Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

on

"alpha-beta testing"

.

Alpha Testing

- Alpha Testing is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users.
- Alpha testing is carried out by the testers who are internal employees of the organization.
- The main goal is to identify the tasks that a typical user might perform and test them.
- The focus of alpha testing is to simulate real users by using a black box and white box techniques.

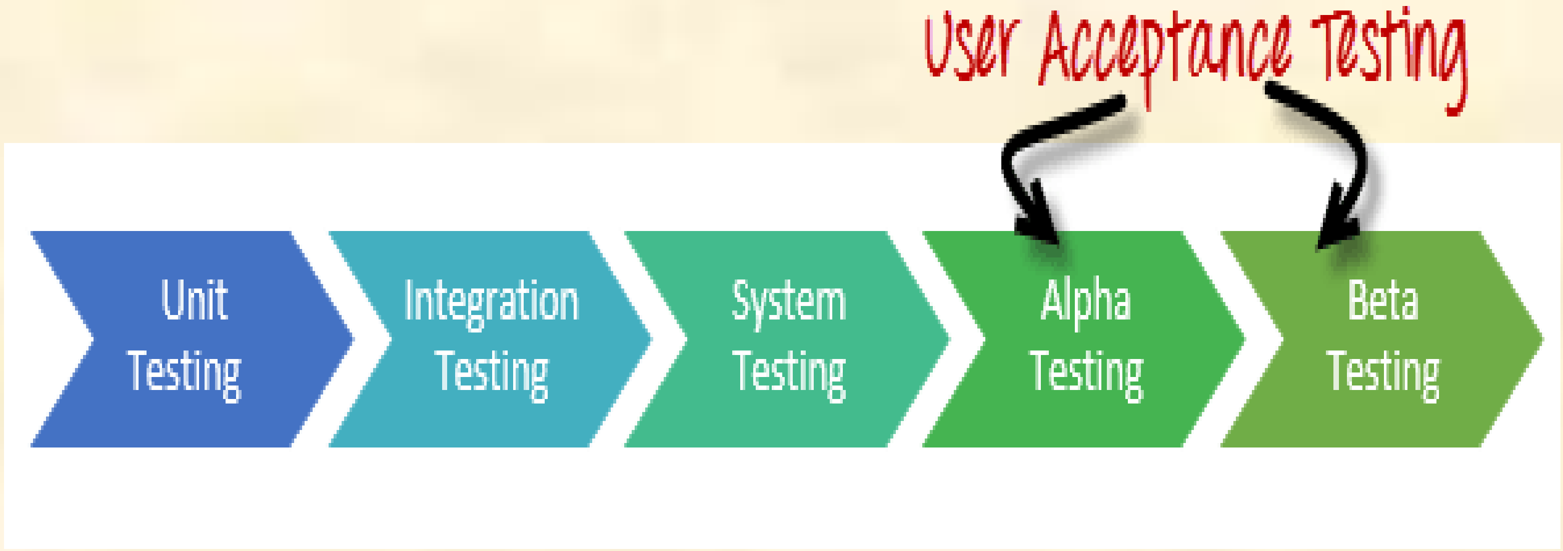
Beta Testing

- ❑ Beta Testing is a type of acceptance testing; it is the final test before shipping a product to the customers.
- ❑ Beta testing of a product is implemented by "real users “of the software application in a "real environment.”
- ❑ Direct feedback from customers is a major advantage of Beta Testing. This testing helps to test products in customer’s environment.

Beta Testing

- ❑ Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality.
- ❑ Beta testing reduces product failure risks and provides increased quality of the product through customer validation.

Alpha Beta Testing



User Acceptance Testing

Unit Testing

Integration Testing

System Testing

Alpha Testing

Beta Testing

Difference between Alpha and Beta Testing

S. No.	Alpha	Beta
1	Alpha testing involves both the white box and black box testing.	Beta testing commonly uses black-box testing.
2	Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.
3	Alpha testing is performed at the developer’s site.	Beta testing is performed at the end-user of the product.
4	Reliability and security testing are not checked in alpha testing.	Reliability, security and robustness are checked during beta testing.
5	Alpha testing ensures the quality of the product before forwarding to beta testing.	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.

Difference between Alpha and Beta Testing

S. No.	Functional	Non-Functional
1	Alpha testing requires a testing environment or a lab.	Beta testing doesn’t require a testing environment or lab.
2	Alpha testing may require a long execution cycle.	Beta testing requires only a few weeks of execution.
3	Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from the beta testing will be implemented in future versions of the product.
4	Multiple test cycles are organized in alpha testing.	Only one or two test cycles are there in beta testing.

Thank - You

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

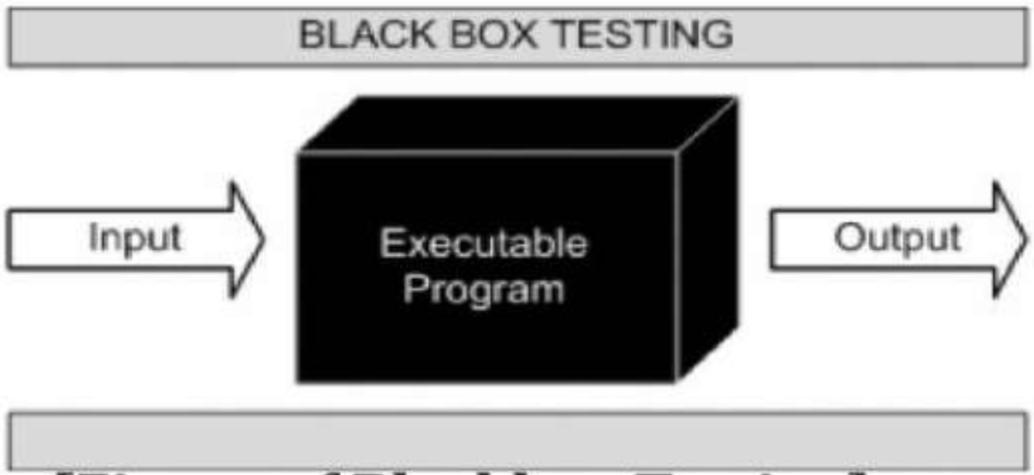
on

“Blackbox & Whitebox Testing”

.

Blackbox Testing

- In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.



[Figure of Blackbox Testing]

- The above Black Box executable program can be any software system you want to test.

Blackbox Testing

- ❑ By applying black-box techniques, you derive a set of test cases that satisfy the following criteria:
- ❑ (1) test cases that reduce, by a count that is greater than one the number of additional test cases that must be designed to achieve reasonable testing, and
- ❑ (2) test cases that tell you something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand.

Blackbox Techniques/ Methods:

- Equivalence partitioning: It is a software test design technique that involves dividing input values into valid and invalid partitions and selecting representative values from each partition as test data.

- Boundary Value Analysis: It is a software test design technique that involves determination of boundaries for input values and selecting values that are at the boundaries and just inside/ outside of the boundaries as test data.

- Cause effect graphing: It is a software test design technique that involves identifying the cases (input conditions) and effects (output conditions), producing a Cause-Effect Graph, and generating test cases accordingly.

Blackbox Testing 1. Equivalence partitioning:

- Equivalence Partitioning also called as equivalence class partitioning. It can be applied at any level of testing and is often a good technique to use first.
- The idea behind this technique is to divide (i.e. to partition) a set of test conditions into groups or sets that can be considered the same (i.e. the system should handle them equivalently), hence ‘equivalence partitioning’.
- Equivalence partitioning is a testing technique where input values set into classes for testing.
- Valid Input Class = Keeps all valid inputs.
- Invalid Input Class = Keeps all Invalid inputs.

- Equivalence classes may be defined according to the following guidelines:
- 1. If an input condition specifies a range, one valid and two invalid equivalence classes are defined.
- 2. If an input condition requires a specific value, one valid and two invalid equivalence classes are defined.
- 3. If an input condition specifies a member of a set, one valid and one invalid equivalence class are defined.
- 4. If an input condition is Boolean, one valid and one invalid class are defined.
- By applying the guidelines for the derivation of equivalence classes, test cases for each input domain data item can be developed and executed.

Blackbox Testing 1. Equivalence partitioning:

- ❑ Example-1:A text field permits only numeric characters.
- ❑ Length must be 6-10 characters long
- ❑ Partition according to the requirement should be like this:

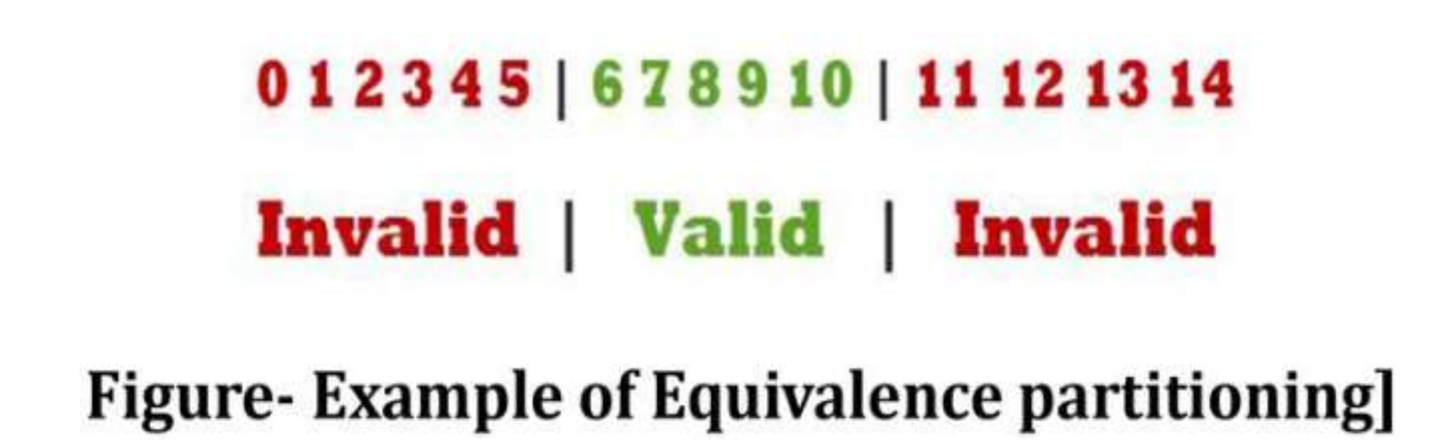


Figure- Example of Equivalence partitioning]

- ❑ While evaluating Equivalence partitioning, values in all partitions are equivalent that’s why 0-5 are equivalent, 6 – 10 are equivalent and are equivalent.

Blackbox Testing 1. Equivalence partitioning:

- ❑ At the time of testing, test 4 and 12 as invalid values and 7 as valid one.
- ❑ It is easy to test input ranges 6–10 but harder to test input ranges
- ❑ Testing will be easy in the case of lesser test cases but you should be very careful.
- ❑ Assuming, valid input is 7.
- ❑ That means, you belief that the developer coded the correct valid range (6-10).

Blackbox Testing 2. Boundary Value Analysis:

- It’s widely recognized that input values at the extreme ends of input domain cause more errors in system. More application errors occur at the boundaries of input domain.

- ‘Boundary value analysis’ testing technique is used to identify errors at boundaries rather than finding those exist in center of input domain.

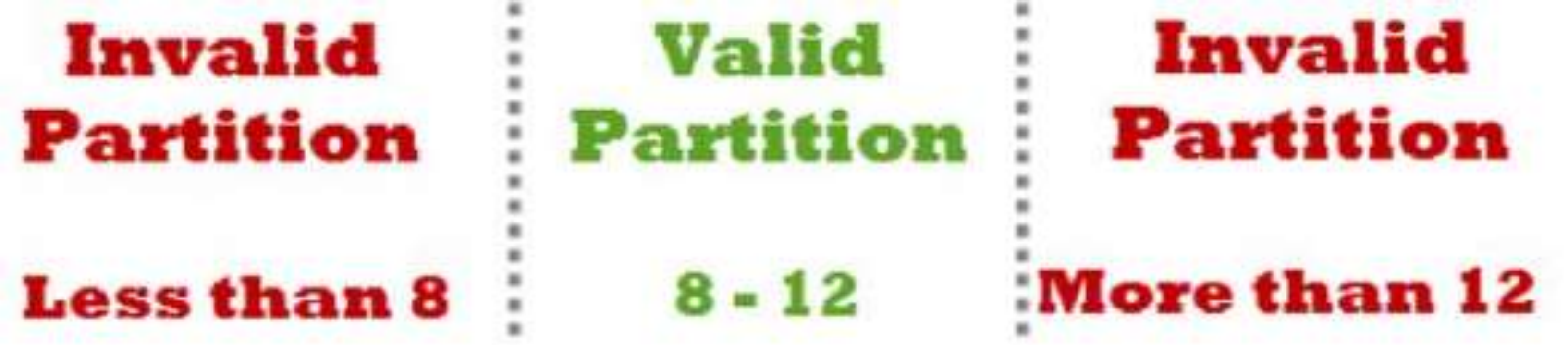
- Boundary value analysis is a next part of Equivalence partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.

- Boundary value analysis is the process of testing between extreme ends or boundaries between partitions' of the input values.

Blackbox Testing 2. Boundary Value Analysis:

- ❑ Boundary value analysis is the process of testing between extreme ends or boundaries between partitions' of the input values.
- ❑ So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called "boundary value analysis testing".
- ❑ Example-1:Suppose you have very important tool at office, accepts valid User Name and Password field to work on that tool, and accepts minimum 8 characters and maximum 12 characters.
- ❑ Valid range 8-12, Invalid range 7 or less than 7 and Invalid range 13 or more than 13.

Blackbox Testing 2. Boundary Value Analysis:



Invalid Partition	Valid Partition	Invalid Partition
Less than 8	8 - 12	More than 12

- Write Test Cases for Valid partition value, Invalid partition value and exact boundary value.
- Test Cases 1: Consider password length less than 8.
- Test Cases 2: Consider password of length exactly 8.
- Test Cases 3: Consider password of length between 9 & 11.
- Test Cases 4: Consider password of length exactly 12.
- Test Cases 5: Consider password of length more than 12.

Blackbox Testing 3. Cause effect graphing:

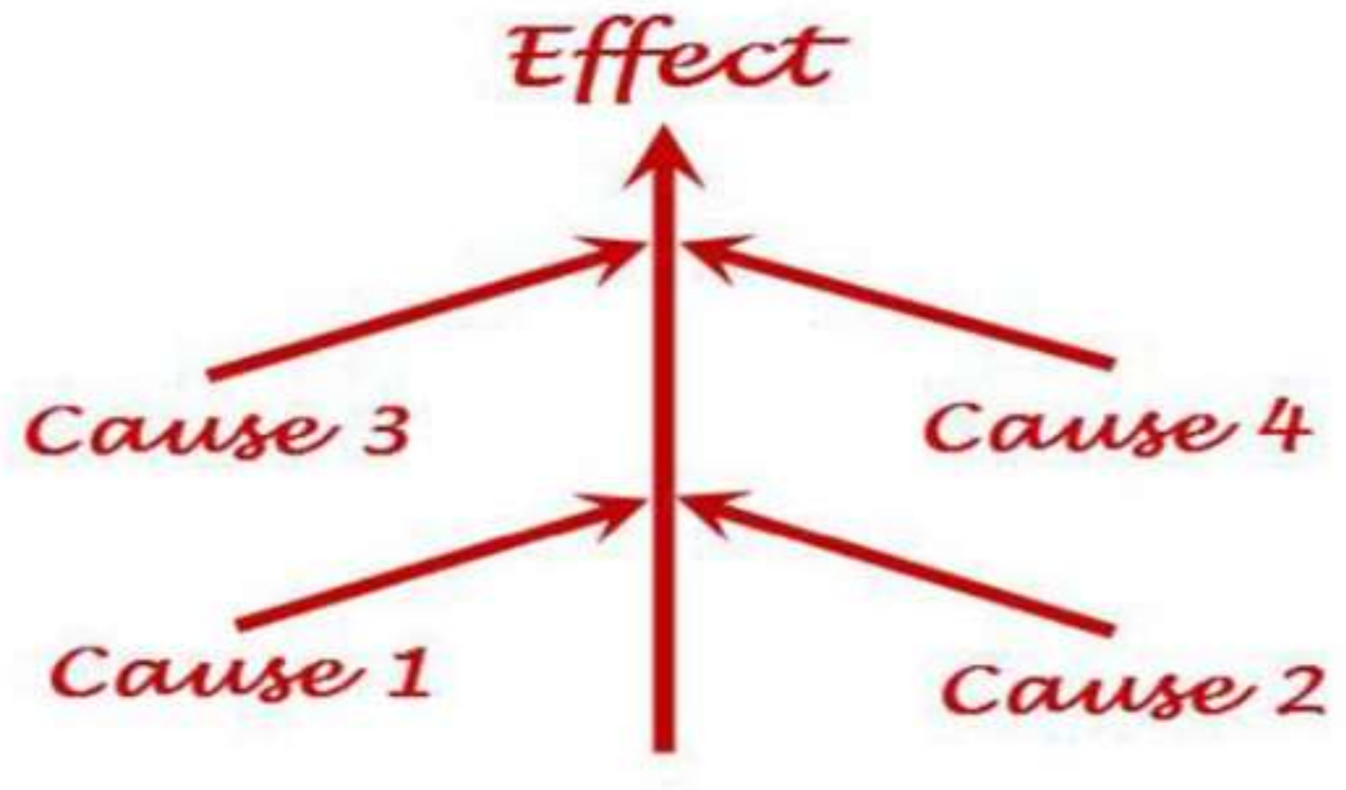
- ❑ A “Cause” stands for a separate input condition that fetches about an internal change in the system.
- ❑ An “Effect” represents an output condition, a system transformation or a state resulting from a combination of causes.
- ❑ It is a testing technique that aids in choosing test cases that logically relate Causes (inputs) to Effects (outputs) to produce test cases.

Blackbox Testing 3. Cause effect graphing:

- ❑ According to Myer Cause & Effect Graphing is done through the following steps:
- ❑ Step – 1: For a module, identify the input conditions (causes) and actions (effect).
- ❑ Step – 2: Develop a cause-effect graph.
- ❑ Step – 3: Transform cause-effect graph into a decision table.
- ❑ Step – 4: Convert decision table rules to test cases. Each column of the decision table represents a test case.

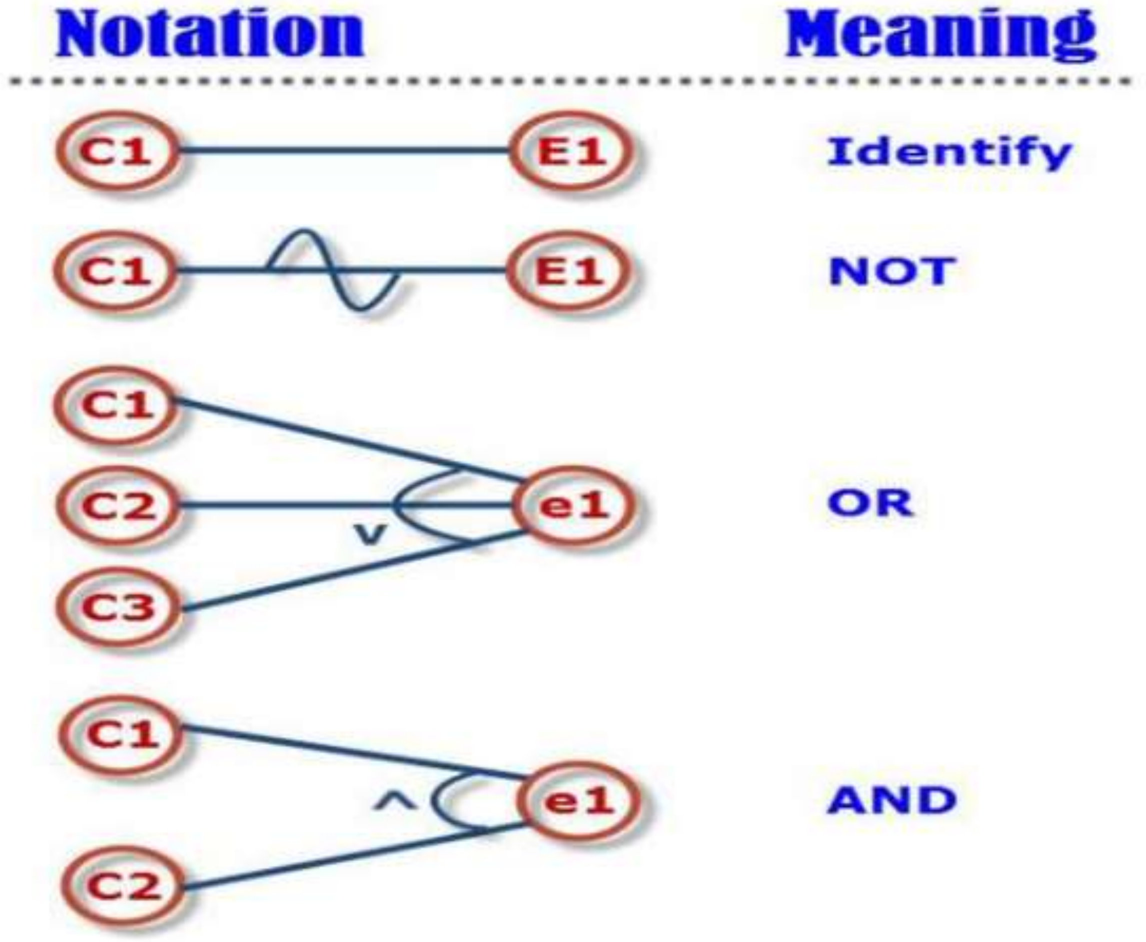
Blackbox Testing 3. Cause effect graphing:

Cause Effect - Flow Diagram



[Figure of Cause Effect-Flow Diagram]

Blackbox Testing 3. Cause effect graphing:



[Symbols used in Cause Effect-Flow Diagram]

Blackbox Testing 3. Cause effect graphing:

- ❑ Just assume that each node having the value 0 or 1 where 0 shows the ‘absent state’ and 1 shows the ‘present state’.
- ❑ The identity function states when c1 = 1, e1 = 1 or we can say if c0 = 0 and e0 = 0.
- ❑ The NOT function states that, if C1 = 1, e1= 0 and vice-versa.
- ❑ Likewise, OR function states that, if C1 or C2 or C3 = 1, e1 = 1 else e1 = 0.
- ❑ The AND function states that, if both C1 and C2 = 1, e1 = 1, else e1 = 0.
- ❑ The AND and OR functions are permitted to have any number of inputs.

Blackbox Testing 3. Cause effect graphing:

- Test cases can be designed for the triangle problem in the following ways
- Firstly: Recognize and describe the input conditions (causes) and actions (effect).
- The causes allocated by letter “C” are as follows,

C1: Side “x” is less than sum of “y” and “z”

C2: Side “y” is less than sum of “x” and “z”

C3: Side “z” is less then sum of “x” and “y”

C4: Side “x” is equal to side “y”

C5: Side “x” is equal to side “z”

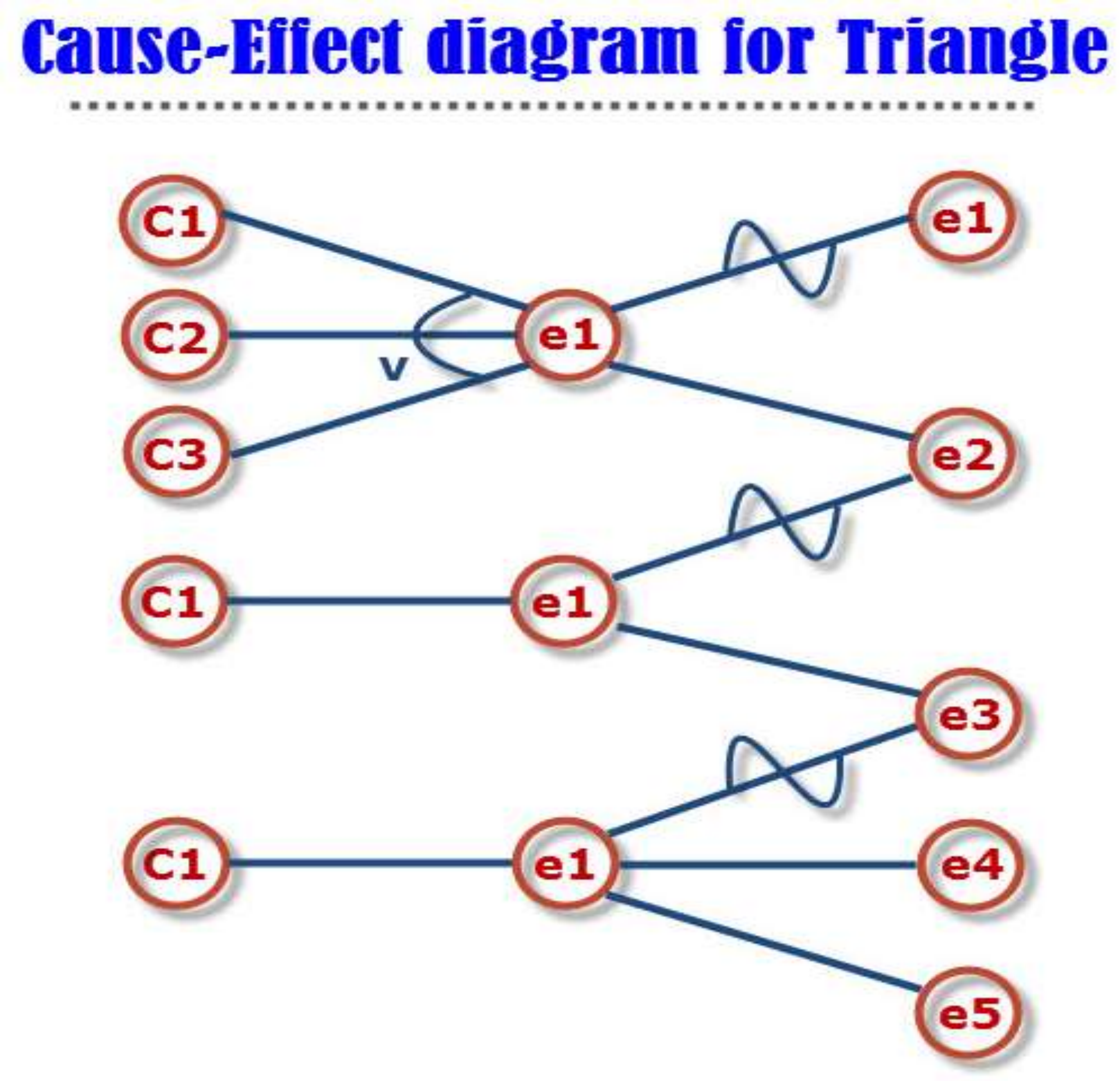
C6: Side “y” is equal to side “z”

Blackbox Testing 3. Cause effect graphing:

- The effects designated by letter “e” are as follows,
- e1: Not a triangle
- e2: Scalene triangle
- e3: Isosceles triangle.
- e4: Equilateral triangle
- e5: Impossible

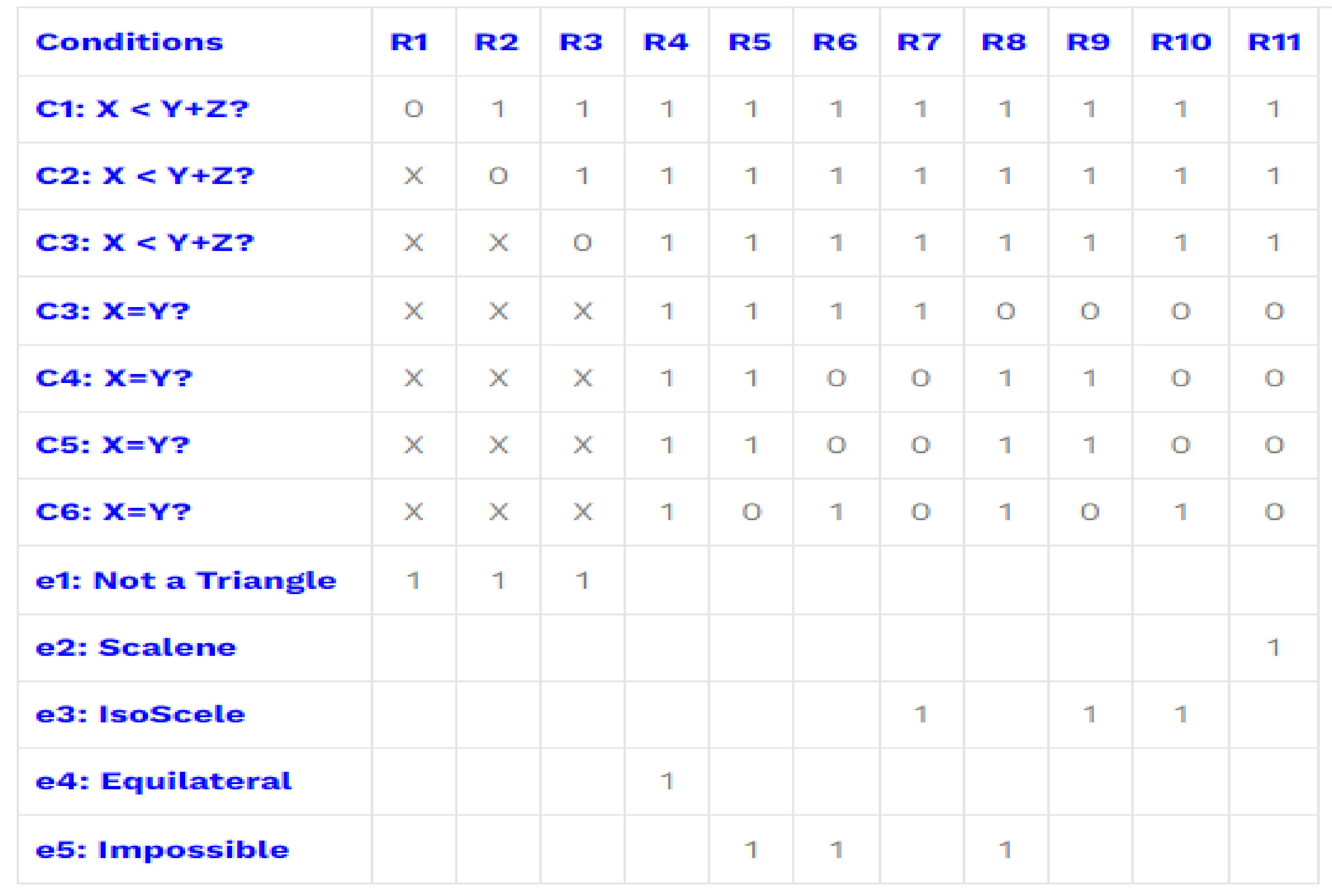
Blackbox Testing 3. Cause effect graphing:

Secondly: Build up a cause-effect graph



Blackbox Testing 3. Cause effect graphing:

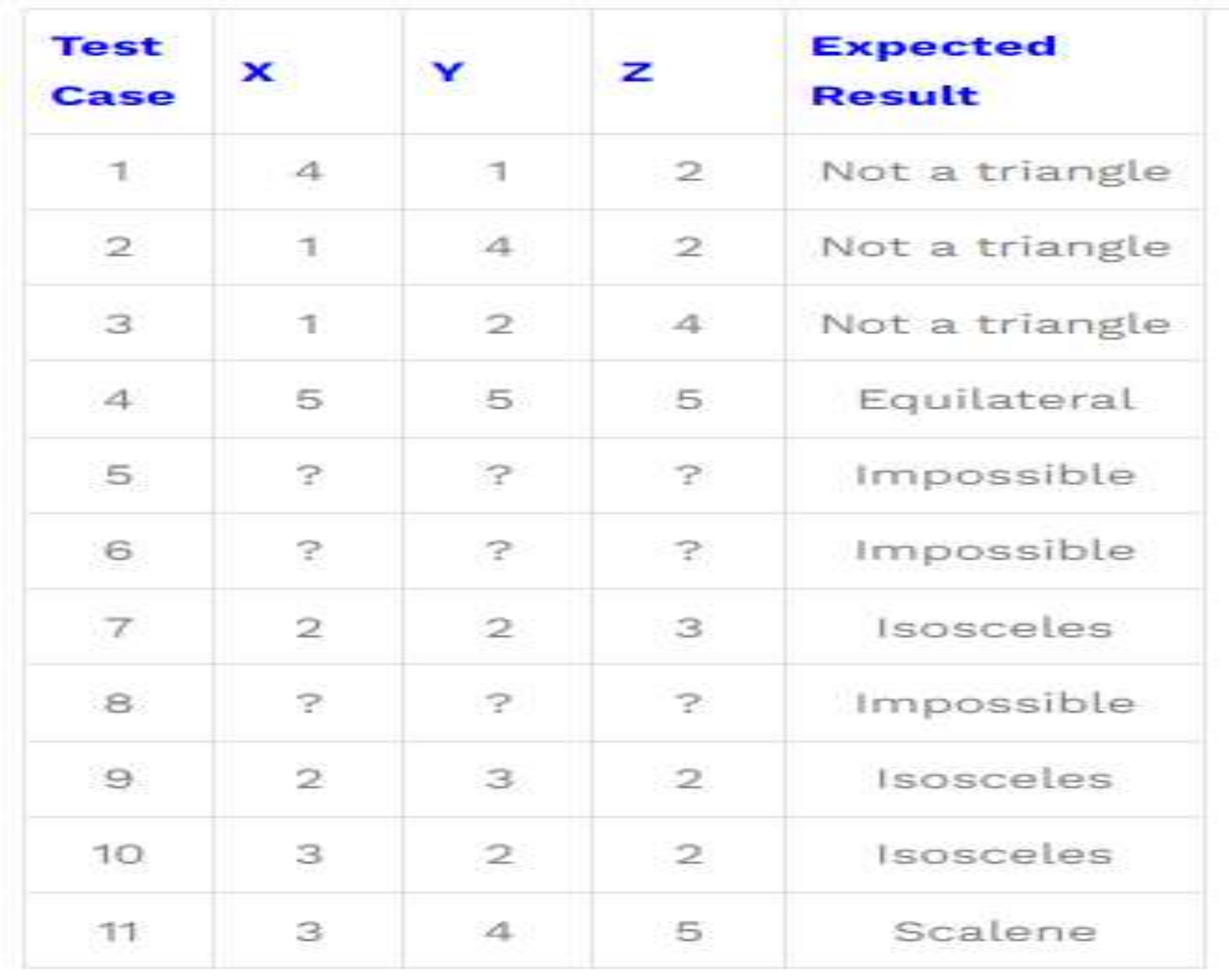
Third: Convert cause-effect graph into a decision table



Conditions	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
C1: X < Y+Z?	0	1	1	1	1	1	1	1	1	1	1
C2: X < Y+Z?	X	0	1	1	1	1	1	1	1	1	1
C3: X < Y+Z?	X	X	0	1	1	1	1	1	1	1	1
C3: X=Y?	X	X	X	1	1	1	1	0	0	0	0
C4: X=Y?	X	X	X	1	1	0	0	1	1	0	0
C5: X=Y?	X	X	X	1	1	0	0	1	1	0	0
C6: X=Y?	X	X	X	1	0	1	0	1	0	1	0
e1: Not a Triangle	1	1	1								
e2: Scalene											1
e3: IsoScele							1		1	1	
e4: Equilateral				1							
e5: Impossible					1	1		1			

Blackbox Testing 3. Cause effect graphing:

Fourth : 11 test cases according to the 11 rules.

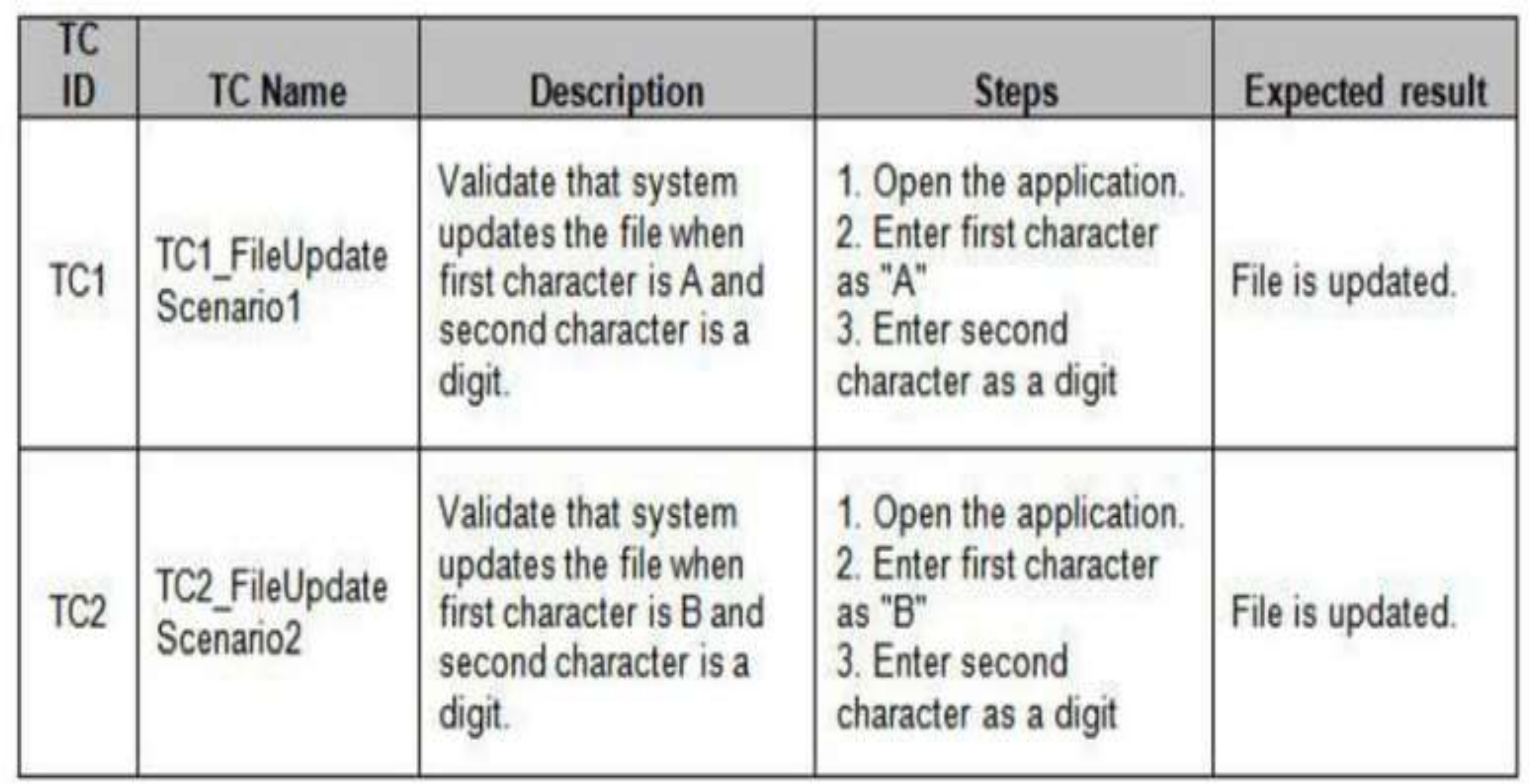


Test Case	X	Y	Z	Expected Result
1	4	1	2	Not a triangle
2	1	4	2	Not a triangle
3	1	2	4	Not a triangle
4	5	5	5	Equilateral
5	?	?	?	Impossible
6	?	?	?	Impossible
7	2	2	3	Isosceles
8	?	?	?	Impossible
9	2	3	2	Isosceles
10	3	2	2	Isosceles
11	3	4	5	Scalene

Blackbox Testing Example:

- The “Print message” is software that read two characters and, depending of their values, messages must be printed.
- The first character must be an “A” or a “B”.
- The second character must be a digit.
- If the first character is an “A” or “B” and the second character is a digit, the file must be updated.
- If the first character is incorrect (not an “A” or “B”), the message X must be printed.
- If the second character is incorrect (not a digit), the message Y must be printed.

Blackbox Testing Example:



TC ID	TC Name	Description	Steps	Expected result
TC1	TC1_FileUpdate Scenario1	Validate that system updates the file when first character is A and second character is a digit.	1. Open the application. 2. Enter first character as "A" 3. Enter second character as a digit	File is updated.
TC2	TC2_FileUpdate Scenario2	Validate that system updates the file when first character is B and second character is a digit.	1. Open the application. 2. Enter first character as "B" 3. Enter second character as a digit	File is updated.

[Test cases for previous example]

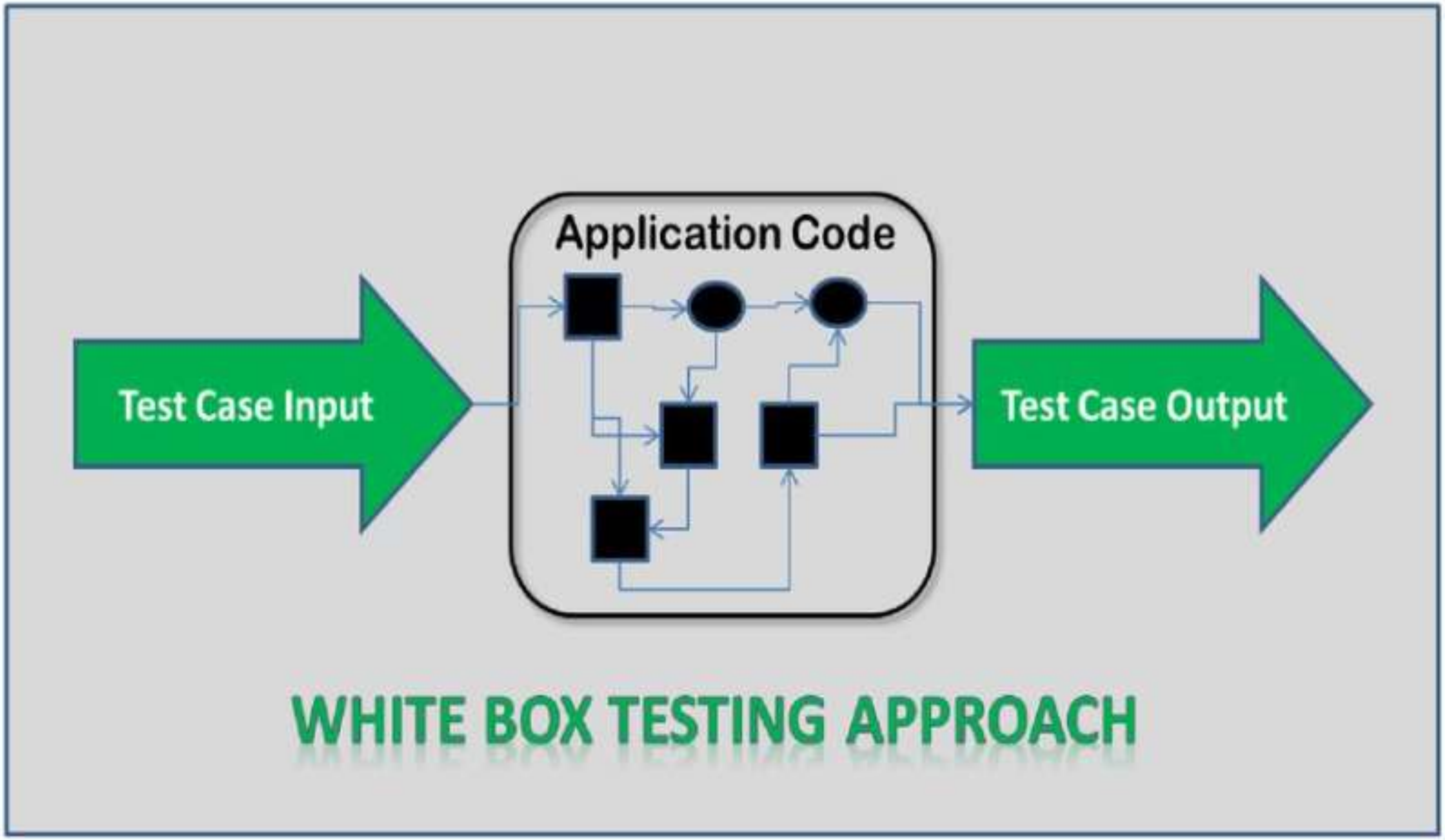
Whitebox Testing :

- White Box Testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
- White Box Testing is like the work of a mechanic who examines the engine to see why the car is not moving.

Whitebox Testing :

- ❑ Using white-box testing methods, you 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.
- ❑ (4) exercise internal data structures to ensure their validity.

Whitebox Testing :



[Figure of Whitebox Testing Approach]

Whitebox Testing :

Why and When White-Box Testing:

- ❑ White box testing is mainly used for detecting logical errors in the program code.
- ❑ It is used for debugging a code, finding random typographical errors, and uncovering incorrect programming assumptions.
- ❑ White box testing is done at low level design and implementable code.
- ❑ It can be applied at all levels of system development especially Unit, system and integration testing.
- ❑ White box testing can be used for other development artifacts like requirements analysis, designing and test cases.

Whitebox Testing Techniques:

- ❑ Following are Whitebox testing techniques:
- ❑ Statement coverage: This technique is aimed at exercising all programming statements with minimal tests.
- ❑ Branch and decision coverage: This technique is running a series of tests to ensure that all branches are tested at least once.
- ❑ Tools: An example of a tool that handles branch coverage testing for C, C++ and Java applications is TCAT-PATH
- ❑ Path coverage: This technique corresponds to testing all possible paths which means that each statement and branch is covered.

Whitebox Testing 1. Statement coverage:

- ❑ Statement coverage is a white box testing technique, which involves the execution of all the statements at least once in the source code.
- ❑ Through statement coverage we can identify the statements executed and where the code is not executed because of blockage.
- ❑ The statement coverage covers only the true conditions. The main drawback of this technique is that we cannot test the false condition in it.
- ❑ The statement coverage is count as per below formula:
- ❑ **(Statement coverage = No of statements Executed/ - Total no of statements in the source code * 100)**

Whitebox Testing Example:

- **Read A Read B**
- **if A>B**
- **Print “A is greater than B”**
- **else**
- **Print "B is greater than A"**
- **endif**

Set1 :If A =5, B =2

- No of statements Executed: 5
- Total no of statements in the source code: 7

  Statement coverage =5/7*100 = 71.00%

Set1 :If A =2, B =5 [False-Not supported by Statement coverage]

- No of statements Executed: 6
- Total no of statements in the source code: 7

  Statement coverage =6/7*100 = 85.20%

Whitebox Testing 2. Branch coverage:

- Branch coverage is also known as Decision coverage or all-edges coverage.
- It covers both the true and false conditions unlikely the statement coverage.
- A branch is the outcome of a decision, so branch coverage simply measures which decision outcomes have been tested.
- This sounds great because it takes a more in-depth view of the source code than simple statement coverage
- The formula to calculate decision coverage is:
- **Decision Coverage=(Number of decision outcomes executed/Total number of decision outcomes)*100%**

Whitebox Testing Example:

- READ X READ Y
- IF (X > Y) PRINT “X is greater that Y” ENDIF
- To get 100% statement coverage only one test case is sufficient for this pseudo-code.
- TEST CASE 1: X=10 Y=5However this test case won’t give you 100% decision coverage as the FALSE condition of the IF statement is not exercised.
- In order to achieve 100% decision coverage we need to exercise the FALSE condition of the IF statement which will be covered when X is less than Y.

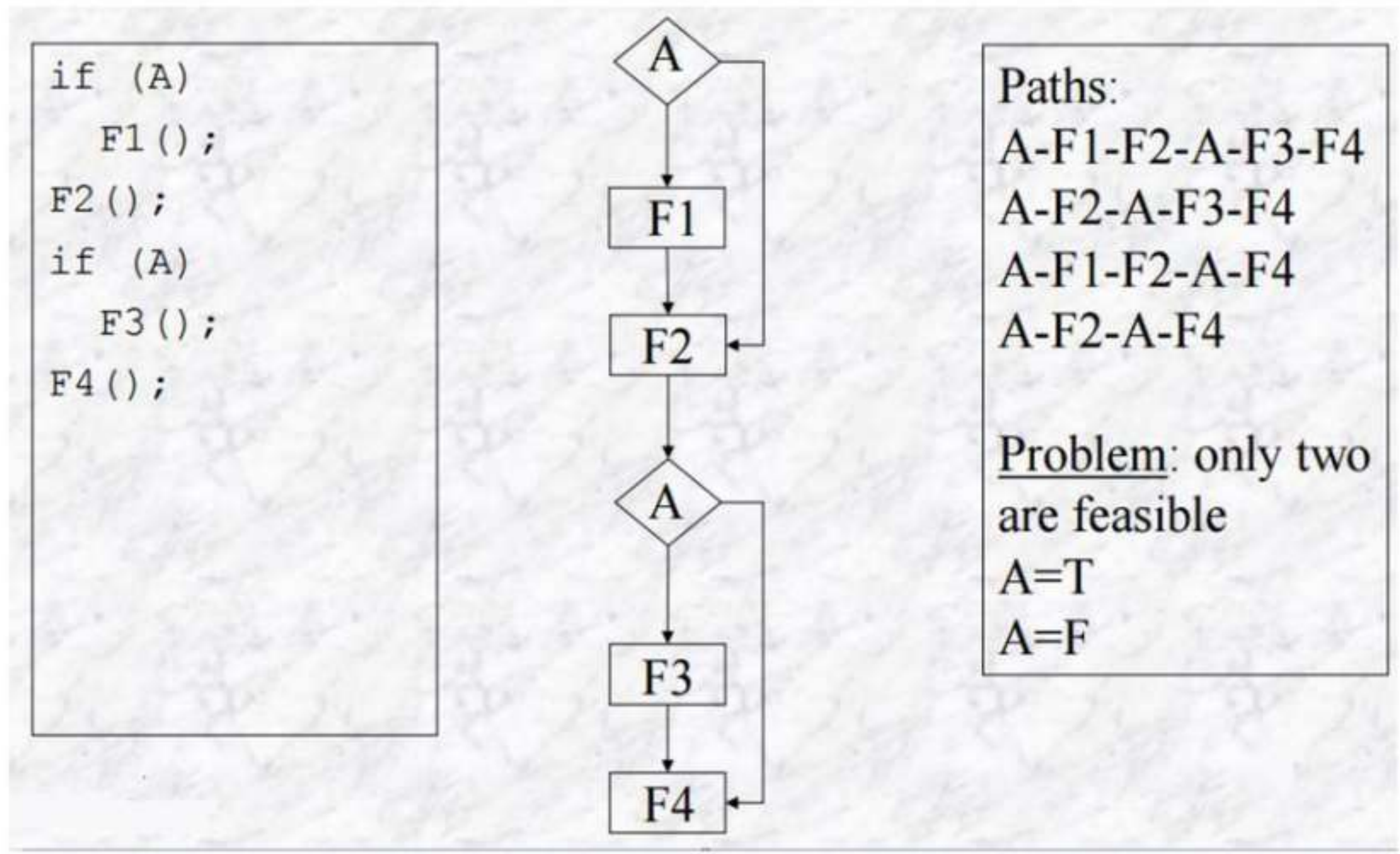
Whitebox Testing Example:

- So the final TEST SET for 100% decision coverage will be:
- TEST CASE 1: X=10, Y=5
- TEST CASE 2: X=2, Y=10
- Note: 100% decision coverage guarantees 100% statement coverage but 100% statement coverage does not guarantee 100% decision coverage.

Whitebox Testing 3. Path coverage:

- ❑ 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.

Whitebox Testing 3. Path coverage:



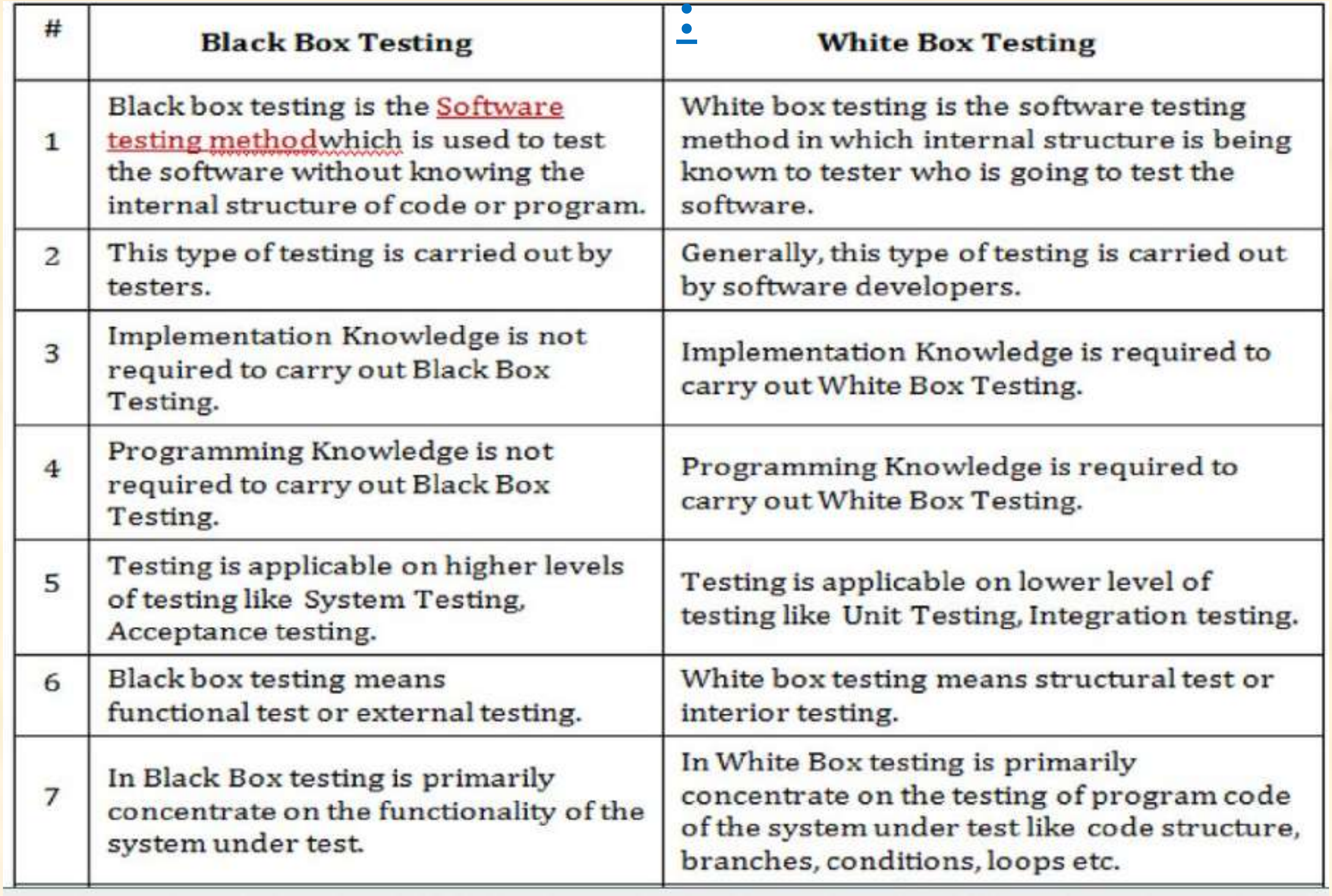
```
if  (A)
   F1();
F2();
if  (A)
   F3();
F4();
```

Paths:
A-F1-F2-A-F3-F4
A-F2-A-F3-F4
A-F1-F2-A-F4
A-F2-A-F4

Problem: only two are feasible
A=T
A=F

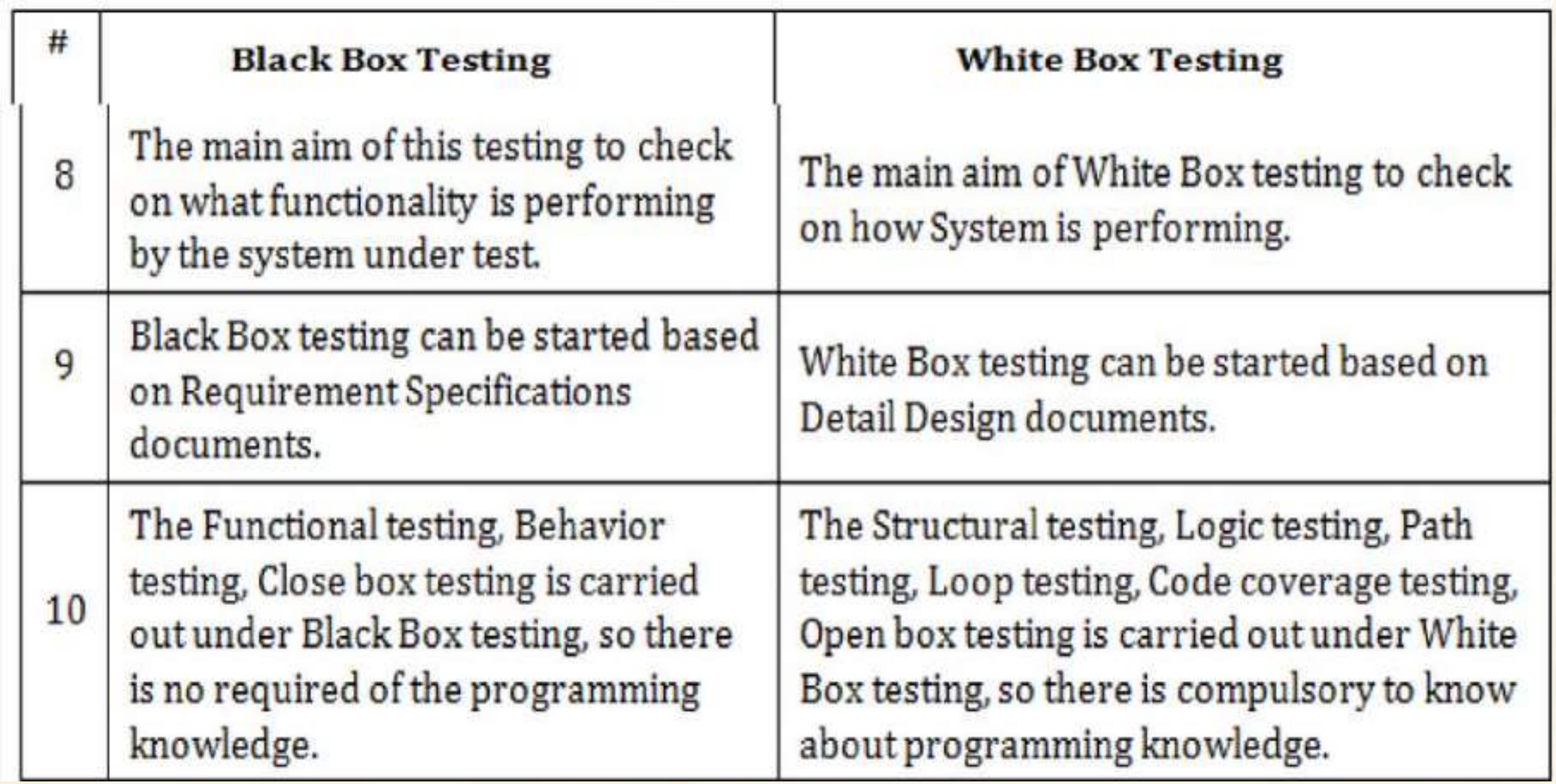
[Figure of Path coverage Example]

Difference between Blackbox &Whitebox Testing :



#	Black Box Testing	White Box Testing
1	Black box testing is the Software testing methodwhich is used to test the software without knowing the internal structure of code or program.	White box testing is the software testing method in which internal structure is being known to tester who is going to test the software.
2	This type of testing is carried out by testers.	Generally, this type of testing is carried out by software developers.
3	Implementation Knowledge is not required to carry out Black Box Testing.	Implementation Knowledge is required to carry out White Box Testing.
4	Programming Knowledge is not required to carry out Black Box Testing.	Programming Knowledge is required to carry out White Box Testing.
5	Testing is applicable on higher levels of testing like System Testing, Acceptance testing.	Testing is applicable on lower level of testing like Unit Testing, Integration testing.
6	Black box testing means functional test or external testing.	White box testing means structural test or interior testing.
7	In Black Box testing is primarily concentrate on the functionality of the system under test.	In White Box testing is primarily concentrate on the testing of program code of the system under test like code structure, branches, conditions, loops etc.

Difference between Blackbox &Whitebox Testing :



#	Black Box Testing	White Box Testing
8	The main aim of this testing to check on what functionality is performing by the system under test.	The main aim of White Box testing to check on how System is performing.
9	Black Box testing can be started based on Requirement Specifications documents.	White Box testing can be started based on Detail Design documents.
10	The Functional testing, Behavior testing, Close box testing is carried out under Black Box testing, so there is no required of the programming knowledge.	The Structural testing, Logic testing, Path testing, Loop testing, Code coverage testing, Open box testing is carried out under White Box testing, so there is compulsory to know about programming knowledge.

Thank - You

Cyclomatic Complexity

Cyclomatic Complexity Measures

Complexity is a software metric that given the quantitative measure of logical complexity of the program.

The Cyclomatic complexity defines the number of independent paths in the basis set of the program that provides the upper bound for the number of tests that must be conducted to ensure that all the statements have been executed atleast once.

There are three methods of computing Cyclomatic complexities.

Method 1: Total number of regions in the flow graph is a Cyclomatic complexity.

Method 2: The Cyclomatic complexity, V (G) for a flow graph G can be defined as

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

Where: E is total number of edges in the flow graph.

N is the total number of nodes in the flow graph.

Method 3: The Cyclomatic complexity V (G) for a flow graph G can be defined as

$$V(G) = P + 1$$

Where: P is the total number of predicate nodes contained in the flow G.

Let us understand computation of Cyclomatic complexity with the help of an

example. Consider following code fragment with line numbered

{

1
.
I
f
(

{

3

.

I

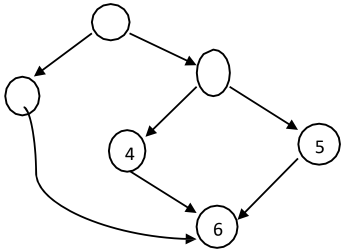
f

(

a

<

c



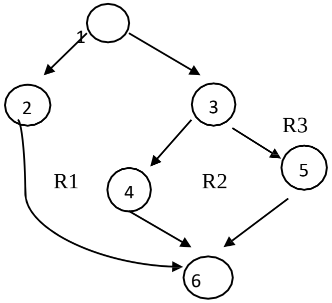
)

4

.

F

Step 2. Compute region, Predicate (i.e decision nodes) edges and total nodes in the flow graph



(

)

;

else

5. F3 () ;

}

6

.

}

- There are 3 regions denoted by R1, R2 and R3.

To compute Cyclomatic complexity we will follow these steps –

Step 1. Design flow graph for given code fragment.

- Nodes 1 and 3 are predicate nodes because which branch to be followed is decided at these points.
- Total edges = 7

  Total nodes = 6

Step 3. Apply formulas in order to compute Cyclomatic complexity.

1) Cyclomatic complexity **V(G)** = Total number of region = 3
2) Cyclomatic complexity **V(G)** = E - N + 2

   Cyclomatic complexity **V (G)** = 7- 6 +2 = 3

3) Cyclomatic complexity **V(G)** = P +1

   V (G) = 2 + 1 = 3

   Where P is predicate nodes (node 1 and node 2) are predicate nodes because from these nodes only the decision of which path is to be followed is taken.

   Thus Cyclomatic complexity is **3** for given code.

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

on

“Debugging”

.

Debugging

- To launch an application into the market, it is very necessary to cross-check it multiple times to deliver an error-free product.
- In the context of software engineering, debugging is the process of fixing a bug in the software. In other words, it refers to identifying, analyzing, and removing errors.
- Software programs undergo heavy testing, updating, troubleshooting, and maintenance during the development process.

Debugging

- Debugging is a developer activity and effective debugging is very important before testing begins to increase the quality of the system.

- Debugging is a developer activity and effective debugging is very important before testing begins to increase the quality of the system.

- Debugging will not give confidence that the system meets its requirements completely, but testing gives confidence.

Why do we need Debugging

- ❑ The process of debugging begins as soon as the code of the software is written. Then, it continues in successive stages as code is combined with other units of programming to form a software product. *Debugging has many benefits such as:*

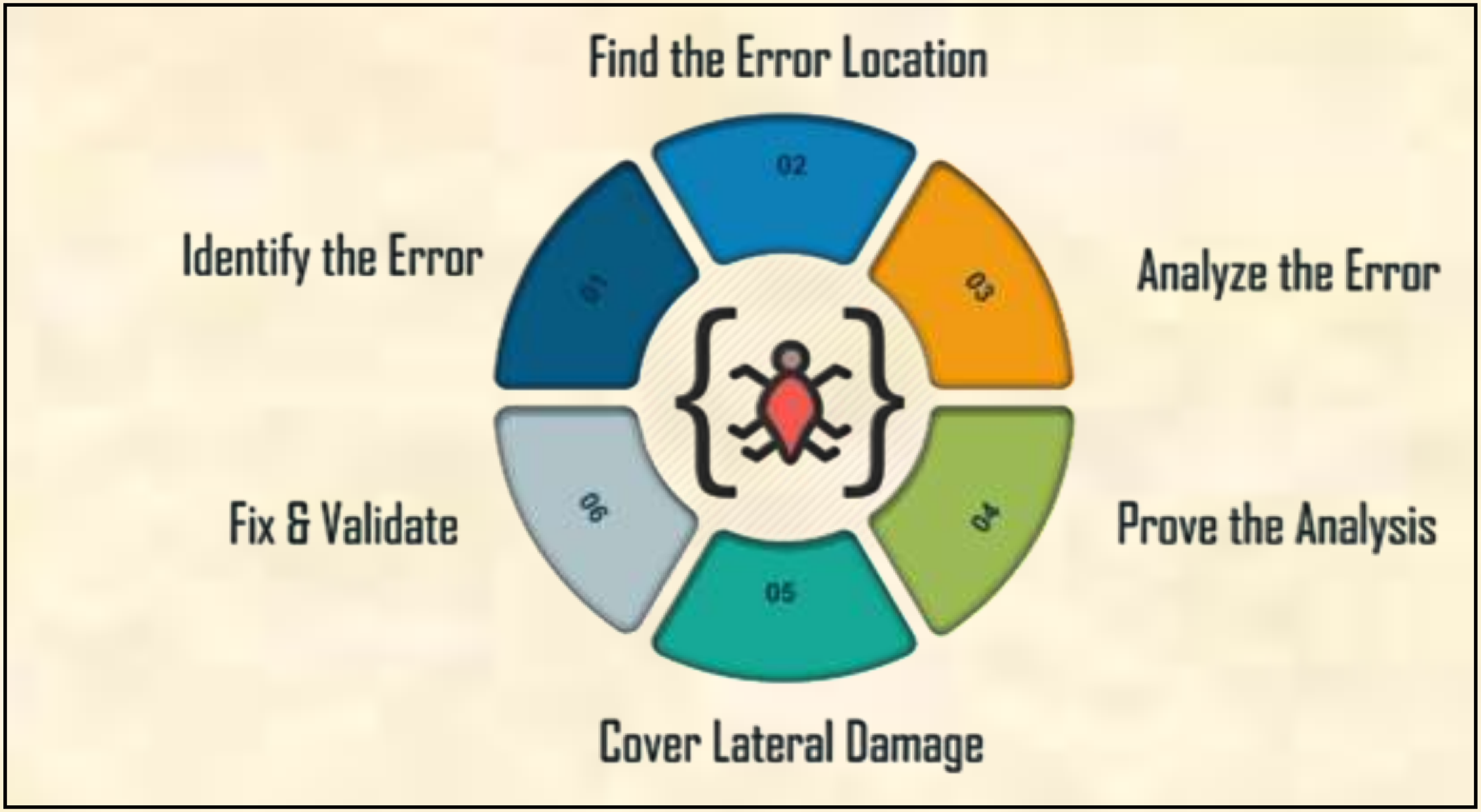
- ✓ It **reports** an **error condition immediately**. This allows earlier detection of an error and makes the process of software development stress-free and unproblematic.

- ✓ It also provides **maximum useful information** of data structures and allows easy interpretation.

Why do we need Debugging

- ✓ Debugging assists the developer in **reducing useless** and **distracting information**.
- ✓ Through debugging the developer can avoid **complex one-use testing code** to save time and energy in software development.

Steps involved in Debugging



Steps involved in Debugging

1) **Identify the Error:** A bad identification of an error can lead, to wasted developing time. It is usual that production errors reported by users are hard to interpret and sometimes the information we receive is misleading. It is import to identify the actual error.

2) **Find the Error Location:** After identifying the error correctly, you need to go through the code to find the exact spot where the error is located. In this stage, you need to focus on finding the error instead of understanding it.

Steps involved in Debugging

3) **Analyze the Error:** In the third step, you need to use a bottom-up approach from the error location and analyze the code. This helps you in understanding the error.

4) **Prove the Analysis:** Once you are done analyzing the original bug, you need to find a few more errors that may appear on the application. This step is about writing automated tests for these areas with the help of a test framework.

Steps involved in Debugging

5) **Cover Lateral Damage:** In this stage, you need to create or gather all the unit tests for the code where you are going to make changes. Now, if you run these unit tests, they all should pass.

6) **Fix & Validate:** The final stage is the fix all the errors and run all the test scripts to check if they all pass.

Debugging Tools

- Debugging tool is a computer program that is used to test and debug other programs. A lot of public domain software like *gdb* and *dbx* are available for debugging. They offer console-based command-line interfaces. Examples of automated debugging tools include *code-based tracers*, *profilers*, *interpreters*, etc. Some of the widely used debuggers are:

- ✓ *Radare2*
- ✓ *WinDbg*
- ✓ *Valgrind*

Thank - You

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

on

“Introduction to Software Testing”

.

Software Testing

- *Software Testing* is a process of evaluating the functionality of a software application to find any software bugs.
- It checks whether the developed software met the specified requirements and identifies any defect in the software in order to produce a quality product.
- It is basically executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Software Testing

- ❑ It is also stated as the process of verifying and validating a software product. It checks whether the software product:

- ✓ *Meets the business and technical requirements that guided its design and development*
- ✓ *Works as per the requirement*
- ✓ *Can be implemented with the same characteristics*

Software Testing

- ❑ Testing includes an examination of code and the execution of code in various environments, conditions as well as all the examining aspects of the code.
- ❑ In the current scenario of software development, a testing team may be separate from the development team so that Information derived from testing can be used to correct the process of software development.

Software Testing

- ❑ The success of software depends upon acceptance of its targeted audience, easy graphical user interface, strong functionality load test, etc.
- ❑ For example, the audience of banking is totally different from the audience of a video game.
- ❑ Therefore, when an organization develops a software product, it can assess whether the software product will be beneficial to its purchasers and other audience.

Software Testing

❑ **Testing objective:**

- ✓ To find any defects or bugs that may have been created when the software was being developed
- ✓ To increase confidence in the quality of the software
- ✓ To prevent defects in the final product
- ✓ To ensure the product meets customer requirements as well as the company specifications
- ✓ To provide customers with a quality product and increase their confidence in the company

Software Testing Principles

1) Software testing can help in detecting bugs: Testing any software or project can help in revealing a few or some defects that may or may not be detected by developers. However, testing of software alone cannot confirm that your developed project or software is error free. Hence, it’s essential to devise test cases and find out as many defects as possible.

Software TestingPrinciples

2) Error free or Bug-free software is a myth: Just because when a tester tested an application and didn’t detect any defects in that project, doesn’t indicate or imply that your software is ready for shipping

3) Testing must be performed in different ways and cannot be tested in a similar way for all modules. All testers have their own individuality, likewise the system under test.

Software Testing Principles

4) Normally a defect is clustered around a set of modules or functionalities. Once they are identified, testing can be focused on the defective areas, and yet continue to find defects in other modules simultaneously.

5) Testing will not be as effective and efficient if the same kinds of tests are performed over a long duration.

Benefits of Software Testing

- *Cost-Effective:* It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- *Security:* It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

Benefits of Software Testing

- *Product Quality:* It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- *Customer Satisfaction:* The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Software Testing Issues

- ❑ Testing the Complete Application
- ✓ It is not possible to test each combination in both Manual as well as Automation Testing. If you try all these combinations, you will never ship the product
- ❑ Misunderstanding of Company Processes
- ✓ There are some myths in testers that they should only go with the company processes even if these processes are not applicable for their current testing scenario. This results in incomplete and inappropriate Application Testing

Software Testing Issues

- ❑ Inadequate schedule of testing:

- ✓ Testing is a time-consuming affair. It must be so since it is done to bring out the defects or inadequacies of the system under different conditions and not to show that it works. Testing needs to go hand in hand with development.

Software Testing Issues

- ❑ Insufficient testing environment and tools:
- ✓ Tools and environments are backbones of proper software testing. However, testing is often carried out in inadequate testing environment. Moreover, some of the environmental components themselves suffer from defects.
- ✓ Team managers must ensure that actual or close enough hardware and software requirements are met in a testing environment.

Software Testing Issues

- ❑ Wrong testing mindset
- ✓ Often the mindset of the software testing team revolves around finding out functionality of the system rather than finding defects in it. This itself prohibits the team from finding out flaws in the software.
- ✓ It is the duty of team lead to instruct the idea that testing is done to find fault with the system or software under different conditions and not to prove that it works

Types of Software Testing

- Testing is an integral part of any successful software project. The type of testing depends on various factors, including project requirements, budget, timeline, expertise, and suitability. Software testing is a huge domain, but it can be broadly categorized into two areas such as

Types of Software Testing

- ❑ ***Manual Testing***
- ✓ Manual Testing is a type of Software Testing where Testers manually execute test cases without using any automation tools.
- ✓ It means the application is tested manually by QA testers.
- ✓ Tests need to be performed manually in every environment, using a different data set and the success or failure rate of every transaction should be recorded.
- ✓ This type of testing requires the tester's knowledge, experience, analytical/logical skills, creativity, and intuition.

Types of Software Testing

- ❑ Some of the tools used for Manual Testing are:Stryka

1. Bugzilla
2. Jira
3. Mantis
4. Trac
5. Redmine
6. Fogbuz
7. Lighthouse

Types of Software Testing

- ❑ **Automated Testing**

- ✓ Automation testing is an Automatic technique where the tester writes scripts by own and uses suitable software to test the software.
- ✓ It is basically an automation process of a manual process.
- ✓ Like regression testing, Automation testing also used to test the application from load, performance and stress point of view

Types of Software Testing

❑ Some of the tools used for Automated Testing are :

1. Selenium
2. TestingWhiz
3. Ranorex
4. Sahi
5. Waitir
6. WaitiN
7. Tosca TestSuite

Thank - You

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

on

"System Testing"

.

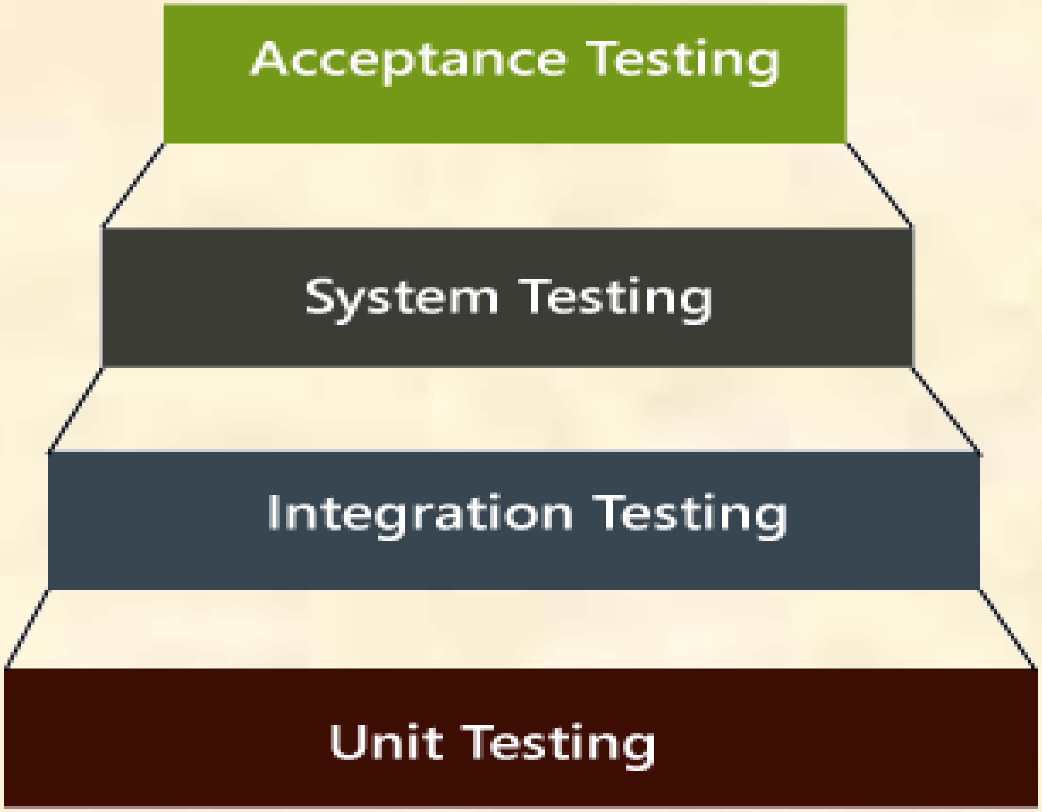
System Testing

- ❑ 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 goal of integration testing is to detect any irregularity between the units that are integrated together.
- ❑ System testing detects defects within both the integrated units and the whole system.

System Testing

- The result of system testing is the observed behavior of a component or a system when it is tested.
- System testing tests the design and behavior of the system and the expectations of the customer.
- It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).
- System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial.

System Testing



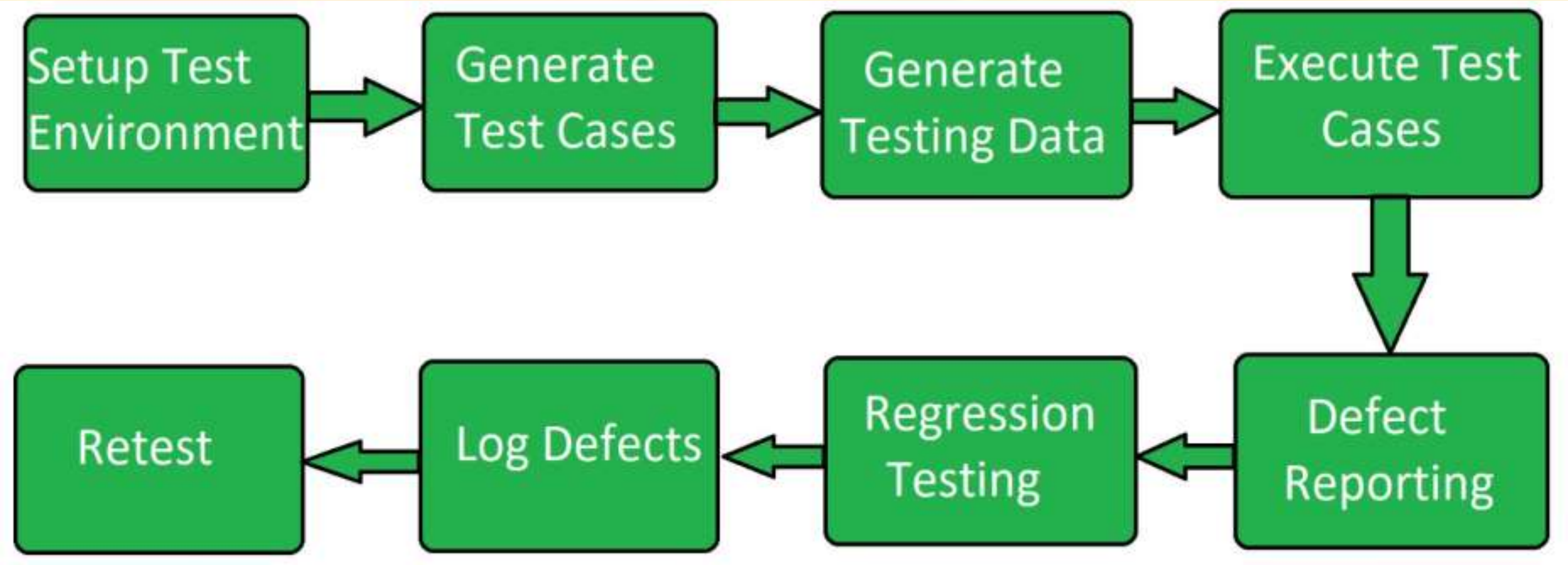
System Testing Process

- *Test Environment Setup:* Create testing environment for the better-quality testing.
- *Create Test Case:* Generate test case for the testing process.
- *Create Test Data:* Generate the data that is to be tested.
- *Execute Test Case:* After the generation of the test case and the test data, test cases are executed.

System Testing

- ❑ *Defect Reporting:* Defects in the system are detected.
- ❑ *Regression Testing:* It is carried out to test the side effects of the testing process.
- ❑ *Log Defects:* Defects are fixed in this step.
- ❑ *Retest:* If the test is not successful then again test is performed.

System Testing



Setup Test Environment

Generate Test Cases

Generate Testing Data

Execute Test Cases

Defect Reporting

Regression Testing

Log Defects

Retest

Types of System Testing

- *Performance Testing:* Performance Testing is a type of software testing that is carried out to test the speed, scalability, stability and reliability of the software product or application.
- *Load Testing:* Load Testing is a type of software Testing which is carried out to determine the behavior of a system or software product under extreme load.

System Testing

- *Stress Testing:* Stress Testing is a type of software testing performed to check the robustness of the system under the varying loads.
- *Scalability Testing:* Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

Thank - You

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

on

"Testing Conventional Applications"

.

Prepared By

Virendra Dani

Testing Conventional Applications

- ❑ **What is it?**
- ✓ Once source code has been generated, software must be tested to uncover (and correct) as many errors as possible before delivery to your customer.
- ✓ Your goal is to design a series of test cases that have a high likelihood of finding errors—but how?
- ✓ That's where software testing techniques enter the picture.

Testing Conventional Applications

- ❑ These techniques provide systematic guidance for designing tests that
- ✓ (1) exercise the internal logic and interfaces of every software component and
- ✓ (2) exercise the input and output domains of the program to uncover errors in program function, behavior, and performance

Testing Conventional Applications

- ❑ **Who does it?**
- ❑ During early stages of testing, a software engineer performs all tests.
- ❑ However, as the testing process progresses, testing specialists may become involved

Testing Conventional Applications

- ❑ **What are the steps?**
- ✓ For conventional applications, software is tested from two different perspectives:
- ✓ (1) internal program logic is exercised using "white box" test-case design techniques and
- ✓ (2) software requirements are exercised using "black box" test-case design techniques.
- ✓ Use cases assist in the design of tests to uncover errors at the software validation level.
- ✓ In every case, the intent is to find the maximum number of errors with the minimum amount of effort and time.

Testing Conventional Applications

- ❑ **What is the work product?**

- ✓ A set of test cases designed to exercise both internal logic, interfaces, component collaborations, and external requirements is designed and documented, expected results are defined, and actual results are recorded.

Testing Conventional Applications

- ❑ **How do you ensure that you've done it right?**
- ✓ When you begin testing, change your point of view.
- ✓ Try hard to "break" the software! Design test cases in a disciplined fashion and review the test cases you do create for thoroughness.
- ✓ In addition, you can evaluate test coverage and track error detection activities.

Thank - You

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

on

“Testing Object Oriented Applications”

.

Prepared By

Virendra Dani

Testing Object Oriented Applications

- ❑ **What is it?**

- ✓ The architecture of object-oriented (OO) software results in a series of layered subsystems that collaborating classes.

- ✓ Each of these system elements (subsystems and classes) performs functions that help to achieve system requirements.

- ✓ It is necessary to test an OO system at a variety of different levels in an effort to uncover errors that may occur as classes collaborate with one another and subsystems communicate across architectural layers

Testing Object Oriented Applications

- ❑ **Who does it?**
- ✓ Object-oriented testing is performed by software engineers and testing specialists.
- ❑ **Why is it important?**
- ✓ You must execute the program before it gets to the customer with the specific intent of removing all errors, so that the customer will not experience the frustration associated with a poor-quality product.
- ✓ In order to find the highest possible number of errors, tests must be conducted systematically, and test cases must be designed using disciplined techniques.

Testing Object Oriented Applications

❑ **What are the steps?**

- ✓ OO testing is strategically analogous to the testing of conventional systems, but it is intentionally different.
- ✓ Because the OO analysis and design models are similar in structure and content to the resultant OO program, “testing” is initiated with the review of these models.
- ✓ Once code has been generated, OO testing begins “in the small” with class testing.
- ✓ A series of tests are designed that exercise class operations and examine whether errors exist as one class collaborates with other classes

Testing Object Oriented Applications

- ❑ **What is the work product?**
- ✓ A set of test cases, designed to exercise classes, them
- ✓ collaborations, and behaviors is designed and documented; expected results are defined, and actual results are recorded.
- ❑ **How do you ensure that you've done it right?**
- ✓ When you begin testing, change your point of view.
- ✓ Try hard to "break" the software! Design test cases in a disciplined fashion, and review the tests cases you do create for thoroughness

Thank - You

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

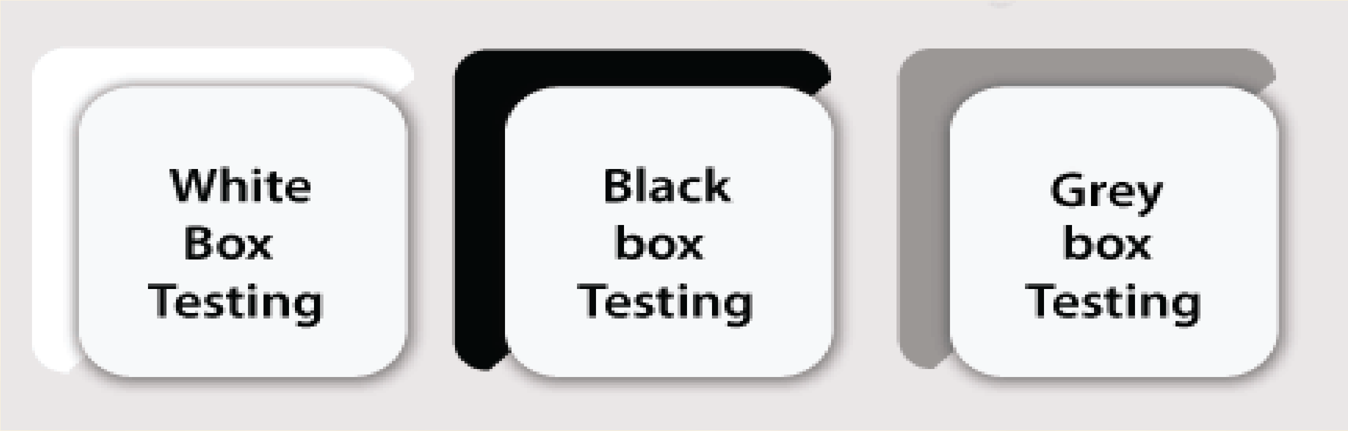
on

“Types of Testing”

.

Classification of Manual Testing

- ❑ In software testing, manual testing can be further classified into three different types of testing, which are as follows



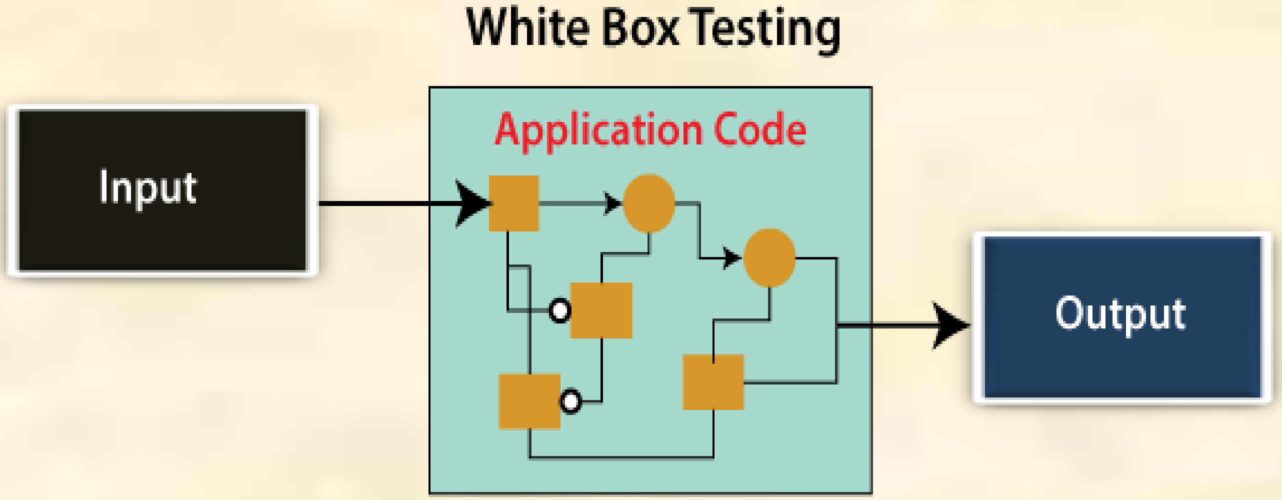
White Box Testing

- ❑ White Box Testing is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security.
- ❑ In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing

White Box Testing

- ❑ Working process of white box testing:
- ❑ *Input:* Requirements, Functional specifications, design documents, source code.
- ❑ *Processing:* Performing risk analysis for guiding through the entire process.
- ❑ *Proper test planning:* Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
- ❑ *Output:* Preparing final report of the entire testing process

White Box Testing

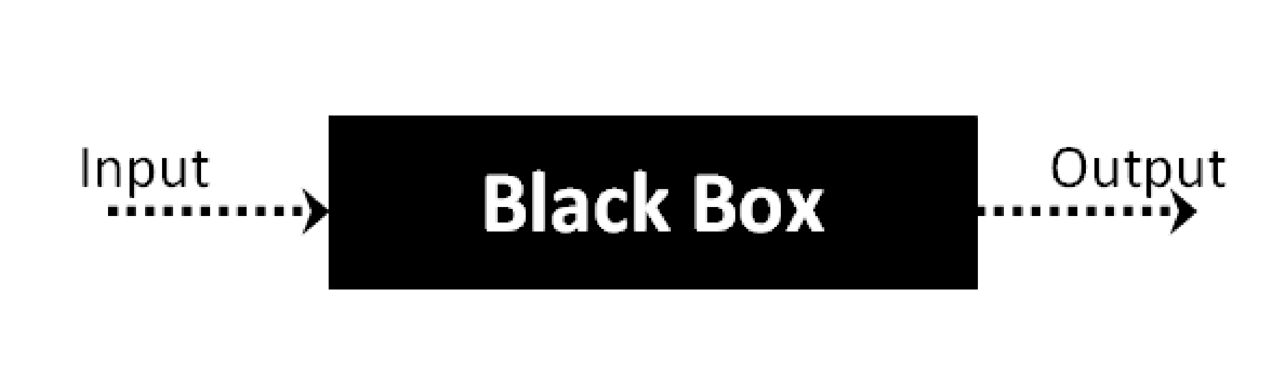


Black Box Testing

- ***Black box*** testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

- Black Box Testing mainly focuses on input and output of software applications, and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

Testing of Testing



Testing of Testing

- ❑ The above ***Black-Box*** can be any software system you want to test.
- ❑ For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application.
- ❑ Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

How to do Black Box Testing

- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT can detect them.
- Tester determines expected outputs for all those inputs.

How to do Black Box Testing

- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

- ❑ *Functional testing* – This black box testing type is related to the functional requirements of a system; it is done by software testers.
- ❑ ***Non-functional testing*** – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- ❑ ***Regression testing*** – Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Tools for Black Box Testing

- ❑ Tools used for Black box testing largely depends on the type of black box testing you are doing.
- ✓ For Functional/ Regression Tests you can use – QTP, Selenium
- ✓ For Non-Functional Tests, you can use – LoadRunner, Jmeter

Functional Testing

- Functional testing is a type of testing which verifies that each function of the software application operates in conformance with the requirement specification.
- This testing mainly involves black box testing, and it is not concerned about the source code of the application.

Functional Testing

- Every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.
- This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application Under Test.
- The testing can be done either manually or using automation

Non-Functional Testing

- *Non-functional* testing is a type of testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application.
- It is explicitly designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.
- It verifies whether the behavior of the system is as per the requirement or not.

Difference between Functional and Non-fuctional Testing

Parameters	Functional	Non-Functional
Execution	It is performed before non-functional testing.	It is performed after the functional testing.
Focus area	It is based on customer's requirements.	It focusses on customer's expectation.
Requirement	It is easy to define functional requirements.	It is difficult to define the requirements for non-functional testing.
Usage	Helps to validate the behavior of the application.	Helps to validate the performance of the application.
Objective	Carried out to validate software actions.	It is done to validate the performance of the software.

Difference between Functional and Non-fuctional Testing

Parameters	Functional	Non-Functional
Manual testing	Functional testing is easy to execute by manual testing.	It's very hard to perform non-functional testing manually.
Functionality	It describes what the product does.	It describes how the product works.
Example Test Case	Check login functionality.	The dashboard should load in 2 seconds.
Testing Types	Examples of Functional Testing Types Unit testing Smoke testing User Acceptance Integration Testing System Testing	Examples of Non-functional Testing Types Performance Testing Volume Testing Scalability Usability Testing Load Testing Portability Testing

Thank - You

Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore (M.P.)

Shri Vaishnav Institute of Information Technology

Department of Computer Science and Engineering

Lecture

on

“Validation Testing and Verification Testing

.

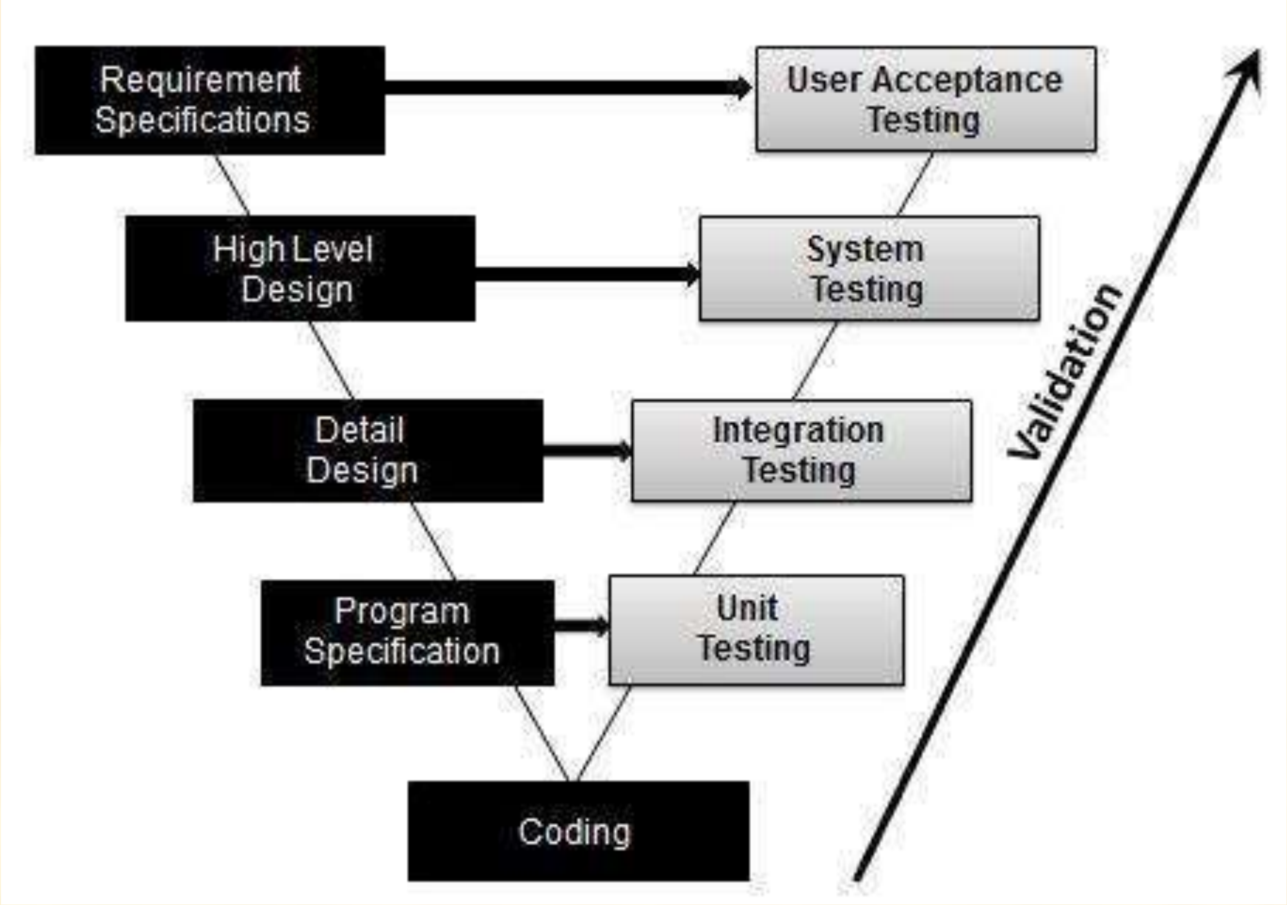
Validation Testing

- The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.
- Validation Testing ensures that the product meets the client's needs.
- It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

Validation Testing

- Validation testing is also known as dynamic testing, where we are ensuring that "we have developed the product right."
- And it also checks that the software meets the business needs of the client.
- It is the process of checking the validation of product i.e., it checks what we are developing is the right product.
- Validation testing can be best demonstrated using ***V-Model***. The Software/product under test is evaluated during this type of testing.

Validation Testing



Validation Testing

- ❑ Activities involved in validation:
- ✓ Black box testing
- ✓ White box testing
- ✓ Unit testing
- ✓ Integration testing

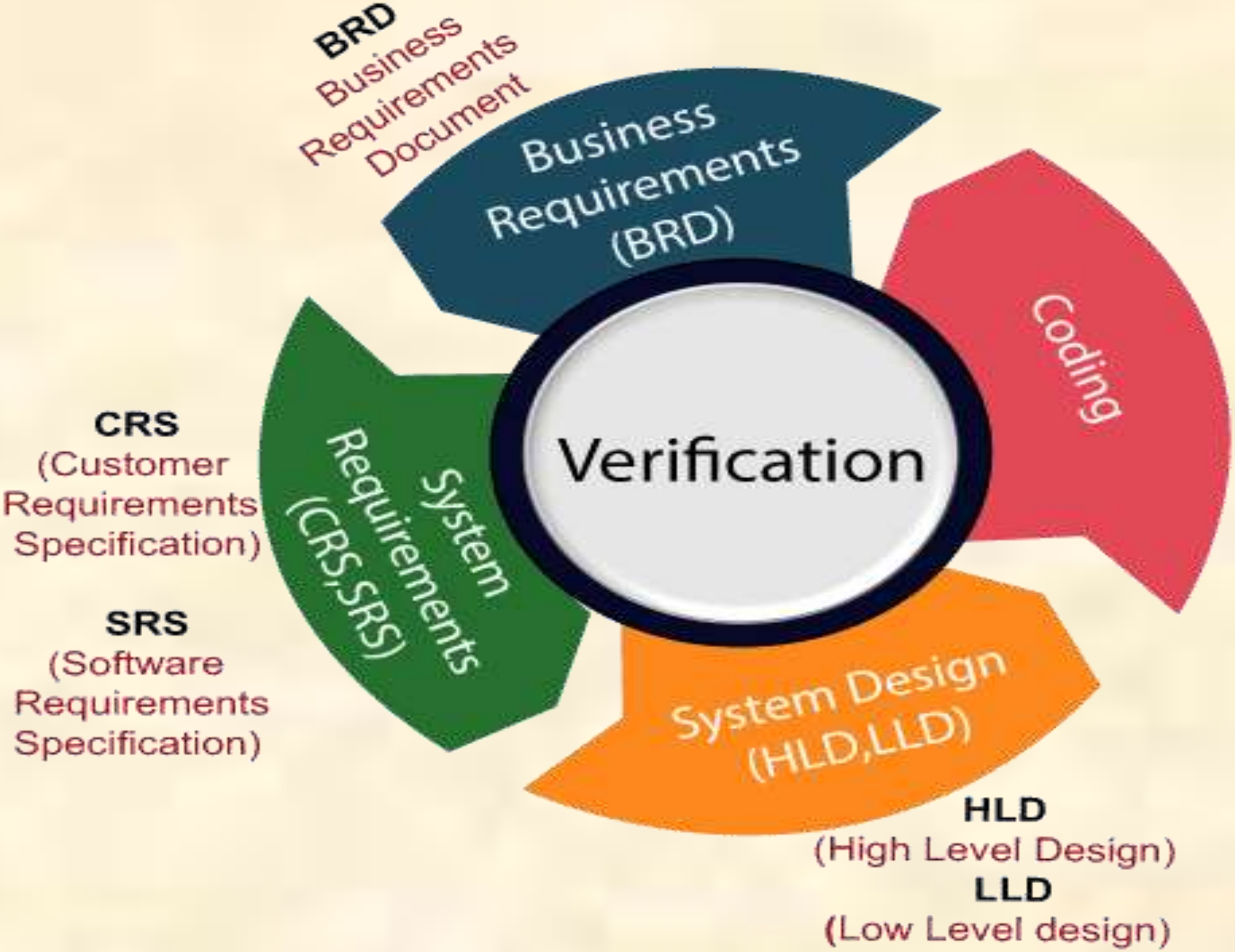
Verification Testing

- ❑ Verification is the process of evaluating work-products of a development phase to determine whether they meet the specified requirements.
- ❑ Verification ensures that the product is built according to the requirements and design specifications.
- ❑ It also answers to the question, Are we building the product, right?

Validation Testing

- Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.
- It is also known as static testing, where we are ensuring that "**we are developing the right product or not**".

Verification Testing



Difference between Verification and Validation Testing

- ❑ Verification and Validation is the process of investigating that a software system satisfies specifications and standards, and it fulfills the required purpose. **Barry Boehm** described verification and validation as the following:
- ❑ *Verification: Are we building the product, right?*
- ❑ *Validation: Are we building the right product?*

Validation Testing

S. No.	Verification	Validation
1	It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.
2	**It comes before validation.**	**It comes after verification**
3	Quality assurance team does verification	Validation is executed on software code with the help of testing team.
4	The goal of verification is application and software architecture and specification	The goal of validation is an actual product.
5	It can find the bugs in the early stage of the development	It can only find the bugs that could not be found by the verification process

Validation Testing

S. No.	Verification	Validation
6	Verification is the static testing.	Validation is the dynamic testing.
7	It does *not* include the execution of the code	Validation is the dynamic testing.
8	QA team does verification and make sure that the software is as per the requirement in the SRS document.	With the involvement of testing team validation is executed on software code.
9	Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.

Thank - You