MODULE -III

SOFTWARE TESTING

Testing is the process of execution a program with the intent of finding errors

TESTING CAN BE CATEGORIES INTO 2 DIFFERENT CATEGORIES

1. FUNCTIONAL TESTING OR BLACK BOX TESTING
2. STRUCTURAL TESTING OR WHITE BOX TESTING

- In Functional testing we test the functionality of a software without considering internal code.
- If we are getting desired output or set of all input condition we will assume functionality of software is correct.

(2) In Structural testing the focus is on internal code & they to find out different types of possible error in internal code.

FUNCTONAL OR BLACK BOX TESTING

- Method of functional testing

(1) BOUNDARY VALUE ANALYSIS

1. Consider a program for checking whether the given no. is even or odd
Suppose the range varii from 1 – 1000
Write the test case Acc to Boundary value Analysis to test this program

n	Exp out		
1	odd	Fail	$1 \le n \le 1000$
2	even	Pass	Yes
500	even	Pass	
999	odd	Pass	
1000	even	Pass	

Total no of Test Cases in Boundary Value formula ⇒ 4n+1

Problem — All the test case not get Range beyond unfit we do not give

Date / Page

2 Consider a program for determining the Previous date. It unfit use term of day, month & year with value in range

$1 \leq month \leq 12$

$1 \leq day \leq 31$

$1900 \leq year \leq 2025$

The Possible output would be Previous date or invalid unfit date Design boundary value test.

Test Case	Month	Day	Year	Expected Output
1	6	15	1900	14 June, 1900
2	6	15	1901	14 June, 1901
3	6	15	1962	14 June, 1962
4	6	15	2024	14 June, 2024
5	6	15	2025	14 June, 2025
6	6	1	1962	31 May, 1962
7	6	2	1962	1 June, 1962
8	6	30	1962	29 June, 1962
9	6	31	1962	Invalid date
10	1	15	1962	14 Jan, 1962
11	2	15	1962	14 Feb, 1962
12	11	15	1962	14 Nov, 1962
13	12	15	1962	14 Dec, 1962

(3) Consider a simple program to classify a triangle. Its input is a triple of positive integer (say x, y, z) and the data type for input parameter ensures that these will be integer greater than 0 and less than or equal to 100 The program output may be one of following word

Scalene : Isosceles : Equilateral : Not a triangle

Design boundary value test case

Test Case	x	y	z	Expected o/p
1	50	50	1	Isosceles
2	50	50	2	"
3	50	50	50	Equilateral
4	50	50	99	Isosceles
5	50	50	100	Not a △
6	50	1	50	Isosceles
7	50	2	50	"
8	50	99	50	"
9	50	100	50	Not a △
10	1	50	50	Isosceles
11	2	50	50	"
12	99	50	50	"
13	100	50	50	Not a △

(3) Consider a program for determining the nature of roots of a Quadratic Equation. Its inputs is triple of positive integer (say a, b, c) & the values may be from interval [0, 100]. The program output may have one of following words

[Not a Quadratic Equation; Real roots; Imaginary roots; Equal roots]

Design a Boundary value Test Case.

$$ax^2+bx+c=0$$

- Roots are real $(b^2-4ac)>0$
- Roots are imaginary if $(b^2-4ac)<0$
- Roots are equal if $(b^2-4ac)=0$
- Roots Eequation is not Quadratic if $a=0$

Test Case	a	b	c	Expected O/p
1	0	50	50	Not Quadratic
2	1	50	50	Real roots
3	50	50	50	Imaginary roots
4	99	50	50	"
5	100	50	50	"
6	50	0	50	"
7	50	1	50	"
8	50	99	50	"
9	50	100	50	Equal roots
10	50	50	0	Real roots
11	50	50	1	Real roots
12	50	50	99	Imaginary roots
13	50	50	100	Imaginary roots

ROBUSTNESS TESTING

In this testing method we would like to see what happens when extreme value are exceeded with a value slightly greater than the maximum and a value slightly less than minimum

Total test case = 6n+1

① Consider a problem for Quadratic roots eq & Wrute the test cases using Robustness testing.

6n+1 = 3 variable 19-test case.

Test Case	a	b	c	Expected O/p
1	-1	50	50	Invalid input
2	0	50	50	Not Quadratic
3	1	50	50	Real roots
4	50	50	50	Imaginary
5	99	50	50	"

Test
1
2
3
4
5
6
7
8
9
10
11
12

6	100	50	50	Imaginary
7	101	50	50	Invalid
8	50	-1	50	Invalid
9	50	0	50	Imagnary
10	50	1	50	"
11	50	99	50	"
12	50	100	50	Equal roots
13	50	101	50	Invalid
14	50	50	-1	Invalid
15	50	50	0	Real roots
16	50	50	1	Real roots
17	50	50	99	Imagnary
18	50	50	100	"
19	50	50	101	Invalid input

Consider the problem for finding the valid date. if date is valid then print the previous date. write test cases using Robustness testing.

Test Case	Month	Day	Year	Expected Output
1	6	15	1899	Invalid date (out of range)
2	6	15	1900	14 June, 1900
3	6	15	1901	14 June, 1901
4	6	15	1962	14 June, 1962
5	6	15	2024	14 June 2024
6	6	15	2025	14 June, 2025
7	6	15	2026	Invalid date (out of range)
8	6	0	1962	Invalid date
9	6	1	1962	31 May, 1962
10	6	2	1962	1 June, 1962
11	6	30	1962	29 June, 1962
12	6	31	1962	Invalid date

13	8	32	1962	Invalid date
14	0	15	1962	Invalid date
15	1	15	1962	14 Jan, 1962
16	2	15	1962	14 Feb 1962
17	11	15	1962	14 Nov, 1962
18	12	15	1962	14 Dec 1962
19	13	15	1962	Invalid date

Worst Case Testing:

In this testing method total no of test case will be 5^n where n is no of variable

Next test case will be 5^n

Consider a program which excepts 2 inputs x & y range for x is 1-100 and range for y 1-500. Write test Cases for worst case testing

$1 \le n \le 100$

$1 \le y \le 500$ $\quad 5^2 = 25$

Test Case	x	y
1	1	
2	2	
3	50	
4	99	
5	100	
6		
7		
8		
9		

Equivalence Class Testing - In this method input domain of a program is partitioned into finite no of equivalence classes such that one can reasonably assume, but not be absolutely sure, that the test of a representative value of each class is equivalent to a test of any other value

In this testing method 2 equivalence classes are formed

- Output Domain Equivalence class
- Input "

Consider the program for determining the previous date in calendar unvalid date. Write the test case for equivalence class method

Output Domain Equivalence Class — $O_1 = \{ < D, M, Y > :$ Previous date of all valid $\}$
$O_2 = \{ < D, M, Y > :$ Invalid date if any input mate the date valid $\}$

Test Case	M	D	Y	Expected Output
1	6	15	1962	14 June 1962
2	6	31	1962	Invalid date

Input Domain Class

I_1 = {month : $1 \le m \le 12$}
I_2 = {month : $m < 1$}
I_3 = {month : $m > 12$}
I_4 = {day : $1 \le D \le 31$}
I_5 = {day : $D < 1$}
I_6 = {day : $D > 31$}
I_7 = {year : $1900 \le Y \le 2025$}
I_8 = {year : $Y < 1900$}
I_9 = {year : $Y > 2025$}

Test Case	M	D	Y	Expected output
1	6	15	1962	14 June, 1962
2	-1	15	1962	Invalid input
3.	13	15	1962	"
4.	6	15	1962	14 June, 1962
5.	6	-1	1962	invalid input
6	6	32	1962	"
7	6	15	1962	14 June, 1962
8	6	15	1899	invalid input (value out of range)
9	6	15	2026	"

(2) Consider the problem for consider Quadratic roots and generate test cases using Equivalen class testing

Output Domain class

O_1 = {< a, b, c > : Not Quadratic eq if $a = 0$}
O_2 = {< a, b, c > : Real roots if $(b^2 - 4ac) > 0$}
O_3 = {< a, b, c > : Imaginary roots if $(b^2 - 4ac) < 0$}
O_4 = {< a, b, c > : Equal roots if $(b^2 - 4ac) = 0$}

Test Case	a	b	c	Expected Output
1	0	50	50	Not a Quadratic
2	1	50	50	Real roots
3	50	50	50	Imaginary
4	50	100	50	Equal.

Input Domain

$I_1 = \{a : a = 0\}$

$I_2 = \{a : a < 0\}$

$I_3 = \{a : 1 \leq a \leq 100\}$

$I_4 = \{a : a > 100\}$

$I_5 = \{b : 0 \leq b \leq 100\}$

$I_6 = \{b : b < 0\}$

$I_7 = \{b : b > 100\}$

$I_8 = \{c : 0 \leq c \leq 100\}$

$I_9 = \{c : c < 0\}$

$I_{10} = \{c : c > 100\}$

Test Case	a	b	c	Expected Output
1	0	50	50	Not Quadratic
2	-1	50	50	Invalid input
3	50	50	50	Imaginary roots
4	101	50	50	Invalid input
5	50	50	50	Imaginary roots
6	50	-1	50	invalid input
4	50	101	50	;;
8	50	50	50	Imaginary roots
9	50	50	-1	invalid input
10	50	50	101	;;

Date
Page

③ Consider the problem for finding triangle using Equivalence class method

Output Domain range: $O_1 = \{ \langle x, y, z \rangle$: Isosceles with sides $x, y, z \}$
$O_2 = \{ \langle x, y, z \rangle$: Not a Δ " $\}$
$O_3 = \{ \langle x, y, z \rangle$: Equilateral triangle " $\}$
$O_4 = \{ \langle x, y, z \rangle$: Scalene triangle " $\}$

Test Cases :	x	y	z	Expected output
1	50	50	50	Equilateral
2	50	50	99	Isosceles
3	100	99	50	Scalene
4	50	100	50	Not a Δ

Input domain class:

$I_1 = \{ x : x < 1 \}$
$I_2 = \{ x : x > 100 \}$
$I_3 = \{ x : 1 \leq x \leq 100 \}$
$I_4 = \{ y : y < 1 \}$
$I_5 = \{ y : y > 100 \}$
$I_6 = \{ y : 1 \leq y \leq 100 \}$
$I_7 = \{ z : z < 1 \}$
$I_8 = \{ z : z > 100 \}$
$I_9 = \{ z : 1 \leq z \leq 100 \}$

Input domain test case.

$I_{10} = \{ \langle x, y, z \rangle : x = y = z \}$
$I_{11} = \{ \langle x, y, z \rangle : x = y, x \neq z \}$
$I_{12} = \{ \quad " \quad x = z, x \neq y \}$
$I_{13} = \{ \quad " \quad y = z \; x \neq y \}$
$I_{14} = \{ \quad " \quad x \neq y, x \neq z, y \neq z \}$
$I_{15} = \{ \quad " \quad x = y + z \}$

$T_{16} = \{$ " : $x > y + z\}$

$T_{17} = \{$ " : $y = x + z\}$

$T_{18} = \{$ " : $y > x + z\}$

$T_{19} = \{$ " : $z = x + y\}$

$T_{20} = \{$ " : $z > x + y\}$

Test Case	x	y	z	Expected output
1	0	50	50	Invalid input
2	101	50	50	"
3	50	50	50	Equilateral
4	50	0	50	Invalid input
5	50	101	50	"
6	50	50	50	Equilateral.
7	50	50	0	~~Equilateral~~ Invalid
8	50	50	101	Invalid input
9	50	50	50	Equilateral.
10	60	60	60	"
11	50	50	60	Isoscales
12	50	60	50	"
13	60	50	50	"
14	100	99	50	Scalene.
15	100	50	50	Not a $\triangle$
16	100	50	25	"
17	50	180	50	"
18	50	100	25	"
19	50	50	100	"
20	25	50	100	"

Decision Table Based Testing

C_1: x,y,z are sides of Δ	N	Y								
C_2: $x=y$?	-	Y		N		Y		N		
C_3: $x=z$?	-	Y	N	Y	N	Y	N	Y	N	
C_4: $y=z$?	-	Y	N	Y	N	Y	N	Y	N	
Not a Δ	X									X
Scalene					X		X	X		
Isosceles										
Equilateral		X								
Impossible			X	X		X				

Condition										
C_1: $x<y+z$?	F	T	T	T	T	T	T	T	T	T
C_2: $y<x+z$?	-	F	T	"	"	"	"	"	"	"
C_3: $z<x+y$?	-	-	F	T	"	"	"	"	"	"
C_4: $x=y$?	-	-	-	T	T	T	T	F	F	F
C_5: $x=z$?	-	-	-	T	T	F	F	T	T	F
C_6: $y=z$?	-	-	-	T	F	T	F	T	F	T
a_1: Not a Δ	X	X	X							X
a_2: Scalene							X	X	X	
a_3: Isosceles				X						
a_4: Equilateral					X	X		X		
a_5: Impossible										

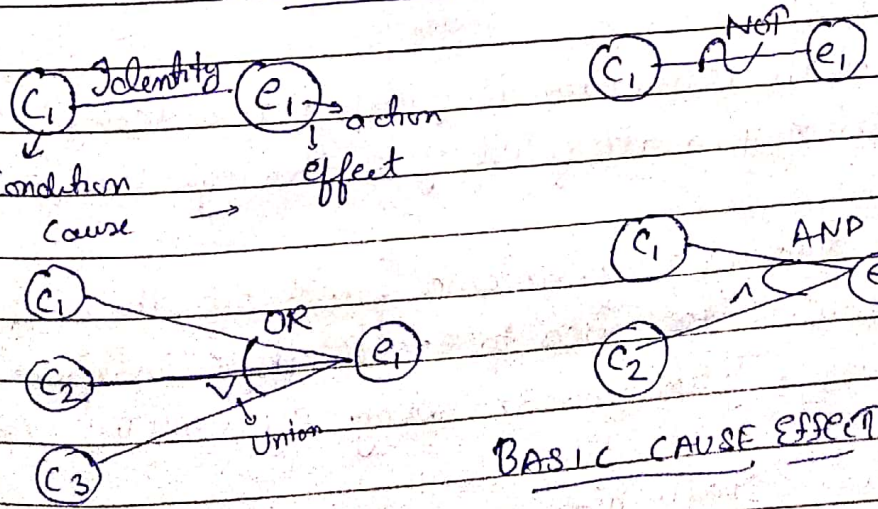
MODIFIED DESICISION TABLE

Test case	X	Y	Z	Expected Output
1	4	1	2	Not a Δ
2	1	4	2	"
3	1	2	4	"
4	5	5	5	Equilateral
5	?	2	?	Impossible
6	?	?	?	"
7	2	2	3	Isosceles
8	?	?	?	Impossible
9	2	3	2	Isosceles
10	3	2	2	"
11	3	4	5	Scalene

Cause Effect Graphing Technique

In this method we form a graph on the basis of different input & output condition

BASIC NOTATION FOR GRAPH



BASIC CAUSE EFFECT SYMBOL

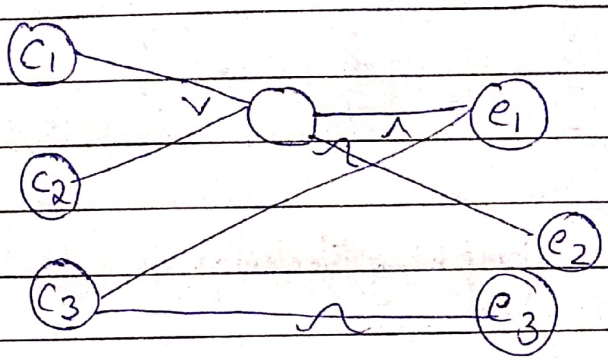
Consider the following example and create a cause effect Graph The character in column 1 must be A or B. The character in column 2 must be a digit. In this situation, the file update is made. If the character in column 1 is incorrect

message x is issued. If character in column 2 is not a digit, message y is issued.

The Causes are: → C_1: character in colum 1 is A
C_2: Character in colum 1 is B
C_3: " 2 is a digit

C_1	C_2	Outputs
A OR B	D- 0-9	file updated
X		x is issued
A OR B	0-9 N	y is issued

Effect: e_1 → update made
e_2: message x is issued
e_3: message y is issued



STRUCTURAL TESTING → A complementary approach to functional testing is called structural / white box testing.

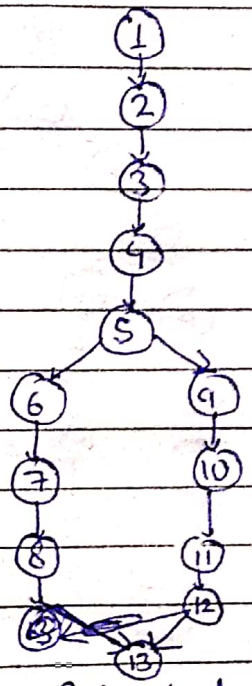
Path Testing is the name given to group of test techniques based on judiciously selecting a set of test paths through the program.

Consider the following program and find out the no of independent paths and write the test cases to check these independent paths and also verify your answer using cyclomatic complexity.

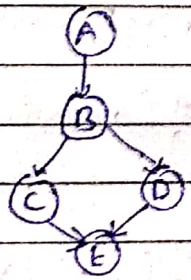
```
#include <stdio.h>
1 void main()
2 {
3 int a, b;
4 scanf("%d %d", &a, &b)
5 if (a > b)
6 {
```

```
7   printf("a is larger")
8   }
9   else
10  printf("b is larger")
11
12  }
13  }
```

Flow graph



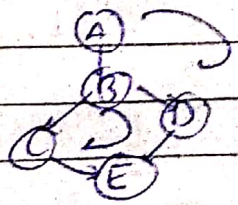
D-D Path Graph



Path

A → B → C → E

A → B → D → E



a	b	Expected output
50	40	a is larger
40	50	b is larger

Cyclometric Complexity $V(G) = e - n + 2p$

e is edges, P = connected component, n = nodes

$V(G) = 5 - 5 + 2$

$= 2$

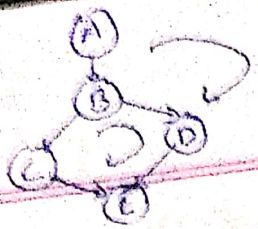
$V(G) = \pi + 1$

π = no of predicate node contained in flow graph

$1 + 1 = 2$

outgoing node

Cyclometric Complexity is equal no of region of flow graph

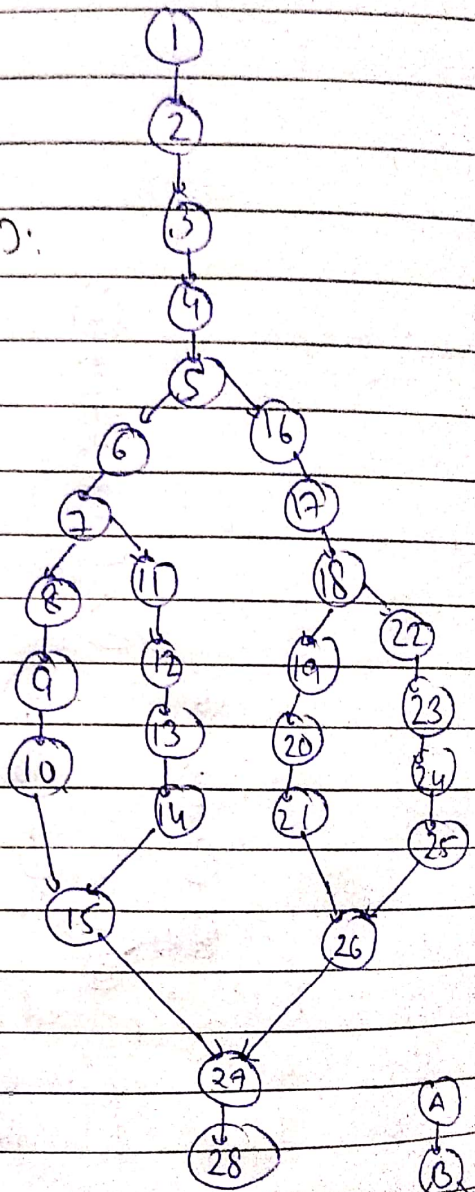


2 region

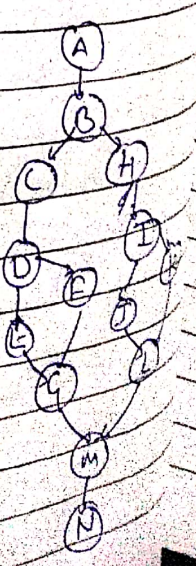
Date
Page

Consider the following program and find out of independent paths test case & verify your answer using cyclomatic complexity.

```
1  void main()
2  {
3  int a, b, c
4  scanf("%d %d %d", &a, &b, &c);
5  if (a>b)
6  {
7  if (a>c)
8  {
9  printf("a is larger")
10 }
11 else
12 {
13 printf("c is largest")
14 }
15 }
16 else
17 {
18 if (b>c)
19 {
20 printf("b is larger")
21 }
22 else
23 {
24 printf("c is largest");
25 }
26 }
27 getch()
28 }
```



Independent path



Test Case

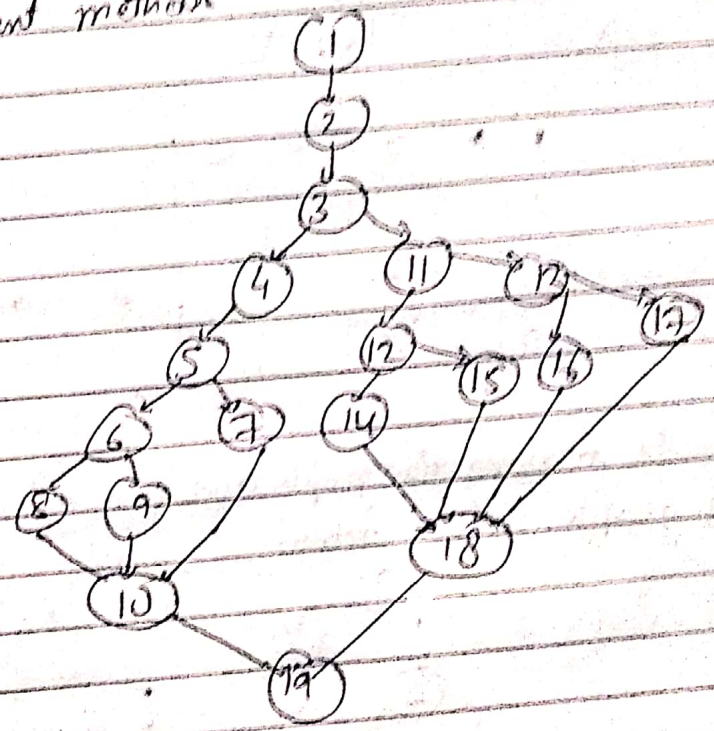
a	b	c	Expected Output
50	40	30	
50	40	60	
40	50	30	
40	50	60	

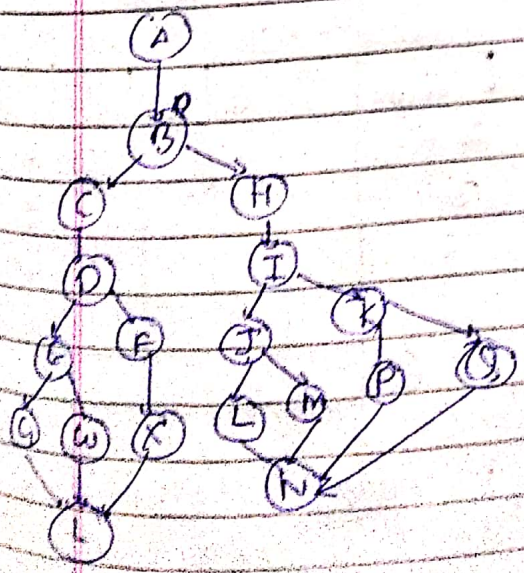
(i) $V(G) = e - n + 2p = 16 - 14 + 2 = 4$

(ii) $V(G) = \pi + 1 = 3 + 1 = 4$

(iii) $V(G) = 4$

(iii) Consider the following flowgraph and calculate complexity by all 3 different method





$V(G) = e - n + 2p$

$= 24 - 19 + 2$

$= 7$

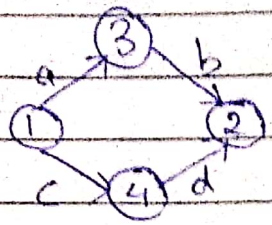
$V(G) = 7 = \pi + 1$

$V(G) = 7$

Graph Matrices:

Consider the following graph and find out cyclomatic complexity using graph matrice method

$4 - 4 + 2 = 2$



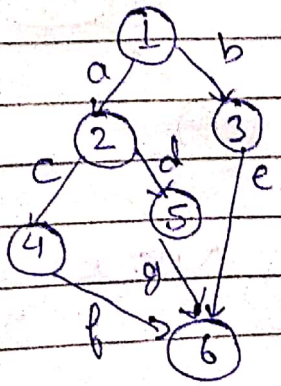
$\Sigma + 1$

$= 1 + 1 = 2$

1	♦	a	a^3	c^4	$2 - 1 = 1$
2					
3		b			$1 - 1 = 0$
4		d			$1 - 1 = 0$

$1 + 1 = 2$

Consider the following flow graph and find out cyclomatic complexity using graph matrix method



$\Rightarrow V(G) = e - n + 2p$

1		a	b		
2				c	d
3					e
4					f
5					g
6					

$= 2 - 1 = 1$

$2 - 1 = 1$

$1 - 1 = 0$

$1 - 1 = 0$

$1 - 1 = 0$

$= 3$

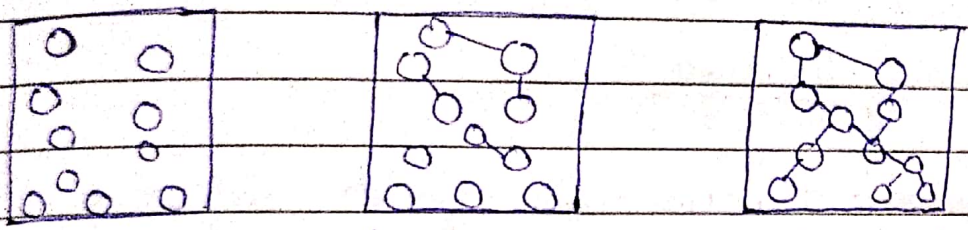
$2 + 1 = 3$

Date
Page

Levels of Testing

3 level of testing

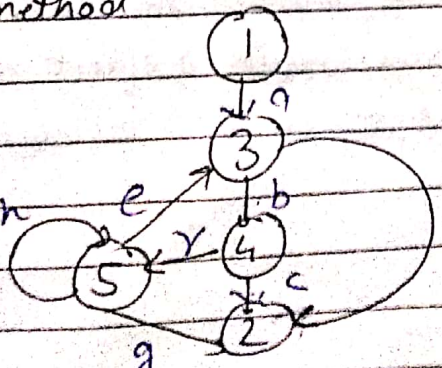
(i) Unit testing
(ii) Integration Testing
(iii) System testing



Unit Testing Integration Testing System Testing

(1) Alpha and Beta testing (2) Exceptance testing.

Consider the following flow graph & Calculate cyclomatic complexity using all four method



a. $V(G) = e - n + 2p$
$= 8 - 5 + 2$
$= 5$

b. $V(G) = \pi + 1$
$4 + 1 = 5$

$V(G) = 5$

1			a		
2					
3		d		b	
4		c			r
5		g	e		h

$= 1 - 1 = 0$

$2 - 1 = 1$
$2 - 1 = 1$
$3 - 1 = 2$

$= 2 + 1 + 1 + 1$

$= 5$

SOFTWARE MAINTENCE

Software Maintenance is a very broad activity that includes error corrections, enhancements of capabilities, deletion of obsolete capabilities and optimization.

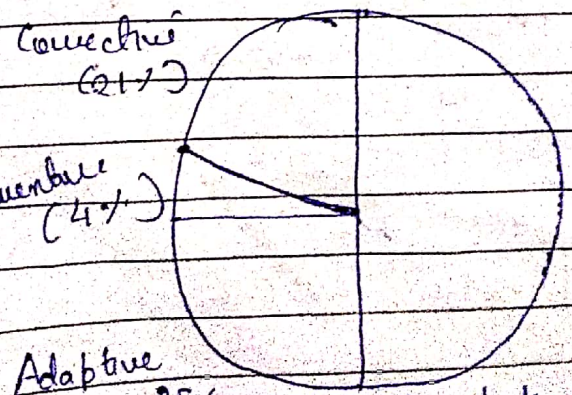
Categories of Maintence :-

(1) Corrective Maintence - This refer to modifications initiated by defects in the Software.

(2) Adaptive Maintence - It includes modifying the Software to match changes in the ever changing environment.

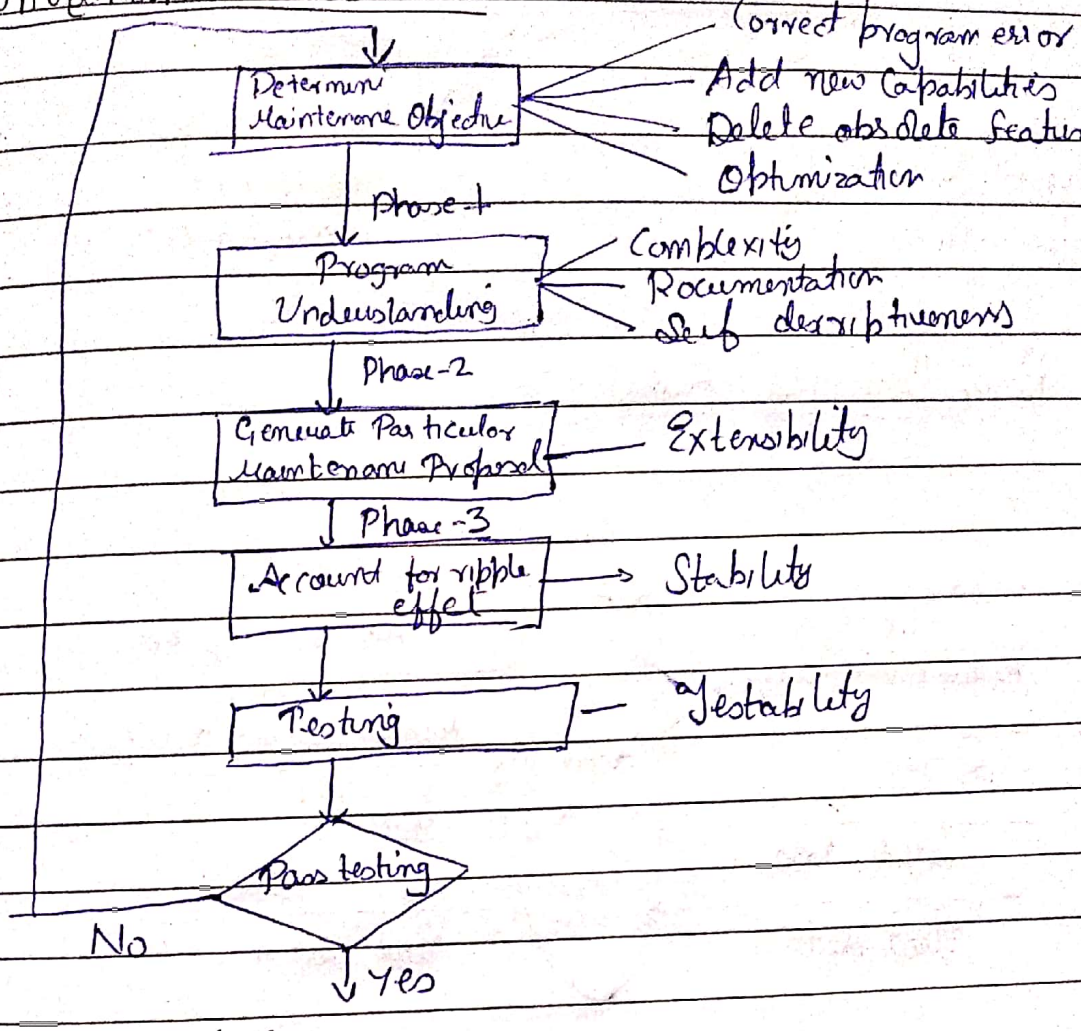
(3) Perfective Maintence - It means improving processing efficiency or performance or restructuring the Software to improve changeability.

Preventive Maintence :- There are long term effects of corrective adaptive and perfective changes. This leads the increase in the complexity of software, which reflect deteriorating structure. The work is required to be to maintain it.



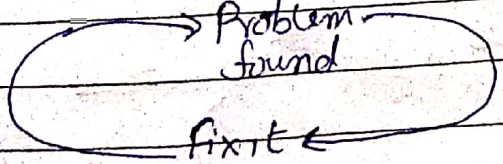
Distribution of maintenance effort

Maintainance Process



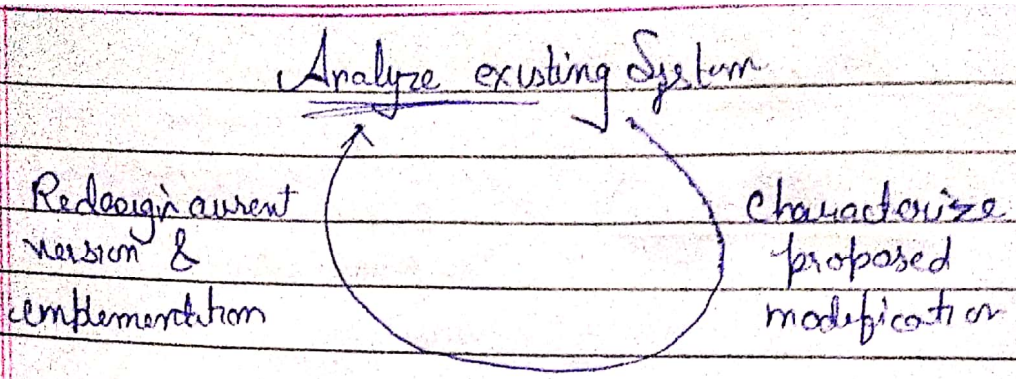
Maintenance Model

(1) Quick-fix Model - This is basically an adhoc approach to maintain software. It is fire fighting approach waiting for problem to occur and then trying to fix it as quickly as possible

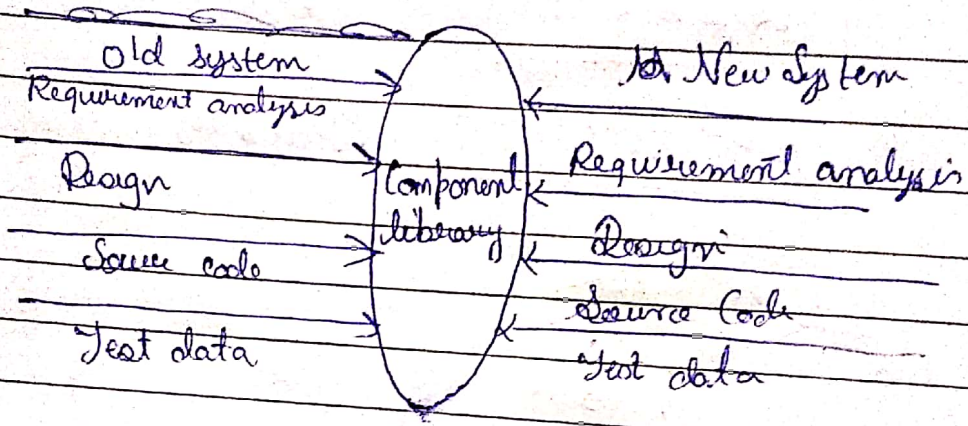


(2) Iterative Enhancement Model
→ Analysis
Characterization of proposed modifications
Redesign & implementation

Ripple Effect - The third phase consisting of accounting for all of the ripple effect as a consequence of program modifications is called ripple effect

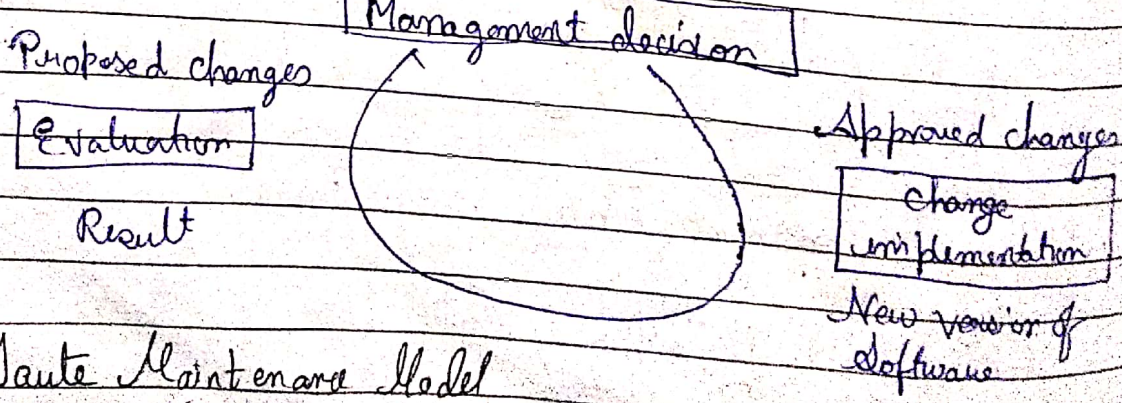


(ii) Reuse Oriented Model



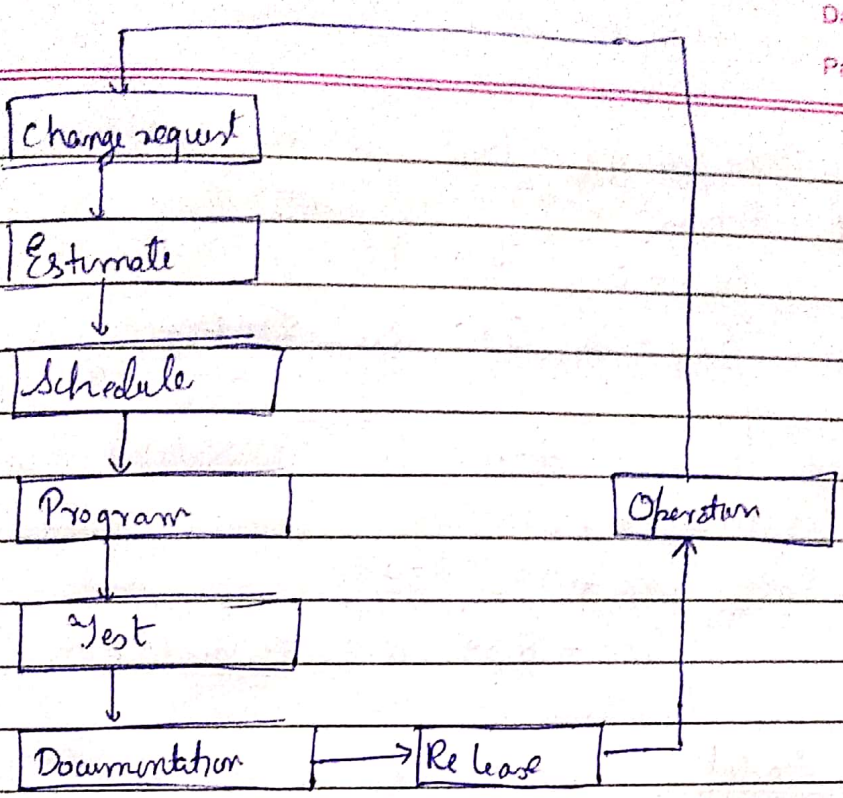
(3) Boehm's Model

Boehm proposed a model for maintenance process based upon the economic model and principle



Taute Maintenance Model

It is typical maintenance model and has eight phases in cyclic fashion.



Belady and Lehman Model

$$M = P + K e^{(c-d)}$$

M = Total effort expended
P = Productive effort
K = An Empirically
C = Complexity measure
d = Degree

Ex: The Development effort for a software project 500 person months. The empirically determine constant (K) is 0.3. The complexity of code is quite high is equal to 8. Calculate Total effort expended (M) if

- maintainer team good level of understanding ($d = 0.9$)
- " poor " ($d = 0.1$)

Development effort (P) = 500PM
$K = 0.3$
$C = 8$

- Maintenance team has good level $d = 0.9$

$$M = P + Ke^{(c-d)}$$
$$500 + 0.3e^{(8-0.9)}$$
$$500 + 363.59 = 863.59 \text{ PM}$$

Maintenance has poor $d = 0.1$

$$M = P + Ke^{(c-d)}$$
$$500 + 0.3e^{(8-0.1)}$$
$$500 + 809.18 = 1309.18 \text{ PM}$$

Boehm Model

↳ Annual Change Traffic (ACT)

$$ACT = \frac{KLOC_{added} + KLOC_{deleted}}{KLOC_{total}}$$

$$AME = ACT \times SDE$$

ACT = Annual Change Traffic
SDE = Software development effort in person months
EAF : Effort Adjustment factor
AME → Annual Maintenance Effort

$$AME = ACT * SDE * EAF$$

Annual Change Traffic (ACT) software System is 15% per year. The development effort is 600PMs. Compute estimate for Annual Maintenance Effort (AME) & life time of project is 10yrs. What is total effort of project

$$ACT = 1.5\% \text{ per year} \quad \frac{15}{100}$$
Development effort 600PM
$$AME = ACT \times SDE$$
$$0.15 \times 600 = 90 \text{ PM}$$
Maintenance effort for 10 year $= 10 \times 90 = 900 \text{ PM}$
Total effort $= 600 + 900 = 1500 \text{ PM}$

A Software project has development effort of 500PM. It assumed 10% code will modified per year. Sum cost of multiplier
- Required Software Reliablity
- Data base size
- Analyst Capability
- Application experience
- Programmer's language experience. (LEXP) high
- Other multiplier are normal. Calculate Annual Maintance Effort (AME)

Annual Change traffic (ACT = 10%)

SDE = 500 PM

Using Table 5 Cocomo Model

RELY = 1.15

ACAP = 0.86

AEXP = 0.82

LEXP = 0.95

DATA = 1.08

EAF = 1.15 × 0.86 × 0.82 × 0.95 × 1.08 = 0.832

AME = ACT * SDE * EAF

0.1 * 500 * 0.832 = 41.6 PM

AME = 41.6PM

Consider a Software of size 1000 KLOC Suppose in maintenance 100 KLOC are added and we are deleted 50 KLOC. Suppose initially the Software was developed in a effect of 500 person. Calculate what will be the effort maintainence effort required for 6 month

SDM = 500 PM

$$ACT = \frac{100+50}{1000} = \frac{150}{1000} = 0.15$$

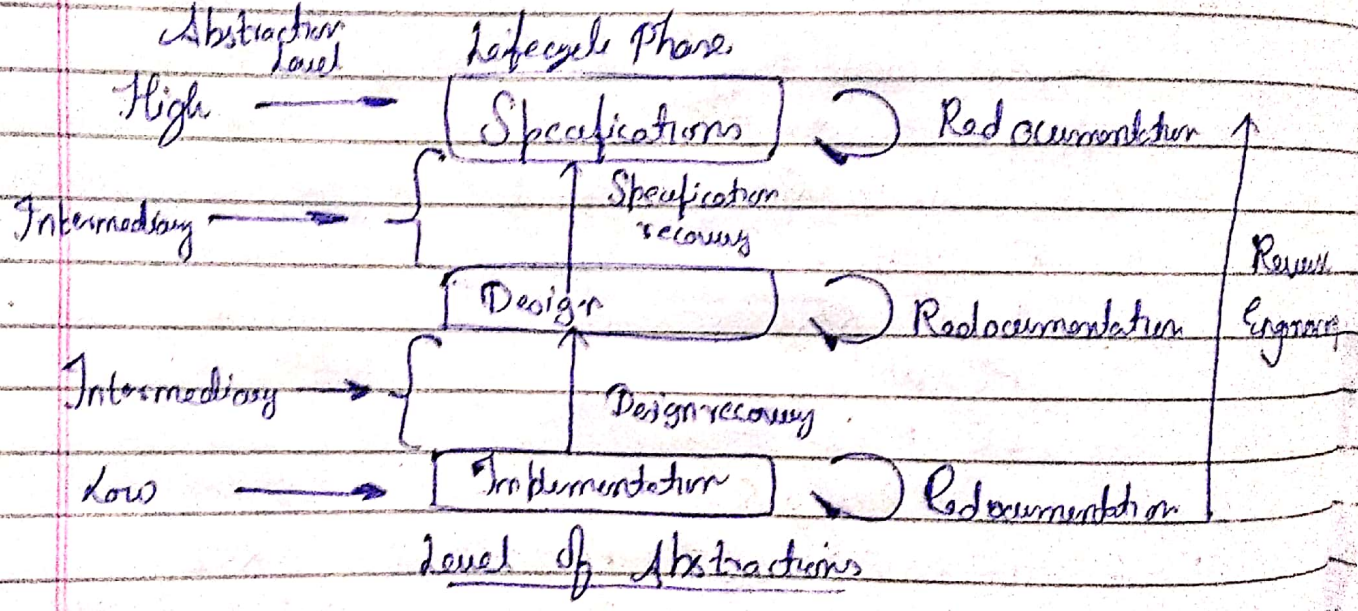
$$AME = 500 \times 0.15 \times \frac{1}{2} = 37.5 PM$$

SOFTWARE RELIABILITY

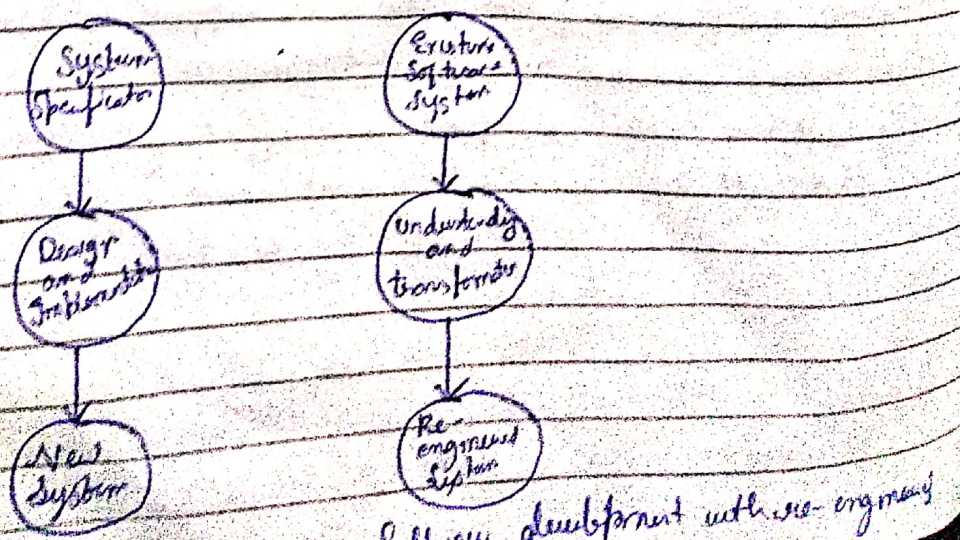
SOFTWARE QUALITY –

1) – McCall Software Quality Model :-

Reverse Engineering – is process to find out followed in order to get difficult unknown and hidden information about a Software system.



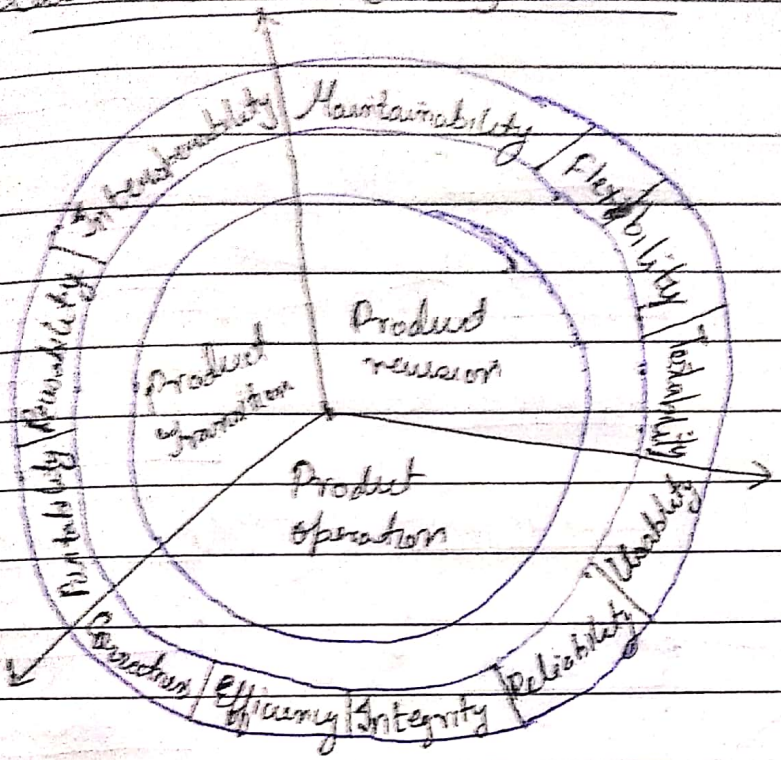
Software RE-Engineering – is concerned with existing



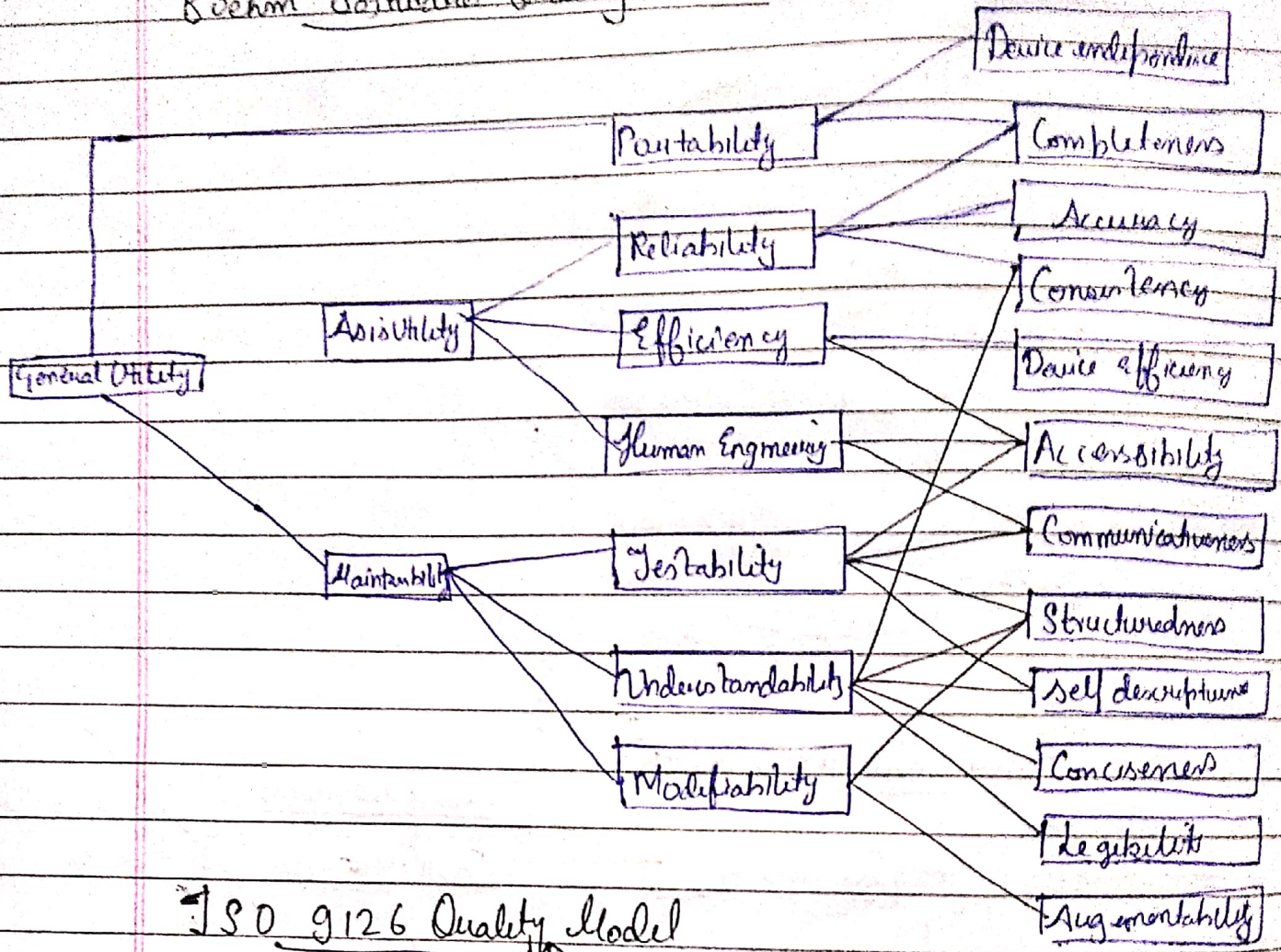
Continuum of new Software development with re-engineering

Software Reusability

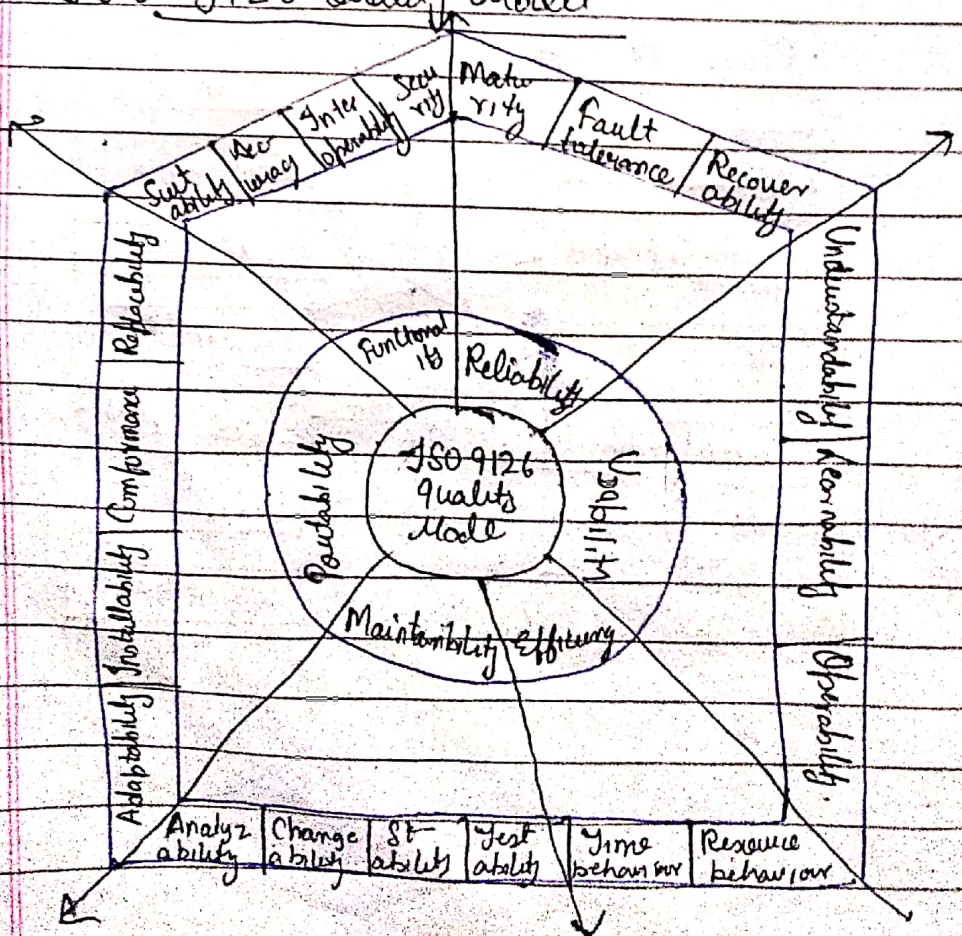
* McCall Software Quality Model



Boehm Software Quality Model

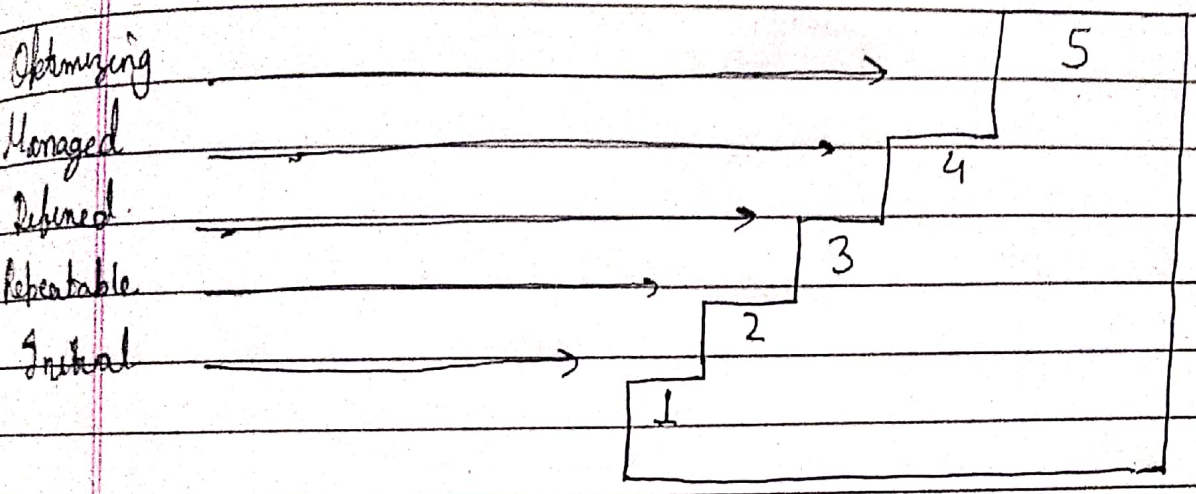


ISO 9126 Quality Model



Capability Maturity Model (CMM)

It is a strategy for improving the software process, irrespective of the actual life cycle model used



ISO 9000 → The SEI capability maturity model initiative is an attempt to improve software quality by improving the process by which software is developed

ISO-9000 series of standards is a set of document dealing with quality system that can be used for quality assurance purpose. It is a series of five related standard that are applicable to a wide variety of industrial activities, including design / development, production, installation and servicing.