Chapter 5

TESTING TECHNIQUES AND MAINTENANCE

5.1. Software Testing

- It is a method to assess the functionality of the software program. The process checks whether the actual software matches the expected requirements and ensures the software is bug-free. The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Software Testing Can be Divided into Two Steps

Verification: It refers to the set of tasks that ensure that the software correctly implements a specific function. It means "Are we building the product right?".

Validation: It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means "Are we building the right product?".

Importance of Software Testing

- Defects can be Identified Early: Software testing is important because if there are any bugs they can be identified early and can be fixed before the delivery of the software.
- Improves Quality of Software: Software Testing uncovers the defects in the software, and fixing them improves the quality of the software.
- Increased Customer Satisfaction: Software testing ensures reliability, security, and high performance which results in saving time, costs, and customer satisfaction.
- Helps with Scalability: Software testing type non-functional testing helps to identify the scalability issues and the point where an application might stop working.
- Saves Time and Money: After the application is launched it will be very difficult to trace and resolve the issues, as performing this activity will incur more costs and time. Thus, it is better to conduct software testing at regular intervals during software development.

5.2. Verification and Validation:

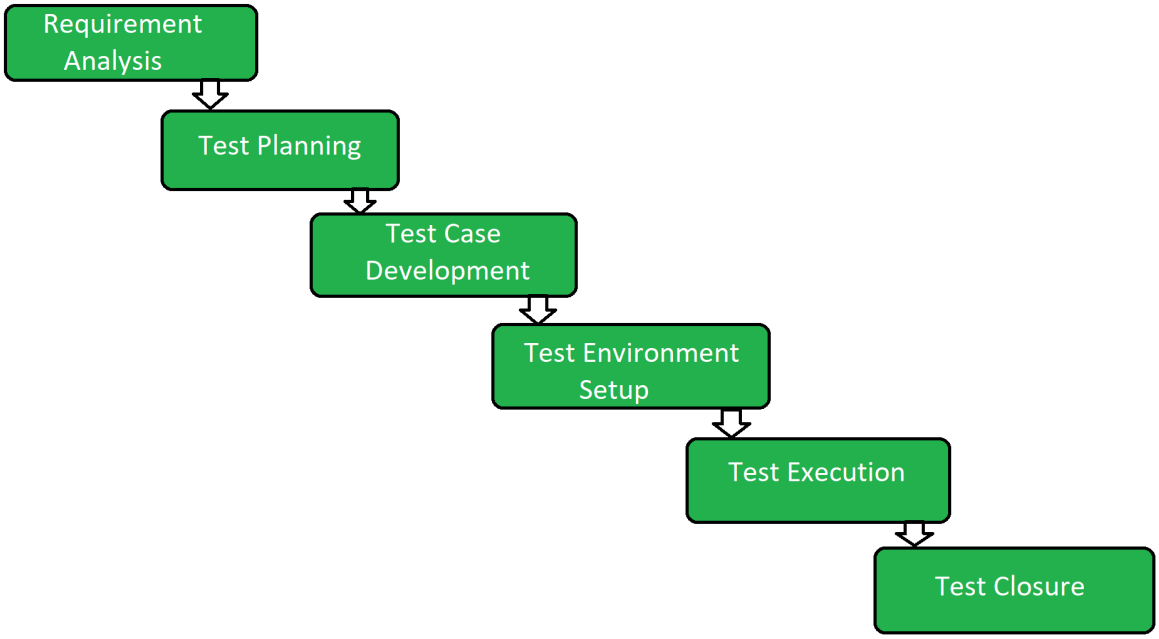
- Verification is the process of checking that software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfils the requirements that we have. Verification is static testing. Verification means Are we building the product right?
- Validation is the process of checking whether the software Product is up to the mark or in other words product has high-level requirements.
- It is the process of checking the validation of the product i.e. it checks what we are developing is the right product. It is validation of the actual and expected products. Validation is dynamic testing. Validation means Are we building the right product?

Differences between Verification and Validation

	Verification	Validation
Definition	Verification refers to the set of activities that ensure software correctly implements the specific function	Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements.
Focus	It includes checking documents, designs, codes, and programs.	It includes testing and validating the actual product.
Type of Testing	Verification is the static testing.	Validation is dynamic testing.
Execution	It does not include the execution of the code.	It includes the execution of the code.
Goal	The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Timing	It comes before validation.	It comes after verification.

5.3. 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.



Phases of STLC

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.

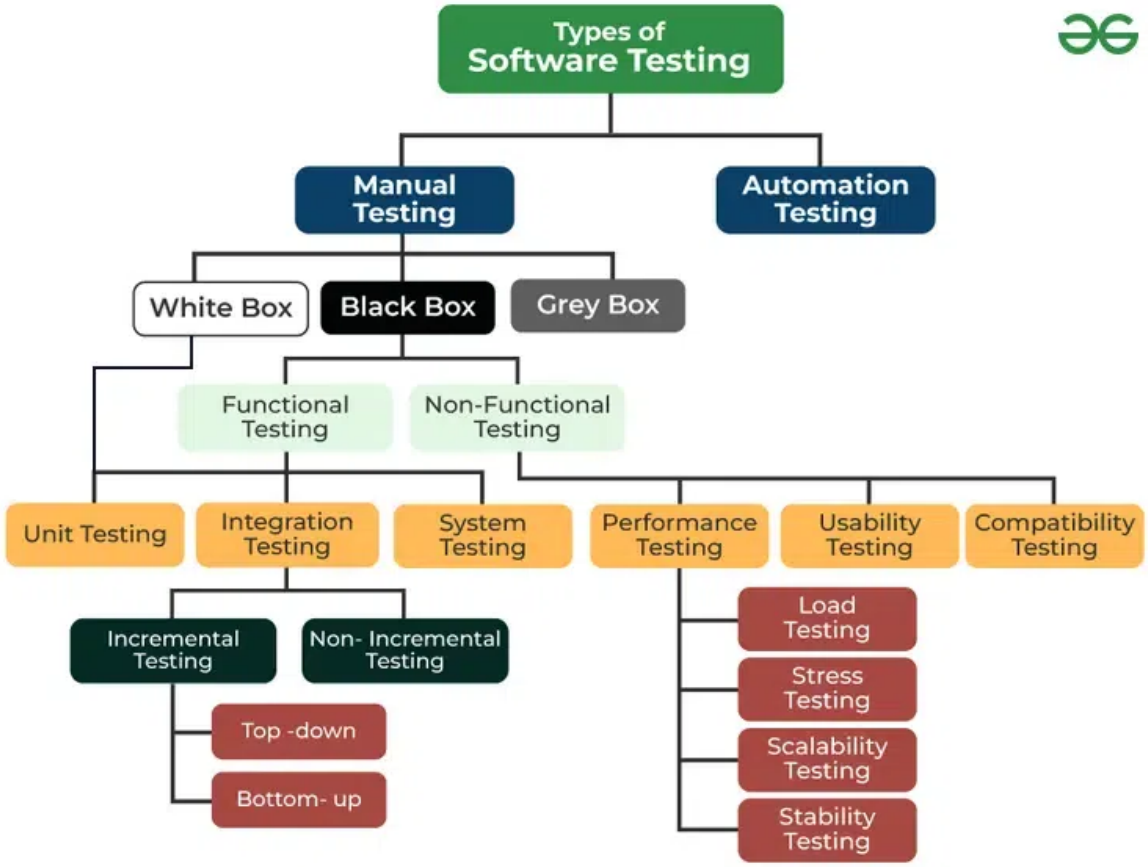
2. Test Planning: Test Planning is the most efficient phase of the software testing life cycle where all testing plans are defined. In this phase manager of the testing, team calculates the estimated effort and cost for the testing work. This phase gets started once the requirement-gathering phase is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



Types of Software Testing

Manual Testing

Automation Testing

1. **Manual Testing**

Manual testing is a technique to test the software that is carried out using the functions and features of an application. In manual software testing, a tester carries out tests on the software by following a set of predefined test cases. In this testing, testers make test cases for the codes, test the software, and give the final report about that software. Manual testing is time-consuming because it is done by humans, and there is a chance of human errors.

2. **Automation Testing**

Automated Testing is a technique where the Tester writes scripts on their own and uses suitable Software or Automation Tool to test the software. It is an Automation Process of a Manual Process. It allows for executing repetitive tasks without the intervention of a Manual Tester.

Here is the table of comparing Manual Testing and Automated Testing:

Parameters	Manual Testing	Automation Testing
Definition	In manual testing, the test cases are executed by the human tester.	In automated testing, the test cases are executed by the software tools.
Processing Time	Manual testing is time-consuming.	Automation testing is faster than manual testing.
Resources requirement	Manual testing takes up human resources.	Automation testing takes up automation tools and trained employees.
Exploratory testing	Exploratory testing is possible in manual testing.	Exploratory testing is not possible in automation testing.
Framework requirement	Manual testing doesn't use frameworks.	Automation testing uses frameworks like Data Drive, Keyword, etc.

Types of Manual Testing

White Box Testing

Black Box Testing

Gray Box Testing

1. **White Box Testing**

White box testing techniques analyze the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing clear box testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

Advantages of White box Testing:

- Thorough Testing: White box testing is thorough as the entire code and structures are tested.
- Code Optimization: It results in the optimization of code removing errors and helps in removing extra lines of code.
- Early Detection of Defects: It can start at an earlier stage as it doesn't require any interface as in the case of black box testing.
- Integration with SDLC: White box testing can be easily started in the Software Development Life Cycle.
- Detection of Complex Defects: Testers can identify defects that cannot be detected through other testing techniques.

2. Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the internal knowledge or implementation details of the software but rather focuses on validating the functionality based on the provided specifications or requirements.

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.
- It is used to find the ambiguity and contradictions in the functional specifications.

3**. Gray Box Testing**

- Gray Box Testing is a software testing technique that is a combination of the Black Box Testing technique and the White Box Testing technique.
- In the Black Box Testing technique, the tester is unaware of the internal structure of the item being tested and in White Box Testing the internal structure is known to the tester.

Advantages of Gray Box Testing:

- Clarity of goals: Users and developers have clear goals while doing testing.
- Done from a user perspective: Gray box testing is mostly done from the user perspective.
- High programming skills not required: Testers are not required to have high programming skills for this testing.
- Non-intrusive: Gray box testing is non-intrusive.
- Improved product quality: Overall quality of the product is improved.

Types of Black Box Testing

Functional Testing

Non-Functional Testing

1**. Functional Testing**

Functional Testing is a type of Software Testing in which the system is tested against the functional requirements and specifications. Functional testing ensures that the requirements or specifications are properly satisfied by the application. This type of testing is particularly concerned with the result of processing. It focuses on the simulation of actual system usage but does not develop any system structure assumptions. The article focuses on discussing function testing.

Benefits of Functional Testing

- Bug-free product: Functional testing ensures the delivery of a bug-free and high-quality product.
- Customer satisfaction: It ensures that all requirements are met and ensures that the customer is satisfied.
- Testing focused on specifications: Functional testing is focused on specifications as per customer usage.

2**. Non-Functional Testing**

Non-functional Testing is a type of Software Testing that is performed to verify the non-functional requirements of the application. It verifies whether the behavior of the system is as per the requirement or not. It tests all the aspects that are not tested in functional testing. Non-functional testing is a software testing technique that checks the non-functional attributes of the system. Non-functional testing is defined as a type of software testing to check non-functional aspects of a

Benefits of Non-functional Testing

- Improved performance: Non-functional testing checks the performance of the system and determines the performance bottlenecks that can affect the performance.
- Less time-consuming: Non-functional testing is overall less time-consuming than the other testing process.
- Improves user experience: Non-functional testing like Usability testing checks how easily usable and user-friendly the software is for the users. Thus, focus on improving the overall user experience for the application.

Types of Functional Testing

Unit Testing

Integration Testing

System Testing

End-to-end Testing

Acceptance testing

1. Unit Testing

Unit testing is a method of testing individual units or components of a software application. It is typically done by developers and is used to ensure that the individual units of the software are working as intended. Unit tests are usually automated and are designed to test specific parts of the code, such as a particular function or method. Unit testing is done at the lowest level of the software development process , where individual units of code are tested in isolation.

Advantages of Unit Testing:

It helps to identify bugs early in the development process before they become more difficult and expensive to fix.

It helps to ensure that changes to the code do not introduce new bugs.

It makes the code more modular and easier to understand and maintain.

It helps to improve the overall quality and reliability of the software.

2. Integration Testing

Integration testing is a method of testing how different units or components of a software application interact with each other. It is used to identify and resolve any issues that may arise when different units of the software are combined. Integration testing is typically done after unit

testing and before functional testing and is used to verify that the different units of the software work together as intended.

Different Ways of Performing Integration Testing:

Top-down integration testing: It starts with the highest-level modules and differentiates them from lower-level modules.

Bottom-up integration testing: It starts with the lowest-level modules and integrates them with higher-level modules.

Big-Bang integration testing: It combines all the modules and integrates them all at once.

Incremental integration testing: It integrates the modules in small groups, testing each group as it is added.

Advantages of Integrating Testing

It helps to identify and resolve issues that may arise when different units of the software are combined.

It helps to ensure that the different units of the software work together as intended.

It helps to improve the overall reliability and stability of the software.

3. **System Testing**

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

Advantages of System Testing:

The testers do not require more knowledge of programming to carry out this testing.

It will test the entire product or software so that we will easily detect the errors or defects that cannot be identified during the unit testing and integration testing.

The testing environment is similar to that of the real-time production or business environment.

It checks the entire functionality of the system with different test scripts and also it covers the technical and business requirements of clients.

After this testing, the product will almost cover all the possible bugs or errors and hence the development team will confidently go ahead with acceptance testing.

4. End-to-end Testing

End-to-end testing is the type of software testing used to test entire software from starting to the end along with its integration with external interfaces. The main purpose of end-to-end testing is to identify system dependencies and to make sure that the data integrity and communication with other systems, interfaces and databases to exercise complete production.

5. Acceptance Testing

It is formal testing according to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers, or other authorized entities to determine whether to accept the system or not.

Advantages of Acceptance Testing

This testing helps the project team to know the further requirements from the users directly as it involves the users for testing.

Automated test execution.

It brings confidence and satisfaction to the clients as they are directly involved in the testing process.

It is easier for the user to describe their requirement.

It covers only the Black-Box testing process and hence the entire functionality of the product will be tested.

Types of Integration Testing

Incremental Testing

Non-Incremental Testing

1. Incremental Testing

Like development, testing is also a phase of SDLC (Software Development Life Cycle) . Different tests are performed at different stages of the development cycle. Incremental testing is one of the testing approaches that is commonly used in the software field during the testing phase of integration testing which is performed after unit testing. Several stubs and drivers are used to test the modules one after one which helps in discovering errors and defects in the specific modules.

Advantages of Incremental Testing

Each module has its specific significance. Each one gets a role to play during the testing as they are incremented individually.

Defects are detected in smaller modules rather than denoting errors and then editing and re-correcting large files.

It's more flexible and cost-efficient as per requirements and scopes.

The customer gets the chance to respond to each building.

There are 2 Types of Incremental Testing

Top-down Integration Testing

Bottom-up Integration Testing

1. Top-down Integration Testing

Top-down testing is a type of incremental integration testing approach in which testing is done by integrating or joining two or more modules by moving down from top to bottom through the control flow of the architecture structure. In these, high-level modules are tested first, and then low-level modules are tested.

Advantages Top Down Integration Testing

There is no need to write drivers.

Interface errors are identified at an early stage and fault localization is also easier.

Low-level utilities that are not important are not tested well and high-level testers are tested well in an appropriate manner.

Representation of test cases is easier and simpler once Input-Output functions are added.

2. Bottom-up Integration Testing

Bottom-up Testing is a type of incremental integration testing approach in which testing is done by integrating or joining two or more modules by moving upward from bottom to top through the control flow of the architecture structure. In these, low-level modules are tested first, and then high-level modules are tested. This type of testing or approach is also known as inductive reasoning and is used as a synthesis synonym in many cases.

This testing results in high success rates with long-lasting results.

Advantages of Bottom-up Integration Testing

It is easy and simple to create and develop test conditions.

It is also easy to observe test results.

It is not necessary to know about the details of the structural design.

Low-level utilities are also tested well and are also compatible with the object-oriented structure.

Types of Non-functional Testing

Performance Testing

Usability Testing

Compatibility Testing

1. Performance Testing

Performance Testing is a type of software testing that ensures software applications perform properly under their expected workload. It is a testing technique carried out to determine system performance in terms of sensitivity, reactivity, and stability under a particular workload.

Performance testing is a type of software testing that focuses on evaluating the performance and scalability of a system or application. The goal of performance testing is to identify bottlenecks, measure system performance under various loads and conditions, and ensure that the system can handle the expected number of users or transactions.

Advantages of Performance Testing

Performance testing ensures the speed, load capability, accuracy, and other performances of the system.

It identifies, monitors, and resolves the issues if anything occurs.

It ensures the great optimization of the software and also allows many users to use it at the same time.

It ensures the client as well as the end-customer's satisfaction. Performance testing has several advantages that make it an important aspect of software testing:

Identifying bottlenecks : Performance testing helps identify bottlenecks in the system such as slow database queries, insufficient memory, or network congestion. This helps developers optimize the system and ensure that it can handle the expected number of users or transactions.

2. Usability Testing

You design a product (say a refrigerator) and when it becomes completely ready, you need a potential customer to test it to check it working. To understand whether the machine is ready to come on the market, potential customers test the machines. Likewise, the best example of usability testing is when the software also undergoes various testing processes which is performed by potential users before launching into the market. It is a part of the software development lifecycle (SDLC).

Advantages and Disadvantages of Usability Testing

Usability testing is preferred to evaluate a product or service by testing it with the proper users. In Usability testing, the development and design teams will use to identify issues before coding and the result will be earlier issues will be solved. During a Usability test, you can,

Learn if participants will be able to complete the specific task completely.

identify how long it will take to complete the specific task.

Gives excellent features and functionalities to the product

Improves user satisfaction and fulfills requirements based on user feedback

The product becomes more efficient and effective

3. Compatibility Testing

Compatibility testing is software testing that comes under the non functional testing category, and it is performed on an application to check its compatibility (running capability) on different platforms/environments. This testing is done only when the application becomes stable. This means simply this compatibility test aims to check the developed software application functionality on various software, hardware platforms, networks browser etc. This compatibility testing is very important in product production and implementation point of view as it is performed to avoid future issues regarding compatibility.

Advantages of Compatibility Testing

It ensures complete customer satisfaction.

It provides service across multiple platforms.

Identifying bugs during the development process.

There are 4 Types of Performance Testing

Load Testing

Stress Testing

Scalability Testing

Stability Testing

1. Load Testing

Load testing determines the behavior of the application when multiple users use it at the same time. It is the response of the system measured under varying load conditions.

The load testing is carried out for normal and extreme load conditions.

Load testing is a type of performance testing that simulates a real-world load on a system or application to see how it performs under stress.

The goal of load testing is to identify bottlenecks and determine the maximum number of users or transactions the system can handle.

Advantages of Load Testing:

Load testing has several advantages that make it an important aspect of software testing:

Identifying bottlenecks: Load testing helps identify bottlenecks in the system such as slow database queries, insufficient memory, or network congestion. This helps developers optimize the system and ensure that it can handle the expected number of users or transactions.

Improved scalability: By identifying the system's maximum capacity, load testing helps ensure that the system can handle an increasing number of users or transactions over time. This is particularly important for web-based systems and applications that are expected to handle a high volume of traffic.

2. Stress Testing

In Stress Testing, we give unfavorable conditions to the system and check how it perform in those conditions.

Example:

Test cases that require maximum memory or other resources are executed.

Test cases that may cause thrashing in a virtual operating system.

Test cases that may cause excessive disk requirement Performance Testing.

It is designed to test the run-time performance of software within the context of an integrated system. It is used to test the speed and effectiveness of the program. It is also called load testing. In it, we check, what is the performance of the system in the given load.

3. Scalability Testing

Scalability Testing is a type of non-functional testing in which the performance of a software application, system, network, or process is tested in terms of its capability to scale up or scale down the number of user request load or other such performance attributes. It can be carried out at a hardware, software or database level. Scalability Testing is defined as the ability of a network, system, application, product or a process to perform the function correctly when changes are made in the size or volume of the system to meet a growing need.

Advantages of Scalability Testing

It provides more accessibility to the product.

It detects issues with web page loading and other performance issues.

It finds and fixes the issues earlier in the product which saves a lot of time.

It ensures the end-user experience under the specific load. It provides customer satisfaction.

It helps in effective tool utilization tracking.

4. Stability Testing

Stability Testing is a type of Software Testing to checks the quality and behavior of the software under different environmental parameters. It is defined as the ability of the product to continue to function over time without failure.

Advantages of Stability Testing

It gives the limit of the data that a system can handle practically.

It provides confidence on the performance of the system.

It determines the stability and robustness of the system under load.

Stability testing leads to a better end-user experience.

5. User Acceptance Testing

User Acceptance Testing is a testing methodology where clients/end users participate in product testing to validate the product against their requirements. It is done at the client's site on the developer's site. For industries such as medicine or aerospace, contractual and regulatory compliance testing, and operational acceptance tests are also performed as part of user acceptance tests. UAT is context-dependent and UAT plans are prepared based on requirements and are not required to perform all kinds of user acceptance tests and are even coordinated and contributed by the testing team.

5.4. Re-engineering

- Software re-engineering, is the process of analyzing, designing, and modifying existing software systems to improve their quality, performance, and maintainability.
- This can include updating the software to work with new hardware or software platforms, adding new features, or improving the software's overall design and architecture.

Objective of Re-engineering

- The primary goal of software re-engineering is to improve the quality and maintainability of the software system while minimizing the risks and costs associated with the redevelopment of the system from scratch. Software re-engineering can be initiated for various reasons, such as:
- To describe a cost-effective option for system evolution.
- To describe the activities involved in the software maintenance process.
- To distinguish between software and data re-engineering and to explain the problems of data re-engineering.

Process of Software Re-engineering

The process of software re-engineering involves the following steps:

Process of Software Re-engineering

Planning: The first step is to plan the re-engineering process, which involves identifying the reasons for re-engineering, defining the scope, and establishing the goals and objectives of the process.

Analysis: The next step is to analyze the existing system, including the code, documentation, and other artifacts. This involves identifying the system's strengths and weaknesses, as well as any issues that need to be addressed.

Design: Based on the analysis, the next step is to design the new or updated software system. This involves identifying the changes that need to be made and developing a plan to implement them.

Implementation: The next step is to implement the changes by modifying the existing code, adding new features, and updating the documentation and other artifacts.

Testing: Once the changes have been implemented, the software system needs to be tested to ensure that it meets the new requirements and specifications.

Deployment: The final step is to deploy the re-engineered software system and make it available to end-users.

Why Perform Re-engineering?

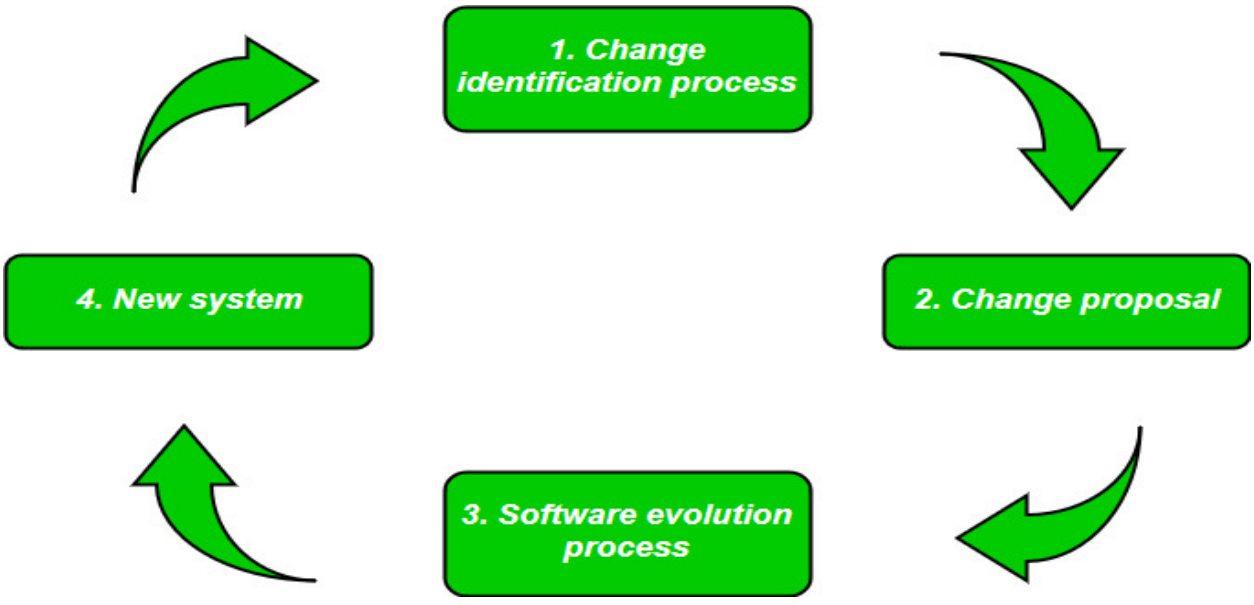
- To improve the software's performance and scalability: By analyzing the existing code and identifying bottlenecks, re-engineering can be used to improve the software's performance and scalability.
- To add new features: Re-engineering can be used to add new features or functionality to existing software.
- To support new platforms: Re-engineering can be used to update existing software to work with new hardware or software platforms.
- To improve maintainability: Re-engineering can be used to improve the software's overall design and architecture, making it easier to maintain and update over time.
- To meet new regulations and compliance: Re-engineering can be done to ensure that the software is compliant with new regulations and standards.
- Improving software quality: Re-engineering can help improve the quality of software by eliminating defects, improving performance, and enhancing reliability and maintainability.
- Updating technology: Re-engineering can help modernize the software system by updating the technology used to develop, test, and deploy the system.

5.5.Software Evolution:

- The software evolution process includes fundamental activities of change analysis, release planning, system implementation, and releasing a system to customers.
- The cost and impact of these changes are accessed to see how much the system is affected by the change and how much it might cost to implement the change.

Necessity of Software Evolution

- Change in requirement with time: With time, the organization's needs and modus Operandi of working could substantially be changed so in this frequently changing time the tools(software) that they are using need to change to maximize the performance.
- Environment change: As the working environment changes the things(tools) that enable us to work in that environment also changes proportionally same happens in the software world as the working environment changes then, the organizations require reintroduction of old software with updated features and functionality to adapt the new environment.
- Errors and bugs: As the age of the deployed software within an organization increases their preciseness or impeccability decrease and the efficiency to bear the increasing complexity workload also continually degrades.
- Security risks: Using outdated software within an organization may lead you to at the verge of various software-based cyberattacks and could expose your confidential data illegally associated with the software that is in use.



Steps in Software Evolution

5.6. Test Case Development Strategies

- Test case development is a critical part of software testing. It involves creating specific scenarios to validate the functionality, performance, and reliability of the software. Below are the key strategies for developing effective test cases:

1. **Boundary Value Analysis (BVA)**

A testing technique that focuses on testing the boundaries between partitions of input values.

Errors often occur at the edges of input ranges.

How it works:

Identify the input range (e.g., 1 to 100).

Test values at the boundaries (e.g., 0, 1, 100, 101).

Include valid and invalid boundary values.

Example:

For an input field accepting values from 1 to 100, test:

Minimum boundary: 0 (invalid), 1 (valid).

Maximum boundary: 100 (valid), 101 (invalid).

Advantages:

Efficient in finding errors at the edges of input domains.

Reduces the number of test cases while maintaining coverage.

2. **Equivalence Partitioning (EP)**

- A technique that divides input data into partitions or classes that are expected to behave similarly.
- Reduces redundancy by testing one value from each partition.

How it works:

Identify valid and invalid partitions.

Select one representative value from each partition for testing.

Example:

For an input field accepting values from 1 to 100:

Valid partition: 1-100 (test with 50).

Invalid partition: <1 (test with 0) and >100 (test with 101).

Advantages:

Reduces the number of test cases.

Ensures coverage of all input scenarios.

3. **Basis Path Testing**

- A white-box testing technique that ensures all possible paths through a program's control flow are tested.
- Ensures complete coverage of the code's logic.

How it works:

Create a control flow graph (CFG) to represent the program's logic.

Calculate the cyclomatic complexity (number of independent paths).

Develop test cases to cover each path.

Example:

For a program with conditional statements (if-else), test all possible outcomes of the conditions.

Advantages:

Ensures thorough testing of all logical paths.

Helps identify untested parts of the code.

4. **Control Structure Testing**

- A testing technique that focuses on the control structures of a program, such as loops and conditionals.
- Ensures that control structures function correctly.

Types:

Condition Testing:

Tests logical conditions (e.g., if-else, switch-case).

Ensures all possible outcomes of conditions are tested.

Loop Testing:

Tests loops for correct iteration and termination.

Includes testing:

Zero iterations (loop not executed).

One iteration.

Multiple iterations.

Maximum iterations.

Example:

For a loop that runs 10 times, test:

Loop not executed (condition false initially).

Loop executed once.

Loop executed 10 times.

Loop executed more than 10 times (if possible).

Advantages:

Ensures control structures work as intended.

Identifies issues like infinite loops or incorrect conditions.