Unit 3

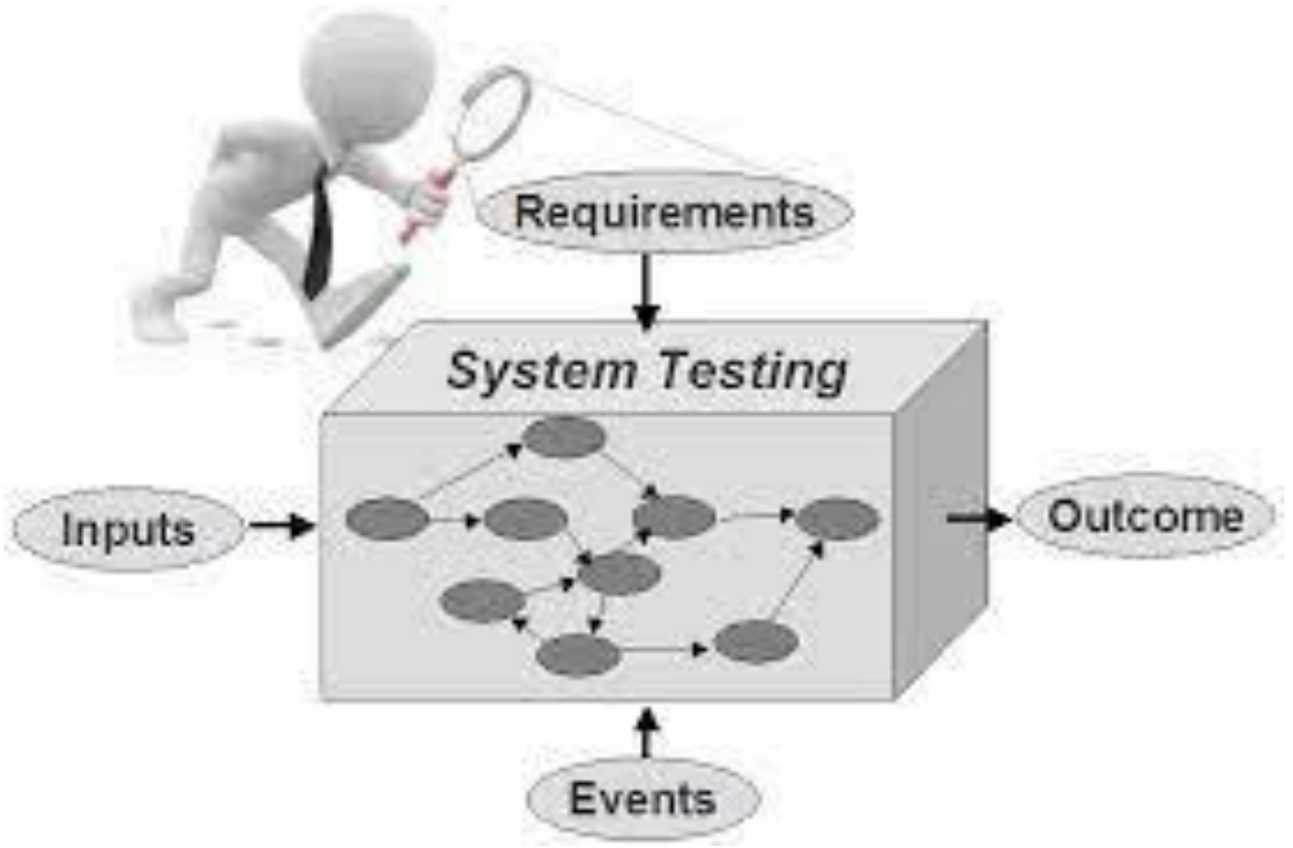
Software Testing

Computer Science and Engineering Department@SoEEC
SQAT(CSE5310)

Outline:

- Specification / Black Box Testing
- Specification-based test construction techniques
- Black Box Testing [Equivalence portioning, Boundary Value Analysis] ,
- White Box Testing and
- Grey Box Testing
- Its Advantages and Disadvantages

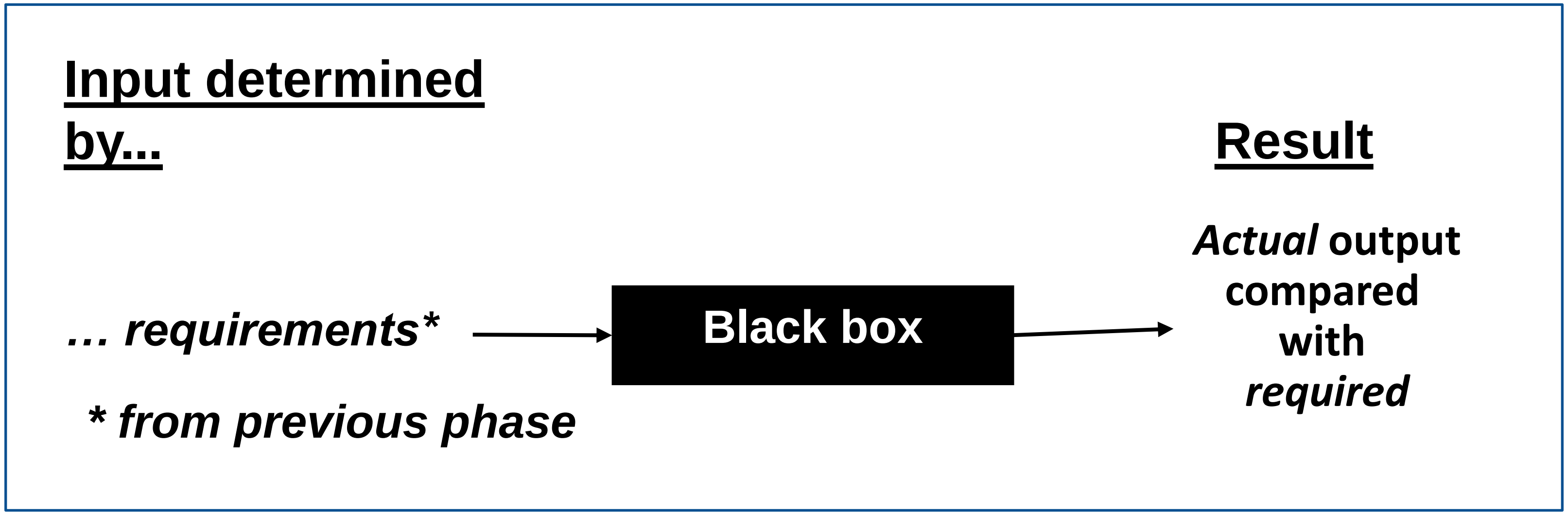
How do you test a system?



Specification Based Testing

- Classification based on the source of information to derive test cases:
  - Black-box testing (functional, specification-based) [

    Equivalence portioning, Boundary Value Analysis]
  - White-box testing (structural, program-based)
  - Grey box testing(Classes, Objects, Design)



Specification Based Testing

- There are numerous forms of testing all with a slightly different purpose and focus.
- We are going to examine and discuss the most common approaches, for instance, “**hand testing”** versus the use of tools and automated CASE style testing equipment. They are also implemented with varying degrees of rigor depending on the integrity level required for the software product being tested.

Black Box Testing

Black box testing verifies the functionality of an application—not necessarily having specific knowledge of the application’s code/internal structure. Black box tests are normally based on **requirements** and the **functionality** specified in them. Their purpose is to confirm that a given input reliably produces the anticipated output condition. This testing process is performed by quality assurance (QA) teams.

Specification Based Testing

White Box Testing

White box testing is based on the knowledge of the internal logic of an application’s code and includes tests such as coverage of code statements, branches, paths, and conditions. This testing process is generally performed by software developers by using a targeted set of prearranged test cases. The purpose of white box testing is to confirm internal integrity and logic of the artifact and code.

Gray Box Testing

Gray-box testing is a combination of white-box testing and black-box testing. The aim of this testing is to search for the defects, if any, due to improper structure or improper usage of applications. A gray-box tester partially knows the internal structure, which includes access to the documentation of internal data structures as well as the algorithms used. Gray-box testers require both high-level and detailed documents describing the application, which they collect in order to define test cases.

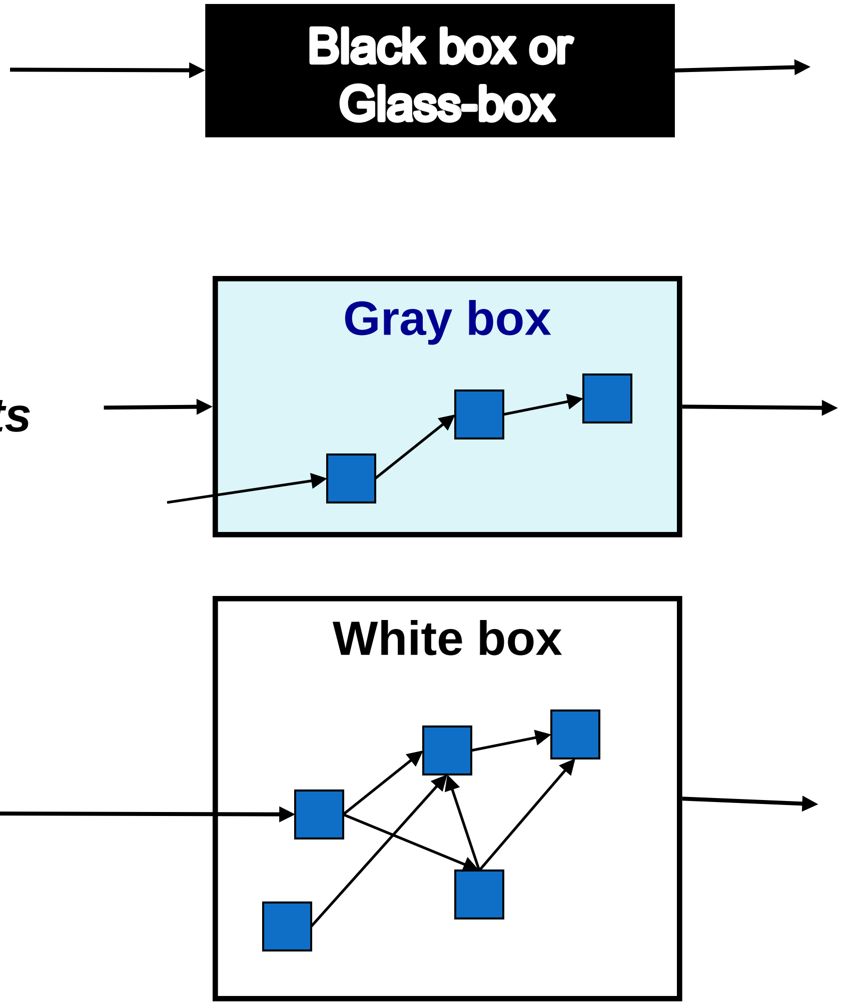
Black, Gray and White-box Testing

Input determined by...

… requirements

… requirements &
key design elements

…design
elements



***Actual* output compared with *required* output**

As for black- and white box testing

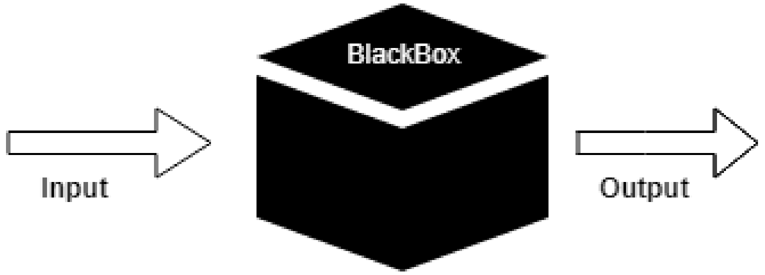
Confirmation of expected behavior

Characteristics of Testable Software

- **Operable:** The better it works (i.e., better quality), the easier it is to test
- **Observable:** Incorrect output is easily identified; internal errors are automatically detected
- **Controllable:** The states and variables of the software can be controlled directly by the tester
- **Decomposable:** The software is built from independent modules that can be tested independently
- **Simple:** The program should exhibit functional, structural, and code simplicity
- **Stable:** Changes to the software during testing are infrequent and do not invalidate existing tests
- **Understandable:** The architectural design is well understood; documentation is available and organized

Black Box Testing

- Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements 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 the expected output or not. If the function produces the correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



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.

Test Procedure

- The test procedure of black box testing is a kind of process in which the tester has specific knowledge about the software's work, and it develops test cases to check the accuracy of the software's functionality.
- It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function.
- A tester knows about the definite output of a particular input, but not about how the result is arising. There are various techniques used in black box testing for testing like decision table technique, boundary value analysis technique, state transition, All-pair testing, cause-effect graph technique, equivalence partitioning technique, error guessing technique, use case technique and user story technique.

Test Cases

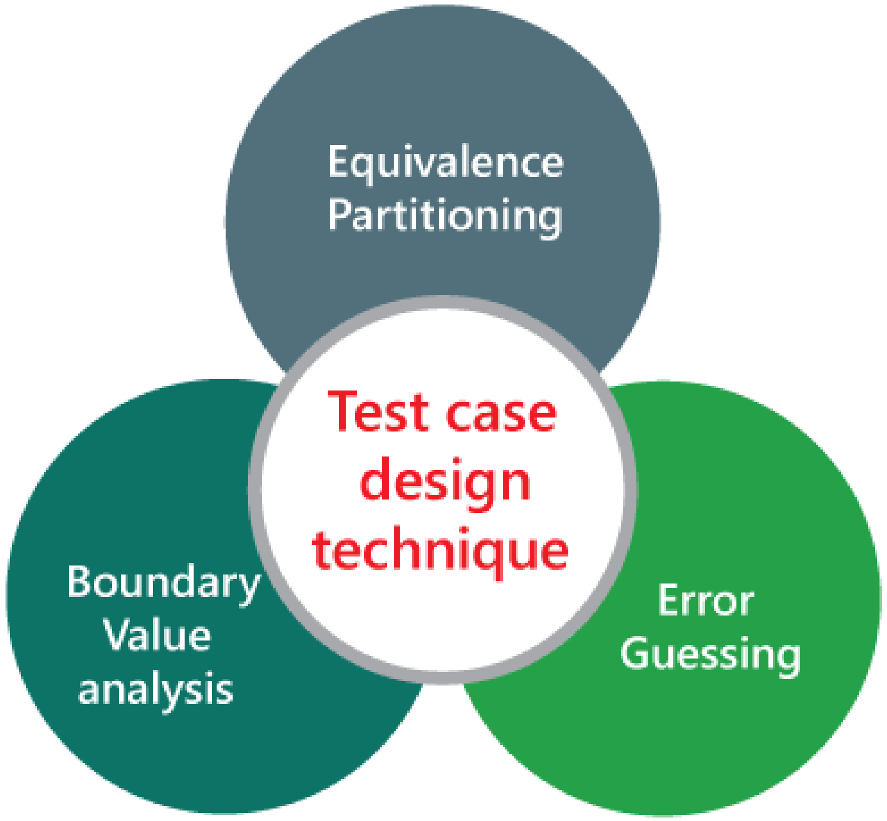
- Test cases are created considering the specification of the requirements. These test cases are generally created from working descriptions of the software including requirements, design parameters, and other specifications.
- For the testing, the test designer selects both positive test scenarios by taking valid input values and adverse test scenarios by taking invalid input values to determine the correct output.
- Test cases are mainly designed for functional testing but can also be used for non-functional testing. Test cases are designed by the testing team, there is no any involvement of the development team of software.

Test Case Design Techniques

- The test case design technique or methods or approaches that need to be followed by every test engineer while writing the test cases to achieve the maximum test coverage. If we follow the test case design technique, then it became process-oriented rather than person-oriented.
- The test case design technique ensures that all the possible values that are both positive and negative are required for the testing purposes. In software testing, there are three major test case design techniques which are as follows:
  1. Error Guessing
  2. Equivalence Partitioning
  3. Boundary Value Analysis[BVA]

Test Case Design Techniques

1: Error Guessing



1. Error Guessing

- In this section, we will see the first test case design technique that is Error guessing techniques.
- **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. It is a type of black box testing technique which does not have any defined structure to find the error.
- In this approach, every test engineer will derive the values or inputs based on their understanding or assumption of the requirements, and we do not follow any kind of rules to perform error guessing technique.
- The accomplishment of the error guessing technique is dependent on the ability and product knowledge of the tester because a good test engineer knows where the bugs are most likely to be, which helps to save lots of time.

Examples of Error guessing method:

- A function of the application requires a mobile number which must be of 10 characters. Now, below are the techniques that can be applied to guess error in the mobile number field:
  - What will be the result, if the entered character is other than a number?
  - What will be the result, if entered characters are less than 10 digits?
  - What will be the result, if the mobile field is left blank?
- After implementing these techniques, if the output is similar to the expected result, the function is considered to be bug-free, but if the output is not similar to the expected result, it is sent to the development team to fix the defects.
- However, error guessing is the key technique among all testing techniques as it depends on the experience of a tester, but it does not give surety of highest quality benchmark. It does not provide full coverage to the software. This technique can yield a better result if combined with other techniques of testing.

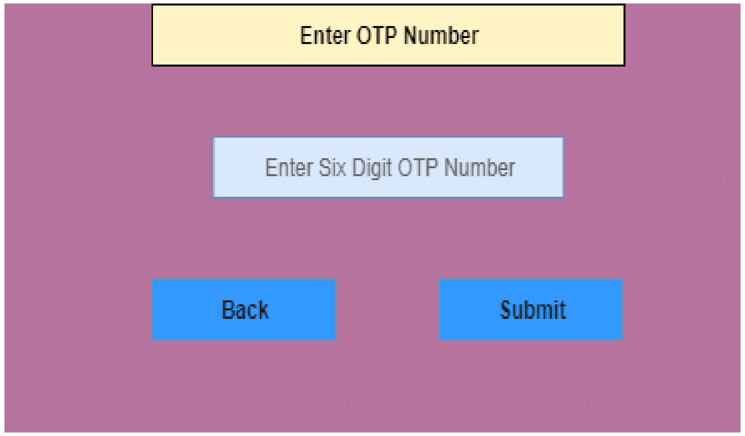
2. Equivalence Partitioning

- Equivalence partitioning is a technique of software testing in which input data is divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior. If a condition of one partition is true, then the condition of another equal partition must also be true, and if a condition of one partition is false, then the condition of another equal partition must also be false. The principle of equivalence partitioning is, test cases should be designed to cover each partition at least once. Each value of every equal partition must exhibit the same behavior as the other.
- The equivalence partitions are derived from the requirements and specifications of the software. The advantage of this approach is, it helps to reduce the time of testing due to a smaller number of test cases from infinite to finite. It is applicable at all levels of the testing process.

Examples of Equivalence Partitioning technique

Assume that there is a function of a software application that accepts a particular number of digits, not greater and less than that particular number. For example, an OTP number which contains only six digits, less or more than six digits will not be accepted, and the application will redirect the user to the error page.

1. OTP(One Time Password) Number = 6 digits

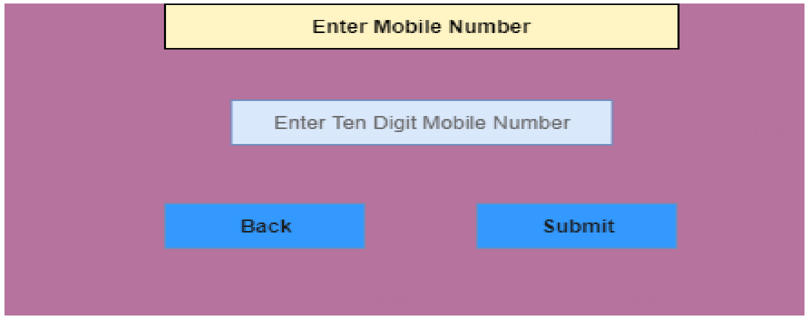


INVALID 1 Test case	INVALID 2 Test case	VALID 3 Test case	VALID
DIGITS >=7	DIGITS<=5	DIGITS = 6	DIGITS = 6
93847262	9845	456234	451483

Examples of Equivalence Partitioning technique

- A function of the software application accepts a 10 digit mobile number.

2. Mobile Number = 10 digits



INVALID 1 Test case	**INVALID** 2 Test case	**VALID** 3 Test case	**VALID**
DIGITS >=11	DIGITS<=9	DIGITS = 10	DIGITS =10
93847262219	984543985	9991456234	9893451483

Examples of Equivalence Partitioning technique

- In both examples, we can see that there is a partition of two equally valid and invalid partitions, on applying valid value such as OTP of six digits in the first example and mobile number of 10 digits in the second example, both valid partitions behave same, i.e. redirected to the next page.
- Another two partitions contain invalid values such as 5 or less than 5 and 7 or more than 7 digits in the first example and 9 or less than 9 and 11 or more than 11 digits in the second example, and on applying these invalid values, both invalid partitions behave same, i.e. redirected to the error page.
- We can see in the example, there are only three test cases for each example and that is also the principal of equivalence partitioning which states that this method intended to reduce the number of test cases.

3. Boundary Value Analysis

- Boundary value analysis is one of the widely used test case design technique for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.
- Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.
- Boundary values are those that contain the upper and lower limit of a variable. Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.
- The basic assumption of boundary value analysis is, the test cases that are created using boundary values are most likely to cause an error.

3. Boundary Value Analysis

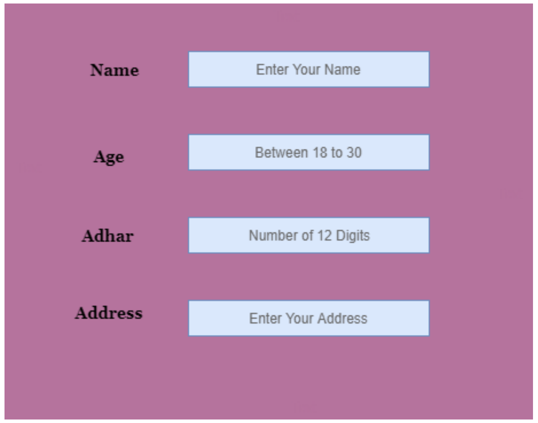
- The 18 and 30 are the boundary values that's why tester pays more attention to these values, but this doesn't mean that the middle values like 19, 20, 21, 27, 29 are ignored. Test cases are developed for each and every value of the range.

Testing of boundary values is done by making valid and invalid partitions. Invalid partitions are tested because testing of output in adverse conditions is also essential.

Example:

- Imagine, there is a function that accepts a number between 18 to 30, where 18 is the minimum and 30 is the maximum value of a valid partition, the other values of this partition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29.

- The invalid partition consists of numbers that are less than 18 such as 12, 14, 15, 16, and 17, and more than 30 such as 31, 32, 34, 36, and 40. Tester develops test cases for both valid and invalid partitions to capture the behavior of the system on different input conditions.



Name	Enter Your Name
Age	Between 18 to 30
Adhar	Number of 12 Digits
Address	Enter Your Address

```
12  14   15   16  17  18  20 22  24  25 26  28  30  31  32  34  36  38  40
--------------------------------|---------------------------------|-----------------------------
Invalid Partition                    Valid Partition                    Invalid Partition
```

Invalid test cases	Valid test cases	Invalid test cases
11, 13, 14, 15, 16, 17	18, 19, 24, 27, 28, 30	31, 32, 36, 37, 38, 39

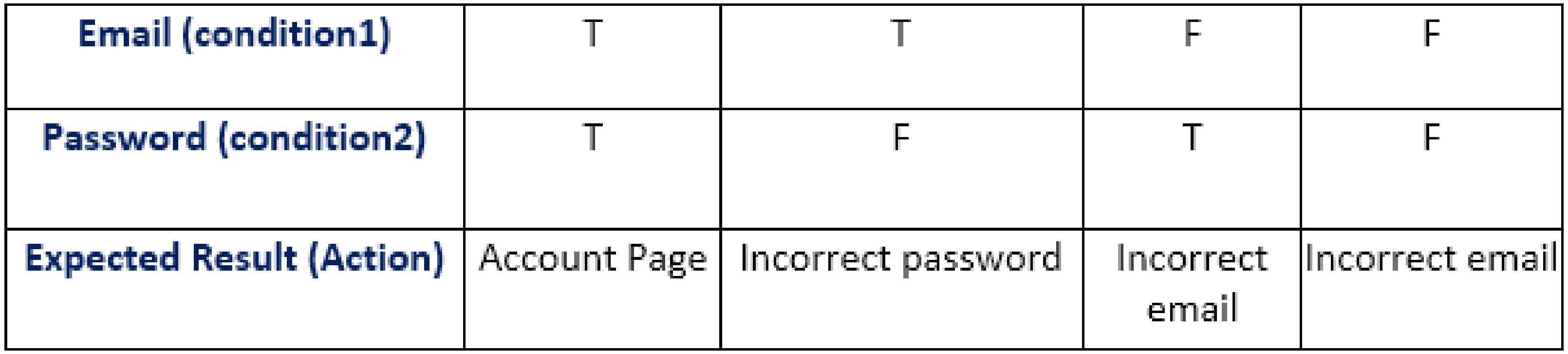
- The software system will be passed in the test if it accepts a valid number and gives the desired output, if it is not, then it is unsuccessful. In another scenario, the software system should not accept invalid numbers, and if the entered number is invalid, then it should display error massage.
- If the software which is under test, follows all the testing guidelines and specifications then it is sent to the releasing team otherwise to the development team to fix the defects.

Decision Table Technique in Black Box Testing

- Decision table technique is one of the widely used test case design techniques for black box testing. This is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form.
- That's why it is also known as a cause-effect table. This technique is used to pick the test cases in a systematic manner; it saves the testing time and gives good coverage to the testing area of the software application.
- Decision table technique is appropriate for the functions that have a logical relationship between two and more than two inputs.
- This technique is related to the correct combination of inputs and determines the result of various combinations of input. To design the test cases by decision table technique, we need to consider conditions as input and actions as output.

.....Decision Table Technique in Black Box Testing

- Most of us use an email account, and when you want to use an email account, for this you need to enter the email and its associated password.
- If both email and password are correctly matched, the user will be directed to the email account's homepage; otherwise, it will come back to the login page with an error message specified with "Incorrect Email" or "Incorrect Password."
- Now, let's see how a decision table is created for the login function in which we can log in by using email and password. Both the email and the password are the conditions, and the expected result is action.



Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

…..Decision Table Technique in Black Box Testing

- In the table, there are four conditions or test cases to test the login function. In the first condition if both email and password are correct, then the user should be directed to account's Homepage.
- In the second condition if the email is correct, but the password is incorrect then the function should display Incorrect Password. In the third condition if the email is incorrect, but the password is correct, then it should display Incorrect Email.
- Now, in fourth and last condition both email and password are incorrect then the function should display Incorrect Email.
- In this example, all possible conditions or test cases have been included, and in the same way, the testing team also includes all possible test cases so that upcoming bugs can be cured at testing level.
- In order to find the number of all possible conditions, **tester uses 2^n formula where n denotes the number of inputs**; in the exampl the number of inputs is 2 (one is true and second is false).
  - **Number of possible conditions = 2^ Number of Values of the second condition**
  - **Number of possible conditions =2^2 = 4**
- While using the decision table technique, a tester determines the expected output, if the function produces expected output, then it is passed in testing, and if not then it is failed. Failed software is sent back to the development team to fix the defect.

Black Box Testing

Advantages	Disadvantages
Well suited and efficient for large code segments.	Limited coverage, since only a selected number of test scenarios is actually performed.
Code access is not required.	Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
Clearly separates user's perspective from the developer's perspective through visibly defined roles.	Blind coverage, since the tester cannot target specific code segments or error prone areas.
Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.	The test cases are difficult to design.

1. Control Flow Testing

- Control flow testing is a testing technique that comes under white box testing. The aim of this technique is to determine 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. It is mostly used in unit testing. Test cases represented by the control graph of the program.
- **Control Flow Graph** is formed from the node, edge, decision node, junction node to specify all possible execution path.

Notations used for Control Flow Graph

1. Node
2. Edge
3. Decision Node
4. Junction node

Node: Nodes in the control flow graph are used to create a path of procedures. Basically, it represents the sequence of procedures in which procedure is next to come so, the tester can determine the sequence of occurrence of procedures.

We can see below in the example the first node represents the start procedure and the next procedure is to assign the value of n after assigning the value there is a decision node to decide the next node of the procedure as per the value of n if it is 18 or more than 18 so Eligible procedure will execute otherwise if it is less than 18 Not Eligible procedure executes. The next node is the junction node, and the last node is the stop node to stop the procedure.

Edge: The edge in the control flow graph is used to link the direction of nodes.

We can see below in the example all arrows are used to link the nodes in an appropriate direction.

Decision node: The decision node in the control flow graph is used to decide next node of the procedure as per the value.

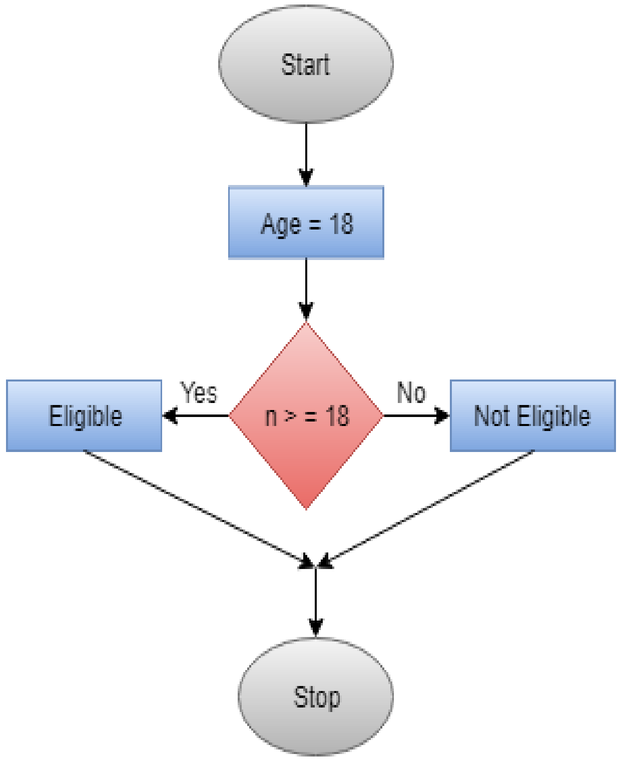
We can see below in the example decision node decide the next node of the procedure as per the value of n if it is 18 or more than 18 so Eligible procedure will execute otherwise if it is less than 18, Not Eligible procedure executes.

Junction node: The junction node in the control flow graph is the point where at least three links meet.

Example:

```
public class VoteEligiblityAge {

public static void main(String []args)
            {
int n=45;
if(n>=18)
            {
            System.out.println("You are eligible for voting");
             }  else
            {
              System.out.println("You are not eligible for voting");
             }
            }
  }
```



The above example shows eligibility criteria of age for voting where if age is 18 or more than 18 so print message "You are eligible for voting" if it is less than 18 then print "You are not eligible for voting."

Program for this scenario is written above, and the control flow graph is designed for the testing purpose.

In the control flow graph, start, age, eligible, not eligible and stop are the nodes, n>=18 is a decision node to decide which part (if or else) will execute as per the given value. Connectivity of the eligible node and not eligible node is there on the stop node.

Test cases are designed through the flow graph of the programs to determine the execution path is correct or not. All nodes, junction, edges, and decision are the essential parts to design test cases.

Bug in Software Testing

we will learn about defect/bug in software testing and why it occurs, basic terminology of a defect, and bug tracking tool.

What is a bug in software testing?

The Bug is the informal name of defects, which means that software or application is not working as per the requirement.

In software testing, a software bug can also be issue, error, fault, or failure. The bug occurred when developers made any mistake or error while developing the product.

….Bug in Software Testing

- While testing the application or executing the test cases, the test engineer may not get the expected result as per the requirement. And the bug had various names in different companies such as error, issues, problem, fault, and mistake, etc.
- Here are different terminology of defect:
  - **Defect**
  - **Bug**
  - **Error**
  - **Issue**
  - **Mistake**
  - **Failure**

Defect/Bug

Why defect/bug occur?

In software testing, the bug can occur for the following reasons:

- **Wrong coding**
- **Missing coding**
- **Extra coding**

- **Wrong coding :** Wrong coding means improper implementation.

  For example: Suppose if we take the Gmail application where we click on the **"Inbox"** link, and it navigates to the **"Draft"** page, this is happening because of the wrong coding which is done by the developer, that's why it is a bug.

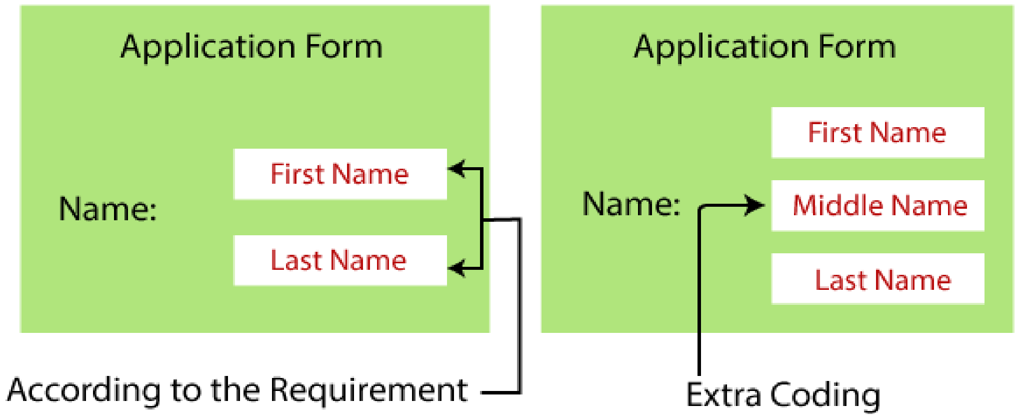
- **Missing coding :** Here, missing coding means that the developer may have developed the code only for that particular feature.

  For example: if we take the above example and open the inbox link, we see that it is not there, which means the feature is not developed.

- **Extra coding :** Here, extra coding means that the developers develop the extra features, which are not required according to the client's requirements.

For example:

- Suppose we have one application form wherein the Name field, the First name, and Last name textbox are needed to develop according to the client's requirement.
- But, the developers also develop the "Middle name" textbox, which is not needed according to the client's requirements as we can see in the below image:



- If we develop an extra feature that is not needed in the requirement, it leads to unnecessary extra effort. And it might also happen that adding up the extra feature affects the other elements too.

Bug Tracking Tool

- We have various types of bug tracking tools available in software testing that helps us to track the bug, which is related to the software or the application.
- Some of the most commonly used bug-tracking tools are as follows:
  - **Jira**
  - **Bugzilla**
  - **Redmine**
  - **MantisBT**
  - **Backlog**

White Box Testing

- White box testing is based on the knowledge of the internal logic of an application’s code and includes tests such as coverage of code statements, branches, paths, and conditions. This testing process is generally performed by software developers/Test Engineers by using a targeted set of prearranged test cases
- White box testing is a form of application testing that provides the tester with complete knowledge of the application being tested, including access to source code and design documents. This in-depth visibility makes it possible for white box testing to identify issues that are invisible to gray and black box testing.

What Does White Box Testing Focus On?

White box testing takes advantage of extensive knowledge of an application’s internals to develop highly-targeted test cases. Examples of tests that might be performed during white box testing include:

Path Checking: White box testing can be used to explore the various execution paths within an application to ensure that all conditional statements are correct, necessary, and efficient.

Output Validation: This enumerates the various potential inputs to a function and ensures that each produces the expected result.

Security Testing: **Static code analysis** and other white box testing techniques are used to identify potential vulnerabilities within an application and validate that it follows secure development best practices.

Loop Testing: Tests the loops within an application to ensure that they are correct, efficient, and properly manage the variables within their scope.

Data Flow Testing: Tracks variables throughout the execution paths of a program to ensure that variables are declared, initialized, used, and properly manipulated.

Types Of White Box Testing

The three types of white box testing are:

- **Unit Testing:** Unit testing is designed to ensure that each component or function of an application works properly. This helps to ensure that the application meets design requirements throughout the development process.
- **Integration Testing:** Integration testing focuses on the interfaces between the various components within an application. Performed after unit testing, it ensures that not only does each component work well in isolation but also that they can work together effectively.
- **Regression Testing:** Changes can break things within an application. Regression testing ensures that the code still passes existing test cases after functionality or security updates are made to an application.

White Box Testing Techniques

One of the main advantages of white box testing is that it makes it possible to ensure that every aspect of an application is tested. To achieve full code coverage, white box testing can use the following techniques:

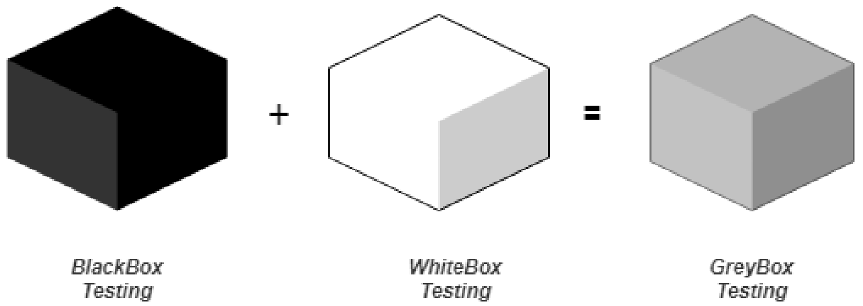
- **Data Flow Testing:** It is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.
- **Statement Coverage:** Statement coverage testing ensures that every line of code within an application is tested by at least one test case. Statement coverage testing can help to identify if portions of the code are unused or unreachable, which can be caused by programming errors, updates, etc. Identifying this dead code enables developers to fix incorrect conditional statements or remove redundant code to improve application performance and security.
- **Branch Coverage:** Conditional statements create branches within an application's execution code as different inputs can follow different execution paths. Branch coverage testing ensures that every branch within an application is covered by unit testing. This ensures that even little-used code paths are properly validated.
- **Path Coverage:** An execution path describes the sequence of instructions that can be executed from when an application starts to where it terminates. Path coverage testing ensures that every execution path through an application is covered by use cases. This can help to ensure that all execution paths are functional, efficient, and necessary.

White Box Testing

Advantages	Disadvantages
As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.	Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased.
It helps in optimizing the code.	Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested.
Extra lines of code can be removed which can bring in hidden defects.	It is difficult to maintain white-box testing, as it requires specialized tools like code analyzers and debugging tools.
Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.	

Grey Box Testing

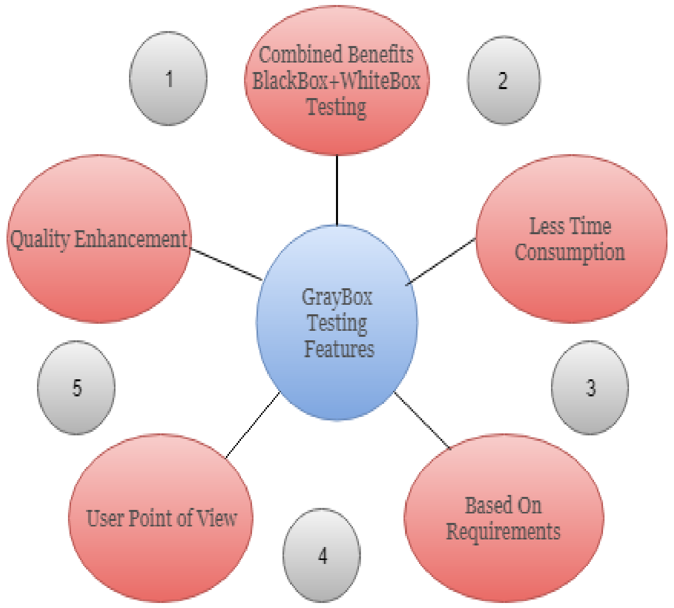
- Grey-box testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a combination of black box and white box testing because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.



- Grey-box testing commonly identifies context-specific errors that belong to web systems. For example; while testing, if tester encounters any defect then he makes changes in code to resolve the defect and then test it again in real time. It concentrates on all the layers of any complex software system to increase testing coverage. It gives the ability to test both presentation layer as well as internal coding structure. It is primarily used in integration testing and penetration testing.

Why Grey Box testing?

- Reasons for GreyBox testing are as follows
  - It provides combined benefits of both Black box and White Box testing.
  - It includes the input values of both developers and testers at the same time to improve the overall quality of the product.
  - It reduces time consumption of long process of functional and non-functional testing.
  - It gives sufficient time to the developer to fix the product defects.
  - It includes user point of view rather than designer or tester point of view.
  - It involves examination of requirements and determination of specifications by user point of view deeply.



Grey Box Testing Strategy

Grey box testing does not make necessary that the tester must design test cases from source code. To perform this testing test cases can be designed on the base of, knowledge of architectures, algorithm, internal states or other high -level descriptions of the program behavior. It uses all the straightforward techniques of black box testing for function testing. The test case generation is based on requirements and preset all the conditions before testing the program by assertion method.

- **Generic Steps to perform Grey box Testing are:**

1. First, select and identify inputs from BlackBox and WhiteBox testing inputs.
2. Second, Identify expected outputs from these selected inputs.
3. Third, identify all the major paths to traverse through during the testing period.
4. The fourth task is to identify sub-functions which are the part of main functions to perform deep level testing.
5. The fifth task is to identify inputs for subfunctions.
6. The sixth task is to identify expected outputs for subfunctions.
7. The seventh task includes executing a test case for Subfunctions.
8. The eighth task includes verification of the correctness of result.

The test cases designed for Greybox testing includes Security related, Browser related, GUI related, Operational system related and Database related testing.

….Grey Box Testing

Advantages	Disadvantages
Offers combined benefits of black-box and white-box testing wherever possible.	Since the access to source code is not available, the ability to go over the code and test coverage is limited.
Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications.	The tests can be redundant if the software designer has already run a test case.
Based on the limited information available, a grey-box tester can design excellent test scenarios especially around communication protocols and data type handling.	Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested.
The test is done from the point of view of the user and not the designer.	

Black-Box Testing	Grey-Box Testing	White-Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as closed-box testing, data-driven testing, or functional testing.	Also known as translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as clear-box testing, structural testing, or code-based testing.
Performed by end-users and also by testers and developers.	Performed by end-users and also by testers and developers.	Normally done by testers and developers.
Testing is based on external expectations - Internal behavior of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
It is exhaustive and the least time-consuming.	Partly time-consuming and exhaustive.	The most exhaustive and time-consuming type of testing.
Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.

Different Types of Software Testing

- In this section, we are going to understand the various types of software testing, which can be used at the time of the Software Development Life Cycle.
- As we know, **software testing** is a process of analyzing an application's functionality as per the customer prerequisite.
- If we want to ensure that our software is bug-free or stable, we must perform the various types of software testing because testing is the only method that makes our application bug-free.

The different types of Software Testing

- The categorization of software testing is a part of diverse testing activities, such as test strategy, test deliverables, a defined test objective, etc. And software testing is the execution of the software to find defects.
- The purpose of having a testing type is to confirm the **AUT** (Application Under Test).
- To start testing, we should have a **requirement, application-ready, necessary resources available**. To maintain accountability, we should assign a respective module to different test engineers.

….different types of Software Testing

- **Acceptance Testing:** Acceptance testing, is the final testing of the product against the specifications of the end user or customer. It is the formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is usually based on "hands-on" use of the product by the end users/customers over some limited period of time. This testing process is usually performed by customer representatives or UAT team.
- **Load Testing:** Load testing puts demand on a system or device and measures its response. It involves testing an application under heavy loads such as testing a website under a range of loads to determine at what point the system's response time degrades or fails. Load testing is frequently performed on an ad hoc basis during the normal developmental process. It is particularly useful because it can quickly and economically identify performance problems without hand checking. It is usually conducted by the performance engineers

….different types of Software Testing

- **Stress Testing:** Stress testing evaluates a system or component at or beyond the limits of its specified requirements. This is a term that is often used interchangeably with "load" and "performance" testing by professionals. It serves the same purpose as load testing in the sense that it is looking to predict failure thresholds. Stress tests normally test system functioning under specific conditions such as unusually heavy loads, heavy repetition of certain actions, input of large numerical values, and large complex queries to a database system.

It differs from load testing in the sense that any potential area of failure under stress is targeted (not just load). This testing process is usually conducted by the performance engineer.

....different types of Software Testing

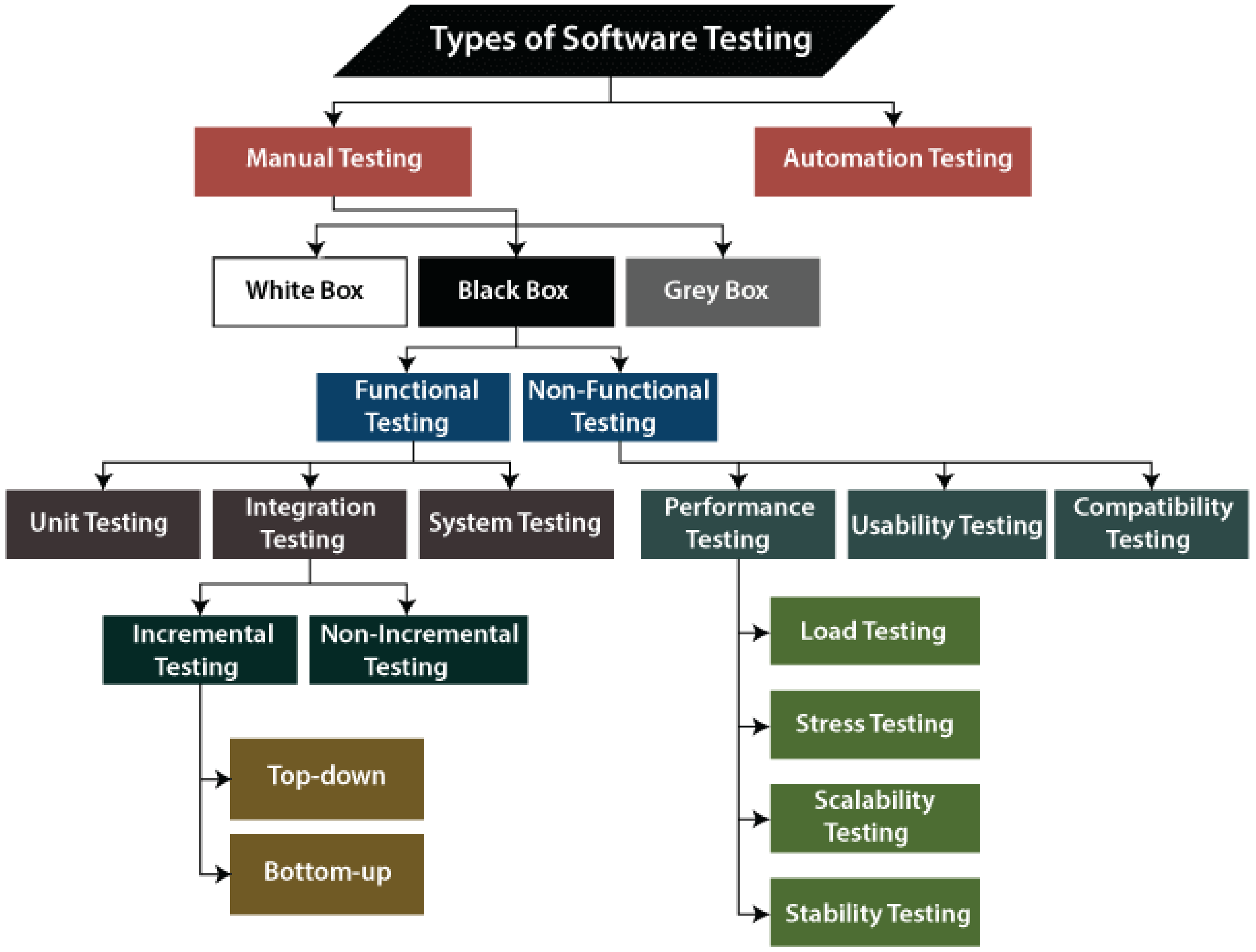
- **Usability Testing:** Usability testing is one of the more common general testing approaches. It verifies the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component. Usability testing could be understood as testing for “user friendliness” or user comfortability. This testing process is usually performed by the end users. User interviews, surveys, video recording of user sessions, and other techniques can be used. Programmers and testers are usually not appropriate as usability testers.
- **Install/Uninstall Testing:** Install/uninstall testing is a type of quality assurance work that focuses on what customers will need to do to install and set up the new software successfully. It may involve full, partial, or upgrades install/uninstall processes. It is typically done by a software testing engineer in conjunction with the configuration manager.

....different types of Software Testing

- **Recovery Testing:** Recovery testing evaluates how well a system recovers from crashes, hardware failures, or other catastrophic problems. This testing process is performed by the testing teams.
- **Security Testing:** Security testing determines that an information system protects data and maintains functionality as intended, and how well the system protects against unauthorized internal or external access. It may require sophisticated testing techniques. Security testing can be performed by testing teams or by specialized security testing companies.
- **Alpha Testing:** Alpha testing is a type of testing of a software product or system conducted at the developer's site. It is typically done by end users or others and definitely not by programmers or testers. It takes place when an application is nearing completion. Some minor design changes can be made as a result of such testing, so it is not as widely circulated.

….different types of Software Testing

- **Beta Testing:** Beta testing is the final testing before releasing the application for commercial purpose. This is by far the most common method of final testing a new product in the real world. Beta testing takes place when development and testing are essentially completed and final bugs and problems need to be found before final release. It is typically done by end users or others and not by programmers or testers.
- **Automated Testing:** Automated testing is the testing technique that uses automation testing tools to control the environmental setup, the execution of tests, and reporting of the results. This testing process is performed by a computer and is used inside the testing teams. This requires a formalized ***"manual testing process"*** that currently exists. The tester must first establish an effective testing process. The real use and purpose of automated test tools is to automate regression testing. This means that a tester must have or must develop a database of detailed test cases that are repeatable and these tests should be run every time there is a change to the application to ensure that the change does not produce unintended consequences.


Types of Software Testing
Manual Testing
Automation Testing
White Box
Black Box
Grey Box
Functional Testing
Non-Functional Testing
Unit Testing
Integration Testing
System Testing
Performance Testing
Usability Testing
Compatibility Testing
Incremental Testing
Non-Incremental Testing
Top-down
Bottom-up
Load Testing
Stress Testing
Scalability Testing
Stability Testing

Thank You!!