Boundary Value Analysis --checking the range the number min and the number max

Decision Table ---eg in login table username and password is crt only it enter into home page like that true table like AND gate

State Transition like flowchart manner from on to off state and off state to on state if any error means it show as fault eg login bank acc first attempt (u,p)=pass,if fail means again (u,P)=pass it goes to home page.



Advantages of black box testing:

Independence: Testers do not require knowledge of the internal workings of the system, making black box testing independent of the development team. This allows for unbiased testing, as testers approach the system from a user's standpoint.

User-centric approach: Black box testing is centered around user requirements and expectations. By focusing on the system's functionalities, inputs, and outputs, it helps ensure that the system meets the user's needs and behaves as expected.

Simplicity: Black box testing is relatively simple to perform, as it does not require detailed knowledge of the system's internal implementation. Testers can focus solely on the system's interfaces and specifications, which simplifies test case design and execution.

Encourages robustness: Black box testing helps identify issues related to incorrect inputs, boundary conditions, missing functionality, and error handling. By exploring different scenarios and inputs, testers can uncover vulnerabilities and enhance the overall robustness of the system.

Disadvantages of black box testing:

Limited coverage: Black box testing primarily focuses on the system's external behavior, which means that it may not reveal defects that lie within the internal structure. Complex interactions or issues related to specific algorithms, data structures, or performance may remain undetected.

Inefficient bug localization: When issues are encountered during black box testing, it can be challenging to pinpoint the exact location of the problem. Since testers lack visibility into the internal workings of the system, debugging and fixing the issues can be time-consuming and require additional efforts.

Redundant test coverage: In some cases, black box testing may overlap with other testing techniques, such as unit testing or white box testing. This can result in redundant test cases, wasting time and resources.

Lack of design insights: Black box testing does not provide insights into the system's design or architecture. This means that it may not be effective in uncovering design flaws or potential optimizations that could enhance the system's overall performance or maintainability.

**White box testing-analysis program structure**

Another name: Glass box testing,open box testing

Checking the internal behaviour of software like statement,decision,all are related to code

**Statement Coverage** ---Statement number if else part is no means there is no flow and does not have statement for condition fail is called total number of

Statement coverage = (no of executed statements/Total no of statement ) \* 100

1. print (**int** a, **int** b) {
2. **int** sum = a+b;
3. **if** (sum>0)
4. print ("This is a positive result")
5. **else**
6. print ("This is negative result")
7. }
8. # a=4,b=5

Total number of statements = 7

Number of executed statements = 5

Statement coverage = 5/7\*100

= 500/7   = 71%

**Branch Coverage**-- not calculator percentage there is no True/false there is only one statement for particular condition there is no flow node and edge are there and there is no else part having only if statement and close by end-if  **all yes one time and all no one time**

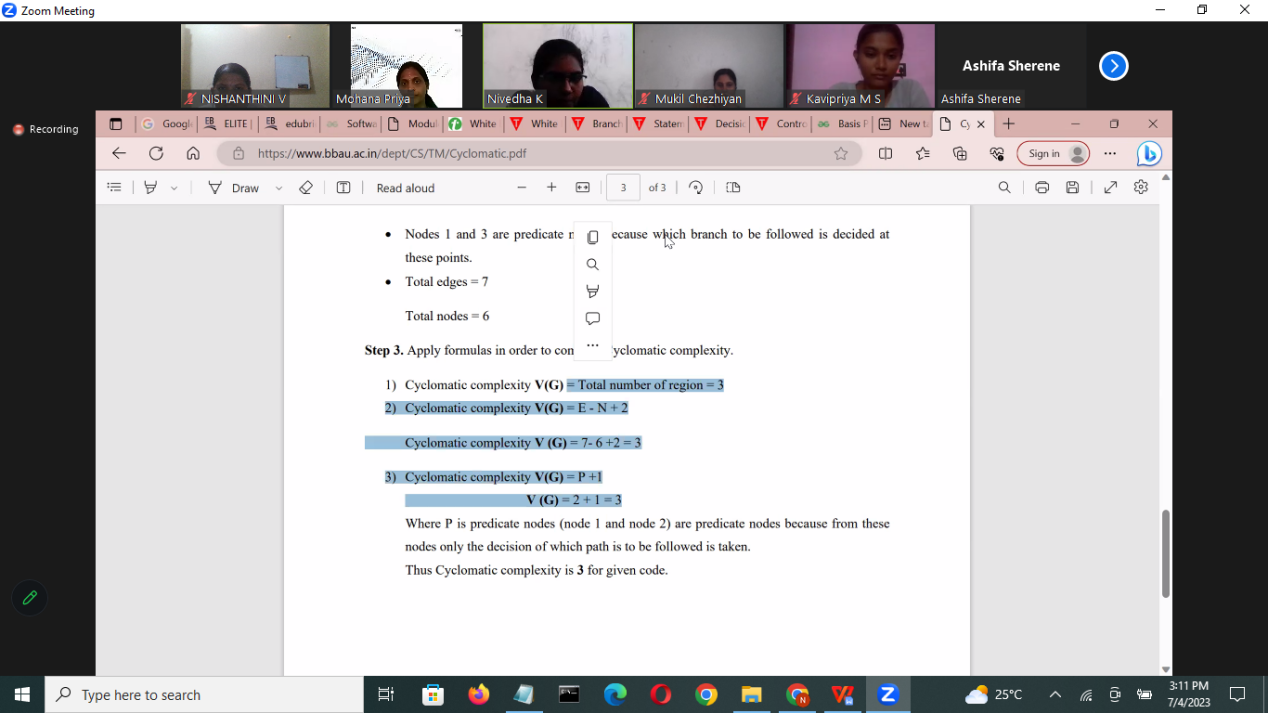
**Path Coverage** ---all the path like yes and no,no and yes, yes and yes,no and no paths

**Decision Coverage** --50% execute only one time

**Cyclomatic complexity** how much complexity level in our program

Formula

Decision d = p



Having some range.

## **Techniques Used in White Box Testing**

|  |  |
| --- | --- |
| [Data Flow Testing](https://www.javatpoint.com/data-flow-testing-in-white-box-testing) | Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events. |
| [Control Flow Testing](https://www.javatpoint.com/control-flow-testing-in-white-box-testing) | Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program. |
| [Branch Testing](https://www.javatpoint.com/branch-coverage-testing-in-white-box-testing) | Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once. |
| [Statement Testing](https://www.javatpoint.com/statement-coverage-testing-in-white-box-testing) | Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code. |
| [Decision Testing](https://www.javatpoint.com/decision-coverage-testing-in-white-box-testing) | This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false. |

|  |  |
| --- | --- |
| **White-box testing** | **Black box testing** |
| The developers can perform white box testing. | The test engineers perform the black box testing. |
| To perform WBT, we should have an understanding of the programming languages. | To perform BBT, there is no need to have an understanding of the programming languages. |
| In this, we will look into the source code and test the logic of the code. | In this, we will verify the functionality of the application based on the requirement specification. |
| In this, the developer should know about the internal design of the code. | In this, there is no need to know about the internal design of the code. |

**Advantages of white box**

The transparency of this type of testing allows the testing to be extensive, as the tester can do comprehensive tests covering all the paths and testing the entire structure and code base. It also assesses internal and external vulnerabilities that could help avoid future security threats and attacks.

Less time to perform white box testing – While the time required to perform white box testing may become a con when it comes to an extensive, complex application, which we mention later in the blog, but when all the available information about the application is available, it takes less time for the tester to go in, understand the code swiftly and perform the checks, but with large applications with complex functionalities, the testing can run into weeks or months.

The test cases can be easily automated, and many tools are available for performing automation, which expedites these tasks.

Code optimization – White box testing helps remove extra lines of code and identify hidden errors. There is a higher chance of detecting security risks or bugs due to all the information available about the application.

The process of white box testing can start early in the Software Development Life cycle Process (SDLC), even before the user interface comes about. This gives the developer a chance to fix the bugs early on or add elements to improve security in the early stages of the development process.

No requirement of an interface as is needed in other types of testing, such as black box testing.

**Disadvantage of white box testing**

Certain functionalities could be missed out as only the available code is tested.

Many developers are against it as it is a tedious process.

It is very time-consuming as the codes must be redesigned along with test cases that need to be rewritten.

Requires resources and professional skills as testers with in-depth programming skills are necessary, unlike in black box testing. In white box testing, a tester is expected to understand the mechanism of the code base of the application and have strong programming language knowledge.

There are basically three levels of testing i.e. Unit Testing, Integration Testing andSystem Testing. Various types of testing

come under these levels.

**Unit Testing**

To verify a single program or a section of a single program

**Integration Testing**

To verify interaction between system components Prerequisite: unit testing completed on all components that compose

a system.

**System Testing**

To verify and validate behaviors of the entire system against the original system objectives. Software testing is a process

that identifies the correctness, completeness, and quality of software. Following is a list of various types of software testing

and their definitions in a random order:

• **Formal Testing:** Performed by test engineers

• **Informal Testing:** Performed by the developers

• **Manual Testing:** That part of software testing that requires human input, analysis, or evaluation.

• **Automated Testing:** Software testing that utilizes a variety of tools to automate the testing process. Automated testing

still requires a skilled quality assurance professional with knowledge of the automation tools and the software being tested

All rights reserved.

No part of this document may be reproduced in any material form (including printing and photocopying or storing it in any medium by electronic or other means or not transiently or

incidentally to some other use of this document) without the prior written permission of EduBridge Lear

ning Pvt. Ltd. Application for written permission to reproduce any part of this

document should be addressed to the CEO of EduBridge Learning Pvt. Ltd.

to set up the test cases.

• **Black box Testing:** Testing software without any knowledge of the back-end of the system, structure or language of the

module being tested. Black box test cases are written from a definitive source document, such as a specification or

requirements document.

• **White box Testing:** Testing in which the software tester has knowledge of the back-end, structure and language of the

software, or at least its purpose.

• **Unit Testing:** Unit testing is the process of testing a particular complied program, i.e., a window, a report, an interface,

etc. independently as a stand-alone component/program. The types and degrees of unit tests can vary among modified

and newly created programs. Unit testing is mostly performed by the programmers who are also responsible for the

creation of the necessary unit test data.

• **Incremental Testing:** Incremental testing is partial testing of an incomplete product. The goal of incremental testing is

to provide an early feedback to software developers.

• **System Testing:** System testing is a form of black box testing. The purpose of system testing is to validate an application's

accuracy and completeness in performing the functions as designed.

• **Integration Testing:** Testing two or more modules or functions together with the intent of finding interface defects

between the modules/functions.

• **System Integration Testing:** Testing of software components that have been distributed across multiple platforms (e.g.,

client, web server, application server, and database server) to produce failures caused by system integration defects (i.e.

defects involving distribution and back-

• **Functional Testing:** Verifying that a module functions as stated in the specification and establishing confidence that a

program does what it is supposed to do.

• **End-to-end Testing:** Similar to system testing - testing a complete application in a situation that mimics real world use,

such as interacting with a database, using network communication, or interacting with other hardware, application, or

system.

**Sanity Testing:** Sanity testing is performed whenever cursory testing is sufficient to prove the application is functioning

according to specifications. This level of testing is a subset of regression testing. It normally includes testing basic GUI

functionality to demonstrate connectivity to the database, application servers, printers, etc.

• **Regression Testing:** Testing with the intent of determining if bug fixes have been successful and have not created any

new problems.

• **Acceptance Testing:** Testing the system with the intent of confirming readiness of the product and customer acceptance.

All rights reserved.

No part of this document may be reproduced in any material form (including printing and photocopying or storing it in any medium by electronic or other means or not transiently or

incidentally to some other use of this document) without the prior written permission of EduBridge Lear

ning Pvt. Ltd. Application for written permission to reproduce any part of this

document should be addressed to the CEO of EduBridge Learning Pvt. Ltd.

Also known as User Acceptance Testing.

• **Adhoc Testing:** Testing without a formal test plan or outside of a test plan. With some projects this type of testing is

carried out as an addition to formal testing. Sometimes, if testing occurs very late in the development cycle, this will be

the only kind of testing that can be performed – usually done by skilled testers. Sometimes adhoc testing is referred to as

exploratory testing.

• **Configuration Testing:** Testing to determine how well the product works with a broad range of hardware/peripheral

equipment configurations as well as on different operating systems and software.

• **Load Testing:** Testing with the intent of determining how well the product handles competition for system resources.

The competition may come in the form of network traffic, CPU utilization or memory allocation.

• **Stress Testing:** Testing done to evaluate the behaviour when the system is pushed beyond the breaking point. The goal

is to expose the weak links and to determine if the system manages to recover gracefully.

• **Performance Testing:** Testing with the intent of determining how efficiently a product handles a variety of events.

Automated test tools geared specifically to test and fine-tune performance are used most often for this type of testing.

• **Usability Testing:** Usability testing is testing for 'user-friendliness'. A way to evaluate and measure how users interact

with a software product or site. Tasks are given to users and observations are made.

• **Installation Testing:** Testing with the intent of determining if the product is compatible with a variety of platforms and

how easily it installs.

• **Recovery/Error Testing:** Testing how well a system recovers from crashes, hardware failures, or other catastrophic

problems.

• **Security Testing:** Testing of database and network software in order to keep company data and resources secure from

mistaken/accidental users, hackers, and other malevolent attackers.

• **Penetration Testing:** Penetration testing is testing how well the system is protected against unauthorized internal or

external access, or wilful damage. This type of testing usually requires sophisticated testing techniques.

• **Compatibility Testing:** Testing used to determine whether other system software components such as browsers,

utilities, and competing software will conflict with the software being tested.

• **Exploratory Testing:** Any testing in which the tester dynamically changes what they're doing for test execution, based

on information they learn as they're executing t heir tests.

• **Comparison Testing:** Testing that compares software weaknesses and strengths to those of competitors' products.

All rights reserved.

No part of this document may be reproduced in any material form (including printing and photocopying or storing it in any medium by electronic or other means or not transiently or

incidentally to some other use of this document) without the prior written permission of EduBridge Lear

ning Pvt. Ltd. Application for written permission to reproduce any part of this

document should be addressed to the CEO of EduBridge Learning Pvt. Ltd.

• **Alpha Testing:** Testing after code is mostly complete or contains most of the functionality and prior to reaching

customers. Sometimes a selected group of users are involved. More often this testing will be performed in-house or by an

outside testing firm in close cooperation with the software engineering department.

• **Beta Testing:** Testing after the product is code complete. Betas are often widely distributed or even distributed to the

public at large.

• **Gamma Testing:** Gamma testing is testing of software that has all the required features, but it did not go through all the

in-house quality checks.

• **Mutation Testing:** A method to determine to test thoroughness by measuring the extent to which the test cases can

discriminate the program from slight variants of the program.

• **Independent Verification and Validation (IV&V):** The process of exercising software with the intent of ensuring that the

software system meets its requirements and user expectations and doesn't fail in an unacceptable manner. The individual

or group doing this work is not part of the group or organization that developed the software.

• **Pilot Testing:** Testing that involves the users just before actual release to ensure that users become familiar with the

release contents and ultimately accept it. Typically involves many users, is conducted over a short period of time and is

tightly controlled.

• **Parallel/Audit Testing:** Testing where the user reconciles the output of the new system to the output of the current

system to verify the new system performs the operations correctly.

• **Glass Box/Open Box Testing:** Glass box testing is the same as white box testing. It is a testing approach that examines

the application's program structure, and derives test cases from the application's program logic.

• **Closed Box Testing:** Closed box testing is same as black box testing. A type of testing that considers only the functionality

of the application.

**Bottom-up Testing:** Bottom-up testing is a technique for integration testing. A test engineer creates and uses test drivers

for components that have not yet been developed, because, with bottom-up testing, low-level components are tested

first. The objective of bottom-up testing is to call low-level components first, for testing purposes.

• **Smoke Testing:** A random test conducted before the delivery and after complete testing

**TEST DATA MANGEMENT & TEST ENVIRONMENT MANAGEMENT**

Ways to improve TDM

* Favor unit Tests - easy and useful and individual
* Minimize Reliance on test data
* Isolate your test data
* Minize reliance on test data stored in db

Make test data readily available

**VALIDATION & VERIFICATION**

verfication -Are we building product right ?step by step verfication process -product oriented approach

validation - Are we building right product ?at last verfifcation -process oriented approach

Eg checking hall ticket before exam hall

Assurance means wearing mask before attack from covid and verfifcaion

Control means after affecting covid wearing mask and validate

**Performance Testing Attributes**

Speed

Scalability -multiple user

Stability -number of user Up/down user at that time stable

Reliability -one particular fault does not affect whole system to work

Reuireent Gsthering

Tool Delection

Performance Test Plan

**Types of performance**

Load- no of user

Stress- checking the capacity

Soak- endurance Testing how long it survive?

Spike - up/down

Volume-total number of space

Scalability-multiple user

Connection - with different software

Production- final report day by day ,week by week,find the response time

**2 types of tool**

JMeter find level an stress

LoadRunner to find the performance of software

Cross docking testing

Lamda testing

Sauce lab

**SECURITY TESTING**

**Six Basic Security Concepts**

**Confidentiality** – Information should be accessible to only those with authorized access. In secure way

**• Integrity** – A measure intended to allow the receiver to determine that the information which it is providing is correct.

**• Authentication** – Establish the identity of the user.

**• Authorization** – The user should receive a service or perform an action for which he has permission. Giving permission

**• Availability** – Information and communication services should be ready at any time, as needed. Available of data

**• Non-repudiation** – Prevent later denial that an action happened. Late service

Vulnerability

URL Manipulation - additional features to communiate bw server and browser.

SQL Injection - code injection technique that might destroy your database. Hacking techniques,

XSS-(Cross Site Scripting)-one web page that affect another web page

**Rolerss**

Hackers

Cracker

Ethical HAcker

Scrript Kiddies or packet monkey