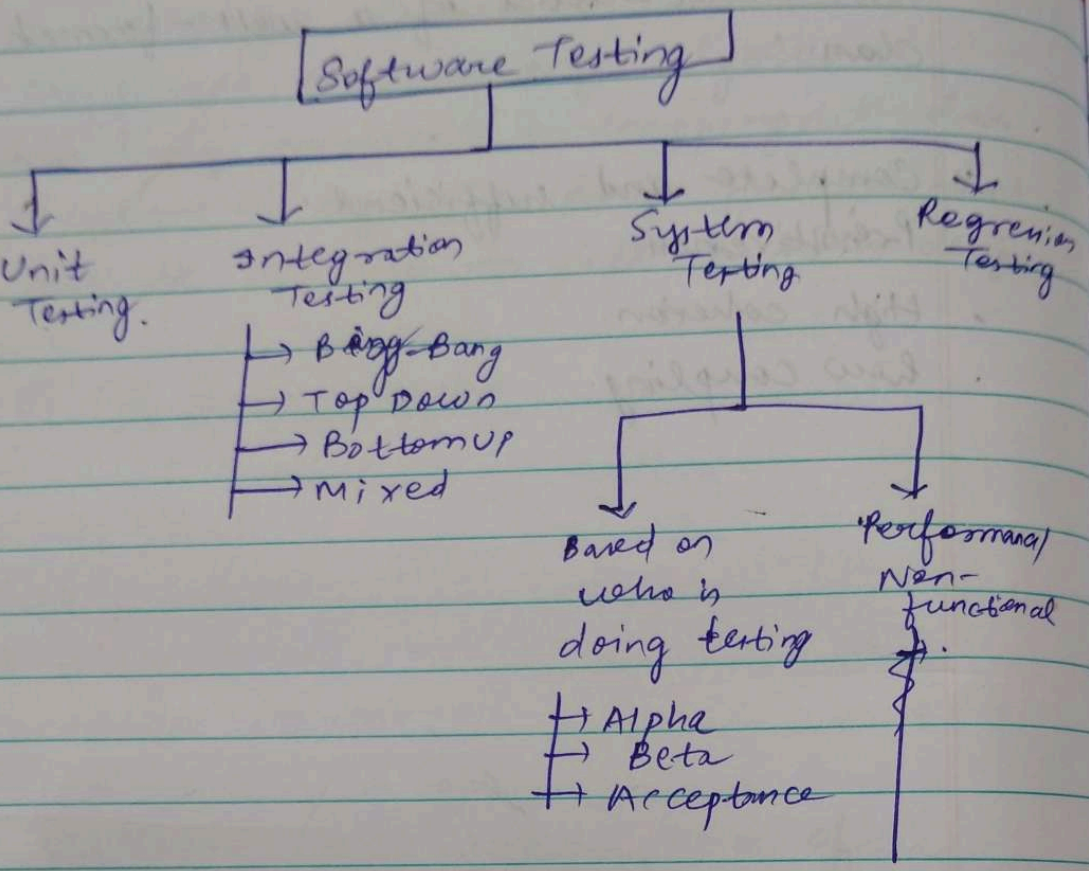
Module - 3

Software Testing

Date: __/__/____



* What is Software Testing?

- Software Testing is a process of executing a program with the intention of finding or identifying bugs or faults in the code.

→ Approach to Testing :-

- Verification and Validation (V&V)

* Verification refers to the set of tasks that ensure that software correctly implements a specific function. Validation refers to a different ~~to~~ set of tasks that ensure that the software that has be

built is traceable to customer requirements

Error seeding :-

Q) A software was tested using the error seeding strategy in which 20 errors were seeded in the code. 16 of the seeded codes were detected. The same test suite also detected 200 non-seeded errors. What is the estimated no. of undetects errors in the code?

A S = Total errors in seeded code = 20
s = Total seeded errors detected during testing = 16
N = Total errors
n = Total non-seeded errors = 200

$$\frac{n}{N} = \frac{s}{S}$$

$$N = \frac{n \times S}{s} = ?$$ → Total errors.

$$\frac{20 \times 200}{16} = 250$$

Undetected error : $\frac{n \times (S-s)}{s}$

$$= \frac{200 \times (20-16)}{16}$$

$$= 50.$$

Total - Non Seeded =

250 - 200 = 50

Page No.:

Functional

1. Unit Testing :-

- Here, individual units of software i.e. group of computer program & module, usage procedures, and operating procedures are tested to determined whether they are suitable for use or not.
- It is the first level of testing.
- It is usually performed by developers.

→ Objectives :-

- To isolate a section of code.
- To verify the correctnen of the code.
- To test every function and procedure.
- To fix bugs early in the development cycle and to save costs.

Functional

2. Integration Testing :-

- Integration Testing is carried out after all the modules have been unit tested.
- The objective of integration testing is to check whether the different modules of a program interface with each other properly.

→ Big-bang approach :-

- It is a simple approach, where all separately tested modules are integrated first and then tested as a single module.

Advantage :-

- Suitable for small project where no. of modules are less.
- No additional cost for writing test drivers and stubs.

Disadvantage :-

- May miss important data flow between modules.
- Difficult to find the root cause of any detected/Bugs.
- Generally not useful for large project.

→ Incremental Testing approach :-

- Testing is done by integrating two or more modules that are logically related and then tested for proper functioning.

(i) Top-Down Approach :-

- This approach begins with the main module and moves downwards integrating

and testing its lower-level modules.

Advantage:-
- Major control and decisions are considered early before testing the lower level modules.

Disadvantages:-
- Testing may be delayed for a module if lower-level modules are not available.

(ii) Bottom-up Approach:-

- Reverse procn of top-down approach.
- lower level modules are integrated function-wise to form a subsystem and sub-system are integrated to form main module.

→ # Sandwich Approach:-
- Combination of both top-down and bottom-up approach.
- Top-down approach forces the lower level modules to be available and bottom-up approach requires upper-level modules.

Date: __.__.____

3. System Testing :-

- System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

- System testing detects defects within both the integrated units and the whole system.

→ Types :-

- Performance testing - speed, scalability, stability
- Volume testing
- Stress testing
- Security testing
- Recovery testing
- Compatibility testing
- Configuration testing
- Installation testing
- Documentation testing

→ Acceptance Testing :-

- This testing is performed before the system is released in the market.
- System is tested in customer point of view.

→ Types :-

- User Testing
- Regulation testing
- Acceptance
- Alpha Testing - Conducted at the developer's site by a representative group of end users.
- Beta Testing - Conducted at one or more end-user sites. The developer generally is not present.

★ Principles of Testing :-

1. Testing shows the presence of defect
2. Exhaustive testing is not possible.
3. Early Testing
4. Defect Clustering
5. Pesticide Paradox
6. Testing is context - ~~development~~ dependent
7. Absence of errors fallacy.

★ White-Box Testing :- (Glass box)

- In white box testing each testing strategy enentially designs test cases based on analysis of some aspect of source code and is based on some heuristic.

Types :-

(1) Statement-Coverage Technique :-

```
1. main()
2. {
3. int n1, n2, n3;          (30, 20, 10)
4. if (n1>=n2 && n1>=n3)
5.     printf ("n1 is largest");
6. else if (n2>=n1 && n2>=n3)
7.     printf ("n2 is the largest");
8. else
9.     printf ("y.d n3 is the largut")
10. }
```

T.C-1 → ~~30~~ n1, n2, n3
30 20 10

Then, 1, 2, 3, 4, 5, 10 = 6/10 = 60%.

T.C-2 → n1, n2, n3
10 20 30

Then, 1, 2, 3, 4, 6, 8, 9, 10 = 8/10 = 80%.

Page No.:

(2) Branch- Coverage Testing :-

Q) Design branch coverage -based test suite for the following Euclid's GCD computation program:

```
int compute GCD(x,y)
int x,y;
{
while (x != y) {
if (x > y) then
x = x-y;
else y = y-x;
}
return x;
}
```

x=3, y=2

x = 1

Ans- x = 3, y = 3
x = 3, y = 2
x = 4, y = 3
x = 3, y = 4

Q)
```
int a;
if (a<0)
{
   a = a+5;
}
printf ("a = %d", a);
```

Ans- a = -2, → statement coverage test case.

a = -2, a = 4 → both are branch coverage test case.

(3) Multiple condition coverage testing :-

All the conditions must be checked.

```
read a, b, c;
if (a==0 || b==0)
{
  print 1;
}
else
{
  if (c==0 && d==0)
  {
    print 2;
  }
}
```

Tc-1 a=0, b=0, c=0, d=0 · 25%.

Tc-2 a=1 b=0 c=0 d=0 25%.

Tc f||f = T =2 ⇒ 2/4 = 50%.

TC 3. a=1, b=1, c=0, d=0 100%.

★ Data Flow Testing :-

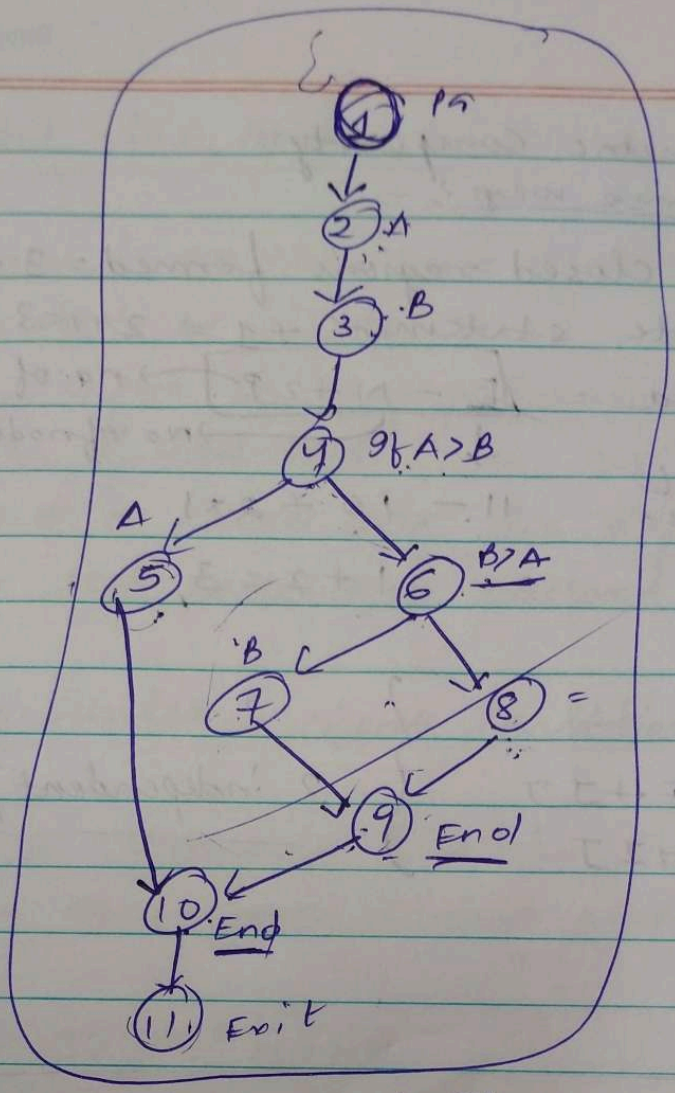
```
1. read a, b, c;
2. if (a > b)
3.     x = a + 1
4.     print x;
   else
5.     x = b - 1
6. Print z;
```

	Define	Use
a	1	2, 3
b	1	2, 5
c	1	NA
x	3	4
z	NA	6

★ Path-Coverage base testing :-

```
1. Program greatest {
2. Input (A) :-
3. Input (B) -
4. If (A > B)
5.     Then Print (A)
6. Else if (B > A)
7.     Then Print B
8. Else
       Print Both (=)
9. End if
10. End if
11. Exit
```

apsara
Date:



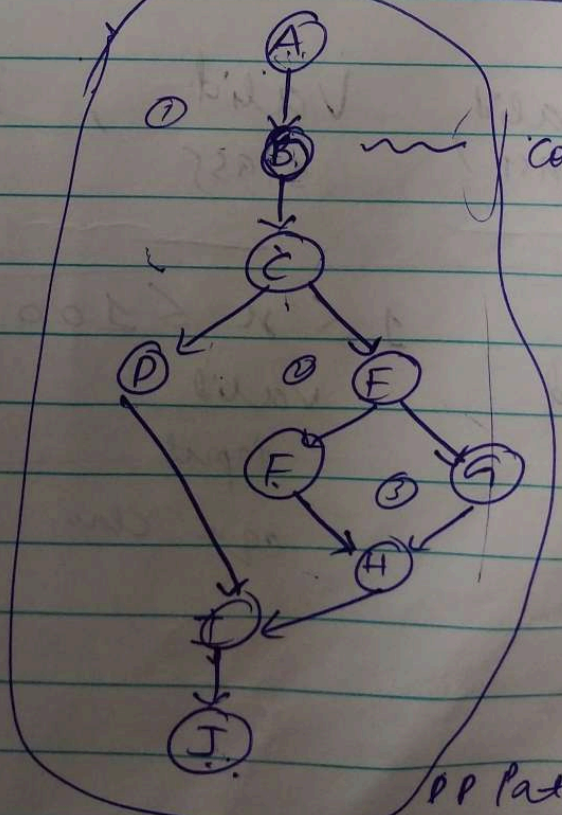
Control Flow graph.

Cyclomatic Complexity

↓ ↑ If/else

No. of predicate statements + 1

TL = 3



DD Path Graph.

→ Cyclometric Complexity.

Three ways :-

∴ No. of closed regions formed = 3.

• Predicate statements + 1 = 2 + 1 = 3

• Formula : E − N + 2P → no. of ways program can exit
(E → Edges, N → no. of nodes)

Edges. 11 − 10 + 2×1

= 1 + 2 = 3,

→ ABCDIJ
→ ABCEFHIJ
→ ABCEGHIJ

} 3 independent paths.

Invalid class | Valid class | Invalid class

$x < 1$ invalid , $1 \leq x \leq 100$ valid , $x > 100$ invalid

input

eg: class pwd.

★ Black - box Approach :-

- performed on the basis of software functions or features.
- Only input values are considered for the design of test cases.
- Output values that the software provides for test cases are observed.

(1) Equivalence class Partitioning :-

Q) If a variable lies b/w 1 and 100 then equivalence classes will be :

all neg nos, 0 | $1 \leq x \leq 100$ | all +ves > 100

invalid | valid | invalid

Q) online application form : age criteria of 16-60 . If you are b/n 16 and 60, you can fill otherwise you can't get a membership

$x < 16$, $16 \leq x \leq 60$, $x > 60$

invalid | valid | invalid

age 5 . X

18 ✓

30 ✓

65 X

Q) Design equivalence class partitioning test suite for a function that reads a character string of size less than five characters and displays whether it is a palindrome?

Ans- Valid Input :-

→ Palindrom, string len than 5 :- a, aa, aba

→ Non-palindrome, string len than 5 :- abc, abcd

Invalid Input :-

→ string size more than 5, abcde

→ Empty string

→ special character : a@!, 12a

eg :-

Test cases	Input	Expected output palindrome	Equivalence class valid/invalid
TC1	aba	Palindrom	Valid

(v - not)

Date: 40, 41, 79, 80

* Boundary Value Analysis :-

Q) For a function that computes the square root of the integer values in the range of 0 and 5000, determine the boundary value test suite.

Ans- There are three equivalence classes :-

$x < 0$; $0 \leq x \leq 5000$, $x > 5000$

invalid. valid invalid

~~Possible test suit : {x = 5, 500, 6000}~~

Boundary value test suit : { 0, -1, 5000, 5001 }

Q) Car speed min 40, max 80.

Boundary Value Test Case

Invalid (Min - 1)	valid T.C. (Min, +Min, Max, -Max)	Invalid TC. Max + 1
39	40, 41, 80, 79	81

Boundary value suite {39, 40, 80, 81}

★ Regression Testing :-

- Regression Testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.
- Each time a new module is added as part of integration testing, the software changes.
- New data & flow paths are established, new I/O may occur, and new control logic is invoked.

→ How to conduct Regression Testing?

- manually, by re-executing a subset of all test cases.

→ Advantages :-

- Helps to maintain the quality of the source code.
- It ensures that no new bugs have been introduced after adding new functionality to the system.
- ~~Test cases used in~~

Date: __.__.____

→ Disadvantage :-

- It can be time and resource conuming if automated tools are not used.
- It is required even after a very small changes in the code.

Types
- Full Regression Testing
- Partial " "

Module-4

Function-Point

* **Function-Point Analysis :-**
  - It is a standardized method used to estimate the size, complexity and functionality of software projects.

* **Unadjusted Function Points (UFP) :-**
  - Calculate UFP by multiplying the number of each type of component by its corresponding weight and summing them up.

* **Value Adjustment Factor (VAF) :-**
  - A factor th

Q) No. of ~~external~~ internal inputs (I) = 30
No. of external inputs (O) = 60
No. of external inquiries (E) = 23
No. of files (F) = 08
No. of external interfaces (N) = 2

I, O, E, F, N are 4, 5, 4, 10 and 7.
(Weights)

Out of 14, 4 are not available, each of the other factor has value 3, and each of the remaining factor has value 4. ~~The~~ co FP?

Ans- $$FP = UAF \times VAF$$

$$UAF = I \times (W) + O \times (W) + E \times W + F \times (W) + N \times (W)$$

$$= 30 \times 4 + 60 \times 5 + 23 \times 4 + 8 \times 10 + 2 \times 7 = 606$$

$$VAF = 0.65 + \left(\frac{\text{Sum of all factors}}{100} \right)$$

Sum of all factors = $4 \times 3 + 6 \times 4$
$= 12 + 24$
$= 36$

$$VAF = 0.65 + \frac{36}{100} = 1.01$$

$$FP = 606 \times 1.01$$

$$\boxed{FP = 612.06}$$