Mod 1.2

Verification:

Verification is checking that software achieves its goal without any bugs. It is the process to ensure whether the developed product is correct or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.

- It includes checking documents, designs, codes, and programs.
- Verification is the static testing.
- It does *not* include the execution of the code.
- Methods used in verification are reviews, walkthroughs, inspections, and desk-checking.
- It checks whether the software conforms to specifications or not.
- It can find the bugs in the early stage of the development.
- The goal of verification is application and software architecture and specification.
- The quality assurance team does verification.

High-Level Design:

High-Level Design in short HLD is the general system design refers to the overall system design. It describes the overall description/architecture of the application. It includes a description of system architecture, database design, brief description of systems, services, platforms, and relationships among modules. It is also known as macro level/system design. It is created by a solution architect. It converts the Business/client requirement into a High-Level Solution.

Low-Level Design:

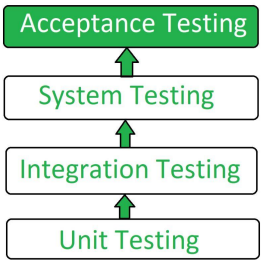
Low-level Level Design in short LLD is like detailing HLD to the component-level design process. It describes a detailed description of each module means it includes actual logic for every system component and it goes deep into each module specification. It is also known as micro-level/detailed design. It is created by designers and developers. It converts the High-Level Solution into a Detailed solution.

Mod 1.3

Validation

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 a validation of actual and expected products. Validation is dynamic testing.

- It includes checking documents, designs, codes, and programs.
- Validation is dynamic testing.
- It includes the execution of the code.
- Methods used in validation are Black Box Testing, White Box Testing, and non-functional testing.
- It checks whether the software meets the requirements and expectations of a customer or not.
- It can only find the bugs that could not be found by the verification process.
- The goal of validation is an actual product.
- Validation is executed on software code with the help of a testing team.



Unit Testing

- Unit testing involves the testing of each unit or an individual component of the software application.
- It is the first level of functional testing.
- The aim behind unit testing is to validate unit components with their performance.
- A unit is a single testable part of a software system tested during the development phase of the application software.
- The purpose of unit testing is to test the correctness of isolated code.
- A unit component is an individual function or code of the application.
- The white box testing approach is used for unit testing and is usually done by the developers.

Advantages

- Unit testing uses a module approach due to that any part can be tested without waiting for the completion of another part testing.
- The developing team focuses on the provided functionality of the unit and how functionality should look in unit test suits to understand the unit API.
- Unit testing allows the developer to refactor code after several days and ensure the module still working without any defects.

Disadvantages

- It cannot identify integration or broad-level errors as it works on units of the code.
- In unit testing, evaluation of all execution paths is not possible, so unit testing is not able to catch each error in a program.
- It is best suitable for conjunction with other testing activities.

Integration Testing

- Integration testing is the second level of the software testing process comes after unit testing.
- In this testing, units or individual components of the software are tested in a group.
- The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.
- The Software is developed with several software modules that are coded by different coders or programmers.
- The goal of integration testing is to check the correctness of communication among all the modules

Advantages

- It can enhance the quality and functionality of software by verifying the integration points and interfaces between components
- Reduce the cost and time of debugging and fixing errors by detecting them early on
- Improve the performance and reliability of software by testing the load, stress, and concurrency of components

Disadvantages

- Integration testing can be challenging due to its complexity and difficulty, as it involves multiple components, dependencies, and scenarios.
- This type of testing requires more resources and tools to set up, execute, and maintain the integration test environment and cases.

System Testing

- System Testing is a type of software testing that is performed on a completely integrated system to evaluate the compliance of the system with the corresponding requirements.
- In system testing, integration testing passed components are taken as input.
- System testing detects defects within both the integrated units and the whole system.
- The result of system testing is the observed behavior of a component or a system when it is tested.
- System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or the context of both.
- System testing tests the design and behavior of the system and also the expectations of the customer.

Advantages

- 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.

Disadvantages

- Can be time-consuming and expensive.
- Requires adequate resources and infrastructure.
- Can be complex and challenging, especially for large and complex systems.
- Dependent on the quality of requirements and design documents.
- Limited visibility into the internal workings of the system.
- Can be impacted by external factors like hardware and network configurations.

Acceptance Testing

- Acceptance Testing is a method of software testing where a system is tested for acceptability.
- The major aim of this test is to evaluate the compliance of the system with the business requirements and assess whether it is acceptable for delivery or not.

Advantages

- This testing helps the project team to know the further requirements of 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.

Disadvantages

- Users should have basic knowledge about the product or application.
- Sometimes, users don’t want to participate in the testing process.
- The feedback for the testing takes a long time as it involves many users and the opinions may differ from one user to another user.
- The development team does not participate in this testing process.

Mod 2.1

Static Testing

- **Static Testing** is a type of Software Testing method that is performed to check the defects in software without actually executing the code of the software application.
- Static testing is performed in an early stage of development to avoid errors as it is easier to find sources of failures and it can be fixed easily.
- The errors that cannot be found using Dynamic Testing, can be easily found by Static Testing.
- Static testing is the verification process.

Inspection

Inspection is the verification of documents by the higher authority like the verification of software requirement specifications (SRS).

Review

In static testing, review is a process or technique that is performed to find potential defects in the design of the software. It is a process to detect and remove errors and defects in the different supporting documents like software requirements specifications. People examine the documents and sort out errors, redundancies, and ambiguities.

Walkthrough

It is performed by an experienced person or expert to check the defects so that there might not be problems further in the development or testing phase.

Mod 2.2

Dynamic Testing

- **Dynamic Testing** is a type of Software Testing that is performed to analyze the dynamic behavior of the code.
- It includes the testing of the software for the input values and output values that are analyzed. Dynamic Testing is performed to describe the dynamic behavior of code.
- It refers to the observation of the physical response from the system to variables that are not constant and change with time.
- To perform dynamic testing the software should be compiled and run.
- It includes working with the software by giving input values and checking if the output is as expected by executing particular test cases which can be done either manually or with an automation process.
- Dynamic Testing is a validation process.

There are various levels of Dynamic Testing. They are:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

Advantages

- It discloses very difficult and complex defects.
- It detects the defects that can’t be detected by static testing.
- It increases the quality of the software product or application being tested.
- Dynamic testing detects security threats and ensures a better secure application.
- It can be used to test the functionality of the software at the early stages of development.
- It is easy to implement and does not require any special tools or expertise.

Disadvantages

- It is a time-consuming process as in dynamic testing whole code is executed.
- It increases the budget of the software as dynamic testing is costly.
- Dynamic testing may require more resources than static testing.
- Dynamic testing may be less effective than static testing in some cases.
- It is difficult to cover all the test scenarios.
- It is difficult to find out the root cause of the defects.

White 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.
- White box testing is also known as structural testing or code-based testing, and it is used to test the software's internal logic, flow, and structure.
- The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.

Features

- White box testing helps to analyze the code coverage of an application, which helps to identify the areas of the code that are not being tested.
- White box testing requires access to the application's source code, which makes it possible to test individual functions, methods, and modules.
- Testers performing white box testing must know programming languages like Java, C++, Python, and PHP to understand the code structure and write tests.
- White box testing helps to identify logical errors in the code, such as infinite loops or incorrect conditional statements.
- White box testing can help to optimize the code by identifying any performance issues, redundant code, or other areas that can be improved.
- White box testing can also be used for security testing, as it allows testers to identify any vulnerabilities in the application's code.

Advantages

- White box testing is thorough as the entire code and structures are tested.
- It results in the optimization of code removing errors and helps in removing extra lines of code.
- It can start at an earlier stage as it doesn't require any interface as in the case of black box testing.
- Easy to automate.
- White box testing can be easily started in the Software Development Life Cycle.
- Easy Code Optimization.

Disadvantages

- It was very expensive.
- Redesigning code and rewriting code needs test cases to be written again.
- Testers are required to have in-depth knowledge of the code and programming language as opposed to black-box testing.
- Missing functionalities cannot be detected as the code that exists is tested.
- Very complex and at times not realistic.
- Much more chances of Errors in production.

Basis Path Testing

- Basis Path Testing is a white-box testing technique based on the control structure of a program or a module.
- Using this structure, a control flow graph is prepared and the various possible paths present in the graph are executed as a part of testing.
- Therefore, by definition, Basis path testing is a technique of selecting the paths in the control flow graph, that provide a basis set of execution paths through the program or module.
- Since this testing is based on the control structure of the program, it requires complete knowledge of the program's structure.

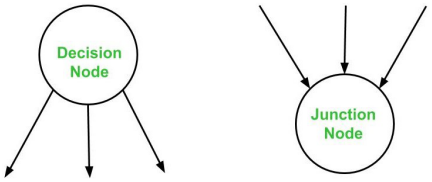
To design test cases using this technique, four steps are followed :

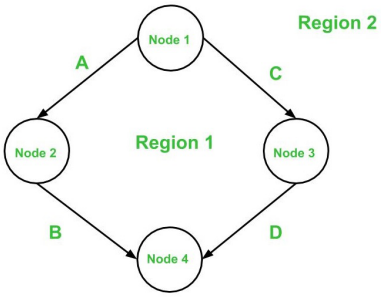
- Construct the Control Flow Graph
- Compute the Cyclomatic Complexity of the Graph
- Identify the Independent Paths
- Design Test cases from Independent Paths

Control Flow Graph

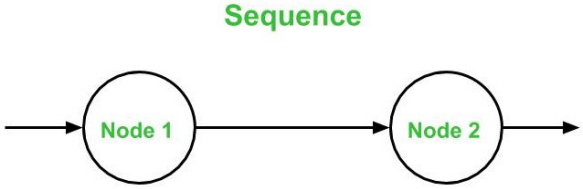
A control flow graph (or simply, flow graph) is a directed graph that represents the control structure of a program or module. A control flow graph (V, E) has V number of nodes/vertices and E number of edges in it. A control graph can also have :

- **Junction Node –** a node with more than one arrow entering it.
- **Decision Node –** a node with more than one arrow leaving it.
- **Region –** area bounded by edges and nodes (area outside the graph is also counted as a region.).

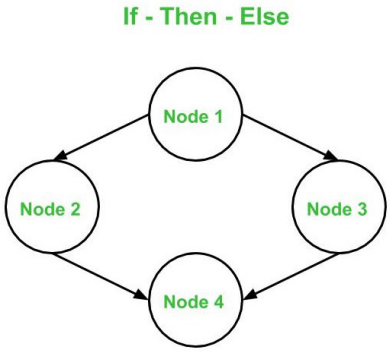




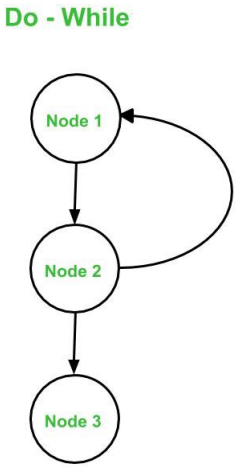
Sequential Statements



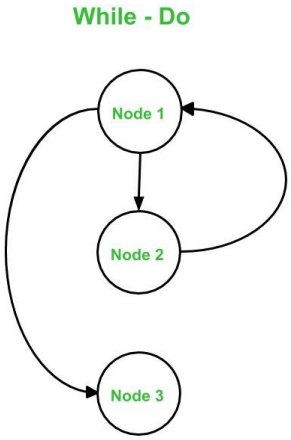
If-Then-Else



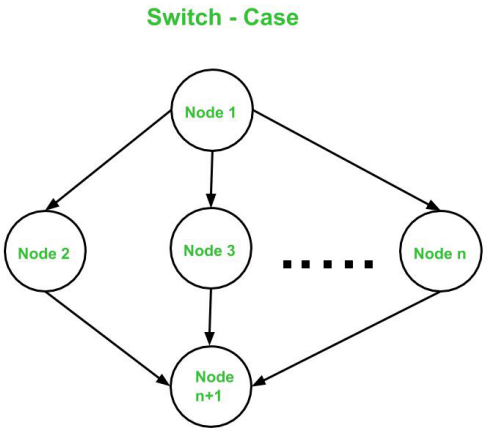
Do-While



While-Do



Switch-Case



Cyclomatic Complexity

The cyclomatic complexity V(G) is said to be a measure of the logical complexity of a program

V(G) = e - n + 2*P

Where e is the number of edges, n is the number of vertices, and P is the number of connected components.

V(G) = d + P

Where d is the number of decision nodes, and P is the number of connected nodes.

V(G) = number of regions in the graph

Advantages

- More Coverage
- Maintenance Testing
- Unit Testing
- Integration Testing
- Testing Effort

Example Video

https://www.youtube.com/watch?v=0Eix0yYVapw

Loop Testing

- Loop Testing is a type of software testing type that is performed to validate the loops.
- It is one of the types of Control Structure Testing.
- Loop testing is a white box testing technique and is used to test loops in the program.

The objective of Loop Testing is:

- To fix the infinite loop repetition problem.
- To know the performance.
- To identify the loop initialization problems.
- To determine the uninitialized variables.

Types

Simple Loop Testing:

- Testing performed in a simple loop is known as Simple loop testing.
- A simple loop is a normal “for”, “while” or “do-while” in which a condition is given, and the loop runs and terminates according to the true and false occurrence of the condition respectively.
- This type of testing is performed basically to test the condition of the loop and whether the condition is sufficient to terminate the loop after some point in time**.**

For simple loops of size n, test cases are designed that:

- Skip the loop entirely
- Only one pass through the loop
- 2 passes
- m passes, where m < n
- n-1 ans n+1 passes

Nested Loop Testing:

- Testing performed in a nested loop is known as Nested loop testing.
- A nested loop is one loop inside another loop.
- In a nested loop there can be a finite number of loops inside a loop and there a nest is made.
- It may be either of any of three loops i.e., for, while, or do-while.

For nested loops:

- All the loops are set to their minimum count and we start from the innermost loop.
- Simple loop tests are conducted for the innermost loop and this is worked outwards till all the loops have been tested.

Concatenated Loop Testing:

- Testing performed in a concatenated loop is known as Concatenated loop testing.
- It is performed on the concatenated loops.
- Concatenated loops are loops after the loop.
- It is a series of loops.
- The difference between nested and concatenated is that in nested loop is inside the loop but here loop is after the loop.

For concatenated loops:

- Simple loop tests are applied for each.
- If they’re not independent, treat them like nesting.

Unstructured Loop Testing:

- Testing performed in an unstructured loop is known as Unstructured loop testing.
- An unstructured loop is a combination of nested and concatenated loops.
- It is a group of loops that are in no order.

Advantages

- Loop testing limits the number of iterations of the loop.
- Loop testing ensures that the program doesn’t go into an infinite loop process.
- Loop testing endures the initialization of every used variable inside the loop.
- Loop testing helps in the identification of different problems inside the loop.
- Loop testing helps in the determination of capacity.

Disadvantages

- Loop testing is mostly effective in bug detection in low-level software.
- Loop testing is not useful in bug detection.

Data Flow Testing

- Data Flow Testing is a type of structural testing.
- It is a method that is used to find the test paths of a program according to the locations of definitions and uses of variables in the program.

It is concerned with:

- Statements where variables receive values,
- Statements where these values are used or referenced.

For a statement number S-

DEF(S) = {X | statement S contains the definition of X}

USE(S) = {X | statement S contains the use of X}

If a statement is a loop or if condition then its DEF set is empty and the USE set is based on the condition of statements.

Reference or defined anomalies in the flow of the data are detected at the time of associations between values and variables.

The anomalies are:

- A variable is defined but not used or referenced,
- A variable is used but never defined,
- A variable is defined twice before it is used

Advantages

- To find a variable that is used but never defined,
- To find a variable that is defined but never used,
- To find a variable that is defined multiple times before it is used,
- Deallocating a variable before it is used.

Disadvantages

- It is a time-consuming and costly process
- Requires knowledge of programming languages

Example Video

https://www.youtube.com/watch?v=2HiLfyZbpRU

Mutation Testing

- Mutation Testing is a type of Software Testing that is performed to design new software tests and also evaluate the quality of already existing software tests.
- Mutation testing is related to the modification of a program in small ways.
- It focuses on helping the tester develop effective tests or locate weaknesses in the test data used for the program.
- Mutation testing can be applied to design models, specifications, databases, tests, and XML.
- It is a structural testing technique, which uses the structure of the code to guide the testing process.
- It can be described as the process of rewriting the source code in small ways to remove the redundancies in the source code.

Objectives

- To identify pieces of code that are not tested properly.
- To identify hidden defects that can't be detected using other testing methods.
- To discover new kinds of errors or bugs.
- To calculate the mutation score.
- To study error propagation and state infection in the program.
- To assess the quality of the test cases.

Types

- **Value Mutations:** In this type of testing the values are changed to detect errors in the program. A small value is changed to a larger value or a larger value is changed to a smaller value. In this testing basically, constants are changed.
- **Decision Mutations:** In decisions, mutations are logical, or arithmetic operators are changed to detect errors in the program.
- **Statement Mutations:** In statement mutations, a statement is deleted or it is replaced by some other statement.

Advantages

- It brings a good level of error detection to the program.
- It discovers ambiguities in the source code.
- It finds and solves the issues of loopholes in the program.
- It helps the testers to write or automate the better test cases.
- It provides more efficient programming source code.

Disadvantages

- It is highly costly and time-consuming.
- Some mutations are complex and hence it is difficult to implement or run against various test cases.
- Here, the team members who are performing the tests should have good programming knowledge.
- The selection of the correct automation tool is important to test the programs.

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
- It focuses on validating the functionality based on the provided specifications or requirements.

Black box testing can be done in the following ways:

- **Syntax-Driven Testing –** This type of testing is applied to systems that can be syntactically represented by some language. For example- compilers, language that can be represented by context-free grammar. In this, the test cases are generated so that each grammar rule is used at least once.
- **Equivalence partitioning –** It is often seen that many types of inputs work similarly so instead of giving all of them separately we can group them and test only one input of each group. i.e., if a test case in one class results in some error, other members of the class would also result in the same error.

Features

- Black box testing is performed by testers who are not involved in the development of the application, which helps to ensure that testing is unbiased and impartial.
- Black box testing is conducted from the perspective of an end user, which helps to ensure that the application meets user requirements and is easy to use.
- Testers performing black box testing do not have access to the application's internal code, which allows them to focus on testing the application's external behavior and functionality.
- Black box testing is typically based on the application's requirements, which helps to ensure that the application meets the required specifications.
- Black box testing can be performed using various testing techniques, such as functional testing, usability testing, acceptance testing, and regression testing.
- Black box testing is easy to automate using various automation tools, which helps to reduce the overall testing time and effort.
- Black box testing can be scaled up or down depending on the size and complexity of the application being tested.
- Testers performing black box testing have limited knowledge of the application being tested, which helps to ensure that testing is more representative of how the end users will interact with the application.

Advantages

- 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.

Disadvantages

- 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.

Boundary Value Analysis Testing

- Boundary Value Analysis is based on testing the boundary values of valid and invalid partitions.
- The behavior at the edge of the equivalence partition is more likely to be incorrect than the behavior within the partition, so boundaries are an area where testing is likely to yield defects.
- It checks for the input values near the boundary that have a higher chance of error.
- Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.
- A boundary value for a valid partition is a valid boundary value.
- A boundary value for an invalid partition is an invalid boundary value.
- For each variable we check-
    - Minimum value.
    - Just above the minimum.
    - Nominal Value.
    - Just below the maximum value.
    - Max value.

Advantages

- It is easier and faster to find defects using this technique because the density of defects at boundaries is higher.
- Instead of testing will all sets of test data, we only pick the one at the boundaries. So, the overall test execution time was reduced.

Disadvantages

- It works well when the product is under test.
- It cannot consider the nature of the functional dependencies of variables.
- BVA is quite rudimentary.

Example Video

https://www.youtube.com/watch?v=29V-Oby0JaE

Equivalence Class Testing

- **Equivalence class testing (Equivalence class Partitioning)** is a black-box testing technique used in software testing.
- This testing technique is better than many of the testing techniques like boundary value analysis, and worst-case testing in terms of time consumption and terms of precision of the test cases.
- The test cases generated are below the minimum boundary, above the maximum boundary, and within the minimum and maximum boundary.

Advantages

- Helps reduce the number of test cases, without compromising the test coverage.
- Reduces the overall test execution time as it minimizes the set of test data.
- Enables the testers to focus on smaller data sets, which increases the probability of uncovering more defects in the software product.
- Used in cases where performing exhaustive testing is difficult.

Disadvantages

- It does not consider the conditions for boundary value.
- The identification of equivalence classes relies heavily on the expertise of testers.
- Testers might assume that the output for all input data sets is correct, which can become a great hurdle in testing.

Example Video

https://www.youtube.com/watch?v=cN5nFS8qNf0

https://www.youtube.com/watch?v=zDMxb13ibqc

Usability testing

- Usability testing refers to evaluating a product or service by testing it with representative users.
- Typically, during a test, participants will try to complete typical tasks while observers watch, listen, and take notes.
- The goal is to identify any usability problems, collect qualitative and quantitative data, and determine the participant's satisfaction with the product.
- To run an effective usability test, you need to develop a solid test plan, recruit participants, and then analyze and report your findings.

Advantages

- Usability testing lets the design and development teams identify problems before they are coded.
- The earlier issues are identified and fixed, the less expensive the fixes will be in terms of both staff time and possible impact on the schedule.

During a usability test, you will:

- Learn if participants are able to complete specified tasks successfully and
- Identify how long it takes to complete specified tasks.
- Find out how satisfied participants are with your Web site or other product
- Identify changes required to improve user performance and satisfaction
- And analyze the performance to see if it meets your usability objectives

Accessibility testing

- Accessibility Testing is one of the Software Testing, in which the process of testing the degree of ease of use of a software application for individuals with certain disabilities.
- It is performed to ensure that any new component can easily be accessible by physically disabled individuals despite any respective handicaps.
- Accessibility testing is part of the system testing process and is somehow similar to usability testing.
- In the accessibility testing process, the tester uses the system or component as it would be used by individuals with disabilities.
- Individuals can have disabilities like visual disability, hearing disability, learning disability, or non-functional organs.
- Accessibility testing is a subset of usability testing where the users under consideration are specific people with disabilities.

This testing focuses on verifying both usability and accessibility. Some examples of such software are:

- Speech recognition software: This software changes the spoken words to text and works as an input to the computer system.
- Screen reader software: This software is used to help low vision or blind individuals to read the text on the screen with a braille display or voice synthesizer.
- Screen magnification software: This software is used to help vision-impaired persons as it will enlarge the text and objects on the screen, thus making reading easier.
- Special keyboard: Some specially designed keyboards are for individuals with motor control problems. These keyboards help them to type quickly.

Factors to Measure Web Accessibility

- Pop-ups: Pop-ups can confuse visually disabled users. The screen reader reads out the page from top to bottom and a sudden pop-up arrives the reader will start reading it first before the actual content.
- Language: It is very important to make sentences simple and easily readable for cognitively disabled users as they have learning difficulties.
- Navigation: It is important to maintain the consistency of the website and not to modify the web pages regularly. Adjusting to new layouts is time-consuming.
- Marque text: It is best practice to avoid shiny text and keep the text on the website simple.

Purpose of Accessibility Testing

- Cater market for disabled people: To serve to market for disabled people like individuals with blindness, deaf, and handicapped, to support social inclusion for people with disabilities as well as other categories of people like older people, and people living in rural areas.
- Abide by accessibility legislation: Government agencies have come out with legalizations that require IT products to be accessible to disabled people.
- Avoid potential lawsuits: In the past few companies like Netflix, Blue Apron, and Winn-Dixie were sued because their products were not disabled-friendly.

For more content: https://www.geeksforgeeks.org/software-testing-accessibility-testing/

Usability v/s Accessibility testing:

Usability Testing	Accessibility Testing
Here are the quality components of usability testing as defined by **Nielsen Norman Group**: • **Efficiency:** Users can perform tasks quickly. • **Learnability:** Users can accomplish basic tasks the first time they encounter the interface. • **Memorability:** Users can reestablish the same proficiency after not using the design for a long time. • **Errors:** When users make errors, they are provided with helpful information to correct them. • **Satisfaction:** Users have a pleasant experience using the website. Usability testing identifies user pain points and provides insights into whether or not the user perceives the website as elegant and easy to use.	The principles recommended for digital accessibility compliance as defined by the **WCAG** are as follows: • **Perceivable** - Information and user interface components are presented to users in ways they can perceive. • **Operable** - Functionality on any given website is available via keyboard-only navigation or with assistive devices. • **Understandable** - Content must be easy to read and understand for all users. • **Robust** - Compatibility with current and future user agents, including assistive technologies, is maximized. Digital accessibility testing will identify user pain points as well as unmet WCAG requirements. For comprehensive coverage, it is recommended to conduct both manual and automated accessibility testing.

Mod 2.3

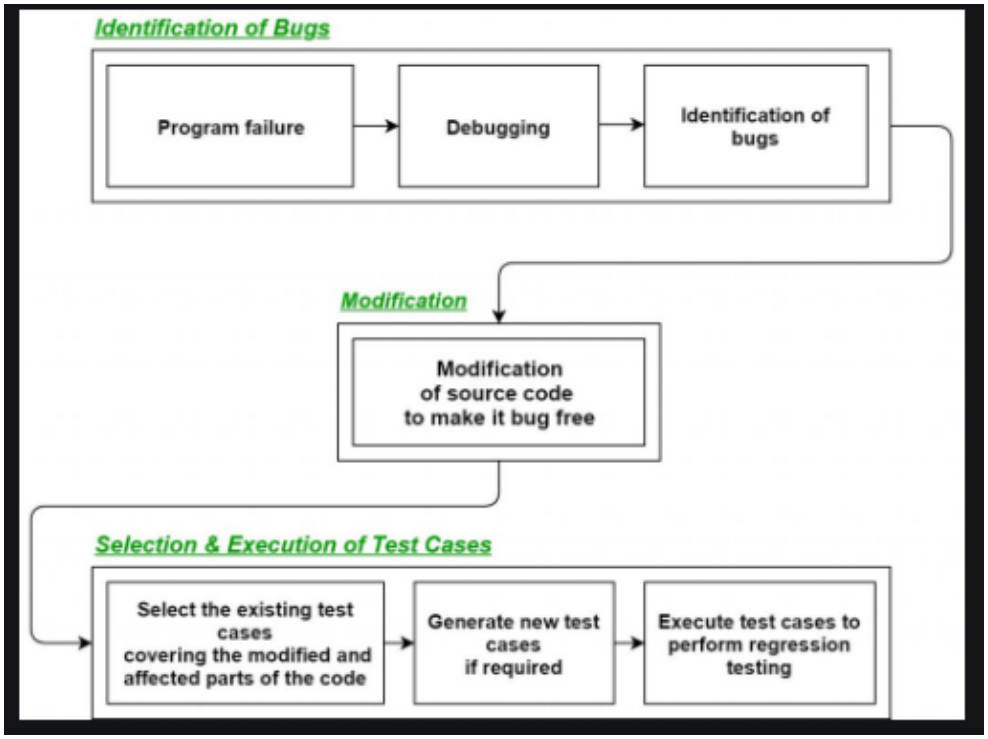
Regression testing

Regression Testing is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made. Regression means the return of something and in the software field, it refers to the return of a bug.

When to do regression testing?

- When a new functionality is added to the system and the code has been modified to absorb and integrate that functionality with the existing code.
- When some defect has been identified in the software and the code is debugged to fix it.
- When the code is modified to optimize its working.

Process of Regression testing



Objectives of regression testing

8.4 Objectives of Regression Testing

It tests to check that the bug has been addressed The first objective in bug-fix testing is to check whether the bug-fixing has worked or not. Therefore, you run exactly the same test that was executed when the problem was first found. If the program fails on this testing, it means the bug has not been fixed correctly and there is no need to do any regression testing further.

If finds other related bugs It may be possible that the developer has fixed only the symptoms of the reported bugs without fixing the underlying bug. Moreover, there may be various ways to produce that bug. Therefore, regression tests are necessary to validate that the system does not have any related bugs.

It tests to check the effect on other parts of the program It may be possible that bug-fixing has unwanted consequences on other parts of a program. Therefore, regression tests are necessary to check the influence of changes in one part on other parts of the program.

Need of regression testing

The purpose of regression testing is to ensure that bug fixes and new functionalities introduced in a new version of the software do not adversely affect the correct functionality inherited from the previous version. I
EEE Software Glossary defines regression testing as follows:
Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.

8.5 When is Regression Testing Done?

Software Maintenance

Corrective maintenance Changes made to correct a system after a failure has been observed (usually after general release).

Adaptive maintenance Changes made to achieve continuing compatibility with the target environment or other systems.

Perfective maintenance Changes designed to improve or add capabilities.

Preventive maintenance Changes made to increase robustness, maintainability, portability, and other features.

Types of Regression Testing

8.6 Regression Testing Types

Bug-Fix regression This testing is performed after a bug has been reported and fixed. Its goal is to repeat the test cases that expose the problem in the first place.

Side-Effect regression/Stability regression It involves retesting a substantial part of the product. The goal is to prove that the change has no detrimental effect on something that was earlier in order. It tests the overall integrity of the program, not the success of software fixes.