

## Software Testing Methodology

- Software Testing Methodology is defined as strategies and testing types used to certify that the Application Under Test meets client expectations.
- Test Methodologies include functional and non-functional testing to validate the AUT.
- Examples of Testing Methodologies are Unit Testing, Integration Testing, System Testing, Performance Testing etc

## Goals of Software Testing:

The main goal of software testing is to find bugs as early as possible and fix bugs and make sure that the software is bug-free.

### Important Goals of Software Testing:

- Detecting bugs as soon as feasible in any situation.
- Avoiding errors in a projects and product's final versions.
- Inspect to see whether the customer requirements criterion has been satisfied.
- Last but not least, the primary purpose of testing is to scale the project and product level of quality.

The goals of software testing may be classified into three major categories as follows:

- Immediate Goals
- Long-term Goals
- Post-Implementation Goals

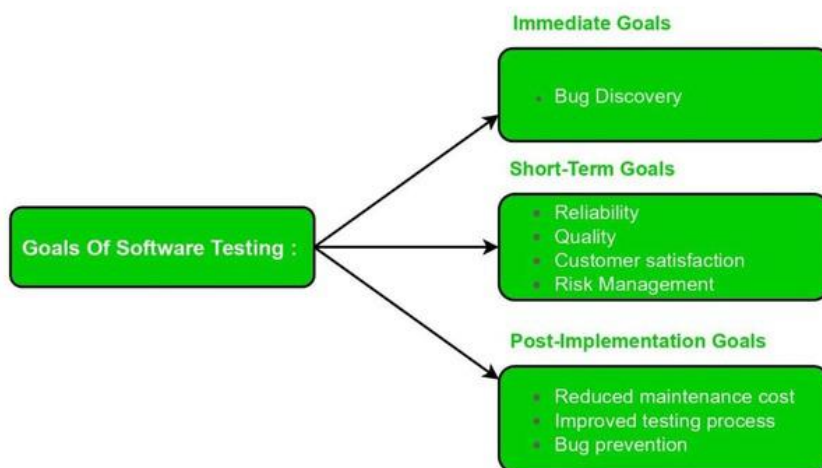


Fig : Software Testing\_Goals

**1. Immediate Goals:** These objectives are the direct **outcomes** of testing. These objectives may be set at any time during the SDLC process. Some of these are covered in detail below:

*Bug Discovery:* This is the immediate goal of software testing to **find errors** at any stage of software development. The number of **bugs** is discovered in the early stage of **testing**. The primary purpose of software testing is to **detect flaws** at any step of the development process.

*Bug Prevention:* This is the immediate action of bug discovery, that occurs as a **result** of **bug discovery**. Everyone in the software development **team** learns how to code from the behavior and analysis of issues **detected**, ensuring that bugs are **not duplicated** in subsequent phases or future projects.

**2. Long-Term Goals:** These objectives have an impact on **product quality in** the long run after one **cycle of the SDLC is completed**. Some of these are covered in detail below:

*Quality:* This **goal enhances the quality** of the software product. Because software is also a product, the user's priority is its quality. Superior quality is ensured by thorough testing. **Correctness, integrity, efficiency, and reliability are all** aspects that influence quality.

*Customer Satisfaction:* This goal **verifies** the customer's satisfaction with a **developed** software **product**. The primary purpose of software testing, from the user's standpoint, is customer satisfaction.

*Reliability:* It is a matter of **confidence** that the software will not fail. In short, reliability means gaining the **confidence** of the customers by **providing** them with a quality product.

*Risk Management:* Risk is the probability of occurrence of uncertain events in the organization and the potential **loss** that could result in negative consequences.

**3. Post-Implemented Goals:** After **the product is released**, these objectives become critical. Some of these are covered in detail below:

*Reduce Maintenance Cost:* Post-released errors are **costlier** to **fix** and difficult to identify. Because effective software does not wear out, the maintenance cost of any software product is not the same as the physical cost. The failure of a **software product due to faults is the only expense of maintenance**. Because they are difficult to discover, post-release mistakes always cost more to rectify.

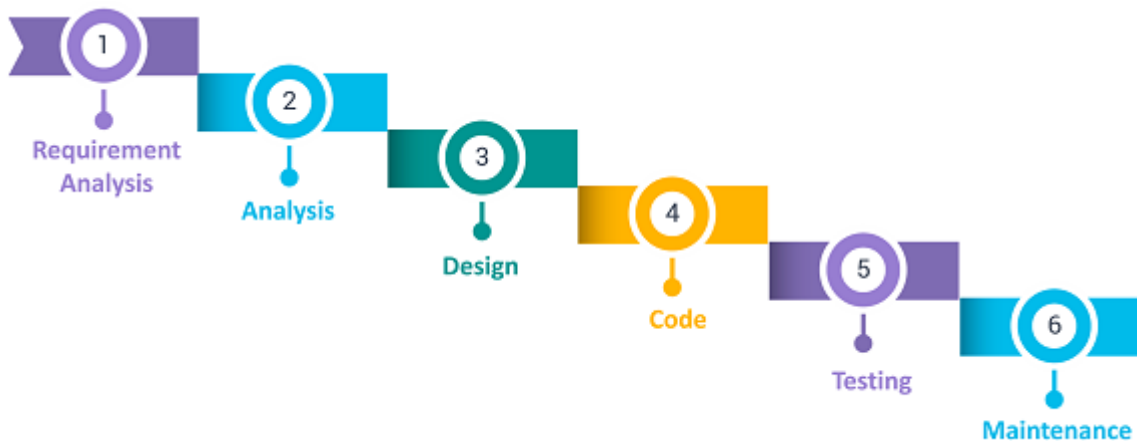
*Improved Software Testing Process:* These goals improve the **testing process** for future use or **software projects**. These goals are known as **post-implementation goals**. A project's testing procedure may not be completely successful, and there may be room for **improvement**.

## Software Testing Models

- [Waterfall Model](#)
- [V Model](#)
- [Agile Model](#)
- [Spiral Model](#)
- [Iterative Model](#)

### Waterfall Model

This is the most basic [software development life cycle](#) process which is followed broadly in the [industry](#). In this model, the developers follow a [sequence](#) of processes downwards [towards](#) the [ultimate](#) goal. It is like a waterfall where there [are](#) various [phases](#) involved. the developer must complete every [phase](#) before the next phase [begins](#). This model is named "**Waterfall Model**",



1. **Requirements Gathering and Analysis:** The first phase involves gathering [requirements](#) from [stakeholders](#) and analyzing them to understand the scope and [objectives](#) of the project.
2. **Design:** Once the requirements are understood, the design phase begins. This involves creating a detailed design [document](#) that [outlines](#) the software architecture, user interface, and system components.
3. **Implementation:** The implementation phase involves [coding](#) the [software](#) based on the design specifications. This phase also includes unit testing to ensure that [each](#) component of the software is working as expected.
4. **Testing:** In the testing phase, the software is tested as a whole to [ensure](#) that it meets the [requirements](#) and is free from defects.
5. **Deployment:** Once the software has been [tested](#) and [approved](#), it is deployed to the [production](#) environment.
6. **Maintenance:** The final phase of the [Waterfall](#) Model is maintenance, which involves fixing [any](#) [issues](#) that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.


## Advantages

- It is **easy** to **implement** and maintain.
- The initial **phase** of rigorous scrutiny of requirements and systems helps in **saving time** later in the developmental phase.
- The requirement of resources is **minimal** and testing is done after each phase has been completed.

## Disadvantages

- It is not possible to **alter** or **update requirements**.
- Once you move into the **next** phase you cannot **make** changes.
- You cannot start the **next** phase until the previous **phase** is completed.

Aspect	Effective Testing	Exhaustive Testing
<b>Definition</b> ①	Selective testing of critical and high-impact areas	Testing all possible combinations and scenarios
<b>Goal</b> ②	Focuses on identifying major defects and risks	Aims to find all possible defects and issues
<b>Coverage</b> ③	Covers a subset of test cases	Covers all possible test cases
<b>Efficiency</b> ④	More efficient; targets key areas	Extremely resource-intensive and time-consuming
<b>Time and Resources</b> \$ ⑤	Requires fewer resources and less time	Requires substantial time and resources
<b>Real-world Usage</b> ⑥	Suitable for most projects and time constraints	Rarely used due to impracticality
<b>Completeness</b> ⑦	Doesn't ensure complete coverage	Theoretically complete but often not feasible

Aspect	Effective Testing	Exhaustive Testing
Bug Discovery 	May miss some rare defects _____	Likely to uncover more defects _____
Common Approach	Used in most software testing scenarios	Rarely applied due to time and resource limits

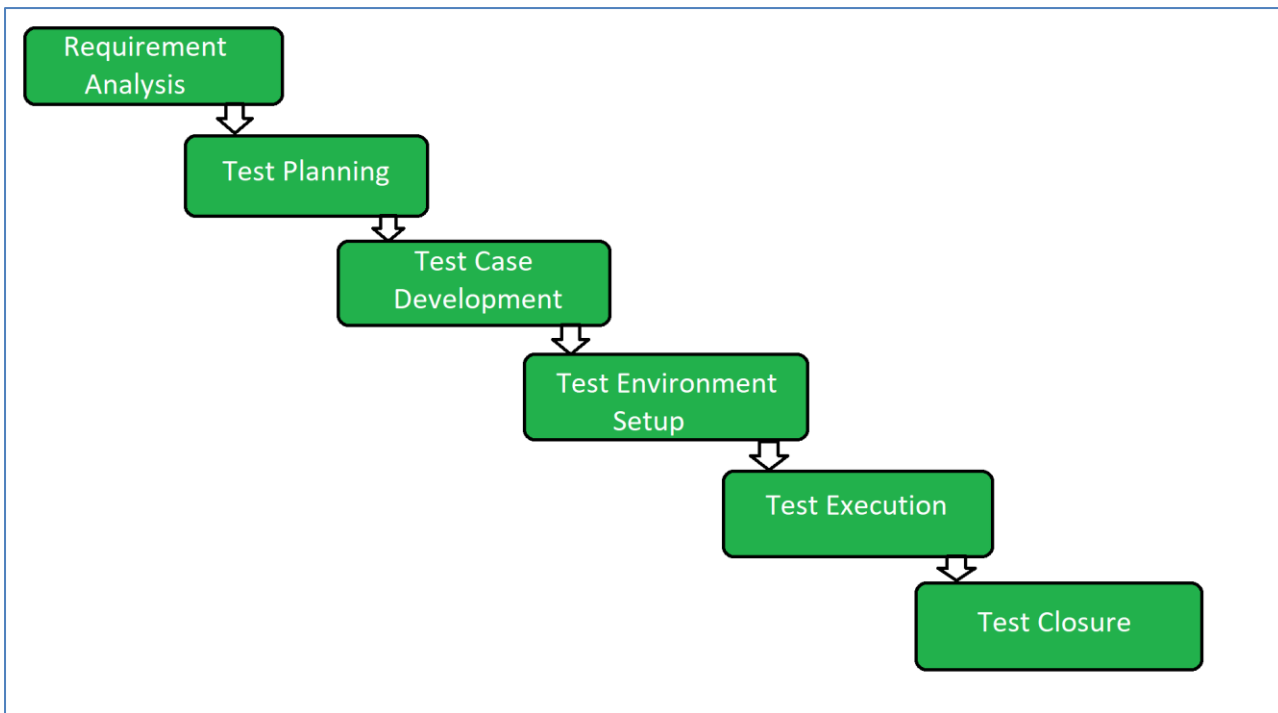
### Software Testing Life Cycle (STLC):

The Software Testing Life Cycle (STLC) is a **systematic approach** to testing a software application to ensure that it meets the **requirements** and is free of **defects**.

- It is a process that follows a **series** of **steps** or phases, and each phase has specific **objectives** and **deliverables**.
- The STLC is used to **ensure** that the software is of high **quality**, **reliable**, and **meets** the needs of the **end-users**.
- The main goal of the STLC is to **identify** and **document** any **defects** or issues in the software **application** as early as possible in the **development** process.
- This allows for issues to be **addressed** and **resolved** before the software is released to the public.

### **Characteristics of STLC**

- STLC is a fundamental part of the [Software Development Life Cycle \(SDLC\)](#) but STLC consists of **only the testing** phases.
- STLC starts as soon as **requirements** are defined or software requirement document is shared by stakeholders.
- **STLC yields a step-by-step process to ensure quality software.**



1. **Requirement Analysis:** Requirement Analysis is the first step of the Software Testing Life Cycle (STLC). In this phase quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then the quality assurance team meets with the stakeholders to better understand the detailed knowledge of requirements.

2. **Test Planning:** Test Planning is the most efficient phase of the software testing life cycle where all testing plans are defined. In this phase manager of the testing, team calculates the estimated effort and cost for the testing work. This phase gets started once the requirement-gathering phase is completed.

3. **Test Case Development:** The test case development phase gets started once the test planning phase is completed. In this phase testing team notes down the detailed test cases. The testing team also prepares the required test data for the testing. When the test cases are prepared then they are reviewed by the quality assurance team.

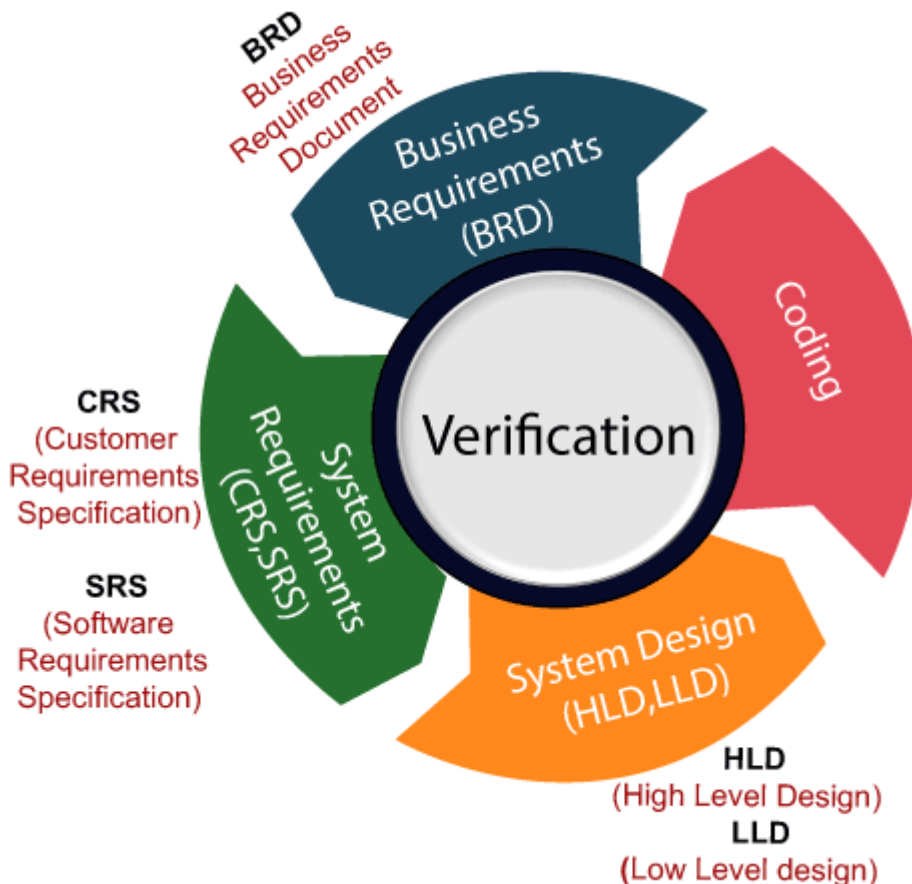
4. **Test Environment Setup:** Test environment setup is a vital part of the STLC. Basically, the test environment decides the conditions on which software is tested. This is independent activity and can be started along with test case development. In this process, the testing team is not involved. either the developer or the customer creates the testing environment.

5. **Test Execution:** After the test case development and test environment setup test execution phase gets started. In this phase testing team starts executing test cases based on prepared test cases in the earlier step.

6. **Test Closure:** Test closure is the final stage of the Software Testing Life Cycle (STLC) where all testing-related activities are completed and documented. The main objective of the test closure stage is to ensure that all testing-related activities have been completed and that the software is ready for release.

**Verification** in Software Testing is a process of checking documents, design, code, and program in order to check if the software has been built according to the requirements or not.

- The main goal of verification process is to ensure quality of software application, design, architecture etc.
- The verification process involves activities like reviews, walk-throughs and inspection.



**Validation in Software Engineering** is a dynamic mechanism of testing and validating if the software product actually meets the exact needs of the customer or not.

- The process helps to ensure that the software fulfils the desired use in a suitable environment.
- The validation process involves activities like unit testing, integration testing, system testing and user acceptance testing.

Verification	Validation
The verifying process includes checking documents, design, code, and program	It is a dynamic mechanism of testing and validating the actual product
It does <b>not</b> involve executing the code	It always involves executing the code
Verification uses methods like reviews, walkthroughs, inspections, and desk- checking etc.	It uses methods like Black Box Testing, White Box Testing, and non-functional testing
It finds bugs early in the development cycle	It can find bugs that the verification process cannot catch
Target is application and software architecture, specification, complete design, high level, and database design etc.	Target is an actual product
It comes before validation	It comes after verification

### Types of Testing Techniques / Terminologies/level of testing

- **Unit Testing** – It refers to individual units or components of the software that are tested. The purpose of unit testing is to see that each software unit performs as per the expected requirement and functionality. It may be termed as first-level testing.
- **Integration Testing** - It refers to testing in which the individual software components or modules are combined together and tested as a group. It is used to see whether there is any fault in the integrated units. It is normally performed after the unit testing.
- **System Testing** – The testing is performed on the system as a whole and to see whether the system is working as per the requirements and functionality specified. It takes the components of the integration testing as input and performs the testing.
- **Regression Testing** – This testing is performed to check whether the changes to the software do not produce undesirable results. It is used to ensure that the previously developed software components and tested components still perform correctly after the changes. It also depicts that the changes have not broken any functionality.
- **UAT – User Acceptance testing** – It is the testing done by the end-user or client to accept the software system before moving the software to the production environment. It is used to check the functionality of the software end to end related to the business flow.

### Q Manual testing

- Manual testing is a software testing process in which test cases are executed manually without using any automated tool.
- All test cases executed by the tester manually according to the end user's perspective.
- It ensures whether the application is working, as mentioned in the requirement document or not.
- Test cases are planned and implemented to complete almost 100 percent of the software application. Test case reports are also generated manually.



**Software Testing** is a method to check whether the actual software product matches **expected requirements** and to **ensure that software product is Defect free**.

- It involves execution of software/system components using manual or automated tools to evaluate **one or more properties of interest**.
- The purpose of software **testing** is to **identify errors, gaps or missing requirements** in contrast to actual requirements.

### **Black Box Testing**

“Black box testing is a technique of software testing which examines the functionality of software without **examining** into its **internal structure** or **coding**. The primary source of black box testing is a **specification of requirements**(SRS) that is stated by the customer.”

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



### **Generic steps of black box testing**

- The black box test is based on the **specification of requirements**, so it is examined in the beginning.
- In the second step, the tester creates a positive **test scenario** and an adverse test scenario by **selecting valid and invalid input** values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester **develops various** test cases such as **decision table, all pairs test, equivalent division, error estimation, cause-effect graph**, etc.
- The fourth phase **includes the execution** of all test cases.
- In the fifth step, the **tester compares the expected output against the actual output**.
- In the sixth and final step, if there is **any flaw in the software**, then it is cured and tested again

## Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones –

- **Functional testing** – This black box testing type is related to the functional requirements of a system; it is done by software testers.
- **Non-functional testing** – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** – Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

### Black Box Testing Techniques:

BDSES

#### **1 Boundary Value Technique**

Boundary Value Technique is used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.

**2.Decision Table Technique** is a systematic approach where various input combinations and their respective system behaviour are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.

**3.State Transition Technique** is used to capture the behaviour of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.

**4.Error guessing is a technique** in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.

**5.State Transition** Technique is used to capture the behaviour of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.

## Advantages of Black Box Testing

- The tester does not need to have more functional knowledge or programming skills to implement the Black Box Testing.
- It is efficient for implementing the tests in the larger system.
- Tests are executed from the user's or client's point of view.
- Test cases are easily reproducible.

## Disadvantages of Black Box Testing

- There is a possibility of repeating the same tests while implementing the testing process.
- Without clear functional specifications, test cases are difficult to implement.
- It is difficult to execute the test cases because of complex inputs at different stages of testing.
- Sometimes, the reason for the test failure cannot be detected.

### White Box Testing

**White Box Testing** is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security.

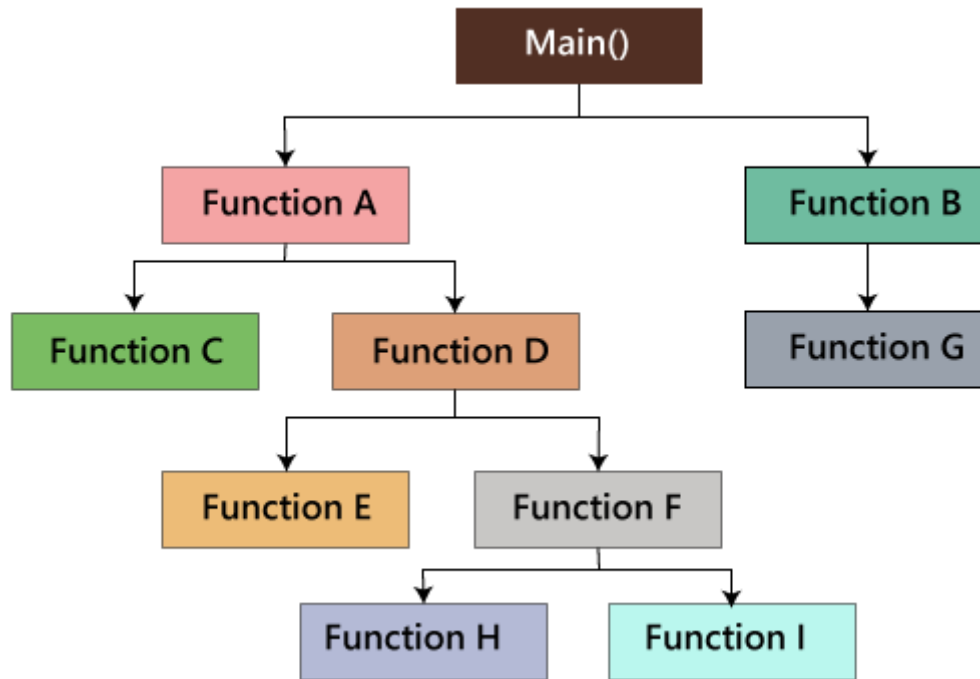
- In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.
- It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs.
- It is based on inner workings of an application and revolves around internal structure testing.

### Generic steps of white box testing:

1. Design all test scenarios, test cases and prioritize them according to high priority number.
2. This step involves the study of code at runtime to examine the resource utilization, not accessed areas of the code, time taken by various methods and operations and so on.
3. In this step testing of internal subroutines takes place. Internal subroutines such as non-public methods, interfaces are able to handle all types of data appropriately or not.
4. This step focuses on testing of control statements like loops and conditional statements to check the efficiency and accuracy for different data inputs.
5. In the last step white box testing includes security testing to check all possible security loopholes by looking at how the code handles security

### **TYPES**

In the **path testing**, we will write the flow graphs and test all independent paths. Here writing the flow graph implies that flow graphs are representing the flow of the program and also show how every program is added with one another as we can see in the below image:



In **the loop testing**, we will test the loops such as while, for, and do-while, etc. and also check for ending condition if working correctly and if the size of the conditions is enough.

#### Condition testing

In this, we will test all logical conditions for both true and false values; that is, we will verify for both if and else condition.

### White Box Testing Techniques:

**1.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

**2. 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

**3. 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.

**4.Decision Coverage 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.

**5.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.

## Advantages of White box testing

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

## Disadvantages of White box testing

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

**“Regression testing** is a black box testing technique. It is used to **authenticate** a code change in the software **does not impact** the existing **functionality** of the **product**.”

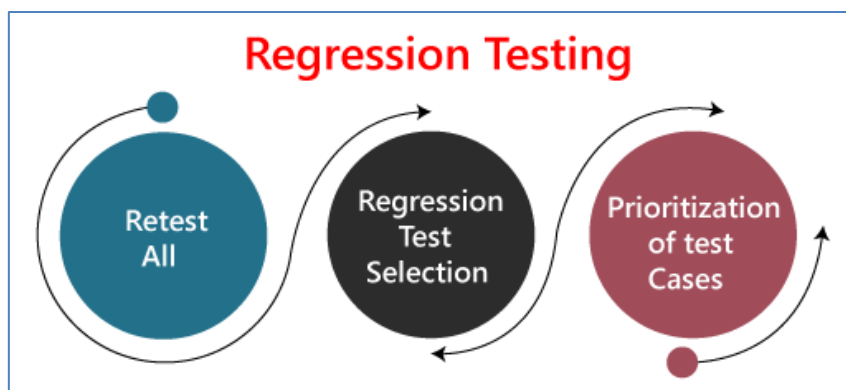
- Regression testing is making sure that the product **works** fine with new **functionality**, **bug fixes**, or any change in the **existing** feature.
- Regression testing is a type of **software testing**. Test cases are re-executed to check the previous functionality of the application is **working fine**, and the new changes have not **produced any bugs**.

### **Need of Regression Testing:**

This testing is required when there is any:

- Change in **requirements** and code is **modified** as per the **changed requirements**
- Added new **features** in product
- **Bug fixing**
- **Fixing of performance related** issues

Regression testing can be performed using the following **techniques**:



### **1. Re-test All:**

Re-Test is one of the approaches to do regression testing. In this approach, all the test case **suits should be re-executed**. Here we can define re-test as when a test **fails**, and we determine the cause of the failure is a **software fault**. The **fault is reported**, we can expect a new version of the software in which defect fixed. In this case, we will need to execute the test **again to confirm** that the fault fixed. This is known as re-**testing**. Some will refer to this as confirmation testing.

The re-**test is very expensive**, as it requires **enormous time and** resources.

### **2. Regression test Selection:**

- In this technique, a selected test-case **suit will execute rather than an entire test-case** suit.
- The selected test case suits divided in two cases
  1. Re**usable** Test cases.
  2. Ob**solete** Test cases.
- Reusable test cases can use in **succeeding regression cycle**.

- Obsolete test cases can't use in succeeding regression cycle.

### 3. Prioritization of test cases:

Prioritize the test case depending on business impact, critical and frequently functionality used. Selection of test cases will reduce the regression test suite.

## Types of Regression Testing:

The different types of Regression Testing are as follows:

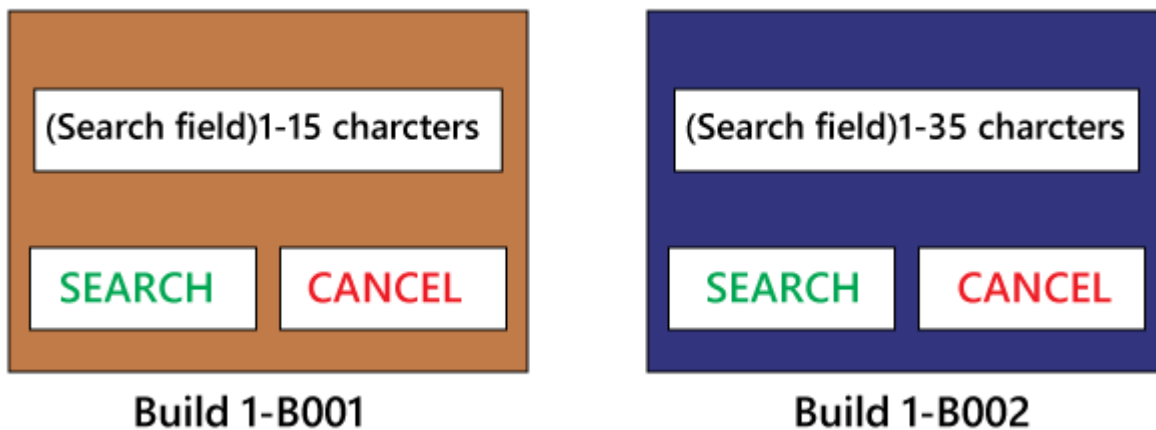
1. Unit Regression Testing [URT]
2. Regional Regression Testing[RRT]
3. Full or Complete Regression Testing [FRT]

### 1) Unit Regression Testing [URT]:

In this, we are going to test only the changed unit, not the impact area, because it may affect the components of the same module.

#### Example1

In the below application, and in the first build, the developer develops the **Search** button that accepts **1-15 characters**. Then the test engineer tests the Search button with the help of the **test case design technique**.



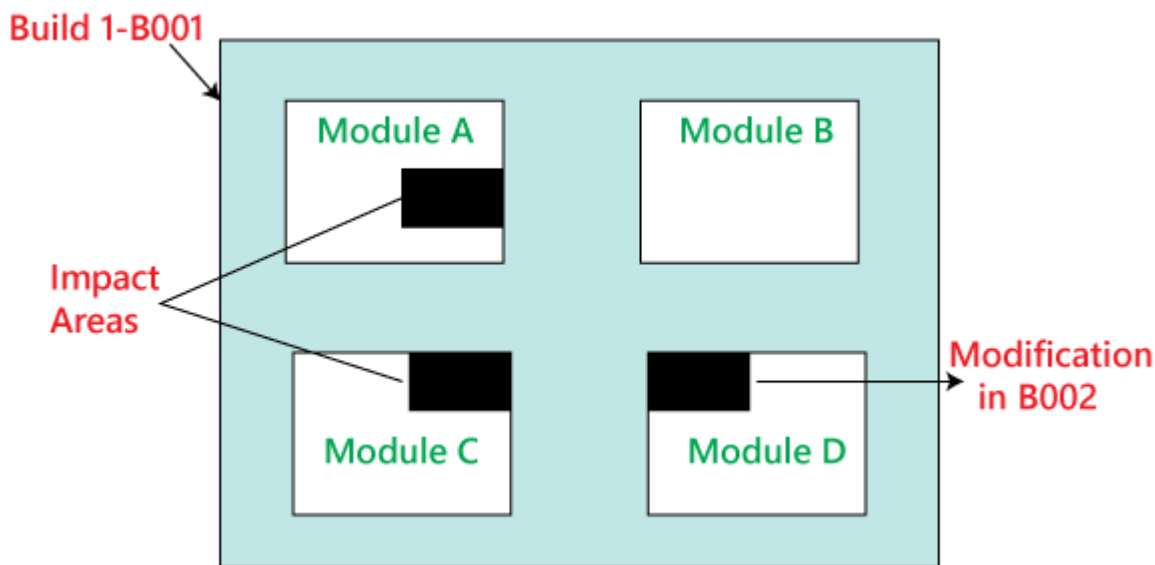
Now, the client does some modification in the requirement and also requests that the **Search button** can accept the **1-35 characters**. The test engineer will test only the Search button to verify that it takes 1-35 characters and does not check any further feature of the first build.

## 2) Regional Regression testing [RRT]:

In this, we are going to test the modification along with the impact area or regions, are called the **Regional Regression testing**. Here, we are testing the impact area because if there are dependable modules, it will affect the other modules also.

### **For example:**

In the below image as we can see that we have four different modules, such as **Module A, Module B, Module C, and Module D**, which are provided by the developers for the testing during the first build. Now, the test engineer will identify the bugs in **Module D**. The bug report is sent to the developers, and the development team fixes those defects and sends the second build.



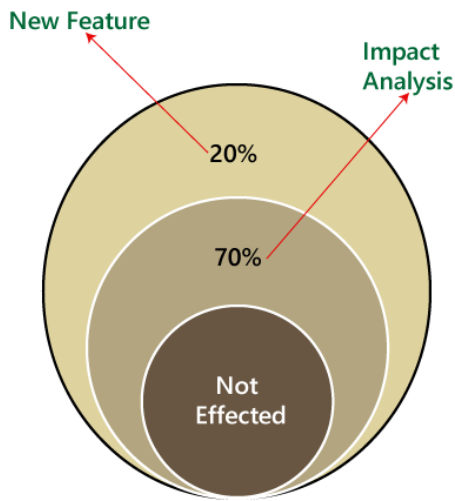
In the second build, the previous defects are fixed. Now the test engineer understands that the bug fixing in Module D has impacted some features in **Module A and Module C**. Hence, the test engineer first tests the Module D where the bug has been fixed and then checks the impact areas in **Module A and Module C**. Therefore, this testing is known as **Regional regression testing**.

## 3.Full Regression testing [FRT]

During the second and the third release of the product, the client asks for adding 3-4 new features, and also some defects need to be fixed from the previous release. Then the testing team will do the Impact Analysis and identify that the above modification will lead us to test the entire product.

Therefore, we can say that testing the **modified features** and **all the remaining (old) features** is called the **Full Regression testing**.





## Advantages of Regression Testing







Advantages of Regression Testing are:

- Regression Testing increases the product's quality.
- It ensures that any bug fix or changes do not impact the existing functionality of the product.
- Automation tools can be used for regression testing.
- It makes sure the issues fixed do not occur again.

## Disadvantages of Regression Testing

There are several advantages of Regression Testing though there are disadvantages as well.

- Regression Testing should be done for small changes in the code because even a slight change in the code can create issues in the existing functionality.
- If in case automation is not used in the project for testing, it will time consuming and tedious task to execute the test again and again.

Aspect	Black Box Testing	White Box Testing
<b>Focus</b> 	External behaviour and functionality 	Internal logic, code structure, and implementation 
<b>Knowledge</b> 	Tester doesn't need knowledge of internal code 	Tester has knowledge of internal code 

Aspect	Black Box Testing	White Box Testing
<b>Objective</b>	Validate correctness as per requirements/specifications	Verify code structure, optimization, and logic
<b>Testing Type</b>	Functional, non-functional, integration, acceptance	Unit testing, code coverage, security testing
<b>Approach</b>	Tests based on inputs and expected outputs	Tests based on code paths and data flow
<b>Test Design</b>	Inputs determined by requirements	Inputs designed to exercise code paths
<b>Test Cases</b>	Designed without knowing internal logic	Designed with knowledge of internal logic
<b>Independence</b>	Can be done by external parties (e.g., users)	Usually performed by developers or testers

- Usually the program fails due to race conditions, as the possibility of preceding B in restricted condition has not been taken care, resulting in a race condition bug.
- In this way, there may be many race conditions in the system, especially in multiprocessing and interactive systems. Race conditions are among the least tested.

## 1.4 SOFTWARE FAILURE CASE STUDIES

- Software systems have become the backbone of almost all the organizations worldwide and how much ever you try to avoid them you end up using software systems in your daily life as a person and as an organization.
- With over 2.9 billion\* of world population on Internet and rapid modernization of countries across the world, it has become inevitable to avoid the software footprint in your everyday life. Adoption of smart phones has made access to software applications more of a convenience and necessity.
- With mission critical and high-risk applications which have human lives and resources on risk depending on software applications, testing not only for expected but aiming for zero defects is required. We have listed the **Top 10 Mega software failures** which resulted in severe disruption and loss of resources in the current year which could have been avoided.

### 1.4.1 Top 8 Mega Software Failures

- |                                      |                                     |
|--------------------------------------|-------------------------------------|
| (1) Amazon Christmas Glitch          | (2) Air traffic control centre NATS |
| (3) Microsoft Azure Crashes          | (4) Bitcoin Exchange Collapse       |
| (5) Screwfix £34.99 Price Glitch     | (6) HMRC's Big Tax Blunder          |
| (7) UK border and immigration system | (8) Sony Pictures Entertainment     |

#### ► (1) Amazon Christmas Glitch

- It was quite a surprise for vendors to see their products on sale for just One penny in Amazon marketplace.
- It was a festive bonanza for shoppers and many picked up items as expensive as mobile phones for just 1 penny.
- This glitch was attributed to a bug in Amazon price comparison software and resulted in \$100,000 for the vendors.

#### ► (2) Air traffic control centre NATS

Another Christmas incident which affected the travel plans of more than 10000 passengers and leading to heavy delays was attributed to one line of software code which was unaltered since late 1960 and written in defunct language Jovial.



**► (3) Microsoft Azure Crashes**

- Microsoft's office 365, Xbox live gaming and Websites using the Azure platform crashed due a bug in the famous cloud computing platform.
- Many customers were unable to access the service due to this glitch which was following a performance update.
- On the Azure blog, Microsoft stated : *"The configuration change for the Blob [Binary Large Object] front-ends exposed a bug in the Blob front-ends, which had been previously performing as expected."* Service was down for more than 11 hours.

**► (4) Bitcoin Exchange Collapse**

Not implementing a feature which keeps track of the transactions proved costly for Mt. Gox, a Bitcoin exchange when lost \$500 million of its virtual currency when someone hacked the exchange.

**► (5) Screwfix £34.99 Price Glitch**

- A price glitch at screwfix's website was the reason for customer's joy as all the items went on Sale for just £34.99. From garden tractors to expensive power tools, all the items priced to the delight of the shoppers.
- All this was attributed to a Data validation error and requires a redo at the testing and validation solutions used by the business. Automation testing could have avoided this glitch as the script would have flagged the glitch immediately.

**► (6) HMRC's Big Tax Blunder**

- Her Majesty's Revenue and Customs (HMRC) which is responsible for collection of taxes in UK was hit by a bug in its PAYE (Pay As You Earn) system which has affected more than 5.7 million people.
- Error in PAYE resulted in wrong tax code allotted to tax payers due to which many paid more than the actual tax and many ended up paying less tax for the last 2 financial years.

**► (7) UK border and immigration system**

- One of the costliest software failure which is estimated to cost up to £1 billion. The system was incapable of dealing with backlog cases and resulted in 29000 applications backlog.
- The department also failed to locate 50000 people when was asked to find about them. What they needed was a comprehensive system wide IT strategy with skilled staff to avoid such issues.