Software Testing

SE-Module 6

By: Mrs. Purvi D. Sankhe

What is Software Testing ?

- **Identifying the correctness of software by considering its all attributes.**
- **Evaluating the execution of software components to find the software bugs or errors or defects.**
- **includes an examination of code and also the execution of code**

Two Ways of Testing

Manual Testing

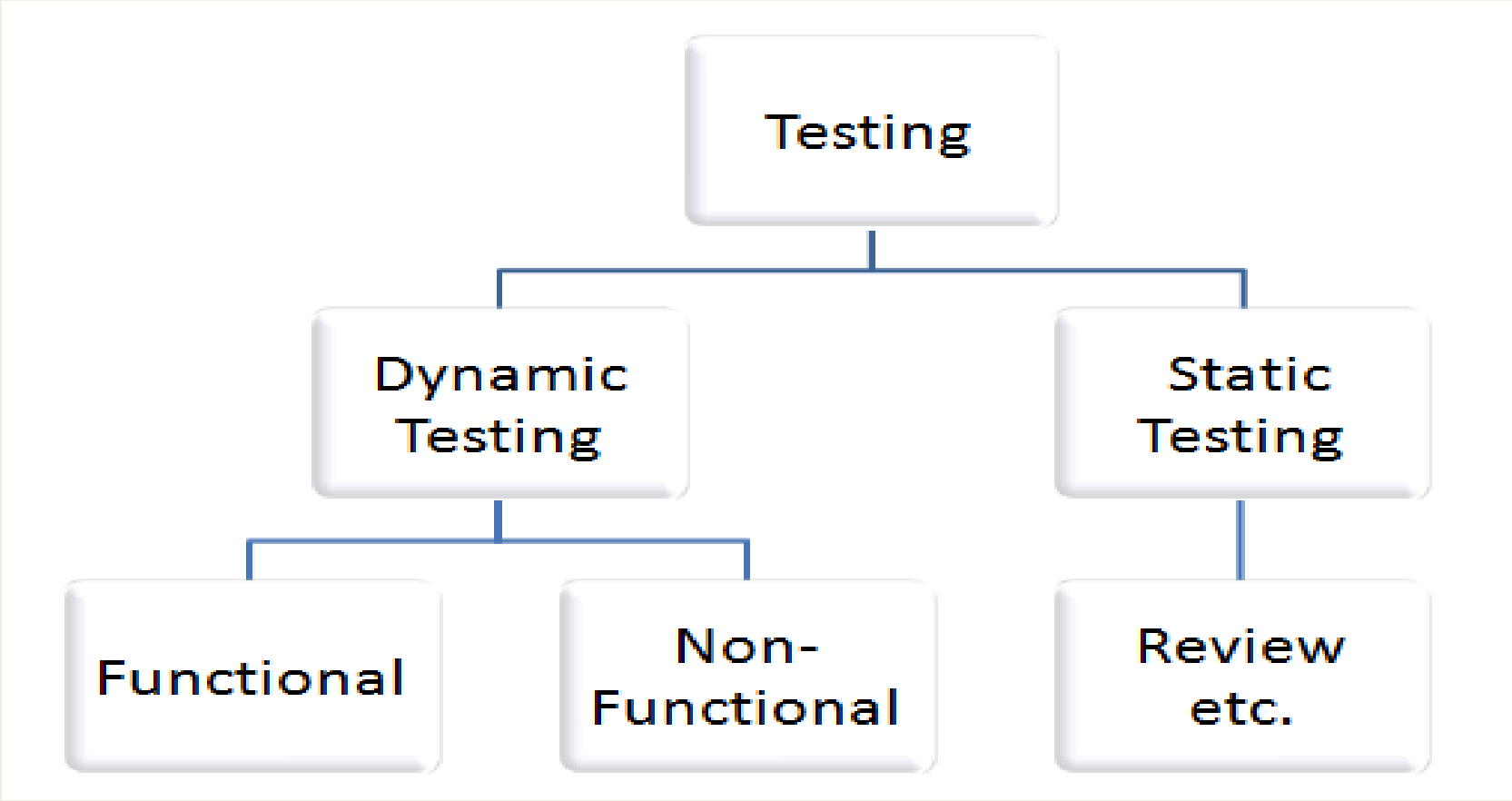
- **Includes testing a software manually,**
- **The tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug.**
- **Different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.**
- **Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing.**

Two Ways of Testing contd..

Automation testing,

- **Also known as Test Automation, is when the tester writes scripts and uses another software to test the product.**
- **This process involves automation of a manual process.**
- **Used to re-run the test scenarios that were performed manually, quickly, and repeatedly.**
- **Also used to test the application from load, performance, and stress point of view.**
- **It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.**

Testing Methods



Testing Methods

1. Static Testing

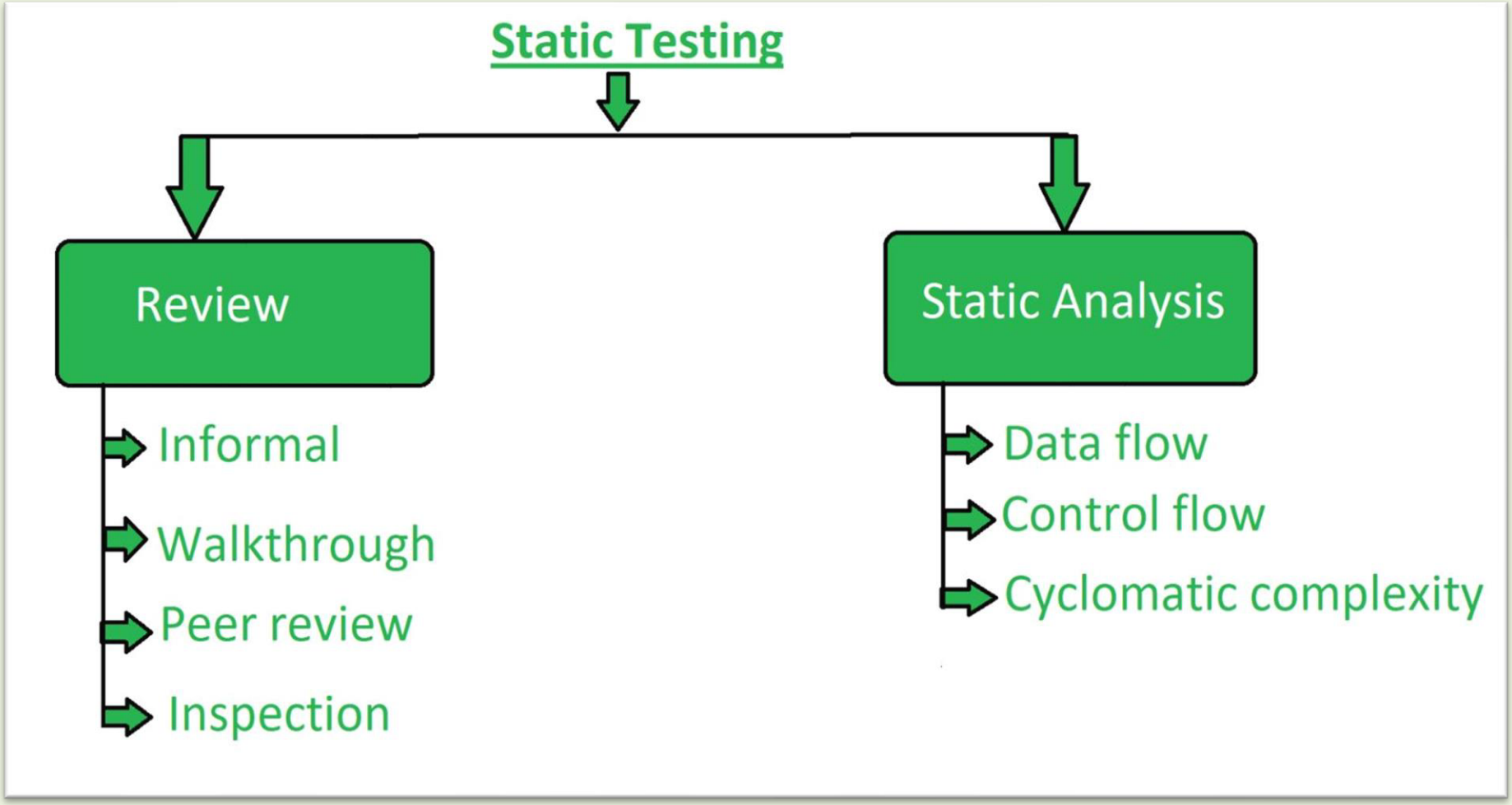
- **software application is tested without code execution.**
- **Manual or automated reviews of code,**
- **requirement documents and**
- **document design are done.**
- **objective of static testing is to improve the quality of software applications by finding errors in early stages of software development process.**
- **Also known as Non-execution testing & Verification in Software Testing.**

Testing Methods

Static Testing contd..

- **Verification is a static method of checking documents and files.**
- **Verification is the process, to ensure that whether we are building the product right i.e., to verify the requirements which we have and to verify whether we are developing the product accordingly or not.**
  - **Examples of Work documents-**
  - **Requirement specifications**
  - **Design document**
  - **Source Code**
  - **Test Plans**
  - **Test Cases**
  - **Test Scripts**
  - **Help or User document**
  - **Web Page content**

Static Testing contd..



Static Testing contd..

Review:

- Process or technique that is performed to find the potential defects in the design of the software.
- Process to detect and remove errors and defects in the different supporting documents like software requirements specifications.
- People examine the documents and sorted out errors, redundancies and ambiguities.
- Review is of four types:

Static Testing contd..

- **Informal:**
- **The creator of the documents put the contents in front of audience and everyone gives their opinion and thus defects are identified in the early stage.**
- **Walkthrough:**
- **Basically performed by experienced person or expert to check the defects**
- **Peer review:**
- **Checking documents of one-another to detect and fix the defects. It is basically done in a team of colleagues.**
- **Inspection:**
- **Verification of document the higher authority like the verification of software requirement specifications (SRS).**

Static Testing contd..

Static Analysis

- Includes the evaluation of the code quality that is written by developers.
- Different tools are used to do the analysis of the code and comparison of the same with the standard.
- It also helps in following identification of following defects:
- Unused variables
- Dead code
- Infinite loops
- Variable with undefined value
- Wrong syntax

Static Testing contd..

- **Static Analysis is of three types:**
- **Data Flow: How Data flows.**
- **Control Flow: Control flow is basically how the statements or instructions are executed.**
- **Cyclomatic Complexity: Cyclomatic complexity defines the number of independent paths in the control flow graph made from the code or flowchart so that minimum number of test cases can be designed for each independent path.**

Testing Methods

2. Dynamic Testing

- **Also known as Validation in Software Testing.**
- **Code is executed.**
- **It checks for functional behavior of software system, memory/CPU usage and overall performance of the system. Hence the name "Dynamic"**
- **The main objective of this testing is to confirm that the software product works in conformance with the business requirements.**
- **This testing is also called an Execution technique or validation testing.**
- **Dynamic testing executes the software and validates the output with the expected outcome.**
- **Dynamic testing is performed at all levels of testing and it can be either black or white box testing.**

KEY DIFFERENCE
Static & Dynamic Testing

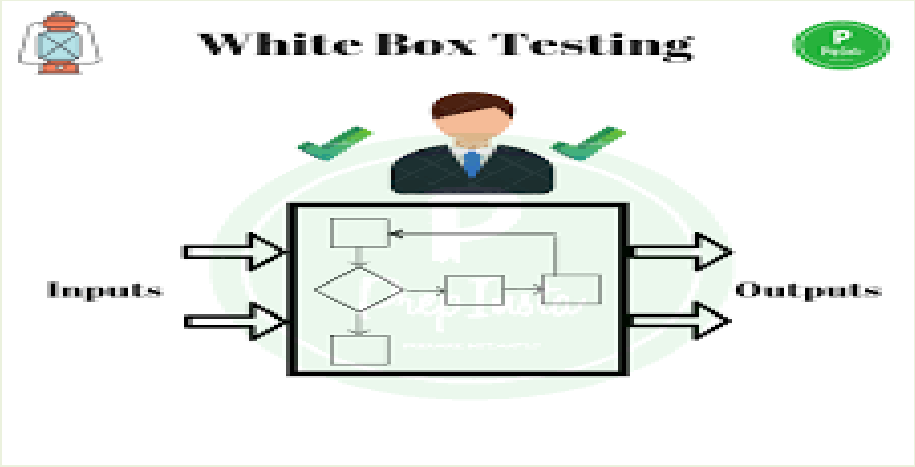
- **Static testing was done without executing the program whereas Dynamic testing is done by executing the program.**
- **Static testing checks the code, requirement documents, and design documents to find errors whereas Dynamic testing checks the functional behavior of software system, memory/CPU usage and overall performance of the system.**
- **Static testing is about the prevention of defects whereas Dynamic testing is about finding and fixing the defects.**

Testing Approaches

- **White Box Testing/ Structural / glass box / clear box testing**
- **Black Box Testing**
- **Grey Box Testing**

White Box Testing

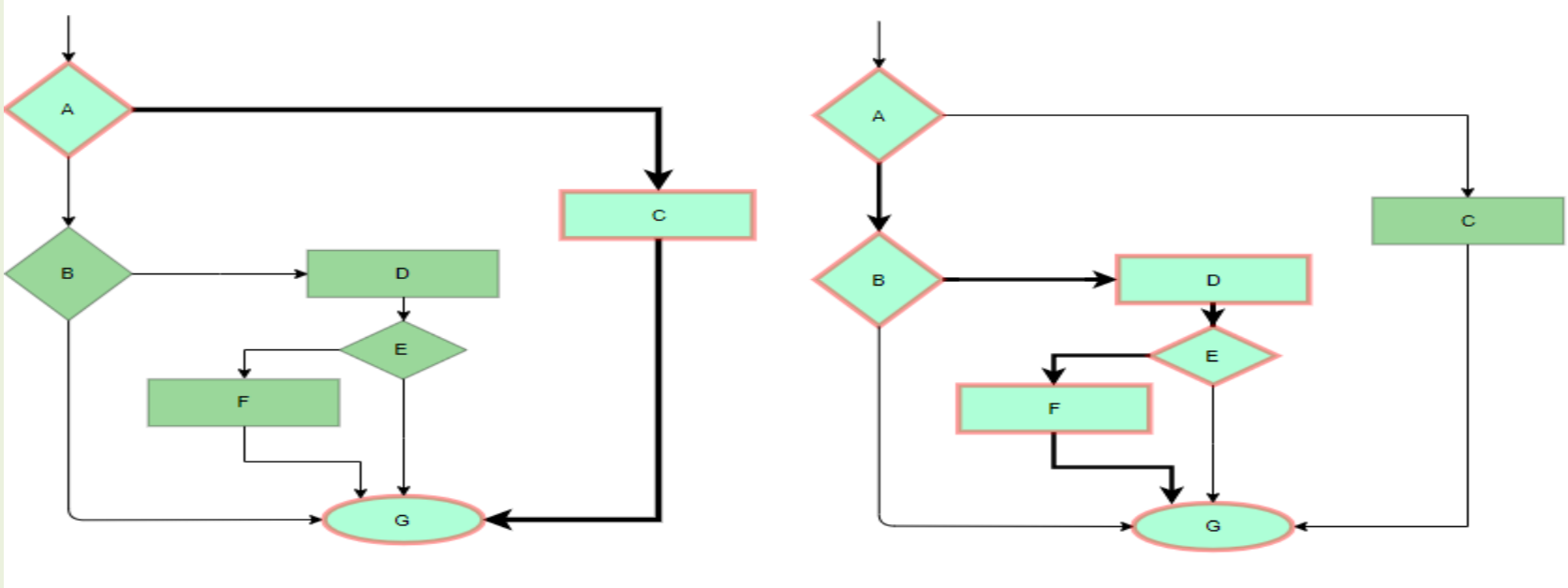
- White box testing is **a software evaluating method used to examine the internal structure, design, coding and inner-working of software**.
- Developers use this testing method to verify the flow of inputs and outputs through the application, improving usability and design and strengthening security.



White Box Testing techniques

Statement coverage

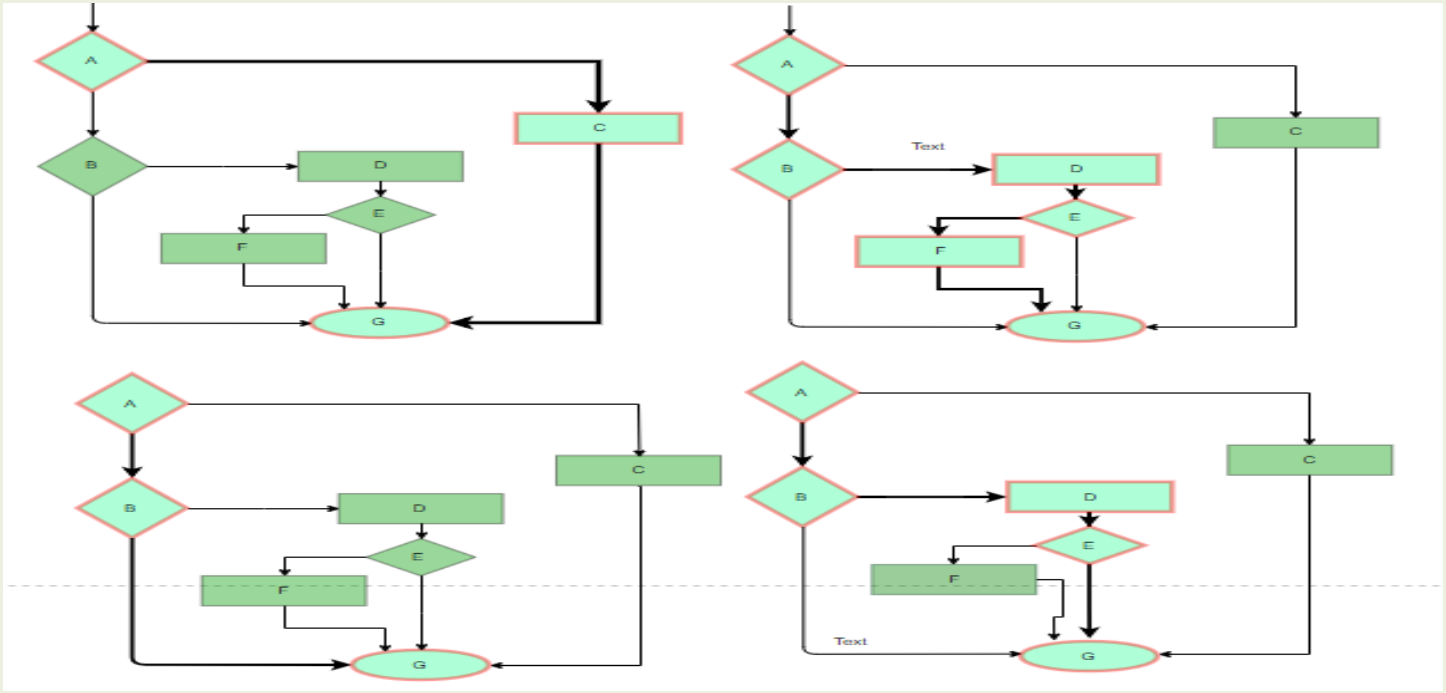
The aim is to traverse all statement at least once. Hence, each line of code is tested. In case of a flowchart, every node must be traversed at least once. Since all lines of code are covered, helps in pointing out faulty code.



White Box Testing techniques contd..

Branch Coverage

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



White Box Testing techniques contd..

Condition Coverage

- **In this technique, all individual conditions must be covered as shown in the following example:**
  - **READ X, Y**
  - **IF(X == 0 || Y == 0)**
  - **PRINT '0'**

White Box Testing techniques contd..

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:

- **READ X, Y**
- **IF(X == 0 || Y == 0)**
- **PRINT '0'**
- **#TC1: X = 0, Y = 0**
- **#TC2: X = 0, Y = 5**
- **#TC3: X = 55, Y = 0**
- **#TC4: X = 55, Y = 5**

Basis Path Testing

- White-box testing technique proposed by Tom McCabe
- Enables the test case designer to derive a logical complexity measure of a procedural design
- Uses 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

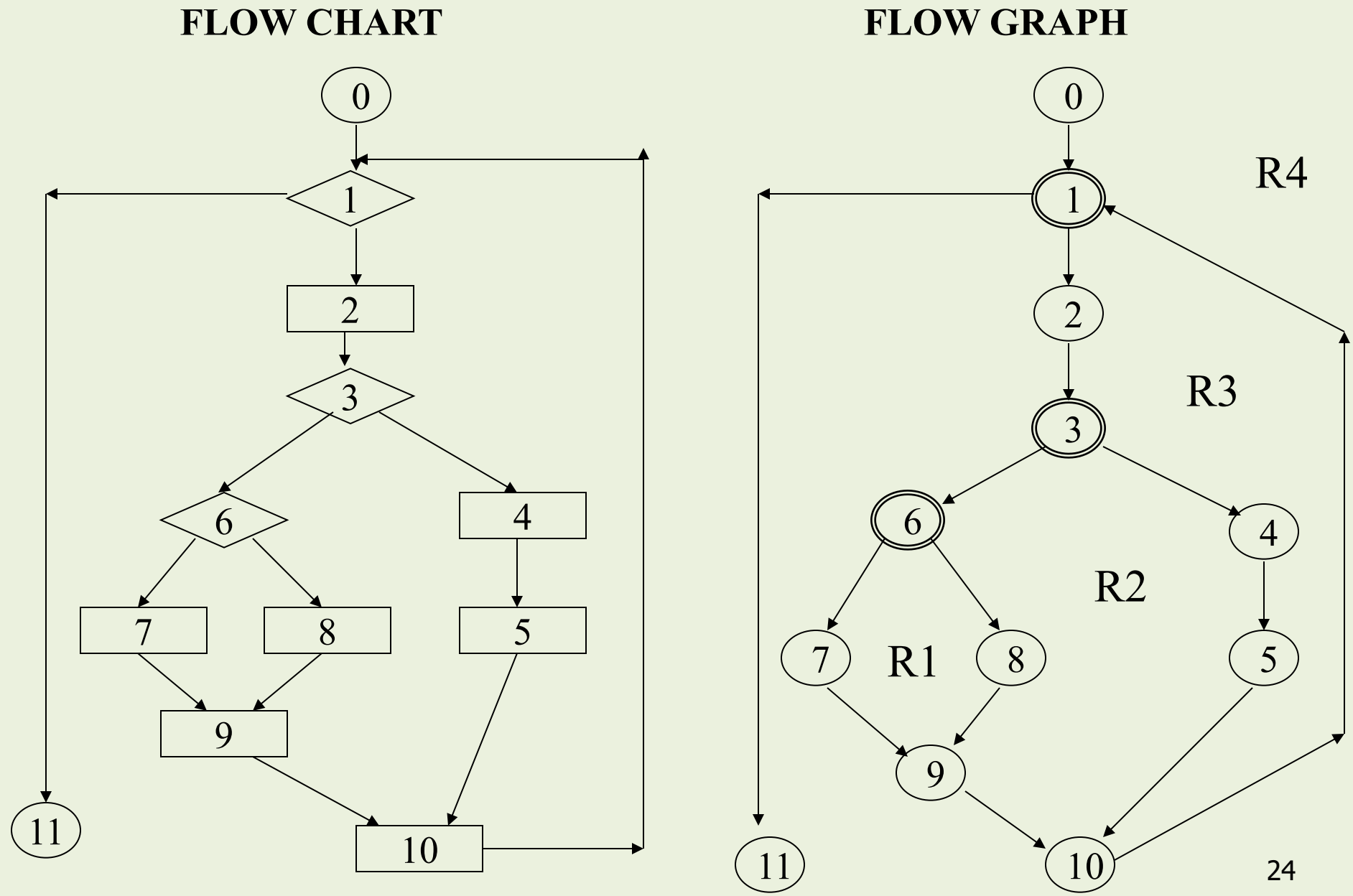
Flow Graph Notation

- A circle in a graph represents a node, which stands for a sequence of one or more procedural statements
- A node containing a simple conditional expression is referred to as a predicate node
  - Each compound condition in a conditional expression containing one or more Boolean operators (e.g., and, or) is represented by a separate predicate node
  - A predicate node has two edges leading out from it (True and False)

Flow Graph Notation

- **An edge, or a link, is a an arrow representing flow of control in a specific direction**
  - **An edge must start and terminate at a node**
  - **An edge does not intersect or cross over another edge**
- **Areas bounded by a set of edges and nodes are called regions**
- **When counting regions, include the area outside the graph as a region, too**

Flow Graph Example



Independent Program Paths

- **Defined as a path through the program from the start node until the end node that introduces at least one new set of processing statements or a new condition (i.e., new nodes)**
- **Must move along at least one edge that has not been traversed before by a previous path**
- **Basis set for flow graph on previous slide**
  - **Path 1: 0-1-11**
  - **Path 2: 0-1-2-3-4-5-10-1-11**
  - **Path 3: 0-1-2-3-6-8-9-10-1-11**
  - **Path 4: 0-1-2-3-6-7-9-10-1-11**
- **The number of paths in the basis set is determined by the cyclomatic complexity**

Cyclomatic Complexity

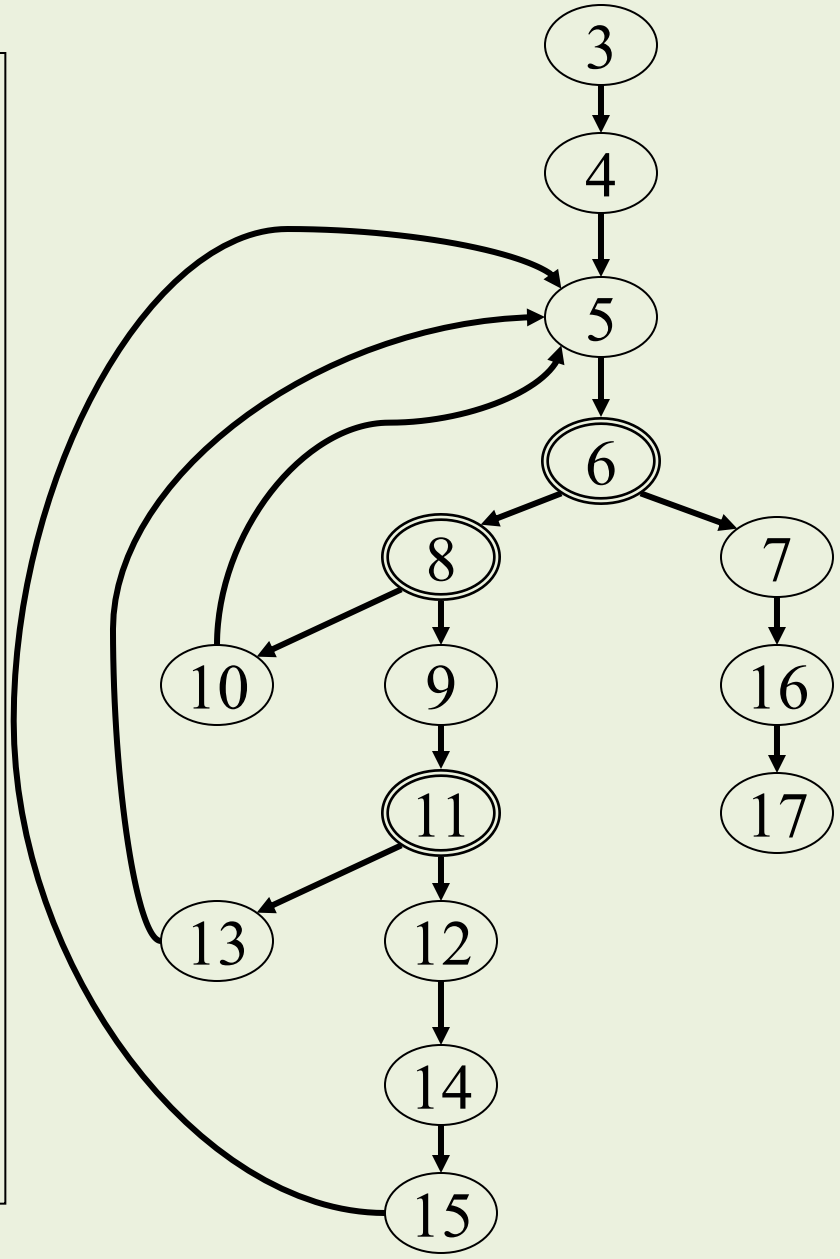
- **Provides a quantitative measure of the logical complexity of a program**
- **Defines the number of independent paths in the basis set**
- **Provides an upper bound for the number of tests that must be conducted to ensure all statements have been executed at least once**
- **Can be computed three ways**
  - **The number of regions**
  - **V(G) = E – N + 2, where E is the number of edges and N is the number of nodes in graph G**
  - **V(G) = P + 1, where P is the number of predicate nodes in the flow graph G**
- **Results in the following equations for the example flow graph**
  - **Number of regions = 4**
  - **V(G) = 14 edges – 12 nodes + 2 = 4**
  - **V(G) = 3 predicate nodes + 1 = 4**

Deriving the Basis Set and Test Cases

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

A Second Flow Graph Example

```
 1  int functionY(void)
 2  {
 3     int x = 0;
 4     int y = 19;

 5  A: x++;
 6     if (x > 999)
 7        goto D;
 8     if (x % 11 == 0)
 9        goto B;
10     else goto A;

11  B: if (x % y == 0)
12        goto C;
13     else goto A;

14  C: printf("%d\n", x);
15     goto A;

16  D: printf("End of list\n");
17     return 0;
18  }
```

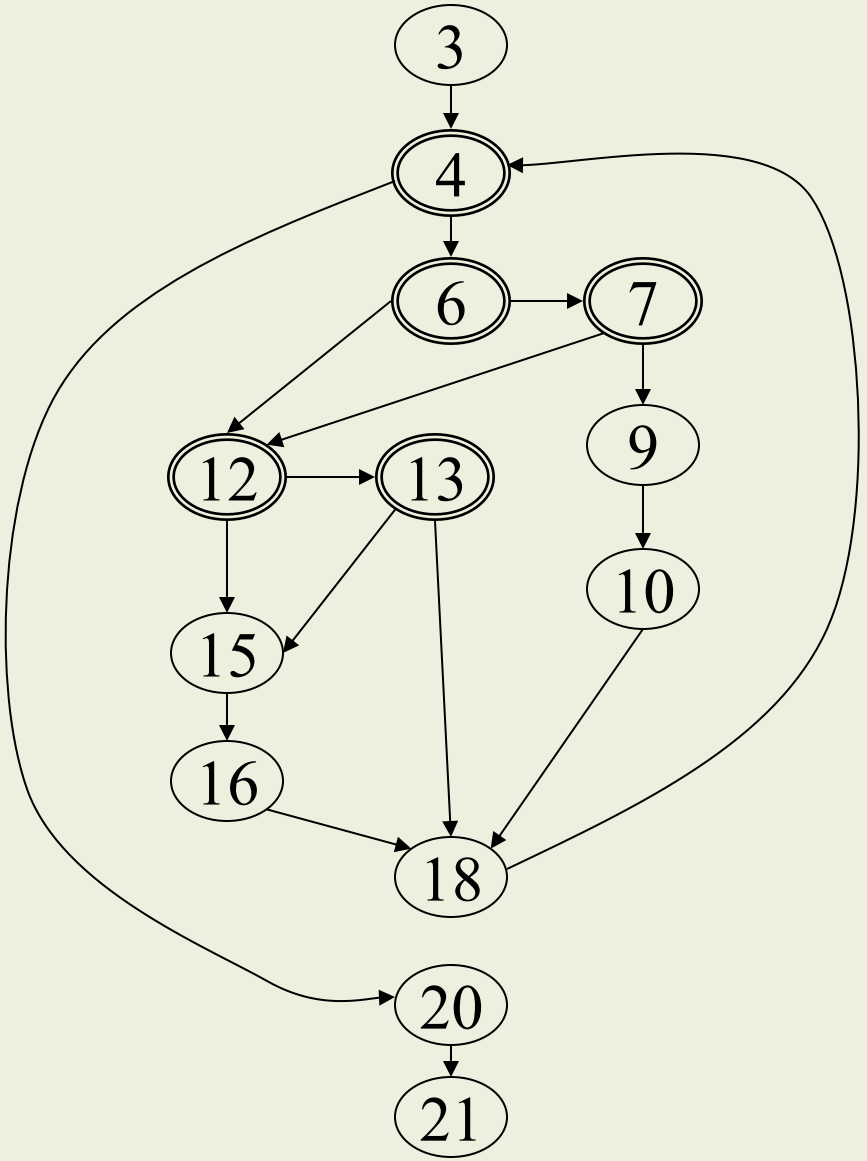


A Sample Function to Diagram and Analyze

```
 1  int functionZ(int y)
 2  {
 3  int x = 0;

 4  while (x <= (y * y))
 5     {
 6     if ((x % 11 == 0) &&
 7         (x % y == 0))
 8        {
 9        printf("%d", x);
10        x++;
11        } // End if
12     else if ((x % 7 == 0) ||
13              (x % y == 1))
14        {
15        printf("%d", y);
16        x = x + 2;
17        } // End else
18     printf("\n");
19     } // End while

20  printf("End of list\n");
21  return 0;
22  } // End functionZ
```

A Sample Function to Diagram and Analyze

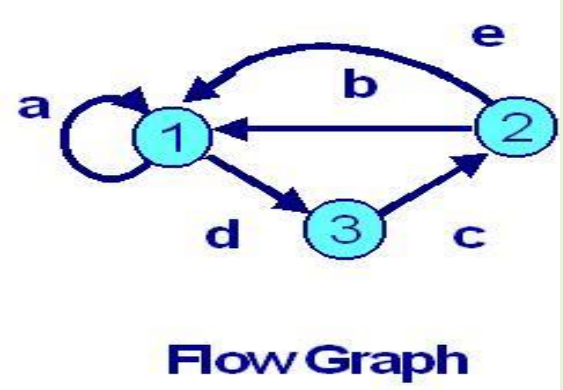
```
 1  int functionZ(int y)
 2  {
 3  int x = 0;

 4  while (x <= (y * y))
 5     {
 6     if ((x % 11 == 0) &&
 7         (x % y == 0))
 8        {
 9        printf("%d", x);
10        x++;
11        } // End if
12     else if ((x % 7 == 0) ||
13              (x % y == 1))
14        {
15        printf("%d", y);
16        x = x + 2;
17        } // End else
18     printf("\n");
19     } // End while

20  printf("End of list\n");
21  return 0;
22  } // End functionZ
```



Graph Matrices

- To develop s/w tool that assist in basis path testing, a DS called graph matrix is quite useful.
- Is a square matrix whose size is equal to no of nodes on the flow graph
- Each node is identified by nos. & each edge is identified by letters
- Letter entry is made to correspond to a connection between two nodes.
- Link weights is a 1 or 0. (connection matrix)
- Provides other way to determine cyclomatic complexity

Graph Matrices contd..



Flow Graph

	1	2	3
1	a		d
2	b+e		
3		c	

Graph Matrix

	1	2	3	4
1			1	1
2				
3		1		
4		1		

Connection Matrix

1			1	1
2				
3		1		
4		1		

2 - 1 = 1

1 - 1 = 0

1 - 1 = 0

1 + 1 = 2 = V(G)

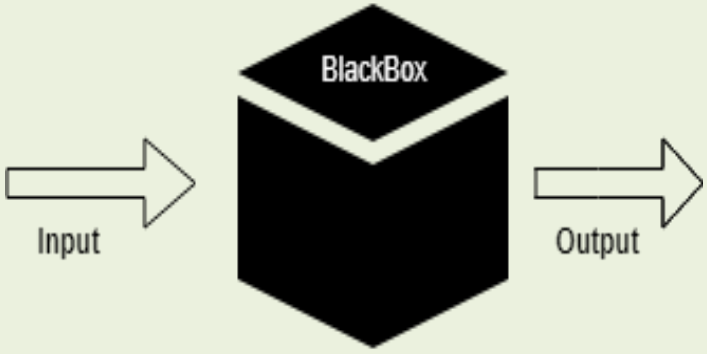
Calculation of V(G)

White Box Testing techniques:

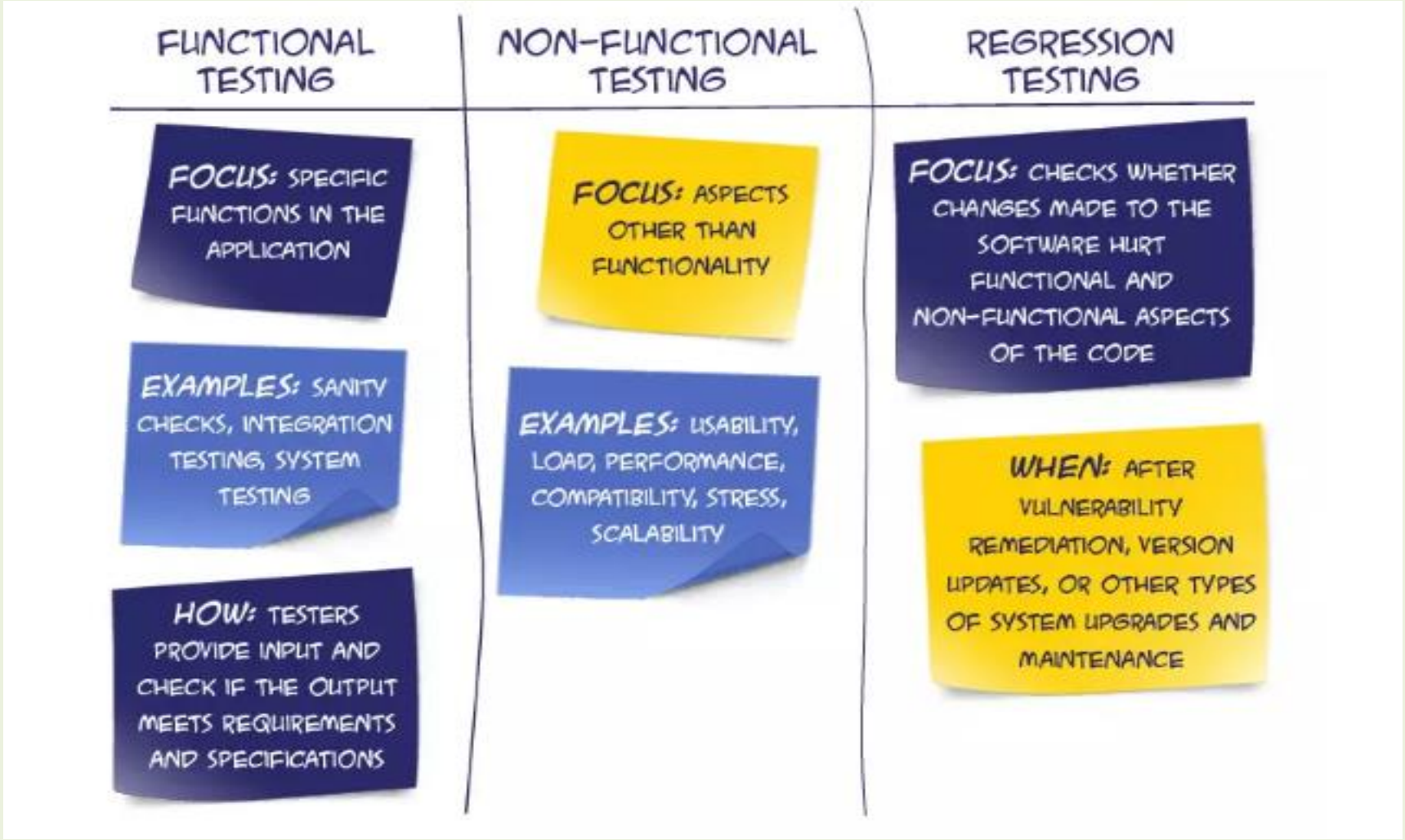
- **Loop Testing:** **Loops are widely used and these are fundamental to many algorithms hence, their testing is very important. Errors often occur at the beginnings and ends of loops.**
  - **Simple loops:** **For simple loops of size n, test cases are designed that:**
    - **Skip the loop entirely**
    - **Only one pass through the loop / 2 passes**
  - **Nested loops:** **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 loops:** **Independent loops, one after another. Simple loop tests are applied for each. If they're not independent, treat them like nesting.**

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.**



Types of Black Box Testing



Functional Testing

- **A type of black box testing that focuses on specific functions in the application.**
- **This includes sanity checks, integration testing, or system testing.**
- **Performed by providing a certain input and checking if the output meets the software requirements and specifications.**

Non-Functional Testing

- This includes a number of black box testing types that don't examine functionality.
- Non functional testing focuses on other aspects and requirements, like usability, load, performance, compatibility, stress, or scalability, etc…

Regression Testing

- **Performed after vulnerability remediation, version updates, or other types of system upgrades and maintenance.**
- **Regression testing checks whether changes made to the software hurt the existing functional or non-functional aspects of the code.**

Black Box Testing Techniques

Equivalence Partitioning

- **Testers can divide possible inputs into groups or “partitions”, and test only one example input from each group.**
- **For example, if a system requires a user’s birth date and provides the same response for all users under the age of 18, and a different response for users over 18, it is sufficient for testers to check one birth date in the “under 18” group and one date in the “over 18” group.**

Example 1: Equivalence partitioning

An ATM allows users to withdraw money with the following conditions: The minimum withdrawal amount is ₹100.The maximum withdrawal amount per transaction is ₹10,000.The withdrawal amount must be in multiples of ₹100.

Equivalence Partitions:

1.Valid Partition (Expected to Pass)

1. **₹100, ₹500, ₹2000, ₹10,000 (within the valid range and multiple of ₹100)**

2.Invalid Partition (Expected to Fail)

1. **Below Minimum: ₹50, ₹99 (less than ₹100)**
2. **Above Maximum: ₹10,500, ₹15,000 (more than ₹10,000)**
3. **Not a Multiple of ₹100: ₹150, ₹1250, ₹3700 (not a valid amount)**

Instead of testing every possible amount, we select representative values from each partition, making testing more efficient while ensuring good coverage.

Black Box Testing Techniques contd…

Boundary Value Analysis

Testers can identify that a system has a special response around a specific boundary value.

For example, a specific field may accept only values between 0 and 99. Testers can focus on the boundary values (-1, 0, 99 and 100), to see if the system is accepting and rejecting inputs correctly.

Example 1: Boundary Value Analysis

Online Shopping - Free Shipping Eligibility An e-commerce website offers free shipping for orders between ₹500 and ₹5000.If the order amount is less than ₹500, a delivery charge applies. If the order amount is more than ₹5000, the order is not accepted.

Boundary Value Test Cases:

Partition	Test Input	Expected Result
Below Minimum Boundary	₹499	Delivery charge applies (Fail)
Minimum Boundary	₹500	Free shipping (Pass)
Just Above Minimum	₹501	Free shipping (Pass)
Valid Range (Middle Value)	₹2750	Free shipping (Pass)
Just Below Maximum	₹4999	Free shipping (Pass)
Maximum Boundary	₹5000	Free shipping (Pass)
Just Above Maximum	₹5001	Order not accepted (Fail)

Instead of testing random values, BVA focuses on edge cases where errors are most likely to occur. It helps detect boundary-related defects effectively.

Black Box Testing Techniques contd..

Decision Table Testing

- **Many systems provide outputs based on a set of conditions. Testers can then identify "rules" which are a combination of conditions, identify the outcome of each rule, and design a test case for each rule.**
- **For example, a health insurance company may provide different premium based on the age of the insured person (under 40 or over 40) and whether they are a smoker or not. This generates a decision table with four rules and up to four outcomes—below is an example with three possible outcomes.**

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Under 40	False	False	True	True
Smoker	False	True	False	True
OUTCOMES				
1: High premium				✔
2: Medium premium	✔	✔		
3: Low premium			✔	

Black Box Testing Techniques contd..

State Transition Testing

- **Needed when the system transits from one state to another.**
- **A common example is a login mechanism which allows users to authenticate, but after a specific number of login attempts, transition to a different state, locking the account.**
- **If testers identify a state transition mechanism, they can design test cases that probe the system when it transitions states.**
- **For example, for a system that locks the account after five failed login attempts, a test case can check what happens at the sixth login attempt.**

Black Box Testing Techniques

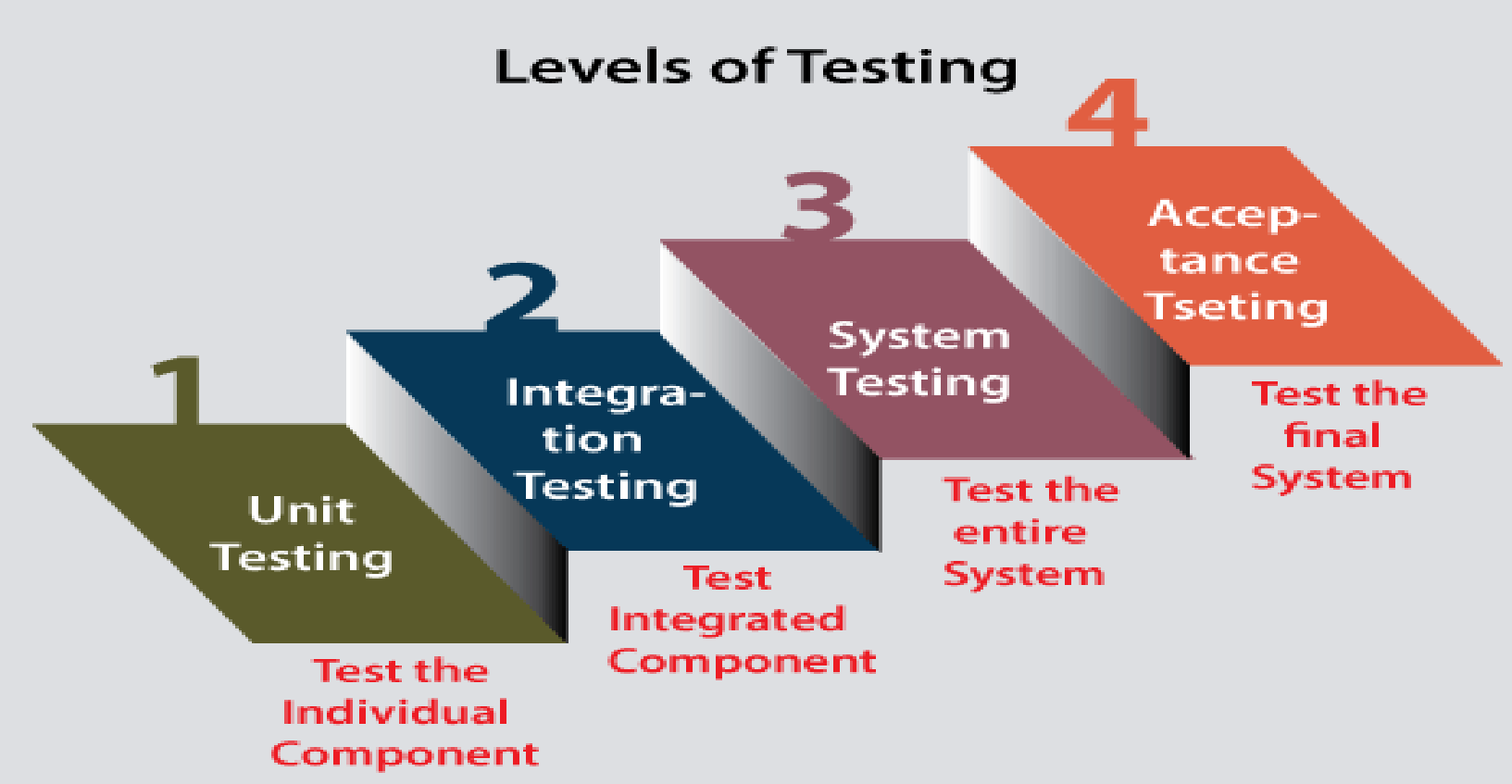
Error Guessing

- **This technique involves testing for common mistakes developers make when building similar systems.**
- **For example, testers can check if the developer handled null values in a field, text in a numeric field or numbers in a text-only field, and sanitization of inputs—whether it is possible to submit user inputs that contain executable code, which has security significance.**

Grey Box Testing

- **Gray Box Testing is a software testing method, which is a combination of both White Box Testing and Black Box Testing method.**
- **In White Box testing internal structure (code) is known**
- **In Black Box testing internal structure (code) is unknown**
- **In Grey Box Testing internal structure (code) is partially known**

Testing Levels



Unit Testing

- **First level of software testing, which is used to test if software modules are satisfying the given requirement or not.**
- **Involves analyzing each unit or an individual component of the software application.**
- **Will help the test engineer and developers in order to understand the base of code that makes them able to change defect causing code quickly. The developers implement the unit.**

Integration Testing

- **Defined as a systematic technique for constructing the software architecture**
  - **At the same time integration is occurring, conduct tests to uncover errors associated with interfaces**
- **Objective is to take unit tested modules and build a program structure based on the prescribed design**
- **Two Approaches**
  - **Non-incremental Integration Testing**
  - **Incremental Integration Testing**

Sample Integration Test Cases for the following scenario:

Application has 3 modules say 'Login Page', 'Mail box' and 'Delete mails' and each of them are integrated logically.

Here do not concentrate much on the Login Page testing as it's already been done in Unit Testing. But check how it's linked to the Mail Box Page.

Similarly Mail Box: Check its integration to the Delete Mails Module.

Test Case ID	Test Case Objective	Test Case Description	Expected Result
1	Check the interface link between the Login and Mailbox module	Enter login credentials and click on the Login button	To be directed to the Mail Box
2	Check the interface link between the Mailbox and Delete Mails Module	From Mail box select the an email and click delete button	Selected email should appear in the Deleted/Trash folder

Non-incremental Integration Testing

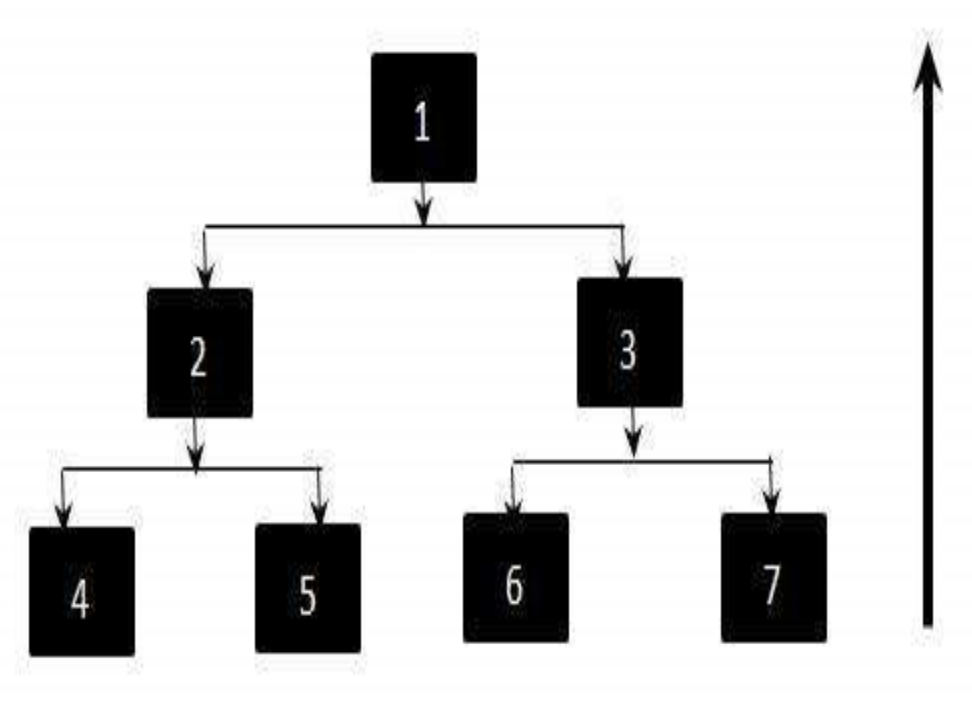
- **Commonly called the “Big Bang” approach**
- **All components are combined in advance**
- **The entire program is tested as a whole**
- **Chaos results**
- **Many seemingly-unrelated errors are encountered**
- **Correction is difficult because isolation of causes is complicated**
- **Once a set of errors are corrected, more errors occur, and testing appears to enter an endless loop**

Incremental Integration Testing

- **Three kinds**
  - **Top-down integration**
  - **Bottom-up integration**
  - **Sandwich integration**
- **The program is constructed and tested in small increments**
- **Errors are easier to isolate and correct**
- **Interfaces are more likely to be tested completely**
- **A systematic test approach is applied**

Bottom Up Integration Testing

Each component at lower hierarchy is tested individually and then the components that rely upon these components are tested.

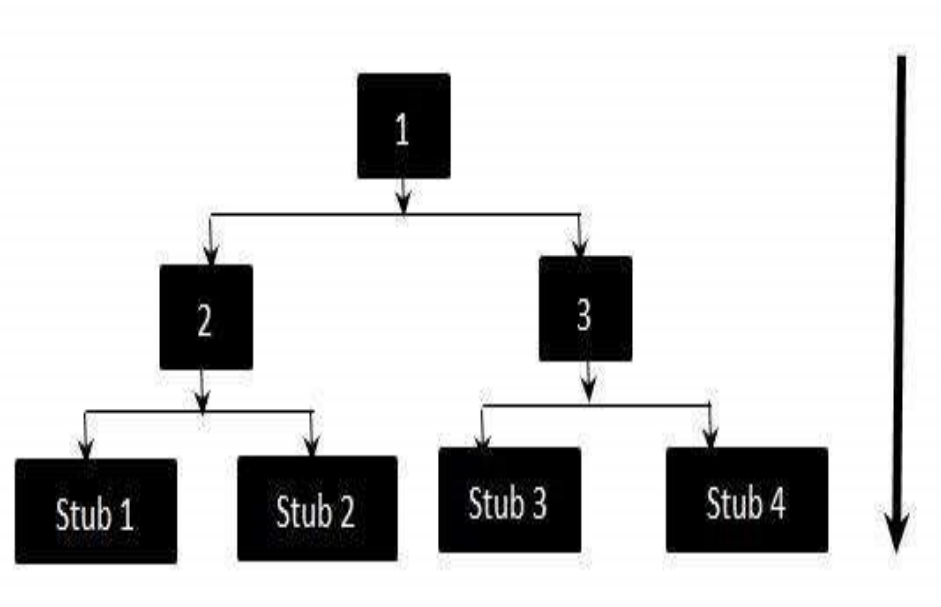


The order of Integration by Bottom-down approach will be:

4,2
5,2
6,3
7,3
2,1
3,1

Top-Down Integration Testing

- Testing technique used in order to simulate the behavior of the lower-level modules that are not yet integrated.
- Stubs are the modules that act as temporary replacement for a called module and give the same output as that of the actual product.
- The replacement for the 'called' modules is known as 'Stubs'.



The above diagrams clearly states that Modules 1, 2 and 3 are available for integration,
whereas, below modules are still under development that cannot be integrated at this point of time.
Hence, Stubs are used to test the modules. The order of Integration will be:

1,2
1,3
2,Stub 1
2,Stub 2
3,Stub 3
3,Stub 4

What are Stubs?

- **A stub is a replica of a module that collects the data and develops many possible data.**
- **it executes like an actual module and is mainly used to test modules.**
- **stubs are created by software developers in order to use them instead of modules if the particular modules are miss or not yet developed.**
- **By using these test stubs, the test engineers can simulate the performance of the lower-level modules, which are not yet joined with the software.**

What are Drivers?

- **The Drivers establish the test environments and takes care of the communication, estimates results, and also sends the reports.**
- **These are just like stubs and used by software test engineers in order to accomplish the missing or incomplete modules/ components requirements.**
- **The drivers are mainly developed in the Bottom-up approach of incremental integration testing.**
- **These can test the lower levels of the code when the upper-level modules or codes are not developed or missing.**

Examples of Stubs and Drivers

- Let us see an example of stubs and drivers, which help us to enhance our knowledge of stubs and drivers.
- Suppose we have one web application that contains **four different modules**, such as:
- **Module-P**
- **Module-Q**
- **Module-R**
- **Module-S**
- And all the modules, as mentioned earlier, are responsible for some individual activities or functionality, as we can observe in the following table:

Different Modules	Individual Activities
Module-P	Login page of the web application
Module-Q	Home-page of the web application
Module-R	Print Setup
Module-S	Log out page

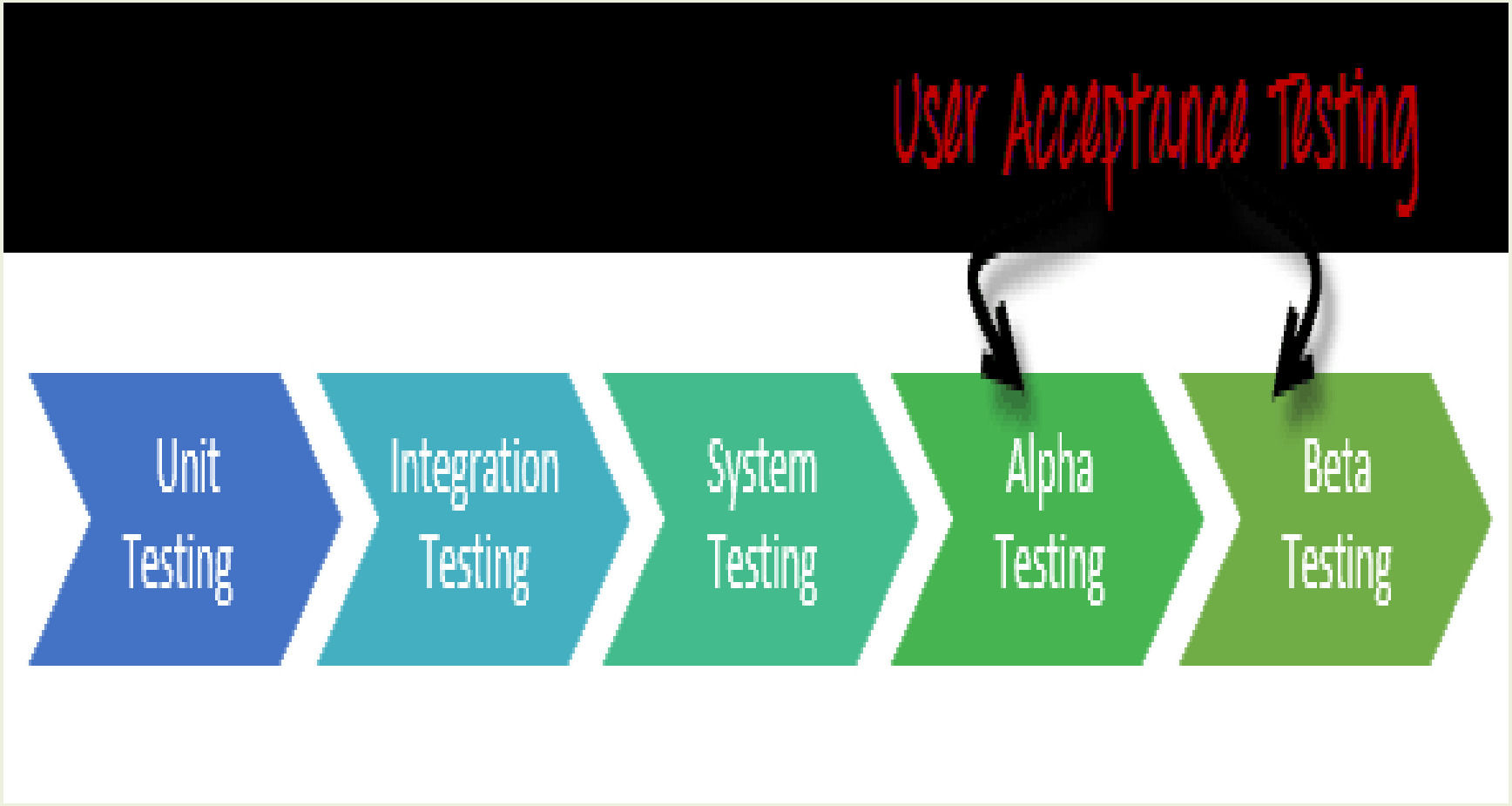
- Once **Module-P** is developed, it will go through the testing process. But, to perform and validate the testing methods regarding **Module-P**, they need **Module-Q**, which is not yet developed entirely and still in the developing process.

- And it is not possible to test **Module-P** on the lack of **Module-Q**. Thus, in such scenarios, we will take the help of **Stubs and Drivers** in the **software testing process.**

- The Stubs and drivers will replicate all the basic functionality and features displayed by the real **Module-Q**. And subsequently, it is being combined with **Module-P** in order to execute the testing process effectively.

Sandwich Integration

- **Consists of a combination of both top-down and bottom-up integration**
- **Occurs both at the highest level modules and also at the lowest level modules**
- **Proceeds using functional groups of modules, with each group completed before the next**
  - **High and low-level modules are grouped based on the control and data processing they provide for a specific program feature**
- **Reaps the advantages of both types of integration while minimizing the need for drivers and stubs**


User Acceptance Testing
Unit Testing
Integration Testing
System Testing
Alpha Testing
Beta Testing

Alpha and Beta Testing

- **Alpha testing**
  - **Conducted at the developer's site by end users**
  - **Software is used in a natural setting with developers watching intently**
  - **Testing is conducted in a controlled environment**
- **Beta testing**
  - **Conducted at end-user sites**
  - **Developer is generally not present**
  - **It serves as a live application of the software in an environment that cannot be controlled by the developer**
  - **The end-user records all problems that are encountered and reports these to the developers at regular intervals**
- **After beta testing is complete, software engineers make software modifications and prepare for release of the software product to the entire customer base**

System Testing

Different Types

- **Recovery testing**
  - Tests for recovery from system faults
  - Forces the software to fail in a variety of ways and verifies that recovery is properly performed
  - Tests reinitialization, checkpointing mechanisms, data recovery, and restart for correctness
- **Security testing**
  - Verifies that protection mechanisms built into a system will, in fact, protect it from improper access
- **Stress testing**
  - Executes a system in a manner that demands resources in abnormal quantity, frequency, or volume
- **Performance testing**
  - Tests the run-time performance of software within the context of an integrated system
  - Often coupled with stress testing and usually requires both hardware and software instrumentation

Which one of the following models is not suitable for accommodating any change?

a) Incremental Model
b) Prototyping Model
c) RAD Model
d) Waterfall Model

Answer: d

What is the major drawback of using RAD Model?

a) Highly specialized & skilled developers/designers are required
b) Increases reusability of components
c) Encourages customer/client feedback
d) Increases reusability of components, highly specialized & skilled developers/designers are required

Answer: d

Choose the correct option according to given below statement.

Statement 1: Umbrella activities are independent of any one framework activity and occur throughout the process.

Statement 2: software quality assurance, software configuration management are umbrella activity.

Statement 3: software quality assurance, software configuration management are not umbrella activity.

a) Only statement 1 is correct.

b) Statement 1 and statement 2 are correct.

c) Only statement 3 is correct.

d) Statement 1 and statement 3 are correct.

Answer:b

Software consists of _____.

a) Set of instructions + operating procedures

b) Programs + documentation + operating procedures

c) Programs + hardware manuals

d) Set of programs

Answer: b

Which is the most important feature of spiral model?

a) Quality management
b) Risk management
c) Performance management
d) Efficiency management

Answer: b

Which document is created by system analyst after the requirements are collected from Various stakeholders?

a)Software requirement specification
b) Software requirement validation
c) Feasibility study
d) Requirement Gathering

Answer: a

What is the meaning of requirement elicitation in software engineering?

a) Gathering of requirement.
b) Understanding of requirement.
c) Getting the requirements from client.
d) All of the above.

Answer: d

In ……. stage of SDLC, the statement of scope and objectives, opportunities and performance criteria are prepared.

a) Problem definition
b) feasibility study
c) system analysis
d) system development

Answer: a

Alpha testing is done at

- a) Developer’s end
- b) User’s end
- c) Developer’s & User’s end
- d) None of the mentioned

- Answer: a
- **Which of the following term describes testing?**
- a) Finding broken code
- b) Evaluating deliverable to find errors
- c) A stage of all projects
- d) None of the mentioned
- Answer: b

A tester is executing a test to evaluate and it complies with the user requirement for a certain field be populated by using a dropdown box containing a list of values, at that time tester is performing _____ .

a. White-box Testing
b. Black-box Testing
c. Load Testing
d. Regression Testing

Answer: b

Testing of individual components by the developers are comes under which testing?

a. Integration Testing
b. Validation Testing
c. Unit Testing
d. System Testing

- Answer: c

What is the main purpose of Integration testing?

- a. Interface errors
- b. Procedure errors
- c. Design errors
- d. None

- Answer: a

Select from which of the following regression testing should be performed.

- a) Every week
  b) After the software has changed
  c) As often as possible
  d) When the environment has changed
  e) Both option (b,d)

- Answer : e

Non-functional test does not belong to which of the following categories?

a) Performance
b) Usability and Security
c) State level Transition
d) all of the above

Answer: c

Which of the following is a form of functional testing?

- a) Security level testing
  b) Boundary value analysis
  c) Performance testing
  d) Usability testing

- Answer: b

- When different combination of input requires different combination of actions,_____ technique is used in such situation.
  a. Boundary Value Analysis
  b. Equivalence Partition
  c. Decision Table
  d. Decision Coverage

- Answer: c