SQE-15 (19/03/24)

Software Metrics:

Measure — cm km (unit)

Measurement — counting.

Metrics — combining 2 d/f measures.

2 km per hour.

100 defects per KLOC.

Availability: (non-func)

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

$$= \frac{10}{10+2} \times 100\% = 83.3\%$$

$$\alpha \propto \text{MTTF} \quad | \quad \alpha \propto \frac{1}{\text{MTTR}}$$

Maintainability.

$$\text{Maintainability} \propto \frac{1}{\text{MTTC}}$$

Correctness:

$$DD = \frac{\text{No of Defects}}{\text{L of code in KLOC}}$$

Usability: - by feedback.

Integrity:

$$\text{Integrity} = \text{Sum}(1 - \text{threat} * (1 - \text{security}))$$

$$\text{Integrity} = \frac{\text{No of successful attack}}{\text{No of total attacks}} \times 100$$

Performance:

latency = actual time - throughput

DAY: ______ DATE: ______

throughput :

Jitter = |delay1 - delay2|.

Calculated Metrics:

- Process Metrics
  ↳ for testing process
- Product Metrics.
  ↳

blocked ⇒ unseen

SQE-16 (22/03/24)

Cyclomatic Complexity

↳ Path Testing ↳ Whitebox.

$$V(G) = \underset{\text{edges}}{E} - \underset{\text{nodes}}{N} + 2$$

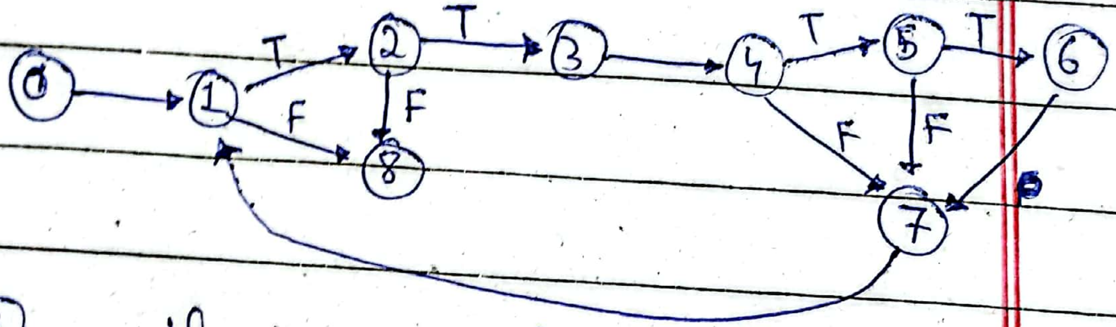
```
(0)
while (value[i] != -999  &&  i < 100)
             (1)               (2)
{
    i++;  (3)
    if (value[i] >= min  &&  value[i] <= max)
              (4)                 (5)
    {
        to Valid++;
        sum = sum + valid[i];  ] (6)
    }
    i++;  (7)
}
(8)
```

No of paths ⇒ by cyclomatic complexity.

blocks ⇒ that may run or not.

mostly conditions are numbered.

⇒ convert code ~~for~~ to graph.



⇒ ③ could be skipped

N = 9 E = 12.

No of paths = 12 - 9 + 2 = 5

0-1-8 input ~~expected~~ | Actual Out | Exp.

value = -999

0-1-2-8 value=1000, i=101

0-1-2-3-4-7-1 value=1000, i=50

0-1-2-3-4-~~3~~5-7 (1-8 explored)

0-1-2-3-4-5-6-7 (7-1 explored)

SQE-17 (26/03/24)

~~SLE → Single Loss Expectancy.~~

~~SLE = Asset Value +~~

(26/02/2

SQE - 17

Loop Testing

Loop   Nested   Concatenated.

```
for (int i = 0 ; i ≤ 5 ; i++){
        print("*");
}
```

↳ 1) Skip input ~~5~~ . i = 5, 6, 10...

2) Loop 1 execute input ~~4~~ i = 5 (only one)

3) max time execute input ~~0~~ i = 0 .. (

4) n+1   i = 6

5) n-1   i = 4.

Nested Loop ::

for (i = input ; i ≤ $\overset{5}{n}$; i++){

        for (j = input ; j ≤ $\overset{5}{m}$; j++){

        }

}

5 x 5 = 25 combinations.

outer loop	inner loop
i = 5	j = 5
= 5	j = 10
= 5	j = 0
= 5	j = 6
= 5	j = 4
= 0	
↓	↓

input2 = m + n - input1 - iter

DAY:________ DATE:________

Concatenated loop:

m = 3;

```
for(i = input1 ; i ≤ n ; i++) {
        cout << "*");
}                      m++; ⇒ dependent loops
for(j = input2 ; j ≤ m ; j++) {
        cout << "*") ;
}
```

If iteration of loop2, does not depend on loop 1's iteration ⇒ independent loops, test separately.

If dependent:

try combinations (by rules)

i	j
n+1: i = 8	skip: j = 8 4
n+1: i = 6	1 : j = 3
n+1: i = 6	m-1 : j = 2
n+1: i = 6	max : j = 0
n+1: i = 6	m+1 : j = 4

SQE-18 (28/03/24)

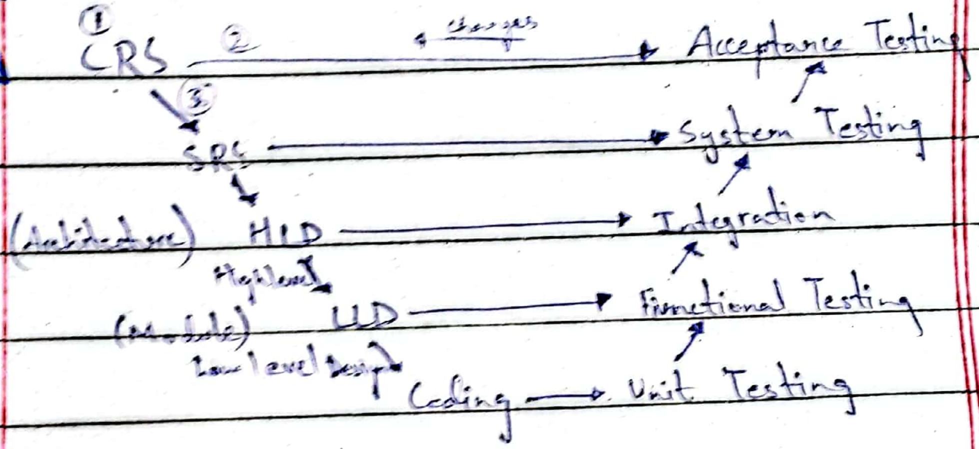
Load Testing (non-func)

static IP → need to be purchased.

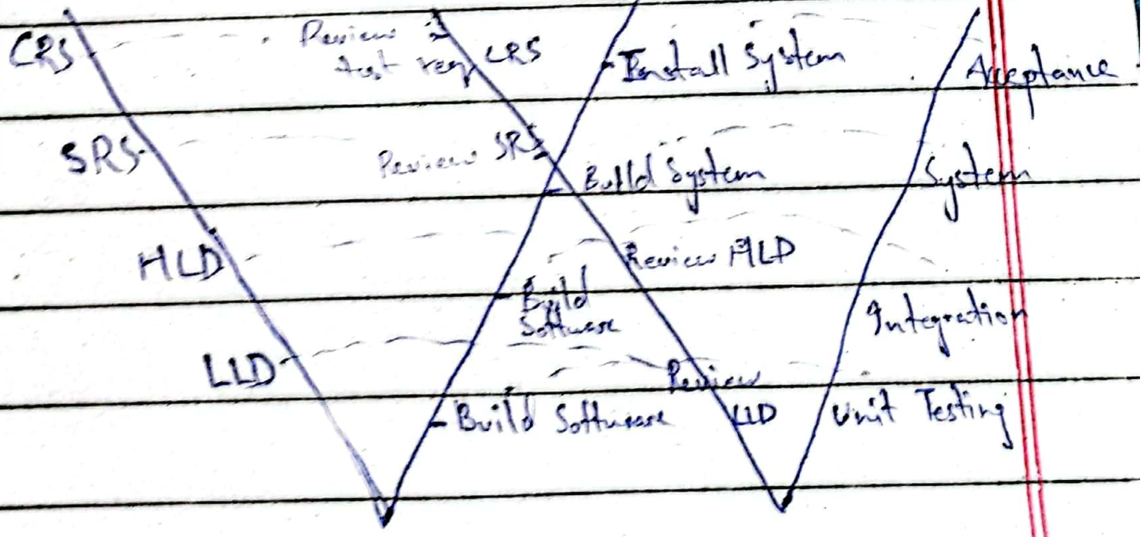
↳ J-meter (tool).

If server down by high traffic → 2nd server + 3rd server (for routing).

V Model Software Testing SQE-19 (02/04/24)



W-Model Software Testing



CRS → Customer Req Specification (non-technical)

Acceptance Testing → (by client only).

All other testing → at software house (white + black box)

SRS → Technical doc (by software house)

↳ all possible req (security)

HLD → Architecture PC, server, ...

LLD → login, signup + (GUI + modules)

↳ Math formulas

⇒ V-model → extension of waterfall model.

↳ don't go to next step before completing first.

↳ no collaboration with testers in waterfall.

→ In V-Model, testing after coding,
till coding, just Review + test plan + test case write.

→ In W-Model, testing in parallel.

SQE-20 (19/04/24)

Big Bang ⇒ Bug tracking is difficult.