Soft ware Engineering

Part B.

Md. Sahil

001710501029

PART (B)

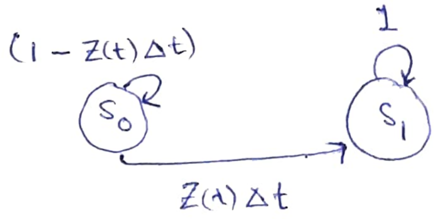
1. a) Iterative model of software development startx with a simple implementation of a small set of the software requirements and iteratively enhances the evoluiy versions ~~unit~~ until the complete system is implemented and ~~red~~ ready to be deployed.

It combiny the advantages of both the waterfall model and the the prototyping model. ~~I~~ Like in the prototypiy model, ~~the~~ an early implementation of the system can be presented to the user at an early state and the developer can gain experience from the user feedback. And like in water fall ~~mode~~ model ~~the~~ ~~at each step goals are clear~~, ~~thus~~ there is no overlaping of steps as each iteration is completed one at a time, the milestones are clear at each state. At the same time we avoid the problem of fixed specificatia in ~~the~~ waterfall model.

b) Markov reliability Model:-

Let ~~the~~ us consider a software system as follows:-

- All states are mutually exclusive
- The system composed of a non-repairable element X_1
- The two possible states are:-
  - ~~S_0~~ S_0 when X_1 is good
  - S_2 when X_1 is bad.



Initial State	Final State	
	S_0	S_1
S_0	$1 - Z(t)\Delta t$	$Z(t)\Delta t$
S_1	0	1

state transition table.

- At $t=0$, the system is in initial state. Final state is reached when the system in in equilibrium.

2. Requirement Engineering is the process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.

Generally the user provided specifications are not enough to guide the ~~due~~ developemet of the system. Requirement engineering involves translating the user provided requirements to form ~~the describ~~ a complete describtion of the system services and calculate the constraints of the system.

Enginering not only involves analysix but also formulating the Standard ~~guidelinse~~ guide lines and tools and the breakdown of requiremet for building the solution.

IEEE specifies the ~~medo~~ standard methods involving Analyses, specification & management.

Types of Requirement :-

(i) User Requirements :-

Ex- In case of a library system,

→ The ability to issue & add books to the library.

(ii) System requirements :-

Ex - In case of an library systen

→ ~~The~~ The system should be able to handle storing of large number of books.

(iii) Software specification :-

Ex → Seperate UI for librarians (admin) & borrowers

→ Intuitive UI.

Classification of requirements on the basis of functionality :-

(i) Functional requirements :-

Ex → A user should be able to issue & return books.

→ An admin should be able to add new books.

(ii) Non functional requirements :-

Ex → The system should be secure

→ Should provide low latency.

→ Should maintain high (99%) uptime.

Classification on the basis of satisficability :-

(i) Normal requirements :-

→ The ability to issue & return books.

(ii) Expected Requirements :-

→ A brrower must have a limit on the number of books that can be issued.

→ error handlig.

(iii) Exciting Requirements :-

→ Sms notifications ~~sho~~ sent to the borrowers of the

3.

a) The factors related to the software quality metric are :-

(i) Product Revision

1. Maintainability
2. Flexibility
3. Testability.

(ii) Product Transition

1. Reusability
2. Interoperability
3. portability.

(iii) Product Operations.

1. Reliability
2. Correctness
3. Efficiency.
4. Usability
5. Integrity.

b). Module Strength & Cohesion dictates the internal activity of a module. ~~In general one module shoud perform one~~. Module cohesion is a measure of the degree to which the elements of a module are functionally related.

In general one module shoud perform a single task. Modules that perform multiple tasks are difficult to maintain and may lead to coupling problems.

Thus a good coupling ensures good cohesion of a module.

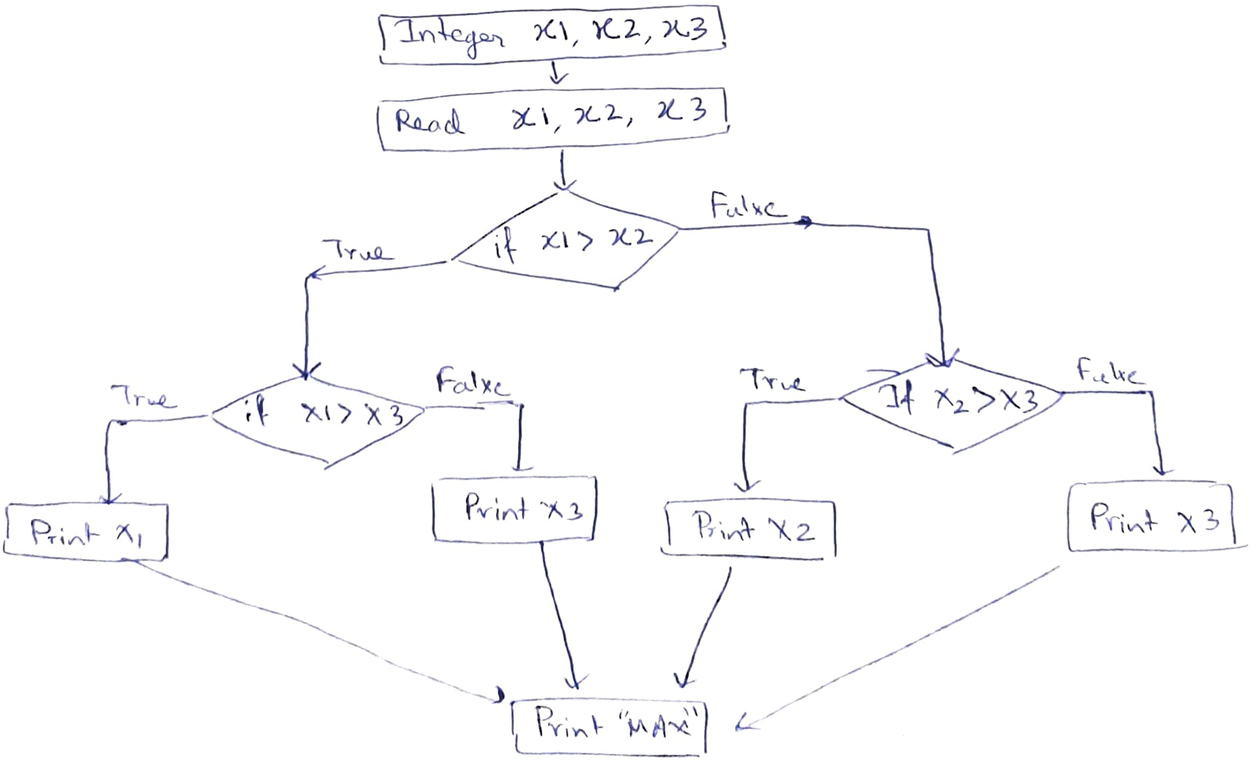
Types of cohexion :-

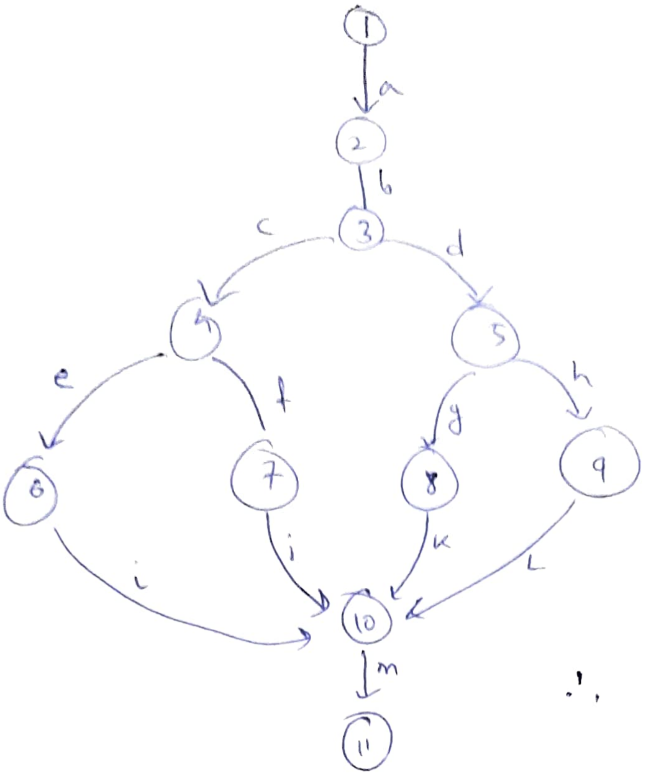
1. coincidental (worst)
2. logical cohesion
3. classical cohesion
4. Procedural cohesion
5. communicational cohesion
6. Functional cohesion
7. Informational cohesion (best).

4.

a) Cyclomatic complexity is a software metric that provides a quantitative mesure of the logical complexily of the program.

Flow graph of the given program :-





Number of Nodes, N = 11

Number of Edges E = 13.

Cyclomatic Complexity

$$V(G) = E - N + 2$$
$$= 13 - 11 + 2 = 4.$$

∴ By flow graph method cyclomatic complexity = 4.

The graph matrix for the given code :-

Connecting Nodes →

Nodes ↓

	1	2	3	4	5	6	7	8	9	10	11	Σ w - 1
1		a										1 - 1 = 0
2			b									1 - 1 = 0
3				c	d							2 - 1 = 1
4						e	f					2 - 1 = 0
5								g	h			2 - 1 = 0
6										i		1 - 1 = 0
7										j		1 - 1 = 0
8										k		1 - 1 = 0
9										L		1 - 1 = 0
10											m	1 - 1 = 0
11												

Sum = 3.

∴ cyclomatic complexity = 3 + 1 = 4.

The basic path set (contains 4 paths) are given as follows :-

{ abccim, abcfim, abdgkm, abdhLm }

5.

a) Software complexity is used to describe the characteristics of code and how its sections interacts with each other.

Three measurements of characteristics determines the software complexity.

Software complexity can be measured as:-

1. Structural complexity → estimated by physical lines of code
   → how many variables, constraints are present.

   It can be calculated using Hailstead's theory of measurement of software complexity [a set of primitive measures].

2. Logical complexity :→ Measuring control flow like decision loops etc.
   → It can be calculated as cyclomatic complexity.

b). Program:-

```
Integer X1, X2, X3
Read X1, X2, X3
if ( X1 > X2) then
      if (X2 > X3) then
            Print X1
      else
            print X3
Else
      if (X2 > X3) then
            print X2
      Else
            print X3
Print "MAX"
Stop.
```

Operators	Occurrences
integer	1
Read	1
if	3
( )	3
then	3
>	3
else	3
print	5
Stop	1
,	4

Operand	Occurrence
x_1	5
x_2	5
x_3	6
"Max"	1

→ Total no of unique operators $n_1 = 10$

→ Total no of operators $N_1 = 27$

→ Total no of distinct operands $\eta_2 = 4$.

→ Total no of operands $N_2 = 17$.

Program length, $N = N_1 + N_2 = 27 + 17 = 44$.

Distinct number of actual i/p and o/p $\eta_2^* = 5$

3 reads for x_1, x_2, x_3
1 value output
1 "Max" print

Estimated program length $N^\wedge = n_1 \log_2 n_1 + n_2 \log_2 \eta_2$

$= 10 \log_2 10 + 4 \log_2 4$

$= 33.219 + 8$

$= 41.219$

Program volume $V = (N_1 + N_2) \log_2 (n_1 + n_2) = 44 \log_2 14 = 44 \times 3.807 = 107.508$ bits

critical volume $V^* = (2 + n_2^*) \log_2 (2 + n_2^*)$

$= 7 \log_2 7 = 19.65$ bits.

Program level, $L = V^*/V = \dfrac{19.65}{167.508} = 0.117$.

Program effort, $E = V/L = \dfrac{167.508}{0.117} = 1431.69$ bits

Program speed $S = E/8 = \dfrac{1431.69}{18} = 79.54$ seconds.

[taking $s = 18$].

Hence required answer

critical volume of program is 19.65 bits.

6.) a) Availability is the probability that the program is performing successfully at a given point of time. Availability essentially means that the system is up and running, according to specifications at any time 't'.

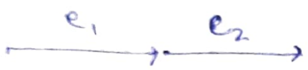
Redundancy is introduced to improve the system reliability. It is achieved by and connecting a duplicate in parallel.

The types of software redundancy are :-

i) Unit redundancy

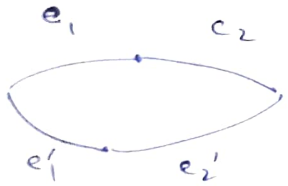
ii) Component redundancy.

Initial system :-



$R_a(P) = P(e_1) \cdot P(e_2) = P^2$

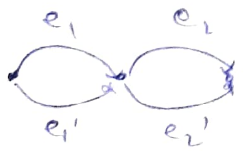
In unit redundancy, additional path for the entire system itself is provided.



In this e_1 and e_2 are two components of the system and two components e_1' & e_2' are provided in parallel thereby improving the reliability of the entire system.

$$R_b(P) = P(e_1e_2 + e_1'e_2') = 2R_a - R_a^2$$
$$= P^2(2-P^2)$$

In component redundancy, additional path for each component of the system is provided.



In this additional components are added in parallel to each existing component. This improves the reliability of each component, thereby for the entire system.

$$R_c(P) = P(e + e_1')P(e_2 + e_2')$$
$$= P^2(2-P^2)$$

$\therefore R_c \geq R_b \geq R_a$

7.

a) Regression Testing :-

Regression testing is a type of software testing to confirm that a recent ~~prgr~~ program or code change has not affected existing features.

Each time a new module is added ~~as~~ ~~a~~ the following changes may occur :-

1. New data flow paths are established.
2. New I/O may occur
3. Functions may not work flawlessly
4. New control logic is invoked.

Regression testing involves reexecution of some subsets of tests that have already been conducted to ensure that changes have not propagated unintentional side effects.

The testing can be done using one of the following ~~ca~~ ~~tech~~ techniques:-

1. Will ~~re~~ exercise all software functions
2. Functions that are likely to be affected by the change.
3. Software components that have been changed.

b). Conservation of data for process and for store refers to :-

1. What comes out of data store must go in.
2. It is not possible for data store to create new data elements.
3. The above points are true for process also.