JAMAL MOHAMED COLLEGE (AUTONOMOUS)

DEPARTMENT OF COMPUTER APPLICATIONS

TRICHY-20.

SEMESTER - VI: CORE XV: SOFTWARE ENGINEERING Course Code: 20UCA6CC15

UNIT- 4

Software testing fundamentals – White-box testing – Basis-path testing – Control structure testing – Black-box testing – Validation testing – System testing.

TOPIC-1 SOFTWARE TESTING FUNDAMENTALS:-

- ✓ Testing presents an interesting anomaly for the software engineer.
- ✓ During earlier software engineering activities, the engineer attempts to build software from an abstract concept to a tangible product.
- ✓ In fact, testing is the one step in the software process that could be viewed (psychologically, at least) as destructive rather than constructive.

Here, we discuss about the following:-

1. Testing Objectives
2. Testing Principles
3. Testability

Testing Objectives:-

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a high probability of finding an as-yet undiscovered error.
3. A successful test is one that uncovers an as-yet-undiscovered error.

Testing Principles:-

Before applying methods to design effective test cases, a software engineer must understand the basic principles that guide software testing.

1. All tests should be traceable to customer requirements.
2. Tests should be planned long before testing begins.
3. The Pareto principle applies to software testing. Stated simply, the Pareto principle implies that 80 percent of all errors uncovered during testing will likely be traceable to 20 percent of all program components. The problem, of course, is to isolate these suspect components and to thoroughly test them.
4. Testing should begin “in the small” and progress toward testing “in the large.”
5. Exhaustive testing is not possible.
6. To be most effective, testing should be conducted by an independent third party.

Testability:-

- Software testability is the degree to which a software system / software module / software requirement – or design document which supports testing in a given test context.

  The following are the characteristics that lead to testable software. They are:-

  a. Operability means “The better it works, the more efficiently it can be tested."

  b. Observability means "What you see is what you test."

  c. Controllability means "The better we can control the software, the more the testing can be automated and optimized."

  d. Decomposability means "By controlling the scope of testing, we can more quickly isolate problems and perform smarter retesting."

  e. Simplicity means "The less there is to test, the more quickly we can test it."

f. Stability means "The fewer the changes, the fewer the disruptions to testing."

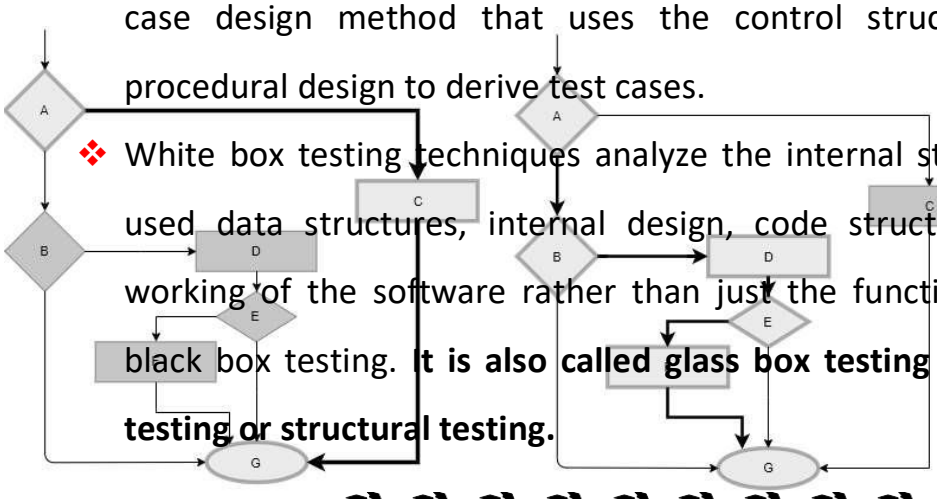
g. Understand ability means "The more information we have, the smarter we will test."

Attributes of a good test:-

1. A good test has a high probability of finding an error. To achieve this goal, the tester must understand the software and attempt to develop a mental picture of how the software might fail.
2. A good test is not redundant. Testing time and resources are limited. There is no point in conducting a test that has the same purpose as another test.
3. A good test should be “best of breed”. In a group of tests that have a similar intent, time and resource limitations may mitigate toward the execution of only a subset of these tests.
4. A good test should be neither too simple nor too complex.

TOPIC-2 WHITE BOX TESTING:-

- White-box testing, sometimes called *glass-box testing* is a test case design method that uses the control structure of the procedural design to derive test cases.
- 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 or clear box testing or structural testing.**

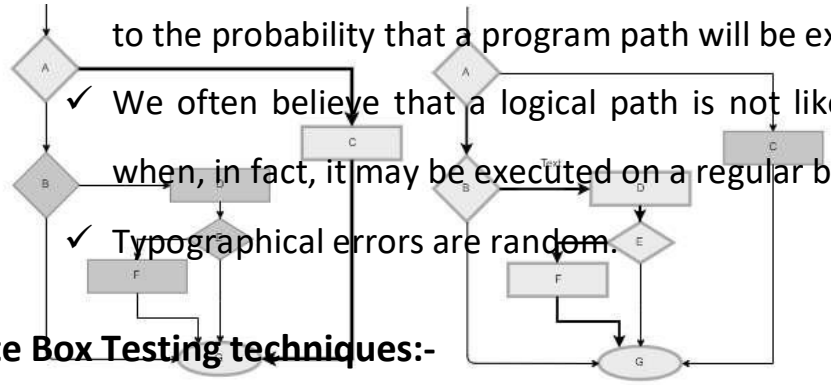


- Using white-box testing methods, the software engineer can derive test cases that (1) Guarantee that all independent paths within a module have been exercised at least once.

(2) Exercise all logical decisions on their true and false sides.

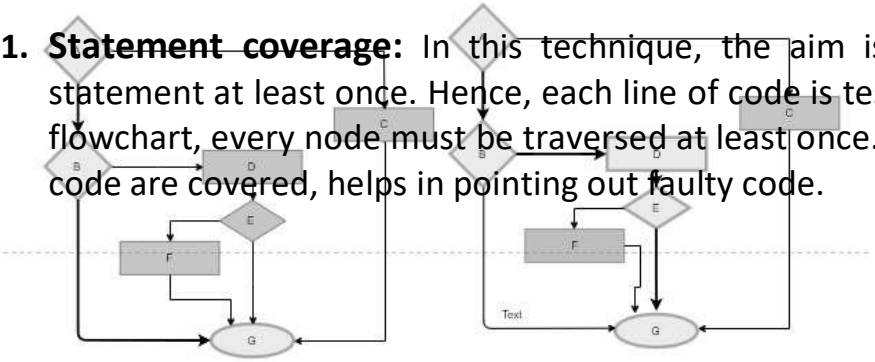
(3) Execute all loops at their boundaries and within their operational bounds, and (4) Exercise internal data structures to ensure their validity.

- ✓ Logic errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed.
- ✓ We often believe that a logical path is not likely to be executed when, in fact, it may be executed on a regular basis.
- ✓ Typographical errors are random.



White Box Testing techniques:-

1. **Statement coverage:** In this technique, the aim is to traverse all statement at least once. Hence, each line of code is tested. In case of a flowchart, every node must be traversed at least once. Since all lines of code are covered, helps in pointing out faulty code.



2. **Branch Coverage:** In this technique, test cases are designed so that each branch from all decision points is traversed at least once. In a flowchart, all edges must be traversed at least once.

3. **Condition Coverage:** In this technique, all individual conditions must be covered as shown in the following example:
   a. READ X, Y
   b. IF(X == 0 || Y == 0)
   c. PRINT '0'

   In this example, there are 2 conditions: X == 0 and Y == 0. Now, test these conditions get TRUE and FALSE as their values. One possible example would be:

   - #TC1 – X = 0, Y = 55
   - #TC2 – X = 5, Y = 0

4. **Multiple Condition Coverage:** In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once. Let's consider the following example:
   a. READ X, Y
   b. IF(X == 0 || Y == 0)
   c. PRINT '0'

   - #TC1: X = 0, Y = 0
   - #TC2: X = 0, Y = 5
   - #TC3: X = 55, Y = 0
   - #TC4: X = 55, Y = 5

Hence, four test cases required for two individual conditions. Similarly, if there are n conditions then 2^n test cases would be required.

Advantages of White box testing:-

- ✓ White box testing is very thorough as the entire code and structures are tested.
- ✓ It results in the optimization of code removing error and helps in removing extra lines of code.
- ✓ It can start at an earlier stage as it doesn't require any interface as in

- ✓ Based of black box testing.
- ✓ Easy to automate.

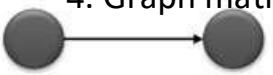
Disadvantages of White box testing:-

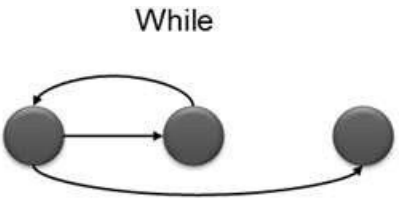
- ✓ Main disadvantage is that it is very expensive.
- ✓ Redesign of code and rewriting code needs test cases to be written again.
- ✓ Testers are required to have in-depth knowledge of the code and programming language as opposed to black box testing.
- ✓ Missing functionalities cannot be detected as the code that exists is tested.
- ✓ Very complex and at times not realistic.

TOPIC-3 BASIS PATH TESTING:-

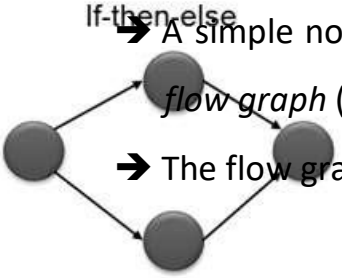
- ✓ *Basis path testing* is a white-box testing technique first proposed by Tom McCabe.
- ✓ The basis path method enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths.
- ✓ Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing.

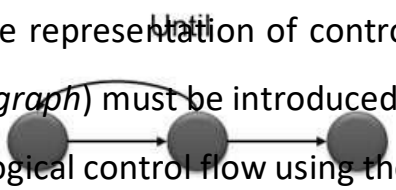
1. Flow graph notation
2. Cyclomatic complexity
3. Deriving test cases
4. Graph matrices





Flow graph notation:-





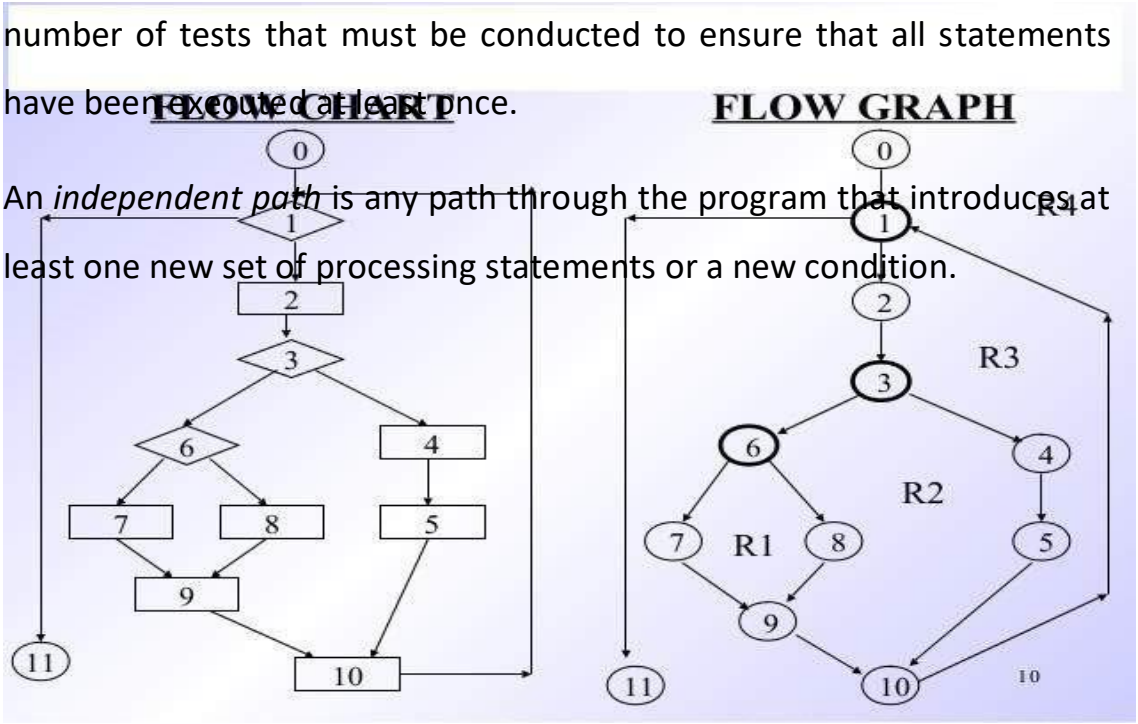
➔ A simple notation for the representation of control flow, called a *flow graph* (or *program graph*) must be introduced.

➔ The flow graph depicts logical control flow using the notation.

Cyclomatic complexity:-

✓ *Cyclomatic complexity* is software metric that provides a quantitative measure of the logical complexity of a program.

- ✓ Cyclomatic complexity defines the number of independent paths in the basis set of a program and provides us with an upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once.



- ✓ An *independent path* is any path through the program that introduces at least one new set of processing statements or a new condition.

Cyclomatic Complexity is computed in one of three ways:

1. The number of regions of the flow graph corresponds to the cyclomatic complexity.

2. Cyclomatic complexity, $V(G)$, for a flow graph, *G,* is defined as

$$V(G) = E - N + 2$$

where *E* is the number of flow graph edges, *N* is the number of flow graph nodes.

3. Cyclomatic complexity, $V(G)$, for a flow graph, *G,* is also defined as

$V(G) = P + 1$ where *P* is the number of predicate nodes contained in the flow graph G.

A predicate node is a node that connects two or more edges.

The cyclomatic complexity can be computed using each of the algorithms just noted:

1. The flow graph has four regions.

2. $V(G)$ = 11 edges _ 9 nodes + 2 = 4.

3. $V(G)$ = 3 predicate nodes + 1 = 4.

Deriving Test cases:-

The basis path testing method can be applied to a procedural design or to source code.

The following steps can be applied to derive the basis set:

1. Using the design or code as a foundation, draw a corresponding flow graph.

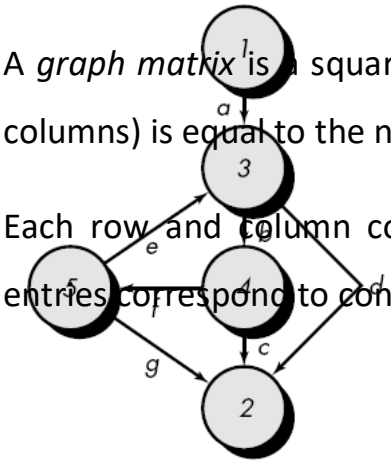
2. Determine the cyclomatic complexity of the resultant flow graph.

3. Determine a basis set of linearly independent paths.

4. Prepare test cases that will force execution of each path in the basis set.

Graph Matrices:-

- To develop a software tool that assists in basis path testing, a data structure, called a *graph matrix,* can be quite useful.
- A *graph matrix* is a square matrix whose size (i.e., number of rows and columns) is equal to the number of nodes on the flow graph.
- Each row and column corresponds to an identified node, and matrix entries correspond to connections (an edge) between nodes.



Flow graph

Node \ Connected to node	1	2	3	4	5
1					
2					
3		d		b	
4		c			f
5		g	e		

Graph matrix

	1	2	3	4	5	6	7	8	Connections
1			1	1					2 – 1 = 1
2									
3					1		1		2 – 1 = 1
4					1				1 – 1 = 0
5						1			1 – 1 = 0
6					1	1	1		3 – 1 = 2
7								1	1 – 1 = 0
8		1			1		1		3 – 1 = 2

- ✓ Each letter has been replaced with a 1, indicating that a connection exists (zeros have been excluded for clarity).
- ✓ Represented in this form, the graph matrix is called a *connection matrix.*

TOPIC-4 CONTROL STRUCTURE TESTING

Control structure testing is used to increase the coverage area by testing various control structures present in the program.

The different types of testing performed under control structure testing are as follows:-

1. Condition Testing

2. Data Flow Testing

3. Loop Testing

Condition testing:-

- ✓ Condition testing is a test cased design method, which ensures that the logical condition and decision statements are free from errors.
- ✓ The errors present in logical conditions can be incorrect Boolean operators, missing parenthesis in a Boolean expression, error in relational operators, arithmetic expressions, and so on.
- ✓ A relational expression takes the form

E1* <relational-operator> *E2

Where *E1* and *E2* are arithmetic expressions and <relational-operator> is one of the following: <, ≤, =, ≠ (non equality), >, or ≥.

A *compound condition* is composed of two or more simple conditions, Boolean operators, and parentheses.

We assume that Boolean operators allowed in a compound condition include OR (|), AND (&) and NOT (¬).

A condition without relational expressions is referred to as a *Boolean expression.*

The types of errors in a condition include the following:-

- Boolean operator error (incorrect/missing/extra Boolean operators).

- Boolean variable error.
- Boolean parenthesis error.
- Relational operator error.
- Arithmetic expression error.

Example-1:-

$C_1 : B_1 \& B_2$

Where B_1 & B_2 are Boolean variables.

Example-2:-

$C_2 : B_1 \& (E_3 = E_4)$

Where B_1 is a Boolean expressions and E_3 and E_4 are Arithmetic expressions.

Example-3:-

$C_3: (E_1 > E_2) \& (E_3 = E_4)$

Where *E1*, E2, *E3,* and *E4* are Arithmetic expressions.

Data Flow Testing:-

The data flow test method chooses the test path of a program based on the locations of the definitions and uses all the variables in the program.

For example, with S as its statement number.

DEF (S) = {X | Statement S has a definition of X}

USE (S) = {X | Statement S has a use of X}

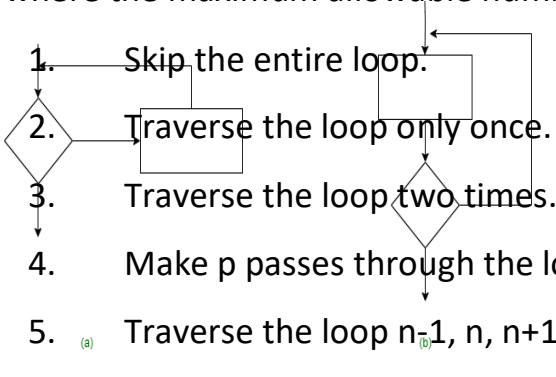
Loop Testing:

Loop testing is actually a white box testing technique. It specifically focuses on the validity of loop construction.

Following are the types of loops:-

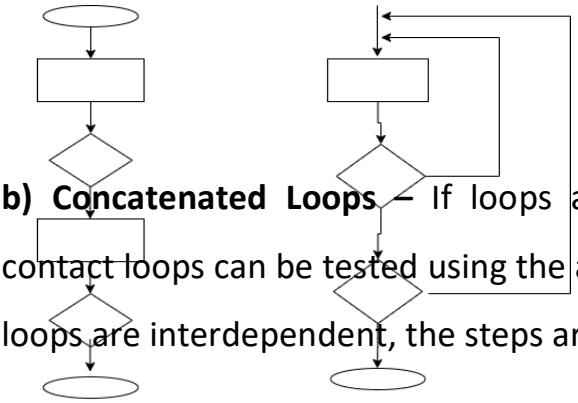
a) Simple Loop – The following set of test can be applied to simple loops, where the maximum allowable number through the loop is n.

1. Skip the entire loop.
2. Traverse the loop only once.
3. Traverse the loop two times.
4. Make p passes through the loop where p<n.
5. Traverse the loop n-1, n, n+1 times.



SIMPLE LOOPS

b) Concatenated Loops – If loops are not dependent on each other, contact loops can be tested using the approach used in simple loops. if the loops are interdependent, the steps are followed in nested loops.



Concatenated Loops

c) Nested loops – Loops within loops are called as nested loops. When testing nested loops, the number of tested increases as level nesting increases.

The following steps for testing nested loops are as follows-

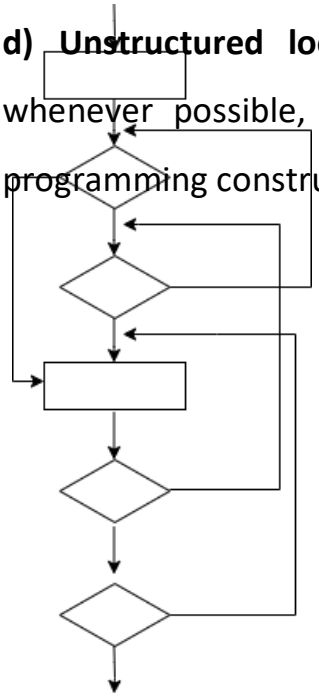
(i) Start with inner loop. Set all other loops to minimum values.

(ii) Conduct simple loop testing on inner loop.

(iii) Work outwards.

(iv) Continue until all loops tested.

d) Unstructured loops – This type of loops should be redesigned, whenever possible, to reflect the use of unstructured the structured programming constructs.



Unstructured Loops

TOPIC-5 BLACK BOX TESTING

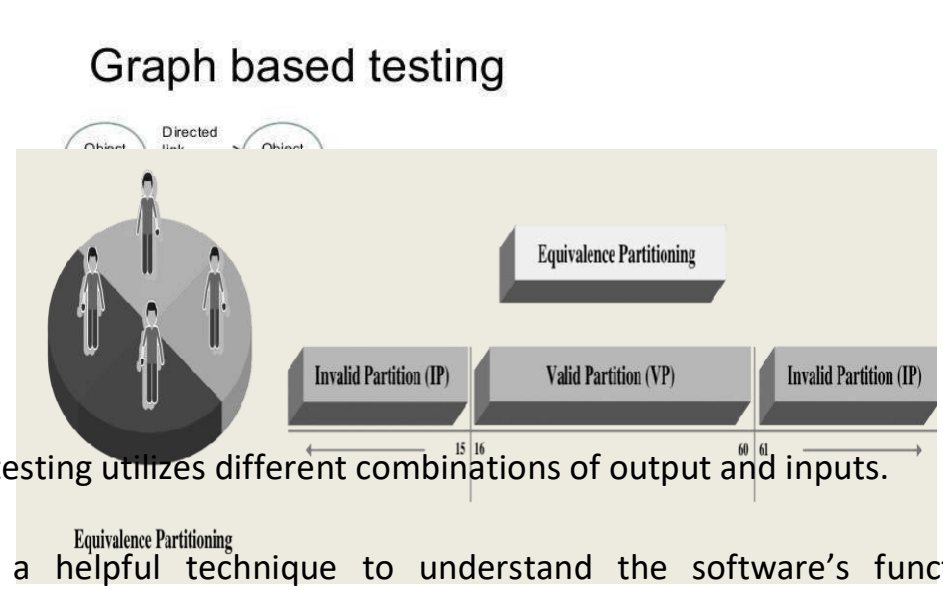
- *Black-box testing,* also called *behavioural testing,* focuses on the functional requirements of the software.
- That is, black-box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.
- Black-box testing is not an alternative to white-box techniques.
- Rather, it is a complementary approach that is likely to uncover a different class of errors than white-box methods.
- Black-box testing attempts to find errors in the following categories: (1) incorrect or missing functions, (2) interface errors, (3) errors in data structures or external data base access, (4) behaviour or performance errors, and (5) initialization and termination errors.

Techniques of Black Box Testing:-

I. Graph based testing method
II. Equivalence Partitioning
III. Boundary Value Analysis
IV. Comparison Testing
V. Orthogonal Array Testing

Graph based testing method:-

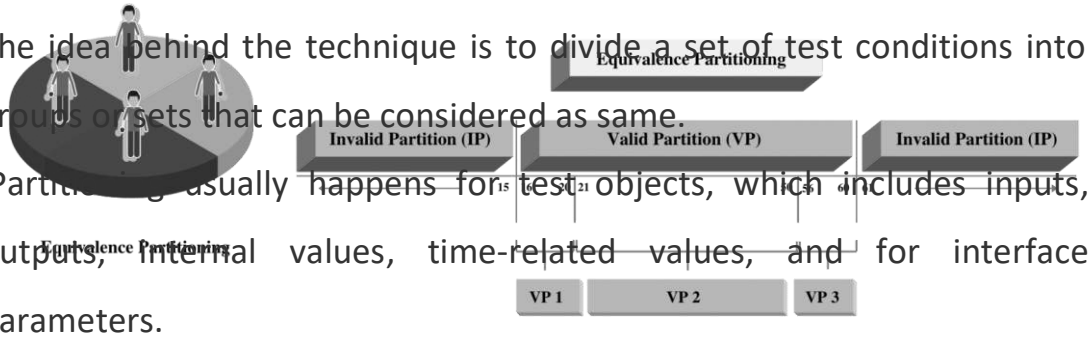
- ✓ The Graph based testing method involves a graph drawing that depicts the link between the causes (inputs) and the effects (output), which trigger the effects.



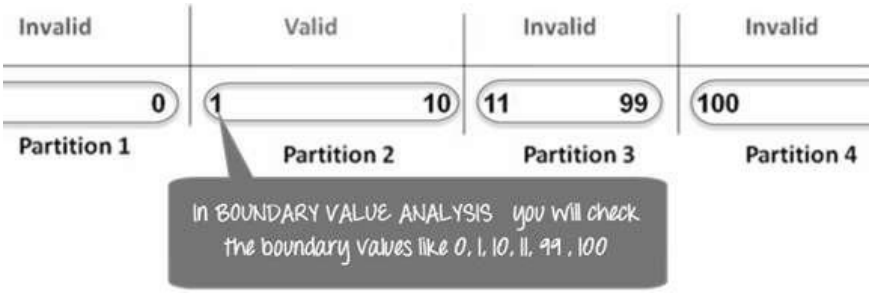
- ✓ This testing utilizes different combinations of output and inputs.
- ✓ It is a helpful technique to understand the software's functional performance, as it visualizes the flow of inputs and outputs in a lively fashion.

Equivalence Partitioning / Equivalence Class Partitioning:-

- ❖ ***Equivalence partitioning is a black-box testing technique*** that applies to all levels of testing.
- ❖ The idea behind the technique is to divide a set of test conditions into groups or sets that can be considered as same.
- ❖ Partitioning usually happens for test objects, which includes inputs, outputs, internal values, time-related values, and for interface parameters.
- ❖ It works on certain assumptions:-
  - The system will handle all the test input variations within a partition in the same way.
  - If one of the input conditions passes, then all other input conditions within the partition will pass as well.
  - If one of the input conditions fails, then all other input conditions within the partition will fail as well.



The First step in Equivalence partitioning is to divide (partition) the input values into sets of valid and invalid partitions.



- ✓ Valid Partitions are values that should be accepted by the component or system under test. This partition is called "**Valid Equivalence Partition.**"
- ✓ Invalid Partitions are values that should be rejected by the component or system under test. This partition is called "**Invalid Equivalence Partition.**"

Boundary Value Analysis:-

- ✓ Boundary value analysis is a test case design technique that complements equivalence partitioning.
- ✓ BVA extends equivalence partitioning by focusing on data at the "edges" of an equivalence class.
- ✓ BVA helps in testing any software having a boundary or extreme values.

Comparison Testing:-

- ✓ Comparison testing comprises of comparing the contents of files, databases, against actual results. They are capable of highlighting the differences between expected and actual results.
- ✓ Usually it is performed by comparing different elements side-by-side. From each of the compared elements, two types of data are collected, namely the performance and preference information. Later, they are compared.
- ✓ Comparison testing is also called as "Back-to-Back testing".

Orthogonal Array Testing:-

Orthogonal array testing is a systematic and statistical way of a black box testing technique used when number of inputs to the application under test is small but too complex for an exhaustive testing.

The following are the characteristics of Orthogonal Array Testing:-

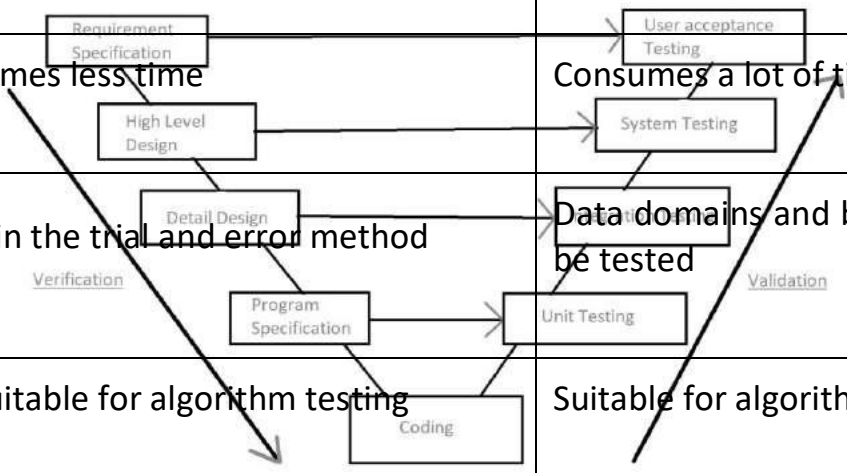
- OAT is a systematic and statistical approach to pair wise interactions.
- Executing a well-defined and a precise test is likely to uncover most of the defects.
- 100% Orthogonal Array Testing implies 100% pair wise testing.

Example:-

If we have 3 parameters, each can have 3 values then the possible Number of tests using conventional method is 3^3 = 27 while the same using OAT, it boils down to 9 test cases.

IMPORTANT DIFFERENCES BETWEEN BLACK BOX TESTING AND WHITE BOX TESTING:-

Black box Testing is used to test software without knowing the internal structure of the software	White Box Testing is performed after knowing the internal structure of the software
Carried out by testers	Performed by developers
Does not require programming knowledge	Requires programming knowledge
Requires implementation knowledge	Does not require implementation knowledge
Higher level testing	Lower level testing
Consumes less time	Consumes a lot of time
Done in the trial and error method	Data domains and boundaries can be tested
Not suitable for algorithm testing	Suitable for algorithm testing



TOPIC-6 VALIDATION TESTING

- **Validation Testing** ensures that the product actually meets the client's needs.
- *Are we building the right product?*

❖ Validation is the Dynamic Testing.

Activities involved in validation:

a. Black box testing

b. White box testing

c. Unit testing

d. Integration testing

Here, we should understand the following three concepts.

1. Validation Test Criteria
2. Configuration Review
3. Alpha and Beta Testing

Validation Test Criteria:-

After each validation test case has been conducted, one of two possible conditions exists:-

❖ The function or performance characteristics conform to specification and are accepted.

❖ A deviation from specification is uncovered and a *deficiency list* is created.

Configuration Review:-

❖ An important element of the validation process is a *configuration*

- ✓ The intent of the review is to ensure that all elements of the software configuration have been properly developed, are catalogued.
- ✓ The configuration review, sometimes called an *audit*

Alpha and Beta Testing:-

Alpha Testing	Beta Testing
1. Alpha testing involves both the white box and black box testing.	1. Beta testing commonly uses black box testing.
2. Alpha testing is performed by testers who are usually internal employees of the organization.	2. Beta testing is performed by clients who are not part of the organization.
3. Alpha testing is performed at developer's site.	3. Beta testing is performed at end-user of the product.
4. Reliability and security testing are not checked in alpha testing.	4. Reliability, security and robustness are checked during beta testing.
5. Alpha testing ensures the quality of the product before forwarding to beta testing.	5. Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users.
6. Alpha testing requires a testing environment or a lab.	6. Beta testing doesn't require a testing environment or lab.
7. Alpha testing may require long execution cycle.	7. Beta testing requires only a few weeks of execution.

8. Developers can immediately address the critical issues or fixes in alpha testing.	8. Most of the issues or feedback collected from beta testing will be implemented in future versions of the product.

TOPIC-7 SYSTEM TESTING:-

- System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system.
- **SYSTEM TESTING** is a level of **testing** that validates the complete and fully integrated software product. The purpose of a **system test** is to evaluate the end-to-end **system** specifications.
- The various types of system testing is as follows:-
    1. Recovery testing
    2. Security testing
    3. Stress testing
    4. Performance testing

Recovery testing:-

- ✓ *Recovery testing* is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed.
- ✓ If recovery is automatic (performed by the system itself), reinitialization, check pointing mechanisms, data recovery, and restart are evaluated for correctness.

- If recovery requires human intervention, the mean-time-to-repair (MTTR) is evaluated to determine whether it is within acceptable limits.

Security testing:-

- ✓ *Security testing* attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.
- ✓ **Security Testing** is a type of Software **Testing** that uncovers vulnerabilities of the system and determines that the data and resources of the system are protected from possible intruders.
- ✓ It ensures that the software system and application are free from any threats or risks that can cause a loss.

Stress testing:-

- ✓ **Stress Testing** is a type of Software **Testing** that verifies the stability & reliability of the system.
- ✓ This **test** mainly measures the system on its robustness and error handling capabilities under extremely heavy load conditions.
- ✓ **Stress Testing** is done to make sure that the system would not crash under crunch situations.

Performance testing:-

- ✓ **Performance testing** is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload.
- ✓ **Performance testing** can involve quantitative **tests** done in a lab, or occur in the production environment in limited scenarios.

1. Explain the various software testing principles.
2. Write short notes on White box testing (or) Explain the different techniques of White box testing.
3. Compare and Contrast Verification and Validation.
4. How integration testing could be done? Explain.
5. Write a note on System testing (or) How System testing could be done?

PART-C

1. Discuss the various basis path testing in detail.
2. Elucidate the concept of black box testing in detail.
3. Explain about the control structure testing in detail.