Software Metrices:

→ Measure (km, cm, kg etc)

→ Measurement (process of measuring km etc)

→ Metrics (two or more different measures) combination.

1000 defects per kilolines count

Software Quality Measurement

1- Availability:

$$\alpha = \frac{\text{Mean Time to failure}}{\text{Mean Time to failure} + \text{Mean time to repair}}$$

$$\alpha = \frac{10}{10+2} \Rightarrow 0.833 = 83.3\%$$

→ α greater, repair time less

→ α smaller, repair time greater

2- Maintainability:

→ degree of easiness with which a program is corrected if an error occured.

$$\text{Maintainability} = \frac{1}{\text{Mean Time To change}}$$

M T W T F S

→ Lower MTTC means higher maintainability

→ MTTC is mean time to fulfill a change request.

3- Correctness:

→ degree of software to which it performs its required functions.

→ Can be measured by defect density.

$$\text{Defect density} = \frac{\text{NO. of defects}}{\text{Size of program in KLOC}}$$

Code length = 5000 LOC

Defects = 5

$$DD = \frac{5}{5 \text{ KLOC}} = 1 \text{ defect/KLOC}$$

$$\text{Correctness} \propto \frac{1}{\text{Defect density}}$$

4- Usability:

→ Cannot be measured

M T W T F S

Date:

→ Concerned with how easy it is for the user to complete a desired task and user support the system provides.

It can be broken down into:

* Learning System Features
* Using a system efficiently
* Minimizing the impact of errors.
* Adapting the system to user needs
* Increasing confidence & satisfaction.

5- Integrity:

→ ability of the system to withstand attacks on its security.

integrity = Sum [1 - threat * (1 - security)]

$$\text{integrity} = \frac{\text{No. of Successfull attacks}}{\text{Total no. of attempts}} \times 100$$

6- Performance:

→ time to respond when an event occurs.

→ Response can be characterized by: (Latency, Throughput, Jitter)

Latency:

→ delay b/w a user's action and a web application's response to that action.

Latency = response time - event occurance t

Throughput:

→ the no. of transactions the system can process in a second.

10/1 min ⇒ 1/6 sec

Jitter:

→ difference b/w two latency rates.

|15 - 18| = (3) → jitter

System Quality $\propto \frac{1}{\text{jitter}}$

Calculated Metrics:

i- Process Metrics

ii- Product Metrics

M T W T F S

Date:

Process Metrics:

→ Test Tracking Metrics:

Total = 240

Pass = 181

Fail = 28

Blocked = 31

Formula:

$$\frac{\text{No. of Passed (or failed, or blocked) tests}}{\text{No. of tests in total}} \times 100$$

Passed = (181/240) x 100 = 75.42%

Failed = (28/240) x 100 = 11.67%

Blocked = (31/240) x 100 = 12.92%

→ Test Case Preparation Productivity:

$$TCPP = \frac{\text{No. of test cases}}{\text{Time spen for test preparation}}$$

No. of TC = 240

Effort Spent = 40 hours

$$TCPP = \frac{240}{40} \Rightarrow 6 \text{ Tc/hour}$$

M T W T F S

→ Test Design Coverage:

* Kaam kitna cover kia h.

TDC = $\frac{\text{NO. of requirement mapped to test cases}}{\text{NO. of requirement}} \times 100\%$

TC mapped to req. = 92

NO. of req. = 136

TDC = $\frac{92}{136} \times 100\% \Rightarrow 68\%$

→ Test Execution Coverage:

TEC = $\frac{\text{NO. of test cases executed}}{\text{Test cases planned for execution}} \times 100\%$

TC executed = 185

TC Planned to execute = 250

TEC = $\frac{185}{250} \times 100\% \Rightarrow 74\%$

M T W T F S

Date:

→ Test execution/design productivity:

$$TDP = \frac{\text{No. of test cases executed}}{\text{Time Spent for execution}}$$

No. of TC executed = 240

Time Spent = 10 hours.

$$TE|DP = \frac{240}{10} = 24 \text{ Test cases/hour}$$

→ Test Effectiveness:

$$TE = \frac{\text{No. of defects found in test}}{\text{No. of defects found in test} + \text{No. of defects found after shipping}} \times 100$$

No. of defects found in test = 145

" " " after shipping = 11

$$TE = \frac{145}{(145+11)} \times 100\% \Rightarrow 93\%$$

$$TE \propto \frac{1}{\text{No. of defects found after shipping}}$$

M T W T F S SQE Date: 22-03-21

Cyclomatic Complexity

→ White Box Testing

→ Path Testing

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

where E = edges and N = nodes

```
①                                    ②
while (value[i] != -999 && value[i] <= Max)
{
    i++;  ③
                                          ⑤
    if (value[i] >= min && value[i] <= max)   ④
    {
        tovalid++;
        Sum = Sum + valid[i];        ⑥
    }
    i++;  ⑦
}
```

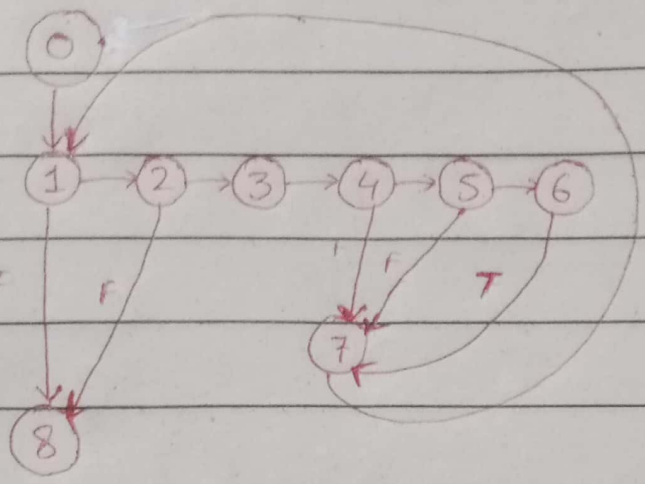
⑧

N = 9

E = 12

V(G) = 12 - 9 + 2

= 5 unique paths



P_1 = 0 - 1 - 8

P_2 = 0 - 1 - 2 - 8

P_3 = 0 - 1 - 2 - 3 - 4 - 7 - 1

P_4 = 0 - 1 - 2 - 3 - 4 - 5 - 7

P_5 = 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7

* Pass from every path at least once.

Path	Input	Expected	Actual
P_1	value = -999		
P_2	value 1000, i = 101		
P_3			
P_4	—		
P_5	—		

M T W T F S

Loop Testing

1- Loop
2- Nested
3- Contatenation

Loop:

```
for(int i=input ; i<5 ; i++) {
    cout << "*";
    }
```

i- Skip loop (input = 5)
ii- Loop executes once (input = 4)
iii- Loop executes maximum times (input=
iv- Loop: n+1 ⇒ 5+1 ⇒ (input = 6)
v- n-1 ⇒ 5-1 ⇒ (input = 4)

Nested:

```
for (i=input ; i ≤ n ; i++)
  {
    for.(j=input ; j≤n ; j++)
    {
    }
  }
```

outerloop Innerloop

Concatenation:

```
for (i=input1; i<=n; i++)
{
    cout << "*";
}
for (j=input2; j<=n; j++)
{
    Cout << "KOKAB";
}
```

→ Both loops are independent so test both loops separately.

m=3

for (i=input1; i<=n; i++) (5)	for (j=input2; j<=m; j++) (4, 3)
{	{
m++;	cout<<"*";
}	}

→ As both are dependent loops on m so test both loops at a time

For 0 times ⇒ i = 6 j = 4

For 1 times ⇒ i = 5 j = 4

i 1 time j max times ⇒ i = 5 j = 0

Date:

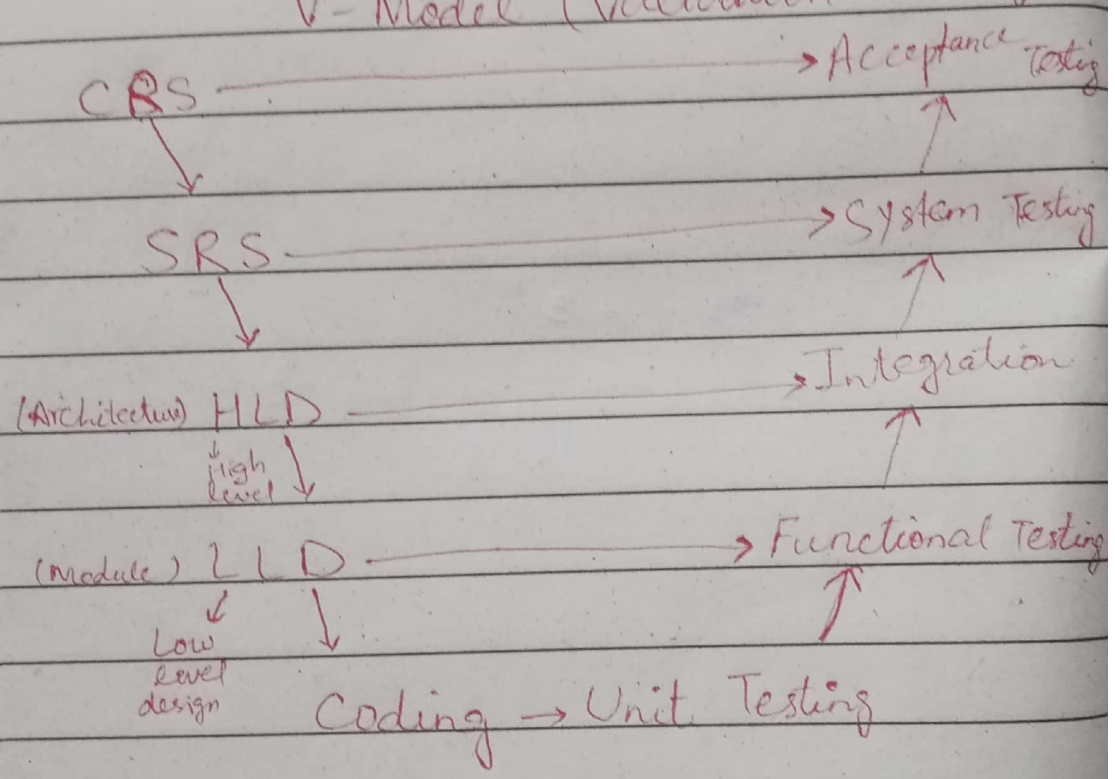
M T W T F S

i	j
n+1	Skip
n+1	1
n+1	n-1
n+1	n+1
n+1	max

02-4=24

i	j
6	4
6	3
6	02
6	0
6	0

V-Model (Validation & Varification



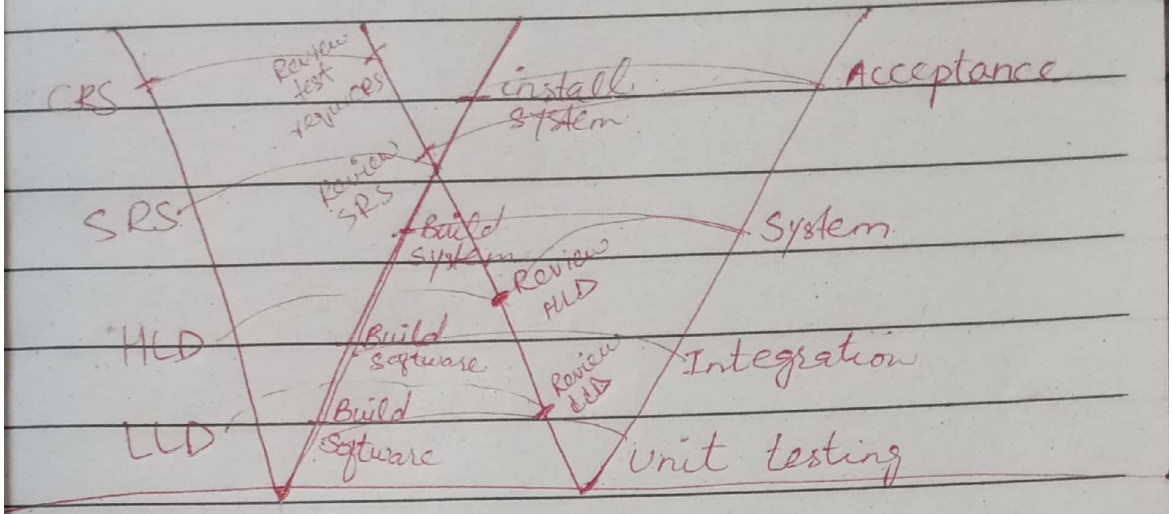
V-Model Software Testing

CRS ⇒ Customer Requirement Specification
non-technical document

M T W T F S

Date:

W-Model Software Testing



Acceptance testing client krta he Baqi Software house mein hoti.

White Black

SRS ⇒ Software Requirement Specification
→ Technical document

HLD ⇒

LLD ⇒ GUI ⇒ choty choty module.
→ jb sara code likh lo us k baad unit testing hoti.

* Choty module ki nae hoti.
* Complete code ki hoti he.