Software Engineering Notes

Contributor: Tanya Garg

KMV CS(H)

Computer Science Notes

2-01-18

Page No.
Date

INTRODUCTION

- provides theoritical background and some procedures in order to find end product.

software → application made for a specific task & is user intractive.

↓

program that may ~~con~~ include a user interface and may have a database

Enginering → to develop software.
↳ defines some procedures.

(end product)

- A computer software is a product that software professional builds and support over a long term.
It includes programs that execute within a machine of any size, architecture and content.

- Software Enginering encompasses a process, a collection of methods and an array of tools that allows a programmer to built high quality software.

Software is a collection of independent Modules

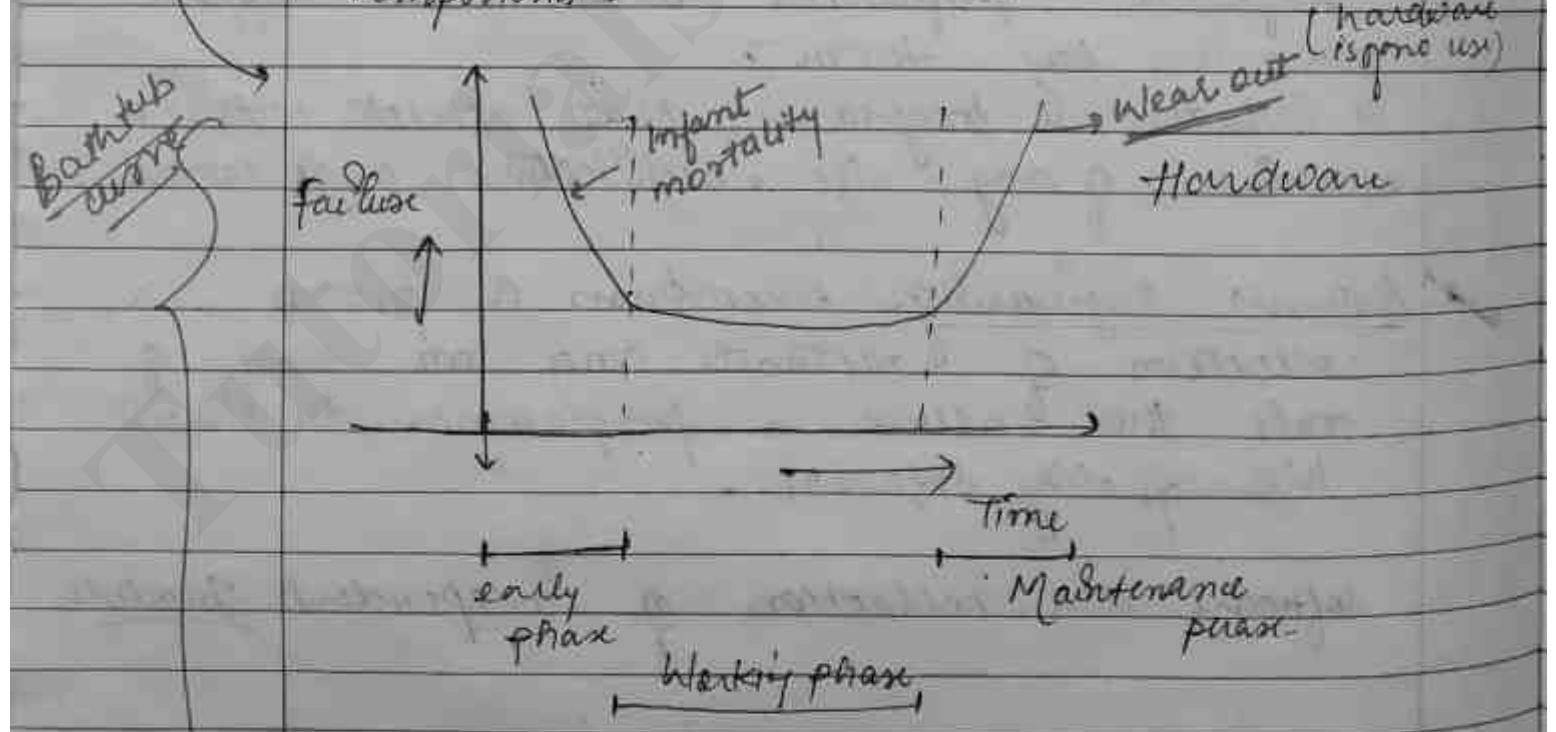
Page No.
Date

⇒ The Nature of Software

→ Software takes on a dual role.
It is a product and at the same time the vehicle for delivering a product.

Q1 ⇒ Characteristics of Software

1) Software is logical rather than physical system element.
2) Software is developed or engineered. It is not manufactured.
3) Software doesn't wear out.
4) A software is composed of several reusable components.

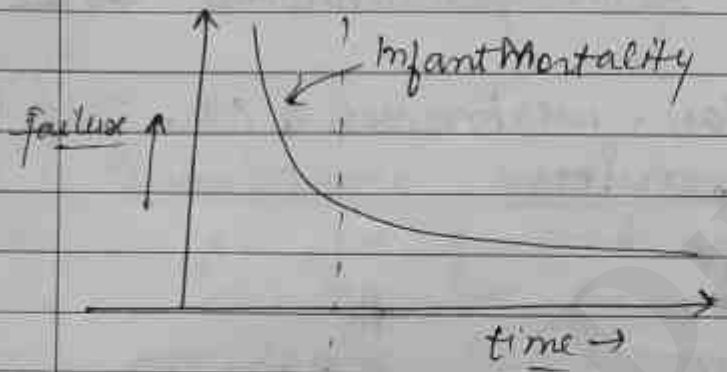


Page No.
Date

- Infant Mortality →
  The product fails in the very initial state/phase

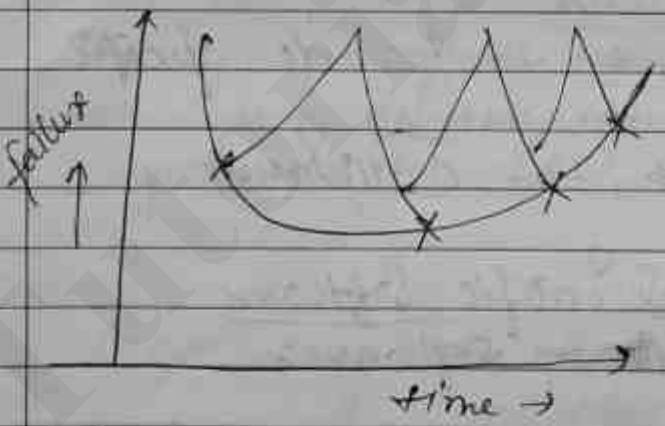
In working phase, possibility of failure is least.

Software Curve eg → windows XP. (theoritical curve)



It might happen that during its initial / dispensing phase, it may fail.

Idealized Curve → But if a software is made



(Practical Software Curve (actual))

Theoritical Curve isn't possible as of changes in technology and change in needs. Thus we always need to modify or ~~changed in~~ update our existing software.

As soon as change in software is done, it is

possible that it might fail, resulting in high failure.

Changes in a software are limited as it leads to increase in cost & time.

→ Software Application Domain

1) System software.
which provide basic functionalities to the system.
Ex → text editors, web browsers, OS, compilers, drivers.
directly interact with hardware & system

2) Application Software
designed in order to provide specific functionalities to user.
eg → Whatsapp, fb, antiviruses.

3) Engineering or Scientific Software
aeronautical Engineering software,

4) Embedded Software
software which are embedded directly into hardware device.
eg → AC, speedometer

Page No.
Date

5) Web Applications → Browse apps, gmail, fb, Orcut, netbanking apps

6) AI Software → Software used in pattern recognisation robotics, etc. Basically used in artificial engineering.

⇒ The unique nature of Web applications completely rely to internet activities.

[Design Issues]

1) Performance and availability.
purpose of web appln must be solved and running all the time that is must be available (eg → gmail). These apps have a wide nature that is desirable.

2) Network Intensiveness (heterogenity)
The apps designed in web can be used by an user on system irrespective of hardware system OS (gmail)

3) Concurrency → simultaneously access to multiple user. eg → gmail accessed by multiple users.

4) Unpredictable load → the no. of user accessing a particular appln is are changing.

↓ primarily with web apps

04/02/18

Page No.
Date

* A problem of concurrency.
① Inconsistency
The data must remain consistent while multiple access web apps.
② fault tolerant.

(5) Data driving
They transfer data from one user to another where data can be text, images, etc. Any type of data can be transported

(6) Content Sensitive
since networking is involved, thus content may leak by any intruder or appl^n, thus content becomes sensitive. Thus, data to be communicated must be secure.

(7) Continuous Evaluation
All the web apps (stand alone or hosted) must be timely updated/evaluated to meet the market needs. If not updated the web page becomes less popular & thus fails.
ex → Orkut is now obsolete.

(8) Security
All webapps must be secure

(9) Immediacy
Software must be launched in market quickly otherwise some other app. the target for its releasing is very short.
Web apps are called "Short duration apps"

(10) Aesthetics
This is associated with ~~look~~ look and feel. The GUI (Interface) of a web App is of prime concern.
The swites must be consistent.

⇒ SOFTWARE ENGINEERING

• By Fritz Bauer, [NAU 69]
According to Fritz,
Standard def" "Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines" (for a long duration of time)
economic in terms of time, human effort & cost.

Effective
- gives out correct output

Efficient
- gives out correct output with less time
- Its output stands on all software

This definition tells reliability but not Technical aspects

tells about qualitative but not a quantitative approach.
It doesn't talk above some parameters.
→ This dfn is modified in

• By IEEE [IEE 93a]

has developed the more comprehensive definition which states that →

Software Engineering is

* (1) The application of systematic, disciplined, quantifiable approach to the development, operation & maintenance of software i.e the application of Engineering to software.

* (2) The study of approaches as in (1)

Disciplined in the sense that the software must lie within defined boundary
Quantifiable in the sense that is performance can be calculated / measured to meet customer's need.
Development - refers to initial stage design and development
Operation → It is the stage when we operate our software

Page No.
Date

Maintenance → continous process until software goes out

→ Software Engineering as Layered Technology (Topic 3)

Layered Structure

→ Product that is to be made. must be a quality product meeting user requirements

→ Process that binds or ~~how~~ to defines how to build tools for a software and methods available to provide a quality product.
Acts as a medium b/w upper 2 layer & lower layer.
- defines a framework with a set of activities
Methods are the activities

→ We have particular Methods at every step of process.

testing is a process
& unit testing, alpha testing are methods

→ Tools

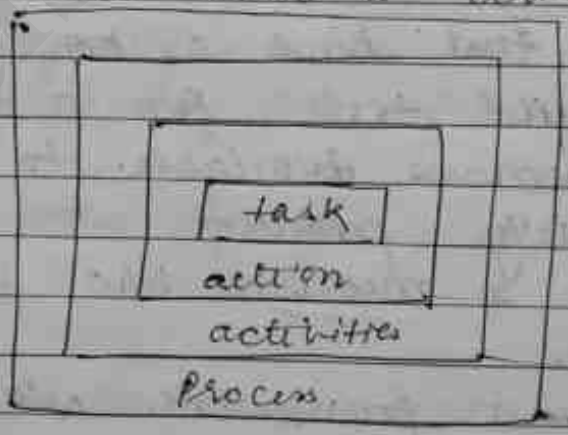
They are automated software or that are software itself.

eg → Android Studio for Android

(Inbuilt software)

⇒ SOFTWARE PROCESS (Topic-4)

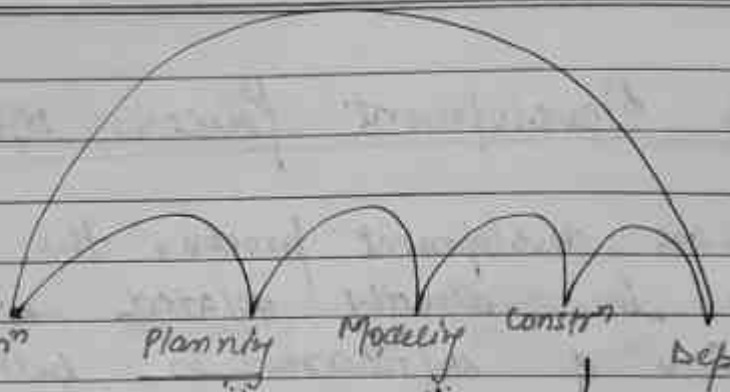
- a process is a collection of activities, actions & tasks that are performed when some work product is to created.
- A process defines who is doing what, when and how to reach a certain goal.



- A process framework establishes the foundation for complete software engineering process by identifying a small number of framework activities that are applicable to all software projects irrespective of their size and complexity. These framework activities are called as

Umbrella activities.

5 Umbrella activities



- developer is communicating with user & vice versa. It is the requirement analysis
- provides interaction b/w client & developer
- It is a necessary & an integral step.

- It is rough structure
- define a model structure of a software
- physical work
- sketches / diagram

- design of coding part (actual part)

- Deployment involves maintenance also.
- last step where we deploy our Software.

- defines how to carry out a development of software
- associated with reliable design of software.
- Understanding of comⁿ & designing a plan.
- logical work

⇒ All the process models follows these 5 activities

Page No.
Date

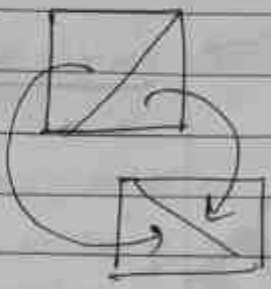
• Software Development Process Model

In software development process, the focus on activities is directly related with production of software is being given.

As the development process specifies the major developments and quality control activities that needs to be performed in the project and for this development process various process models have been proposed each with its strength, weakness and applications.

Page No.
Date

① Waterfall Model



(1) validation / System feasibility → feasibility Report } outcome of each step.

(2) validation → Requirement document & project plan (SRS)

Requirement analysis & project planning

(3) verification / System design → System design doc (SDD)

working on seq. (How & when)

(4) verification / Detail Design → Detail Design doc (DDD)

(5) verification / Coding → Programs

(6) verification / Integrating → Test Plan & Report Manually.

and Testing

(7) Installation → Installation Report

(8) Operation & Maintenance

↔ basic theme of this model is based on 5 umbrella activity

Page No.
Date

⇒ Verification
is a small process or small entity in comparison with validation.
It is valid for a particular process only. & we have verification b/w entry and exit step.

eg → Verification at step (3) has entry pt as Req. document & project plan and exit phase as system design document. Verification ensures that input & output matches each other

- Validity – Composed of entire process or program we check that software is valid or not. Also called as validation project. Validity is checked through feasibility project

Step-7 Installation

Step-1 Feasibility phase
whether project is possible or not.
Time set + make the project.

This model is a linear Model. It has no cycles.

Outcome of 1st phase is feasibility report
{ provides all feasibility features in software }

Step-2 Req. Analysis – what we need & what features can be included in ~~the~~ ~~proj~~ project.

Page No.
Date

Project Plan - small planning of the project.

Step-3 System Design - Rough design about the project. Doesn't involve any coding and has relationship b/w entities. How features are combined with each other.

Step-4 Detail design - how all processes are related.
SDD - modules
Detail design combines all modules.

Step-5 Coding phase - writing the software actually. O/P of coding is programs.

Step-6 Integration and Testing - Combine & test each & every module. O/P is test plan and test report (outcomes of all the test plan)

Step 8 Operation & Maintenance - The software is in execution & is at customer end. (end-user)

Page No.
Date

⇒ Advantages

1) simplicity.
2) clearly divided phases. (clearly divided)
3) easy to execute and implement
4) Intuitive and logical. {each and every step is logically defined}.

no phase conflict

⇒ Limitation

1) It assumes that requirements of a system can be frozen.
2) ~~Process Phase~~ Begin the requirement require selection of hardware at early stage which is not preferred in this era of technology change.
3) It follows the big bang approach.
4) It is a document driven process.
5) All or nothing approach
6) Disallow changes.
7) Cycle time is too long.
8) No feedback is not allowed. (User)

- We must know all the requirements from the very first step that is programmer should be clear about his work.

→ Big-Bang - The user comes to know abt the project after coming on 7th or 8th stage. The whole project gets to know in one step, can't be

Page No.
Date

known abt. steps (not continuous interaction with the user)

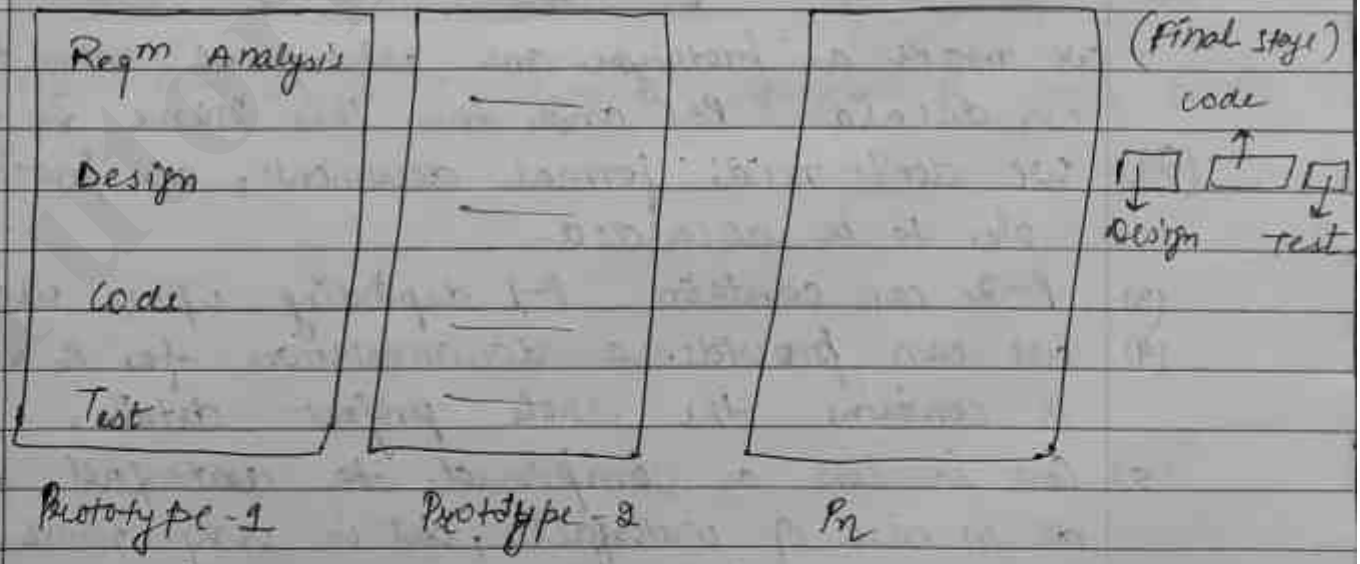
→ All or nothing Approach → project is either accepted or rejected.

⇒ Applications of ~~feedback~~ Waterfall Model

1) for well understood problem.
2) short duration project.
3) Automation of existing manual system.

→ Project made with waterfall model is clear & known beforehand. It is only possible when we enhance an existing product. It is used for short duration as even if the project fails, the investment would be compensated. It is only used when all resources and req. are known.

⇒ ROTOTYPE MODEL



(Small section of whole process)

→ This model overcomes all limitations of waterfall model. Here, we provide a small product of the entire software.

⇒ Design → designing of given req^m giving each prototype to user, we can remove old features & keep on adding new features. Here customer is regularly involved & thus customer can't deny about the made project. We can take continous amt. from user according to prototypes.

⇒ Requirements

(1) Before proceeding, the requirement of prototype is that it must be a short duration prototype. Req^m included must be such that they can be delivered in a timely manner.

Known as "Quick And Dirty Approach".

we make a prototype and when we move to P_i, we discard P-1 and thus this name is given.

(2) we don't need formal document, as prototypes are to be discarded.

(3) P-2 can contain P-1 depending upon users req^m.

(4) We can provide a documentation for P_n as it contains the whole project details.

5) Cost is less as compared to waterfall model as in case of waterfall, cost is very much higher at later stages.

Page No.
Date

⇒ Advantages

1) Helps in requirement elicitation (/gathering)
2) Reduce risk.
3) Leads to a better system.
4) Minimal documentation.
5) Cost effective

⇒ Disadvantages (We need to keep working on every prototype)

1) Front heavy process
2) Possibly higher cost due to continuous change.
3) It is a continous and complicated process

⇒ Applications

→ (new / beginner users)

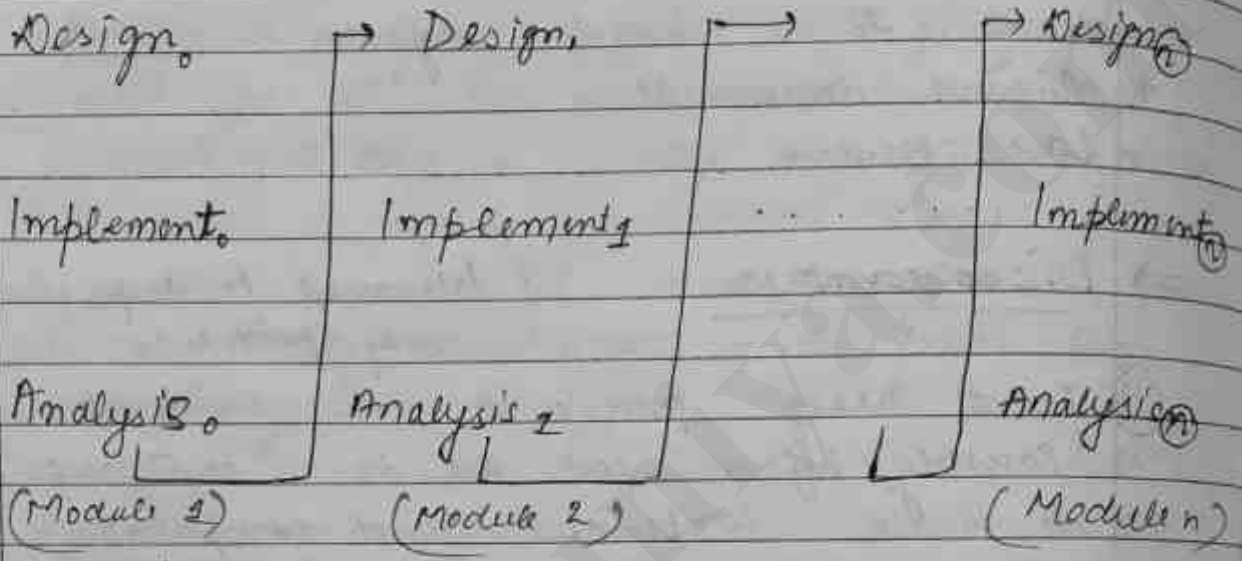
1) System with Novice users.
2) When uncertainity exist in requirements
3) When user interaction is very important.

uncertainities wrt. hardware, software & other reqms.

Page No.
Date

Increments

Iterative Enhancement Model



- Combines the approach of waterfall as well as prototype model
- controls a project control list (PCL) —
  ↓
  (contains overall functionality of process)

Advantages

- It counters the third limitation (Big Bang) of waterfall model & combine the benefits of both Prototype & waterfall model.
- The Basic Idea is software should be developed in increments.
- A PCL is created that contains of all the tasks that must be performed to obtain final implementation.

Page No.
Date

- Redesign of the system is only allowed in initial phases of software designing.

Note →
If we change something at later stages, it gets more costly like waterfall Model.

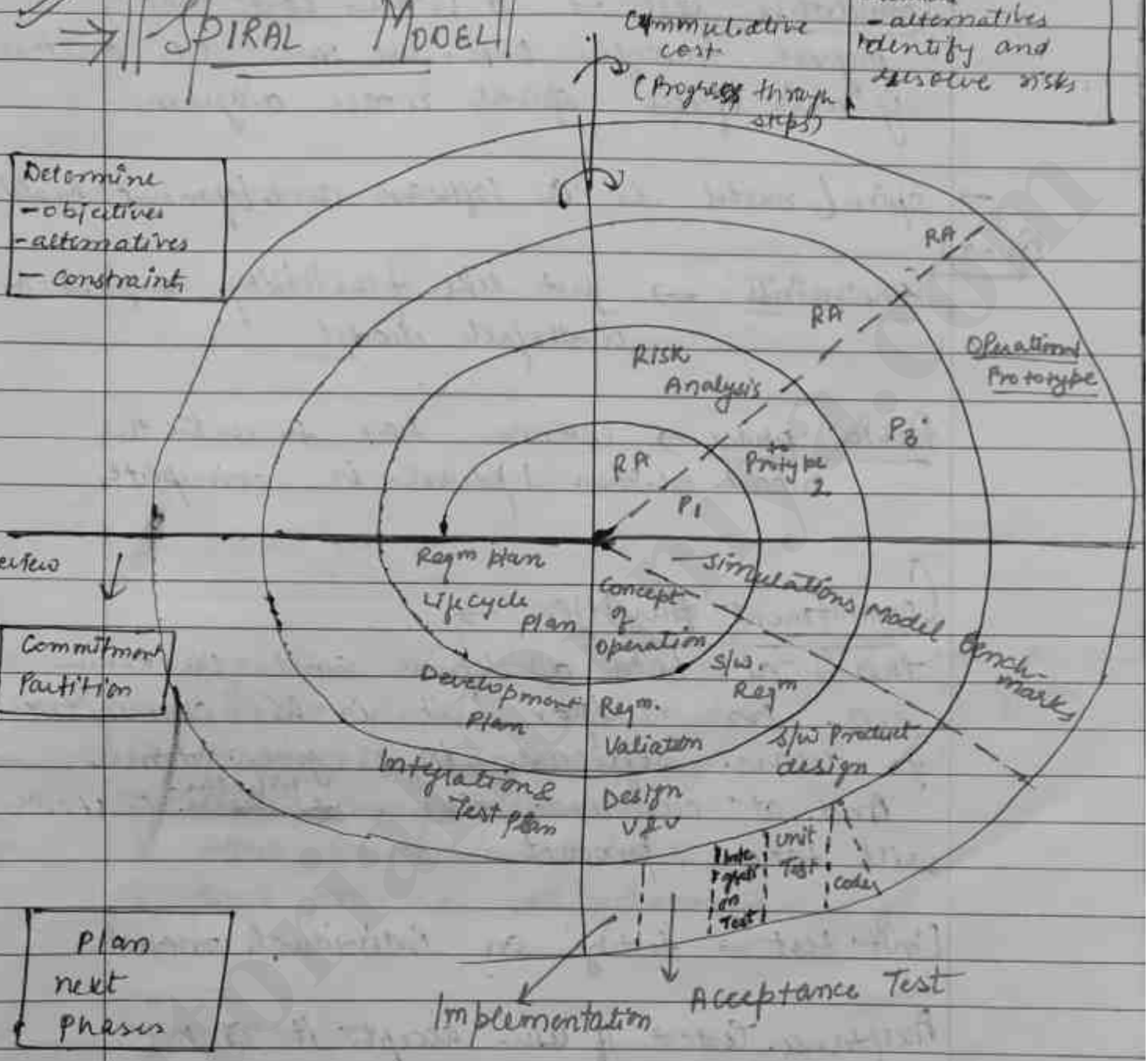
Q⇒ Difference b/w Verification & Validation in S.E

Verification	Validation
1) → Is the process of evaluating software to determine whether the products of a given development phase satisfies the condition imposed at the start of that phase.	1) The process of evaluating software in beginning, during or at the end of the development process to determine whether the complete software meets the Customer's expectation and requirements.
2) is a static practice of verifying documents design, code and programs.	2) is a dynamic mechanism of validating the actual product. (and testing)
3) It generally doesn't involve executing the code.	3) It always involve executing the code.

Verification	Validation
4) It uses methods like inspection, reviews, walk throughs and desk checking etc.	4) uses Methods like functional and str testing.
5) is a low level exercise.	5) It is a high level exercise.
6) is to check whether the software confirms to specifications.	6) is to check whether software meets custom expectation & requirem
7) is used to built high quality software, which is well engineered & error free.	7) Is to make the end product effective

Page No.
Date

⇒ SPIRAL MODEL



We can even increase these spirals acc. to requirements.

Page No.
Date

Comulative cost → It is the cost that progress through steps. we have CC on the y-axis of the spiral model diagram.

→ spiral model is a software development model

Quad 1

Constraints → Just like feasibility report in waterfall model.

Review phase → customer has to review the model when 1 phase is complete.

Commitment partition →

This is at both developer and customer end. For developer, it is his commitment to deliver software in a given time. And at customer side, whether customer will take product.

Unit test → testing on individual model

Acceptance test → If user accepts it or not

Page No.
Date

⇒ STRENGTHS

1) Regular and quick delivery. → (increases customer satisfaction)
2) Reduced Risks.
3) Accomodate changes.
4) Allow user feedback.
5) Allow reasonable exit points.
6) ~~Produ~~ prioritize requirements.

WEAKNESSES

1) Each iteration can have planning overhead
2) Cost may increase as work done in 1 iteration may have to be undone later.
3) System architecture and structure may suffer as frequent changes are made.

Page No.
Date

Both Iterative Enhancement & spiral models are Iterative models.

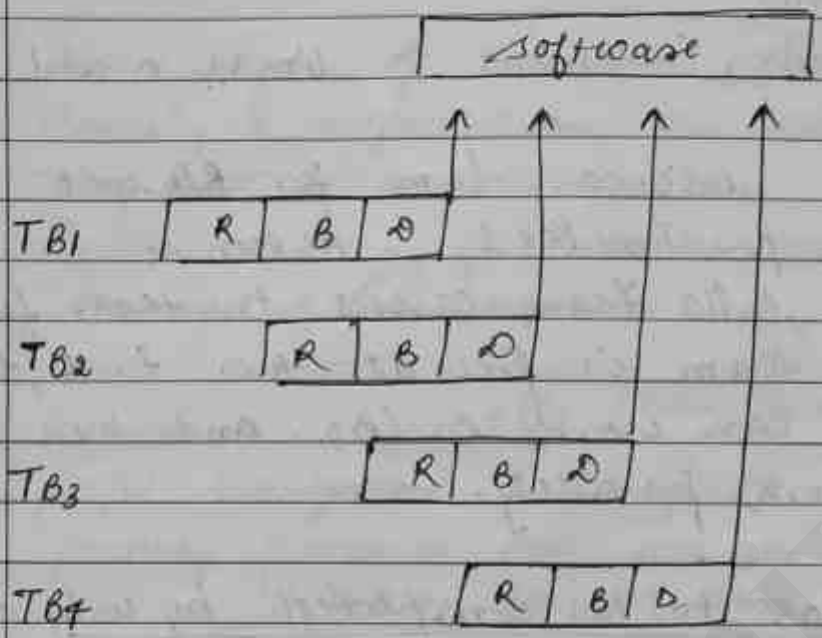
Applications

1. For business, when time is of essence.
2. Where risk of long project can not be taken.
3. Where reqm are not known and will be available only with time.

Unit-1

Page No.
Date

⇒ Time-Boxing Model { Pipeline Model }



(Time Box)

Requirement Team	RA for TB1	RA for TB2	RA for TB3	RA for TB4
Build Team	Build for TB1	Build for TB2	Build for TB3	Build for TB4
Deploy Team	Deploy TB1	Deploy TB2	Deploy TB3	Deploy TB4

R → Requirement
B → Build
D → Deployment

RA → Requirement Analysis

Page No.
Date

- parallel Model

→ uses Iterative, increment & linear model

→ There is a dedicated team for RA that ~~starts~~ works for Time Box 1. As soon as req. finishes, Build Team carries the task further. Req. ~~team~~ team is free at this time, thus Req. team can work on TB2. And thus all teams work parallely.

Product has to be completed by using -

1) duration of time box (overall)
2) duration of each phase of time box, for max. utilization.

- To speed up the development, parallelism b/w different iterations can be employed which uses the pipelining of different iterations within a timebox with stages of equal duration.
- The basic unit of development is Time Box where there is a dedicated team for each process in a time box.
- It is divided into 3 process

Requirement → Build → Deploy

Page No.
Date

Strength

1. All strengths of iterative model.
2. Planning & negotiations are easier.
3. Very short delivery cycle
4. Regular delivery of modules with one module at the end of each module.

Weaknesses (as entry and exit point is defined)

1. Project Management is complex.
2. possibly increased cost due to large team size
   ↳ (due to dedicated team for each process)

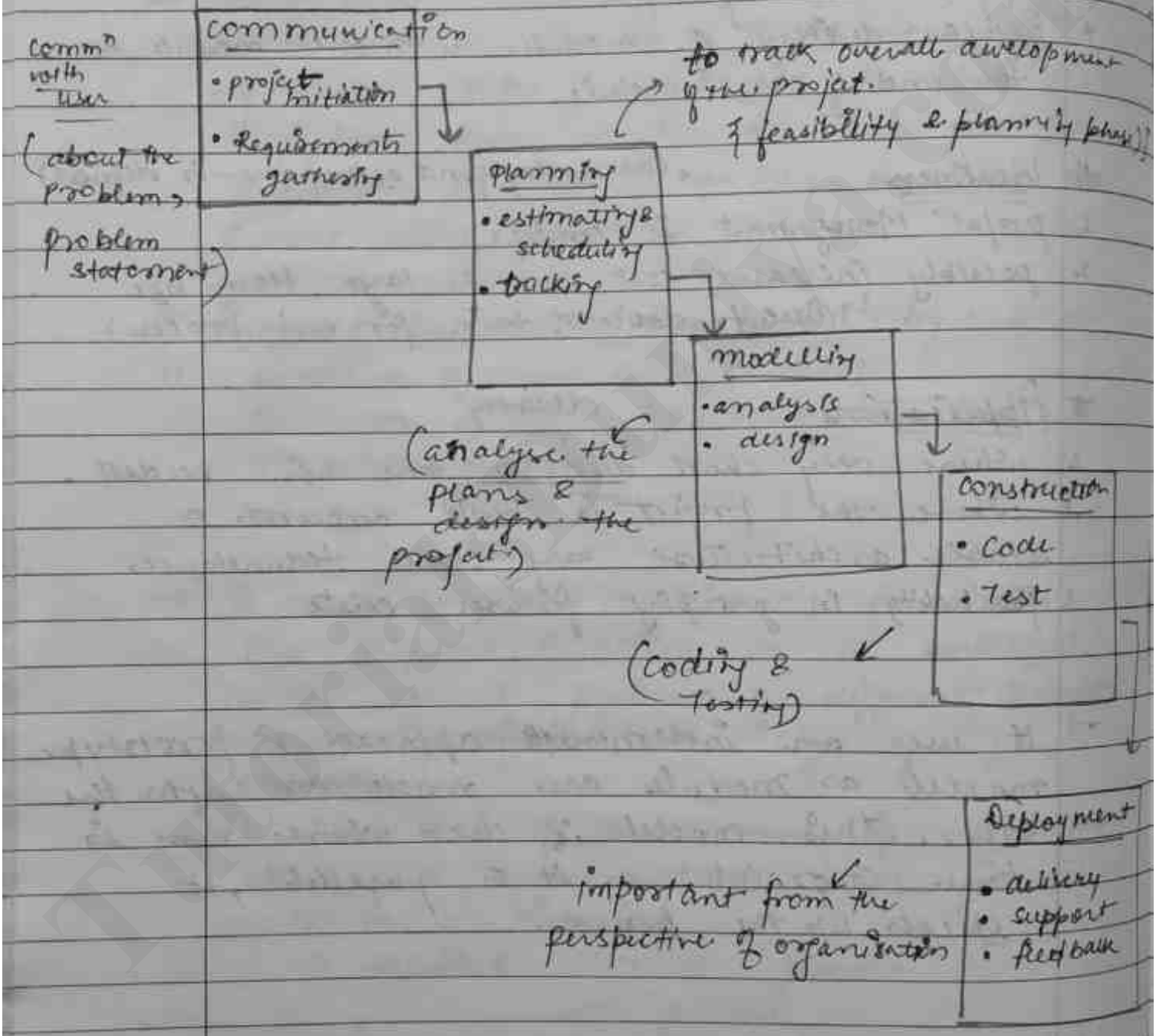
Applications

1. where very short duration (delivery) time is needed.
2. where the project is build around a stable architecture using stable technologies
3. flexibility in grouping features exists.

→ It uses an incremental approach & prototype model as modules are made one after the other. This model is used where there is time constraints as it is parallel, it speeds up the process.

Page No.
Date

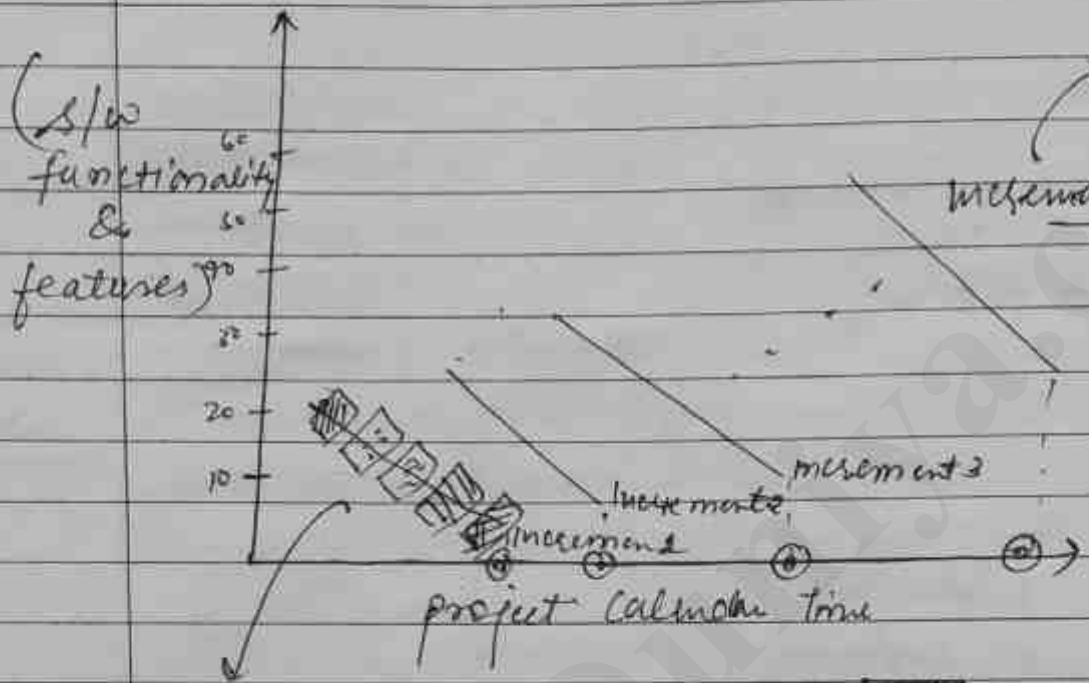
Waterfall Model {(according to Pressman)}



Page No.
Date

Iterative enhancement

⇒ Incremental process Model

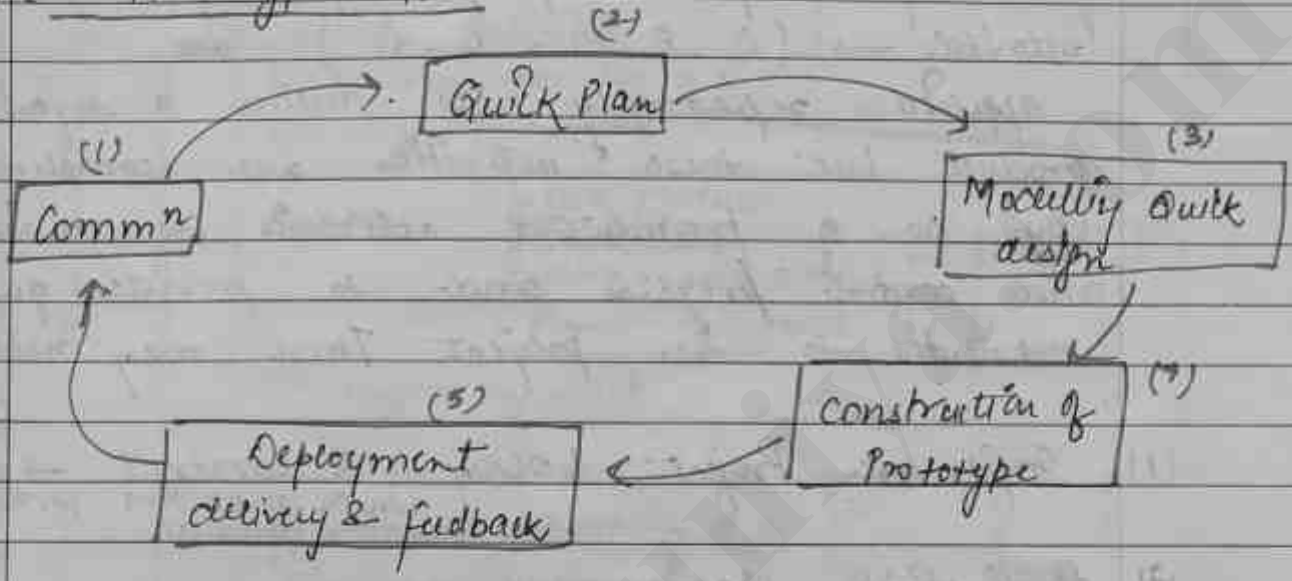


Here keep on increasing the features, more time is required to complete the project

Page No.
Date

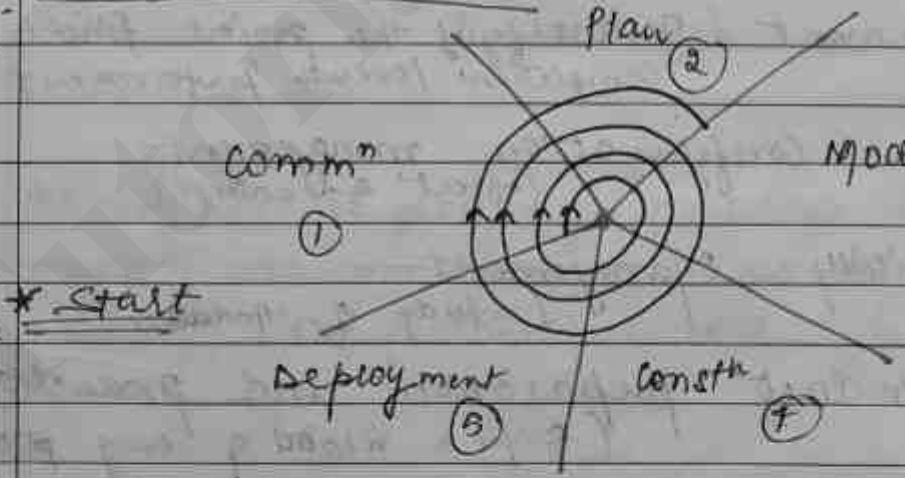
⇒ Evolutionary Process Model

(1) Prototype Model



It is a cyclic product until a product is delivered.

(2) Spiral Model



Page No.
Date

⇒ Defining a framework activity

for major Software projects, the five umbrella activities → (C, P, M, C, D) are applied repeatedly to deliver a software product but these activities are complimented by no. of framework activities to manage and control progress and to provide quality attributes to the project. These may include

(1) Software Project tracking & control → team which monitors progress of a product.

(2) ~~Risk~~ Risk Management. (team that access risks)

(3) Software Quality Assurance (adding on quality feature)

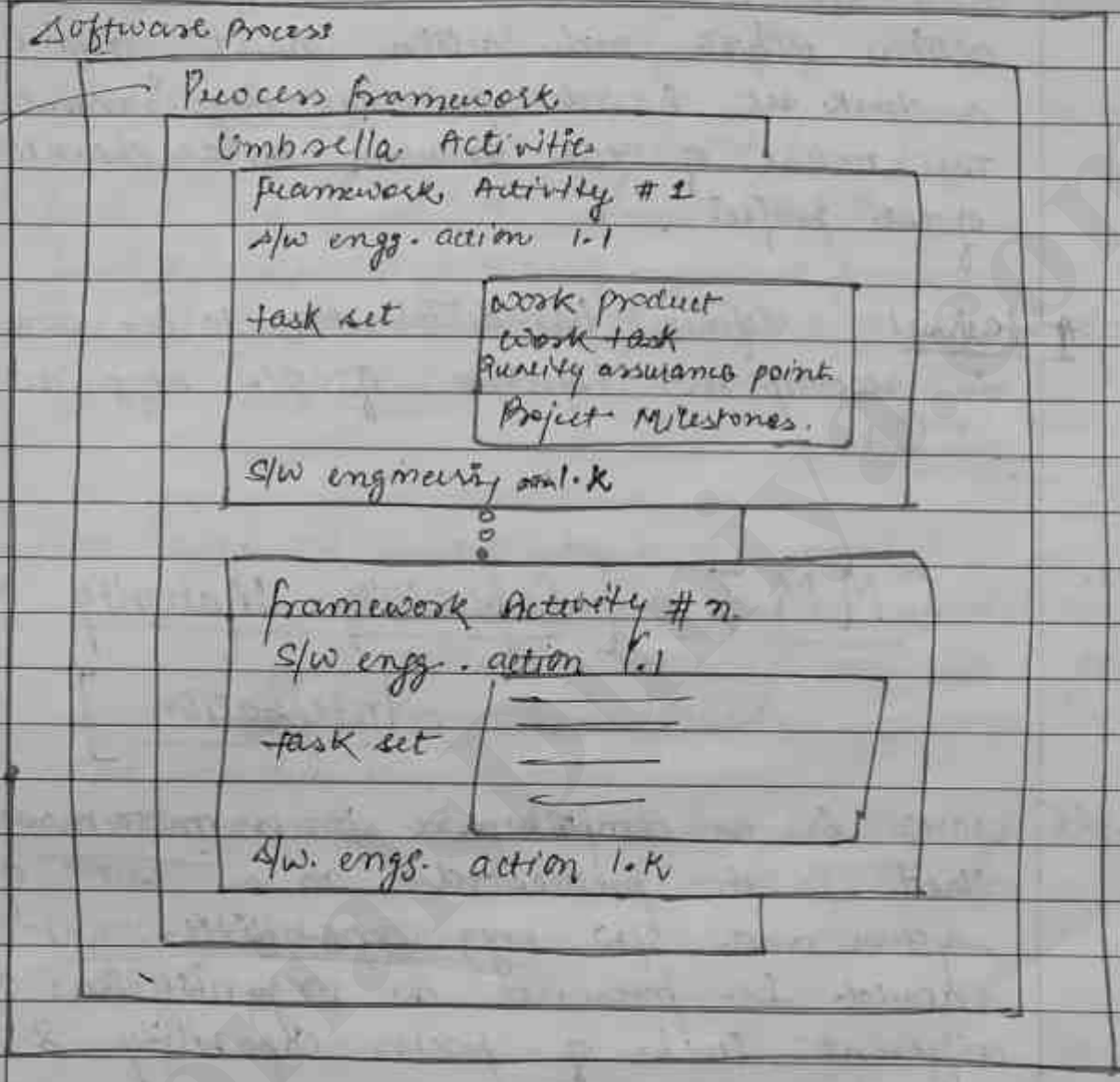
(4) Technical reviews

(5) Measurement (quantifying the project, finds completion / overall performance)

(6) Software Configuration management (effect of change)

(7) Reusability Management. (Study of Modules)

(8) Work product preparation and production. (keeps a record of every outcome of each phase)



→ Activity is a group of multiple actions - In an activity we can have k actions. a Activity is further divided into tasks set. max. of 4 tasks

→ a task set is smallest unit of Execution.

Page No.
Date

⇒ Each activity is divided into k number of actions where each action must define a task set based upon the requirement given The nature of the problem and characteristic of the project.

⇒ A task set defines the actual work to be done to accomplish objectives of S/W engg. actions.

CMMI [Capability Maturity Model Integration]

- CMMI is a comprehensive process meta model that is predicated on a set of system and S/W engg. capabilities that should be present as organisations reach different levels of process capability & maturity.

→ It represents a meta model in 2 different ways.

1st Model

(i) As a continous model which describes a person in two dimensions →
Capability level & process area.

Each process area, is formally assessed agaist

specific goals and practices and is rated according to following capability levels.

capability levels

Level 0 (incomplete) → obj. at entry pt are incomplete.
Level 1 (performed) → executed acc. to.
Level 2 (Managed) → defined policies (quality)
Level 3 (defined) → use of engineered practices.
Level 4 (quantitatively Managed) → —
Level 5 (optimised) → Optimizing the statistics need to meet the customers need.

(usiss)

Meta Model → model about the model
It is some specific practices / principles that must be followed for other models

HE study is known as CMMI. (integration of all other models)

• CMMI Level Diagram

⇒

Capability level

5 (well defined)
4 (Managed)
3 (performed only)
2 (quantitatively Managed)
1 (Managed)
0

PP → project Planning.
REQM → Requirement Manage-ment
MA → Measurement & Analysis.
CM → configuration management
PPQA → process & product quality assurance.

PP | REQM | MA | CM | PPQA → others

Process area

CM → Changes are completely and accurately managed..

book Page → (797 - 801)

→ The CMMI defines each process area in terms of specific goals and specific practices required to achieve these goals. Specific goals establish the characteristics that must exist if the activities adopted by a process area is effective and Specific practices refine a goal into set of process related activities.

for example,

project planning is one of the process area defined by CMMI and the Specific goals (SG) and associated specific practices (SP) defined for a project planning are

→ SG 1 → Establish (Estimates) (Estimates scope of the project)
- SP 1.1 - 1
- SP 1.2 - 1 (establish estimate of work product & task attr)
- SP 1.3 - 1 → defined project life cycle
- SP 1.4 - 1 → determined estimate of effort & cost

Page No.
Date

→ SG 2 – develop a project plan
- SP 2.1 – 1 establish the budget & schedule
- SP 2.2 – 1 identify project risk.
- SP 2.3 – 1 planned for data management
- SP 2.4 – 1 plan for project resources
- SP 2.5 – 1 plan for needed knowledge & skills
- SP 2.6 – 1 plan for stakeholder involvement
- SP 2.7 – 1 establish the project plan.

→ SG 3 – obtain commitment to the plan.
- SP 3.1 – 1 review plans that affect the project
- SP 3.2 – 1 reconcile work and resource level
- SP 3.3 obtain plan commitment.

⇒ In addition to SG and SP, CMMI also defines generic goals & practices for each process area. Each of these generic goals correspond to capability levels. In order to achieve a particular level, the generic goal & practice corresponding to that level must be achieved.

GP (Generic Practice)

→ GG 1 (generic goal one.) – achieve specific goals.
- G.P. 1.1 – perform basic practices.

→ GG 2 – institutionalize a managed process.
- GP 2.1 – establish an organisational policy.
- GP 2.2 – Plan the process
- GP 2.3 – Provide resources

- GP 2.4 - assign responsibility.
- GP 2.5 - train persons.
- GP 2.6 - manage configuration
- GP 2.7 - Identify and involve relevant stakeholders
- GP 2.8 - Monitor and control the Process.
- GP 2.9 - objectively evaluate adherence
- GP 2.10 - review status with higher level management.

→ GG-3 - Institutionalize a defined process
- GP 3.1 - establish a defined process.
- GP 3.2 - collect improvement Information.

→ GG-4 - Institutionalize a quantitatively managed process.
- GP 4.1 - establish quantitatively objectives for the process.
- GP 4.2 - stabilize sub process performance

→ GG-5 - Institutionalize & optimizing process.
- GP 5.1 - ensure continuous process improvement
- GP 5.2 - correct root causes of problems.

Page No.
Date

Q3

(ii) The Staged Model

2nd Model

defines the same process area, goals & practices with a difference that it defines 5 maturity levels rather than 5 Capability level. To achieve a maturity level, the specific goals and practices associated with a given process area must be achieved. The relationship b/w maturity level & process area may be given as →

Level	focus	Process area
• Optimising (last level)	Continuous process improvement	1) Organizational innovation & deployment 2) Causal analysis & resolution.
• Quantitatively Managed	quantitative management	1) organizational process performance 2) quantitative project Management
• Defined	process standardisation	1) Req^m development 2) technical sol^n 3) Product Integration 4) Verification 5) Validation 6) Organizational process focus. 7) Organisational process definition. 8) Organisational training.

Level	focus	Process Area
		9) Integrated project management 10) Integrated Supplier management 11) Risk management. 12) decision analysis & resolution 13) Organisation env. for integration & 14) Integrated learning
• Managed	Basic project Management	a) Req. management b) project planning c) project monitoring & control d) Supplier agreement management e) Measurement & Analysis f) Procss & product quality assurance g) Configuration manage.
• Performed	basic project development	a) CPMCD → umbrella activities b) project development & deployment

18-01-18

Page No.
Date

⇒ Requirement Engineering

1) It is a condition or capability needed by a user to solve a problem or achieve an objective.

2) It is a condition or a capability that must be met or processed by a system to satisfy a contract, standard, specification or other formally imposed document.

Need for SRS → (Software req → Specification)

(i) An SRS establishes the basis for agreement b/w client and the supplier on what the software product will do.

(ii) It provides a reference for validation of the final product.

(iii) A high quality SRS is a pre-requisite for high quality software.

(iv) A high quality SRS reduces the development cost.

Phase	Cost (person-hrs)
Reqm	2
design	5
coding	15
Testing	50
Maintenance	150

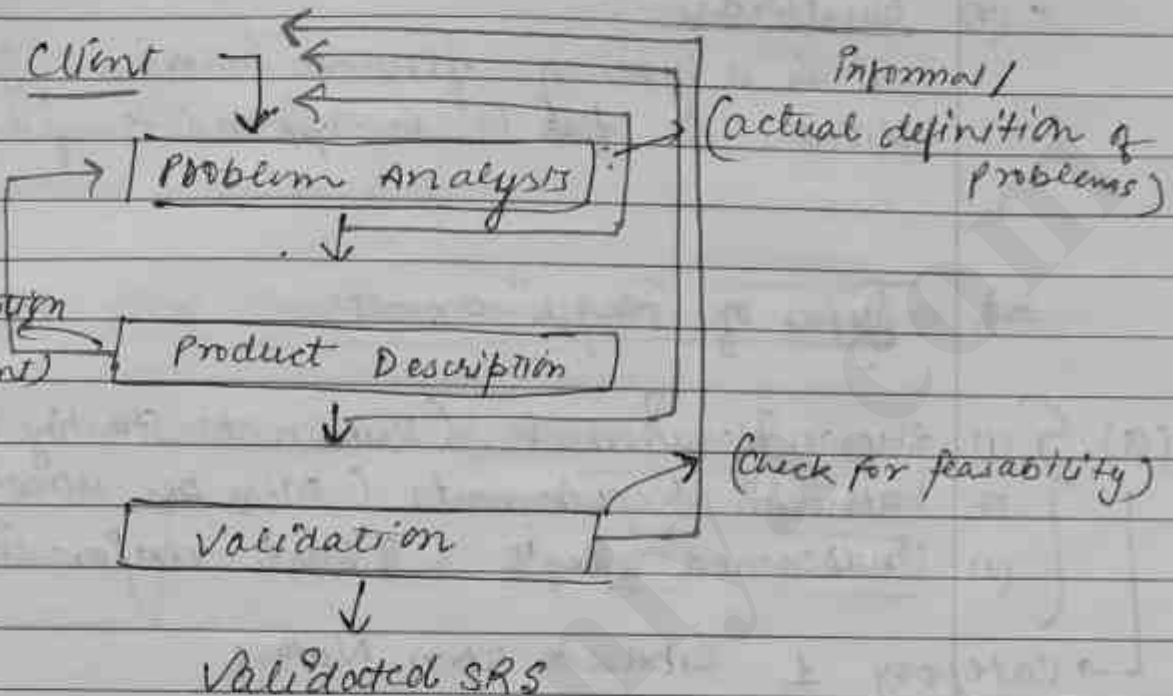
(1) eg → If we have 50 new reqm only at reqm phase
∴ cost = 50 × 2 = 100 person-hrs

(2) • If new reqms are not defined at Reqm phase & instead at other levels → 65% design, 2% coding, 36% at testing & 3% at maintenance
then, (65% of 50 * 5) + (2% of 50 * 15) + (36% of 50 * 50) (+ 3% of 50 * 150)
= 1152.5 person hrs

Introducing changes at later stages can result in more increased no of person-hrs. as compared to changes in only reqm phase as cost at later stages is increasing exponentially.

Page No.
Date

Q:⇒ Requirement process



⇒ This is not a linear process. It is a cyclic process

Techniques for req^m Elicitation

- (1) **Interview**
  To gather / analyse info. from the client.
- (2) **Brainstorming**
  It is a group process / discussion where there are teams at developer and client end.
  Reqm can be defined in a better way. One Problem / req^m can have 'n' no. of scopes & n no. of clients.
- (3) **fast** { facilitated application specification Technique }

Page No.
Date

• (7) Questionare
It is a type of formal interview given to client & he is supposed to fill that questionare

⇒ Type of Requirement

(a) (1) Known Requirements (known at starting)
(2) Unknown Requirements (may be discovered)
(3) Undreamed Req^m (have major impact)
→ category 1 → based upon Nature

(b) Category 2 {based on functionality/feature}
(1) functional Req^m
(Stated already that must be included)

(2) Non functional Req^m
additional qualities to be included in the proje
- availability, testing, usability, maintainability.

(c) Category 3
1) User Req^m & (2) System Req^m.
Specified by end user & client - inherent req^m that exists system.

(d) Category 4
1) Interface requirement.
h/w — s/w — interface API
- (actual view of project)

Page No.
Date

⇒ Problem Analysis

→ (1) Informal approach
- informal discussion
- meeting with client

→ (2) Data-flow Modelling → pictorial representation.

- (a) DFD's (Data flow Diagram)

External Entity

Process

Data store

Data flow

→ Level 0 : contextual Diagram

In level 1 → we elaborate the processes in more detailed manner.

Page No.
Date

22-01-18

Software Requirement Specifications (SRS)

definition of SRS

The final output of Reqm Engineering is software Requirement specification Document (SRS)

It is a formal document with a predefined template which converts the outcome of analysis Space into a well formed specified document.

Characteristics of SRS

To properly satisfy the basic goals, SRS should have certain properties. Some desirable characteristics of SRS are-

Must read

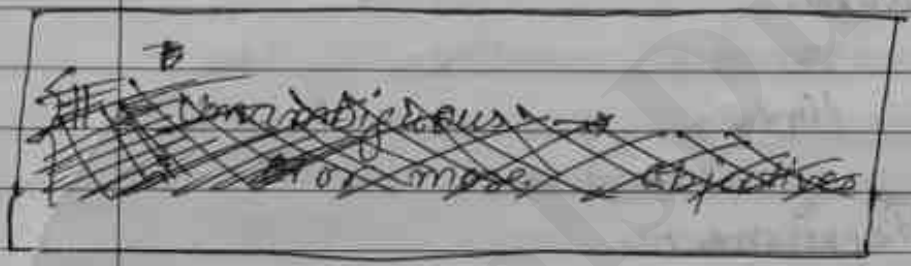
(1) Complete → reqms should be stated initially.
(2) Correct
(3) Unambiguous → conflicts b/w reqms should not be there
(4) Verifiable → Cost effective Measure must exist.
(5) Consistent → (reqm must be consistent with respect to each other)
(6) Ranked for Importance or Stability. → (ranking) Categorize the reqms on the basis of their importance & stability)
(7) Modifiable → (SRS should be changeable provided it ensures all other properties properly)
(8) Traceable → (Should be mapped with some features)

What is correct SRS.

Correct

- When reqms are valid & possible (feasible)
- functionalities should be properly defined.
- Whatever is desired at end or by the software must be properly defined in SRS.

Note: Maintaining complete is almost impossible or difficult because we need to write down all the reqms at the start of the project that are going to be used later for project.



COMPONENTS OF SRS → Design Issues of SRS

SRS should specify some basic system properties and the Issues that may exist while specifying SRS are –

1) Functional Requirements.
2) Performance Requirement
3) External Interface Reqm.
4) Designed Constraints.

Page No.
Date

→ phase after the finalisation of SRS -

Software Project Planning

After the finalisation of SRS, a process to estimate size, cost and development time of the project is carried out & this process is called Software Project Planning.

It includes following activities.

1) Size ~~Acti~~ Estimation : LOC overall size of proj
2) Cost Estimation.
3) Development time.
4) Resource Requirement
5) Project scheduling

LOC → Lines of Code or (KLOC) (Kilo)

Page No.
Date

→ Size Esthmation

• 1) Lines of code

a line of code is any line of program that is not a comment or blank line regardless of no. of statements or fragments of statements on the line.

for ex.→

```
// factorial function.
(1) int facto (int n)
(2) {
(3)     if (n==1)
(4)         return 1;
(5)     else
(6)         return n*facto(n-1);
(7) } // main function.
(8) int main()
(9) {
(10)    int num;
(11)    cout << "enter a number";
(12)    cin >> num;
(13)    cout << "factorial is" << facto(num);
(14)    return 0;
(15) }
    // end of Program.
```

This program have LOC count of 15.

↓ except comments & blank lines

(Numerical)

Function point Analysis (FPA)

this technique is used for size measurement which decomposes a system into 5 functional units.

(1) Input (EI) → information entering into the system.

(2) Output (EO) → Information leaving the system. (Maybe an input or an output)

(3) Enquiries (EQ) → Request for instant access to information. (Combination of input & output)

(4) Internal logical file → (ILF) Inf^n held within the system. (Resources maintained in the system)

(5) External Interface file (EIF) → Inf^n held by other system that is used by the system being analysed. (past & other System used by our System)

⇒ UFP → Unadjusted FP

Function point

Page No.
Date

Novell Corl

[redacted]

	functional unit	weighing factors. Low	Avg	High
✓	EI	3	4	6
To be learned	EO	4	5	7
	EQ	3	4	6
	ILF	7	10	15
	EIF	5	7	10

Q→ Consider a project with following funtional units (1) no. of user Input = 50
(2) no. of user output = 40
(3) no. of user enquiries = 35.
(4) no. of user files = 6
(5) no. of External Interface = 4.
Compute unadjust function point, consider EI, EO, [redacted] as low, EQ as average & ILF, EIF as high.

Sol"

$$\begin{aligned} &50 * 3 \\ &+ 40 * 4 \} \text{low} \\ &+ 35 * 4 \} \text{Avg} \\ &+ 6 * 15 \\ &+ 4 * 10 \} \text{high} \end{aligned} = \begin{aligned} &150 \\ &+160 \\ &+140 \\ &+090 \\ &+040 \end{aligned}$$

$$580 \rightarrow UFP$$

25-01-18

⇒

Q→2 An application has a following → 10 low EI, 12 high EO, 20 low ILF, 15 high EIF & 12 avg EQ. Calculate UFP.

$3 \times 10 + 7 \times 12 + 20 \times 7 + 15 \times 10 + 12 \times 7$

$= 30 + 84 + 140 + 150 + 78$

$= 482$

Q→3 EI → 10 with low = 10 X 3 = 30
15 with avg = 15 X 7 = 60
17 with high = 17 X 6 = 102
= 192

EO → L = 6 = 24
→ A = 6 = 0
→ H = 13 = 91

EQ → L = 3 = 9
A = 7 = 16
H = 2 = 12
→ 429

ILF → L = 0 = 0
A = 2 = 20
H = 1 = 15

IEF → L = 9 = 45
A = 0 = 0
H = 0 = 0

Page No.
Date

General formula

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} Z_{ij} * W_{ij}$$

W_{ij} → weighing factor

Z_{ij} → Total entities available

Z is count of no. of functional unit of type I → J

W_{ij} is the associated way correspond to ith row & jth column.

⇒ Function point is calculated using the following formula

$$FP = UFP * CAF$$

CAF → Complexity adjustment factor.

$$CAF = \left(0.65 + 0.01 \sum_{i=1}^{14} f_i\right)$$

$\forall i = 1$ to 14.

where f_i is rated based upon degree of influence ranging from 0 to 5. with following set of 14 Questions.

Given an impact factor & corresponding Questions we may find some impact on particular question.

	0	No influence
may or may not —	1	Incidental
	2	Moduate
	3	Average
	4	Significant
very essential impact i.e. impact is very high	5	Essential

} Imapct factor

Questions with respect to these 6 Impact factors are

Q→1 Does the system require reliable backup & recovery ?

Q-2 Is data communication required. ?

Q-3 Are these distributed processing Sys

Q4 Is performance critical. ?

Q5 will the system run in an existing heavily utilized operational environment

Q6 Does the system require on line data entry ?

Q7 Does the online data entry require input transaction to be build over multiple screen & Operation ?

Page No.
Date

Q8 Are the master files updated online ?

Q9 Is the Input, output, files or enquiries complex?

Q10 Is the Internal Processing complex?

Q11 Is the code design to be reusable. ?

Q12 Are the conversion & installation included in the design ?

Q13 Is the system designed for multiple installations in different organizations?

Q14 Is the application design to facilitate to change & ease of use by the user ?

Example - 1

Assume all complexity factor as average compute function point.

UFP = 580 (from 1st example)

That means (3) from Impact factor.

CAF = 0.65 + 0.01 (14 X 3)

= 0.65 + 0.01 X 42

= 1.07

Page No.
Date

Now,

$$F \cdot P = UFP * CAF$$
$$= 580 * 1.07$$
$$\boxed{FP = 620.6}$$

Example - 2

Given CAF = 1.10

compute UFP & FP.

⇓

452 (from Q-2)

$$FP = UFP * CAF$$
$$= 452 * 1.10$$
$$= 497.2$$

Example 3

In addition to Q-3, the system requires ① significant data communication ② performance is very critical. ③ designed code may be moderately reusable. ④ system is not designed for multiple installation in different organization. ⑤ other CAF are average.

Page No.
Date

CAF = 0.65 + [10*3 + 4 + 5 + 2 + 0] * 0.01

= 0.65 + 0.01 * 71

= 1.06

FP = 449.44

⇒ COCOMO - I

(Constructive cost model)

Table 1 For screen no. of sources of data tables

no. of views contained	total ≤ 4 Server ≤ 2 Client ≤ 3	total < 8 Server 2-3 client 3-5	Server > 3 total ≥ 8 client > 5
< 3	Simple	Simple	Medium
3-7	simple	Medium	Difficult
> 8	Medium	Difficult	Difficult

For reports

Table 2 no. of sources of data tables

no. of Sections Contained	total < 4 Server < 2 client < 3	total < 8 Server 2-3 client 3-5	Server > 3 total > 8 client > 5
0 or 1	simple	simple	Medium
2 or 3	Simple	Medium	Difficult
4 +	Medium	Difficult	Difficult

table 3

Object type	WI Simple	Medium	Difficult
Screen	1	2	3
report	2	5.	8.
3GL component	—	—	10

8→ Consider a database project with following chrs →

1) The application has 4 screens with 4 views each and 7 data tables for 3 servers & 4 clients.
2) The application may generate 8 reports of 6 sections for 8 database tables, 3 servers & 5 clients.
3) There is 10% reuse of object points.
4) The developers experience & capability in similar environments is low & maturity of environment in terms of capability is also low calculate object point count, new object point & effort to develop this project.

Answer

→ 4 screens – 4 views each / 7 tables ← 3 servers, 4 clients

↓

for this, complexity is medium (my acc. to table.

Page No.
Date

2 reports - 6 sections / 8 tables < 2 servers, 5 clients.

↓

complexity is 0 (Difficult)

(1) To calculate Object point count

OP = no. of screen * weight + no. of report * weight.

= 7 * 2 + 2 * 8 + 0 * 10

= 29

(Complexity of difficult level from table)

we took (0*10) as there was nothing mentioned about the 3GL factor in the question.

NOTE
- Screen represents view of project (in terms of webpages)
- Reports represents tables to be used in project.
- 3GL components are used for graphic components.

defined during project planning

→ to normalize project point

⇒ NOP = O.P × (100 − % reuse) / 100

(New Object Point)

= 24 × (100 − 10) / 100 = 24 × 90 / 100 = 21.6

Effort = NOP / PROD — LOCOMO I

Developer's Maturity / cap.

nominal

Table 1

Developers Maturity/cap	very low	low	nominal	high	very high
Env. maturity/cap.	very low	low	nominal	high	very high
PROD	4	7	13	25	50

Effort E = NOP / PROD = 21.6 / 7

E = 3.086 P.M

↳ (Person Month)

Standard unit for effort

↳ person month

Note → If one is low and one is high, we consider low one as prod. ses and thus effort ies. Thus, always consider the lower cost.

Page No.
Date

Ques:-

⇒ Cocomo is software estimation model and stands for Constructive Cost Model.

It is one of the most widely used cost estimation model which addresses the following areas.

- (1) Application Composition Model.
  It is used during early stages of software engineering when prototyping of user interfaces, consideration of various interactions and assessment of performance and evaluation of technology is considered.

- (2) Early design Stage Model
  It may be used once requirements have been stabilized and basic software architecture is established.
  (SRS has been validated & submitted)

- (3) Post architecture Stage Model
  It is used during construction of the software.

COCOMO - II Model requires sizing information such as object point, function point & lines of code.
It uses the following complexity table to associate weight to calculated object points based upon no of screens, reports & 3gl components (Third Table)

Once complexity is determined, object point is calculated by multiplying the number of object types with their corresponding weights. & to normalise the calculated object point % of reuse is estimated and object point is adjusted to new object point (NOP) with the following relation →

$$NOP = \frac{(OP * (100 - \% reuse))}{100}$$

to derive an estimate of effort based on the computed NOP value, a productivity rate must be derived, which is the ratio of developer's experience and env. maturity / capability rated on a scale from very low to very high as given in table 4

$$\boxed{Effort(E) = \frac{NOP}{PROD}}$$

$$\begin{cases} Effort(E) = 1.4(L)^{0.93} \\ Documentation(DOC) = 30.7(L)^{0.90} \\ Duration(D) = 4.6(L)^{0.26} \end{cases}$$

Page No.
Date

Basic COCOMO

Organic | semi detached | embedded.

Effort $E = a_b (KLOC)^{b_b}$

Development time $D = c_b (E)^{d_b}$

a_b, b_b, c_b, d_b } 4 factors

S/W project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
semi detached	3.0	1.12	2.5	0.35
embedded	3.6	1.20	2.5	0.32

	Organic	Semi detached	Embedded
•	2-50 KLOC	• 50-300 KLOC	• 300 + KLOC
Innovative factor	little	• medium	• significant
•	experienced developer	• Average experience	• real-time sys.
			• no previous experience
•	familiar with environment		• new environment.

Page No.
Date

Q→1

⇒ A project cost estimated to be 900 KLoC. Calculate the effort and development time.

$$E = a_b (KLOC)^{b_b}$$

$$= 3.6 \times (900)^{1.20}$$

$$= 4772.8137 \text{ pm}$$

$$D = C_b (E)^{d_b}$$

$$= 2.5 (E)^{0.32}$$

$$= 2.5 (4772.8137)^{0.32}$$

$$= 38.07 \text{ months}$$

Q-2 A project size of 200 KLoC, calculate the effort & D=?, avg staff size, & prod of the project.

$$\text{Average Staff Size} = \frac{E}{D} \text{ persons}$$

$$\text{PROD} = \frac{KLOC}{E}$$

(no. of lines of code made by person month pm a given time)

Page No.
Date

$E = 3 \times (200)^{1.12} = 1133.12$ p-m

$D = 2.5 \times (E)^{0.35}$

$= 2.5 \times (1133.12)^{0.35}$

$= 29.3$ months

$SS = \frac{E}{D} = \frac{1133.12}{29.30} = 38.67$ persons ≈ 39

$PROD = \frac{200}{1138.12} = 0.17650$ KLOC/pm

$= 176.5$ LOC/pm

⇒ Software Eqn Model

It is a dynamic multivariable model that assumes a specific distribution of effort for software development. It is derived from productivity data collected over 4000 software projects. Based upon these data, the estimation model is given by the eqn.

$$E = \frac{LOC \times B^{0.333}}{P^3} \times \frac{1}{t^4}$$

where E = effort in p-m, t = project duration in months.

B = special skill factor.

Types of project	KLOC	B
small	5-15	0.16
large	>70	0.39

P is the productivity parameter that reflects overall process maturity & management practices & is given as.

Type of project	P
• real time embedded s/w.	2000
• telecomm & sys/ s/w	10000
• business system application	28000.

⇒ To simplify the estimation process, the software eqn model is converted to another estimation model which defines minimum development time & effort as →

$$t_{min} = 8.14 \frac{LOC}{P^{0.43}} * \text{months}, \quad t_{min} \geq 6$$

$$E = 180 B t^3 \quad p\text{-m}, \quad E \geq 20.$$

 Sadella Kotta

Page No.
Date

Q→ For a CAD software the recommended value of P is 12000 with an estimate of 33.2 KLOC find min time & effort required for this project. Given the value of B is 0.28.

$t = ?$

$$t_{min} = \frac{8.14 * 33.2 * 1000}{(12000)^{0.43}}$$

12.6

$$= 4761.17997$$

56.760

Page No.
Date

PROJECT SCHEDULING

Scheduling of a project is necessary to provide timeline & to apply checkpoints to evaluate progress of any software project.

Some common techniques for project scheduling →

1. Gantt chart

is a timeline chart which is developed to fix a schedule for a given project. The horizontal axis is time where each task is assigned its own horizontal band of calendar duration and on the vertical axis, all the activities are listed distinctly and tasks every identified and grouped into categories for a given duration.

The advantages of a gantt chart are

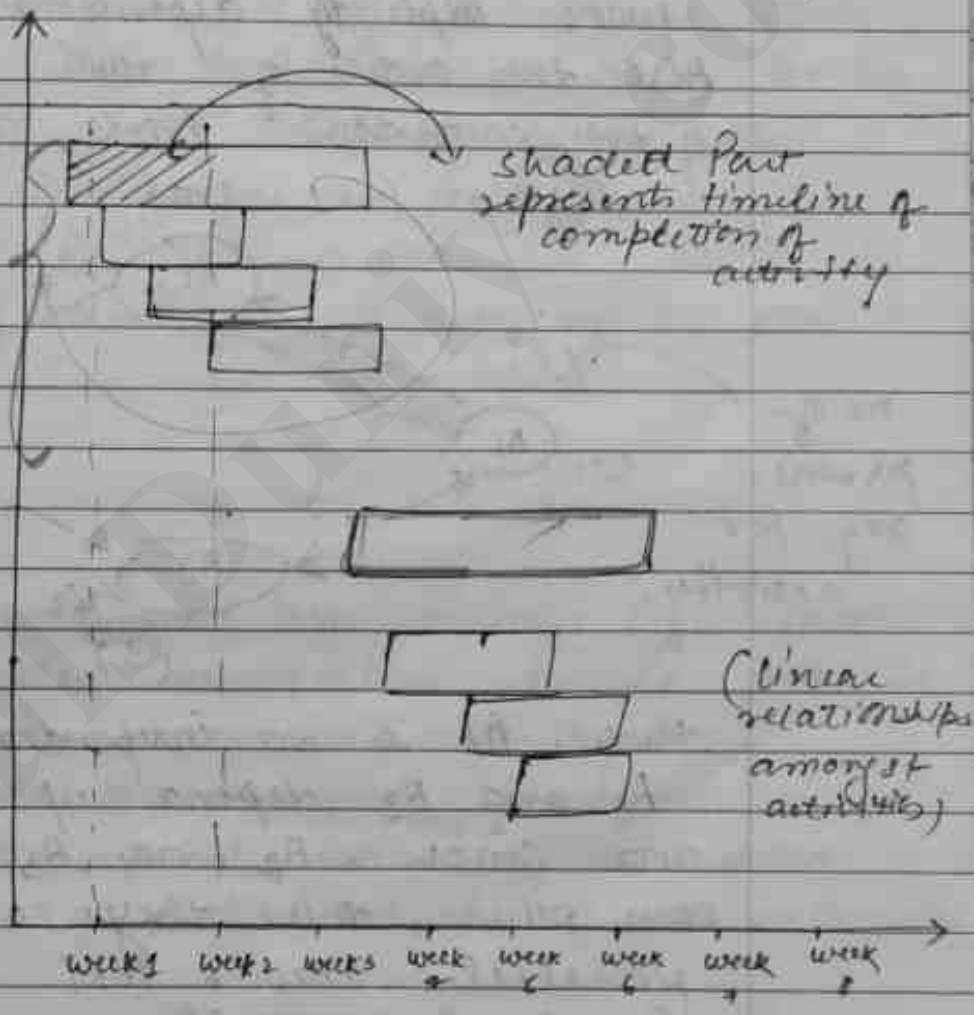
1) The time is explicit & linear
2) All tasks are clearly visible
3) Deadlines are shown.
4) Progress is indicated by filling the task boxes.

It may have following shortcomings

1) task might not be associated with people.
2) Person hours are not indicated.

3) dependencies are not clearly indicated.
4) Critical ~~path~~ paths are not given.

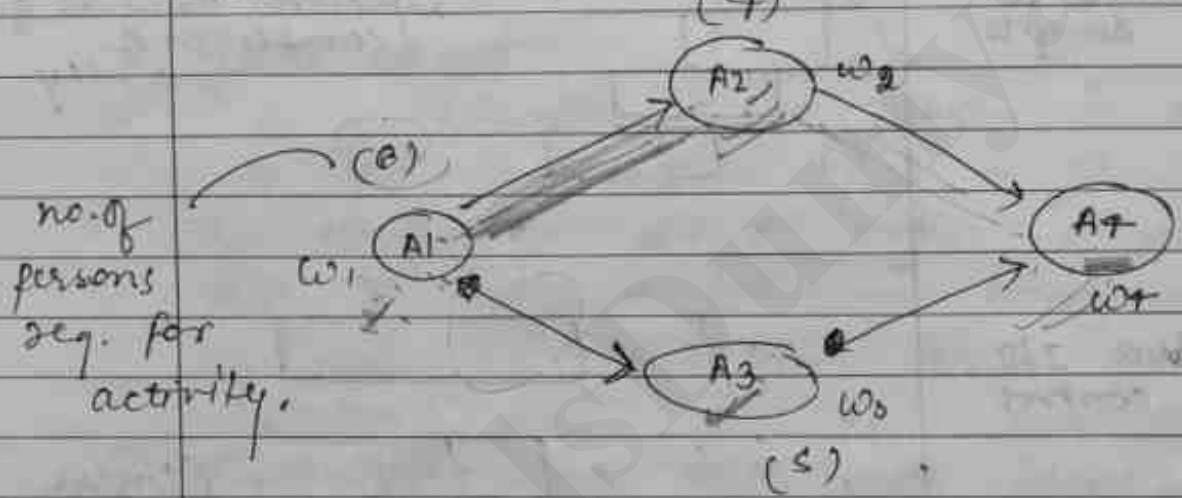
Example



2. PERT

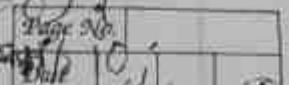
(Program Evaluation & Review technique) in which the shortcoming of gantt chart are resolved.

The Pert uses a collected series of nodes to make explicitly dependencies b/w tasks. Also the order of task is given by flow of the connections from left to right.



Here, A1 is an independent activity but A2 and A3 depend upon outcome of A1 and since, A2 and A3 are independent of each other, thus they can be executed in parallel. Thus, PERT represents both linear & non-linear relationship.

```
reg int = n;
while(n!=0){
rem int = n%10;
rev int = rev int * 10 + rem;
n=n/10;
}
```



3. CPM (Critical Path Method)

is similar to Pert chart but it includes explicit indication of critical path that a sequence of tasks which defines the minimum amount of time required for the project. In other words, it defines all the tasks where a delay will lead to delay for the entire project. CPM share the same strength & weaknesses as Pert.
CPM is crucial where time is critical & delievery of the project must be given on time.

⇒ All the techniques are driven by information already developed in earlier project planning activities such as estimate of effort, decomposition of product function, selection of appropriate process model and identification of task set & its decomposition.

- Interdependencies among the task are defined using work breakdown structures which may be defined for the over-all project or for individual modules.

The objective of project scheduling is to enable a project manager to define work task, establish dependencies, estimate the resource required & developed some graph, chart or table that helps to track & control the progress of s/w project.

Page No.
Date

(WBS)

1) First step is to specify work breakdown structure.
2) assign start & end date & attach human resource & attach other data to each task.

Contingency ⇒ a future event or circumstance which is possible but cannot be predicted with certainty.

5-02-18

Page No.
Date

⇒ Risk Management

Proactive strategy

↓

Identify all the risks before & identify strategies for handling risks.

Drawback → It is not possible to predict all risks.

Reactive strategy.

↓

Easy - approach first allow risk to come then solve, drawback → project failure

⇒ Risk analysis and management are actions that help a software team to understand & manage uncertainty.

A Risk is a potential problem which may or may not occur but its a good practice to identify the risk, assess its probability of occurence, estimate the impact and establish a contigency plan to develop a successfull project. Risk Management is a group of activities that involves →

- (1) Risk identifications.
- (2) Risk Specifications.
- (3) Risk analysis. [ranked for probability & impact]
- (4) Risk Management Plan.

Page No.
Date

⇒ In the context of s/w engineering risk is concerned with future happenings and it may involve some unpredictable change or choice. With respect to future concern, we have to identify what risk might cost the s/w project to change.
With respect to change & choice, the concern is to identify change in customer requirement, development technologies, target environment & methods & tools available and adopted to develop a project.

⇒ Types of Risk Management Strategies

1. Reactive Strategy

This strategy is based on the concept that never take any action about the problem until they appear.
Majority of s/w teams rely on reactive risk strategies which monitors the project for likely risk and resources are set aside to deal with them. More commonly it is observed that software team does nothing about the risk until something goes wrong.

Page No.
Date

Syllabus

2. Pro-active Strategy

A proactive strategy begins long before technical work is initiated. Potential risks are identified, their probability & impact are assessed & they are ranked by their importance. The s/w team establishes the plan for managing the risks.
The primary objective is to avoid risk but bcoz not all the risk can be avoided, or identified the team develops a contingency plan that ~~enable~~ respond in controlled and effective manner to minimise/avoid risk.

Q→

Characteristics of s/w risk.

1. Uncertainity
The first characteristic of s/w is uncertainty. i.e. a risk may or may not happen and there is no 100% probable risk associated with a given project and if they exist they must be considered as constraints during sys analysis phase.

2. Loss
If the risk becomes ~~the~~ a reality, unwanted consequences or losses will occur. Thus, it is required to quantify the level of uncertainity

and degree of loss associated with each risk.

⇒ Effect of S/w Risk:

1. project risk threatened the project plan. ie. if the project risk becomes real, it is likely that project scheduled will slip and time & cost will increase.

2. Technical risk threatened the quality & timeliness of s/w to be produced. ie. if a technical risk becomes a reality, implementation may become difficult or impossible
ability to work

3. Business risk threatened the viability of s/w and it may put the project in danger - some common business risks are:
a) Building a product that noone really wants.
b) Building a product that no longer requires.
(c) Building a product that sales team does not understand how to sell.
(d) Loosing the support or change of focus of people in management.
(e) Loosing budget.

Page No.
Date

Q) Categories of Risk

1. Known Risk
These are the risks that can be uncovered after careful evaluation of project plan, the business & technical environment in which project is developed and other associated resources such as unrealistic ~~delivery~~ delivery date, poor documentation or poor development environment.

2. Predictable Risk { depends on history }
that are identified from past project experience such as staff turnover, poor communication & inadequate effort for a given project.

3. Unpredictable Risk
These are the risks which are extremely difficult to identify in advance.
(Security threat)

13-02-18

⇒ 7 Principles of Risk Management

1. Maintain a global perspective.
2. Take a forward looking view.
3. Encourage open communication.
4. Integrate risk into software process.
5. Emphasize a continuous process.
6. Develop a shared product vision among stakeholders.
7. Encourage team work.

→ Global perspective.
Identify all the risk involved in project, constant with organisation, business. {entities related to project}

→ As implementation of project proceeds, keep on analyzing risks

→ Software involves team work not a single person therefore communicate with involvement every team member of project.
↳ (stakeholders)
Identify overall impact factor Probability

→ Analysation of Risk should be a continuous process.

⑥→ Isolation should not be there, all the tasks must be properly performed in the vision of all stakeholders.

Page No.
Date

⇒ Process of Risk Management

(1) Risk Identification

Risk Identn is a systematic effort to specify threats to the project plan. By Identifying known and predictable risks the project manager may take necessary steps to avoid and control all these risks.

All the identified risks may be broadly classified into two categories.

- 1) Generic Risks
  these are the potential threat to every s/w project

- 2) Project Specific Risk
  It can be identified with clear understanding of technology, people & environment that is specific to the software project that is to be built.

One common Method to identify risk is to create a risk item checklist. All the predictable & know risks are analysed under following subcategories →

(1) product size (p.s)
(2) business Impact (Bu)
(3) Stakeholder characteristics (sc)
(4) Process Definition (pd)
(5) Development environment (de)

(6) Technology (Te)
(7) Staff size and experience (ST)
(8) Cost risks (CU)

15-02-18

Accessing Overall Project Risk.

we have to answer several questions that are formed by a detailed survey of similar type of project, and based upon the answers received, the risk associated with the project ~~mainly~~ may be identified.
If the questions are answered negatively then some steps should be adopted to manage the risk and the degree to which the project is at risk is directly proportional to no. of negative responses received.

Q8→ Risks components & drivers. ✓
The risk components are identified in the following categories.

- (1) Performance risk
  The degree of uncertainity that a product will meet its requirements

- (2) Cost risk
  The degree of uncertainity that a product budget will be maintained.

Page No.
Date

(3) ~~Schedule~~ Support Risk — resultant

The degree of uncertainty that the project is easy to correctly adapt & enhance.

(4) Scheduled Risk

The degree of uncertainty that the project will be delivered on time.

The impact of each risk driver is divided into one of the 4 impact categories

1) Negligible (3) Critical
2) Marginal (4) Catastrophic

↓ Category / Component →	Performance	Support	Cost	Schedule
1. Catastrophic (due to (high) sudden changes)	① Req^n meet ✗		↑cost 500K$, delayed delivery.	
	② limited / no technical feat.	③ non responsive	significant financial crisis	unachievable control list — IOC
2. Critical (wrongly Interpreted)	failure & meet req^n leads to degradation of performance		failure / delay due to ↑cost	
	reduction in technical features	minor delay in modification	shortage of funds	slipping of IOC
3. marginal	failure to meet req^n & degradation in secondary objectives		(1K to 500K) $ cost	
	min / req^n in tech. performance	responsive s/w support	sufficient funds	realistic schedule
4. negligible (low)	no req^n in performance	easily supportive	budget underrun	early delivery

IOC → Unachievable control list.

~~① Failure to~~

→ when cost exceeds
- $500K → catastrophic
- $100K - 500K → critical
- $1K - 100K → marginal
- below $1K → negligible

19-02-

Risk projection

Page No.
Date

Q→ Developing a Risk table

	Risk	category	Prob.	Impact	Rmmm
1.	Size estimate is significant low	PS	60	2	
2.	Lago no. of users	PS	30	3	
3.	Less Reuse	PS	70	2	
4.	Delivery deadline not met	BU	50	3	
5.		C			
6.	Cost Exceeds	CU	40	1	
7.	Customer Req^m Changed	PS	30	2	
8.	Funding lost	BU	40	3	
9.	Lack of Planning	TE	50	2	
10.	Staff Inexperienced	ST	40	3	

Page No.
Date

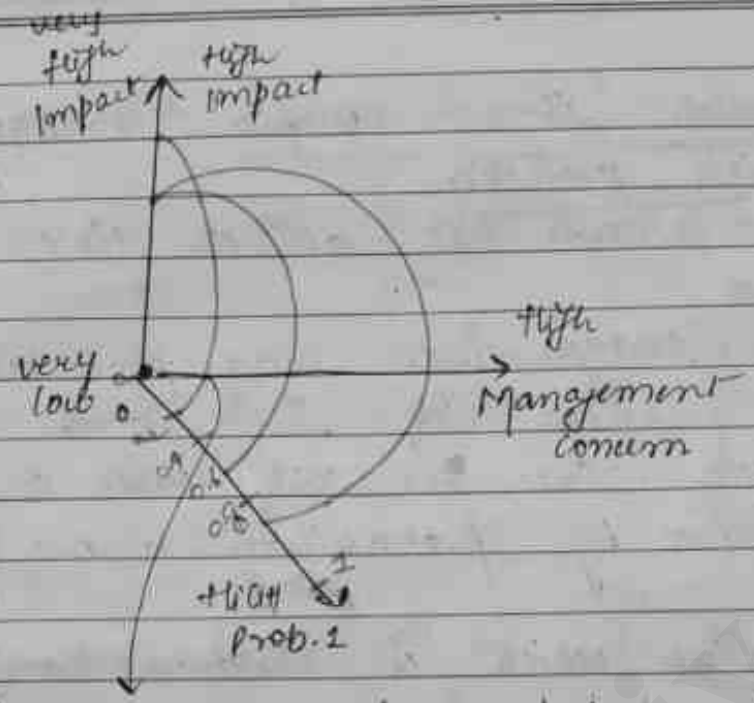
⇒ A risk table is a simple technique for risk projection.
The first column list all the risk that are identified.
They are categorised and probability and impact of each risk is listed in the next column. Once the risk table is competed
⇒ it is sorted by probability and by impact.

A cut off line which is drawn horizontally at some point in table implies that only risk that are above the cut off line will be given further attention.

Rist that falls below the line are re-evaluated to accomplish second order prioritisation.

Risk impact & Probability have a distinct influence on mangement concern.
⇒ A Risk factor with high impact & low probability should not absorb significant amount of management time.
However, high impact risk with moderate to high probability and low impact risk with high probability should be carried forward.

Risk with high probability should be first concerned



A risk with very high impact & with very low probability have very low management concern.

⇒

⇒ Accessing the Risk Impact (Estimate)

Three factors that affects the consequences that are likely if a risk occurs.
nature, scope & tyming.

- Nature of the risk indicate the problems that are likely if it occurs. for ex → poorly defined external Interface will lead to system Integration problems.

- Scope of a risk combines the ~~loss~~ severity with its overall distribution. for ex → how much the project will be affected if the given risk appears.

Page No.
Date

- Timing of a risk concerns when & how far the impact will be felt.

Numerical Example

Risk = Only 70% of software components for reuse will be integrated into applications. The remaining functionality will have to be custom developed. Calculate the risk exposure given that risk probability is 80%. No of reusable components is 60. Average component size is 100 LOC & cost is 19 $.

- We can reuse 70% & rest 30% is custom developed

60 reusable components → 70% / 30% = 18 reusable components will be custom developed

P = 80%.

(Component size) e = 100 LOC

C = 19 $

Risk Exposure (RE) = P * C → (cost to the project if the risk occurs)

(Probability)

Page No.
Date

$C = 18 * 100 * 14$ $\quad P = 80\% = 0.8$
$= 25200$

$\Rightarrow R.E = 0.80 * 25200$
$= 20160$

⇒ Risk Refinement

During early stages of project planning a risk may be stated quite generally and it is possible to refine the risk into a set of more detailed risk which is easier to mitigate, monitor & manage.

One way for refine a risk is to represent it in CTC format i.e. condition, Transition, Consequence.

A general CTC takes the following form:

→ Given that <condition> then there is a concern that (possibly) <consequence>

for eg →

given that < all reusable software components may not confirm to specific design standards > then there is a concern that (70%) < of the reusable modules may be integrated into built-in systems resulting in mainly 30% components to be custom engineered. >

Page No.
Date

→ The risk may be refined in the following manner.

1. certain reusable components were developed by a third party, with no knowledge of internal design standards
2. The design standards has not been solidified and may not confirm to existing reusable components.
3. Implementation language is not supported on the target environment.

(These subconcepts may increase the cost of project)

Imp ⇒ RMMM

Q→ RISK MITIGATION, MONITORING & MANAGEMENT

(to reduce effect of risk) (to continually sense) (adapt a policy to manage risk)

- A risk management strategy is included in the software project plan and it can be organised into separate risk mitigation, monitoring & management.

Risk Mitigation is the risk avoidance activity. and in a proactive approach risk avoidance is the always the best strategy. This is achieved by developing a plan for risk mitigation. Example →

Page No.
Date

Eg →
high staff turnover is noted as project risk r_4 and based upon the past history the likelyhood l_4 is 70% and impact x_4 is critical.
Develop a strategy to mitigate this risk.

From Book

Risk Monitoring is a project tracking activity which proceeds after the commencement of project. The primary objectives of project monitoring are →

→ 1. To assess whether predicted risk occur and to analyse its effects.

→ 2. To ensure that risk aversion steps defined for the risk are being properly applied or

→ 3. To collect the information that can be used for future risk analysis.

Example →
In case of high staff turnover, the monitoring activity includes measurement of project pressure is to calculate the degree of turnover & identify interpersonal relationships & potential problems and to carry out market survey for availability of jobs

Page No.
Date

Risk Management & contingency plan assumes that mitigation efforts have failed and the risk becomes a reality. Continuing with example, the project is well underway and a no. of people announce that they will be leaving. If the mitigation strategy has been followed, then backup is available, information is documented & knowledge has been dispersed across the team.
For this, a risk information sheet is been prepared and the format for RMMM sheet is as follows.

from Book

Risk Information Sheet

Risk ID	Date	Prob.	Impact
Description:			
Refinement: Subcond^n 1 !			
Mitigation/ Monitoring 1. 2. 3			
Risk Mtng/ contingency plan/ trigger			
Current state:			
originated:		assigned:	

Page 402 & 404 from Pressman

Page No.
Date

Unit-5

Software And Quality Management

Software quality can be defined as an effective software process applies in a manner that creates a useful product & produces a measurable value for those who produce it & those who use it.

Q → Software quality Attributes.

(1) Performance & feature quality.
(2) Reliability → provides services without Error
(3) Conformance
(4) Durability & serviceability
(5) Very Perception and Build Perception
(6) Correctness → conforms to specific reqm
(7) Efficiency → min. resources → max. output.
(8) Usability.
(9) Maintainability → used after ~~ab~~ product Released
(10) Flexibility
(11) Portability
(12) Reusability
(13) Inter operability.
(14) Robustness
(15) High Availability / Durability.

Short term

system will work efficiently irrespective of no. of errors (eg. - gmail)

Page No.
Date

(16) Richness { the degree to which product is rich in its features }

⇒ Achieving software quality ? { eg. good management practices & sound engineering principles }

It is the result of good project management and solid s/w engineering practices. Management and practices are applied within the context of following framework activities.

1. S/w Engineering Methods ✓
To build a high quality s/w we must completely understand the problem to be solved. Capable of creating s/w design that conforms to the solid problem & standards & that includes characteristics that lead to s/w that exhibits several quality attributes.

2. Project Management Techniques -
Several project Management technique may be included to develop a high quality software : some of them are -

- Project estimation - estimate cost of project in advance
- Risk Planning -
- Establish Dependencies - (establish relation b/w modules, teams)

Pg - 619 - 623 {FPA numerical

Page No.
Date

- Project Planning

3) Quality Control { testing, feedback
Quality control ~~ensures~~ encompasses a set of of s/w engg. actions that helps to ensure that each work product meet its quality goals. Models are reviewed and inspected to ensure correctness & consistency of the project. The goal is to uncover & correct errors before testing. A series of testing steps is applied to make the software error free. A combination of measurement & feedback allows a software team to tune the process when any of these work product fails to meet quality goals

4) Quality Assurance {auditing}
[Combines all pts stated above] It establishes the infrastructure that supports solid s/w enginnering principles and methods, rational project management & quality control actions to build high quality s/w. It consists of set of auditing and reporting functions that assess the effectiveness & completeness of quality control actions. The goal of quality assurance is to provide management & technical staff with data necessary about the product quality.

- Software Reviews
They are a filter for s/w process which are applied at various pts during s/w engg that aims to uncover errors and defects

Page No.
Date

that can be resolved removed. The need for S/W review is correction and adaption of any changes that may generate during software designing process. • Software Reviews are software engg. work products which are purified at every phase of S/W development.

→ Many diff types of reviews can be conducted as part of S/W engineering.

(1) Informal Meeting. (not cont., checkpoint based)

(2) formal presentation of S/W architecture to customer management or technical staff.

(3) Technical Review - It is the formal & most effective filter for quality control that may include peer reviews, casual reviews, walk throughs & inspections.
(group activities) (can be done by anyone) (more formal and done by a dedicated team)

⇒ Cost impact of Software Defects

A S/W product may include defect or fault that implies a quality problem that is discovered after the software has been released to end users. (It is different from error which depicts a quality problem that is discovered by S/W engineers before the software is released to end users) [The primary objective of technical reviews is to find errors during the S/W development process so that they will not become defects after release of the software]. Early discovery of errors

defect // S/W quality problem // at user

Page No.
Date

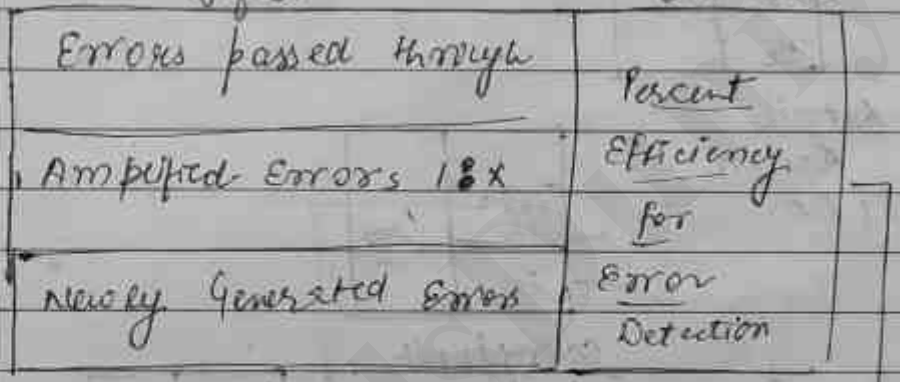
is required so that they do not propagate to defects in the next step in software process.

Page No.
Date

→ 8-02-18

Q5 → DEFECT AMPLIFICATION & REMOVAL

A defect Ampl^{n} model is used to illustrate the generation and detection of errors during design and code generation phase of software process. The model uses a box to represent software engineering action. This model is depicted as.

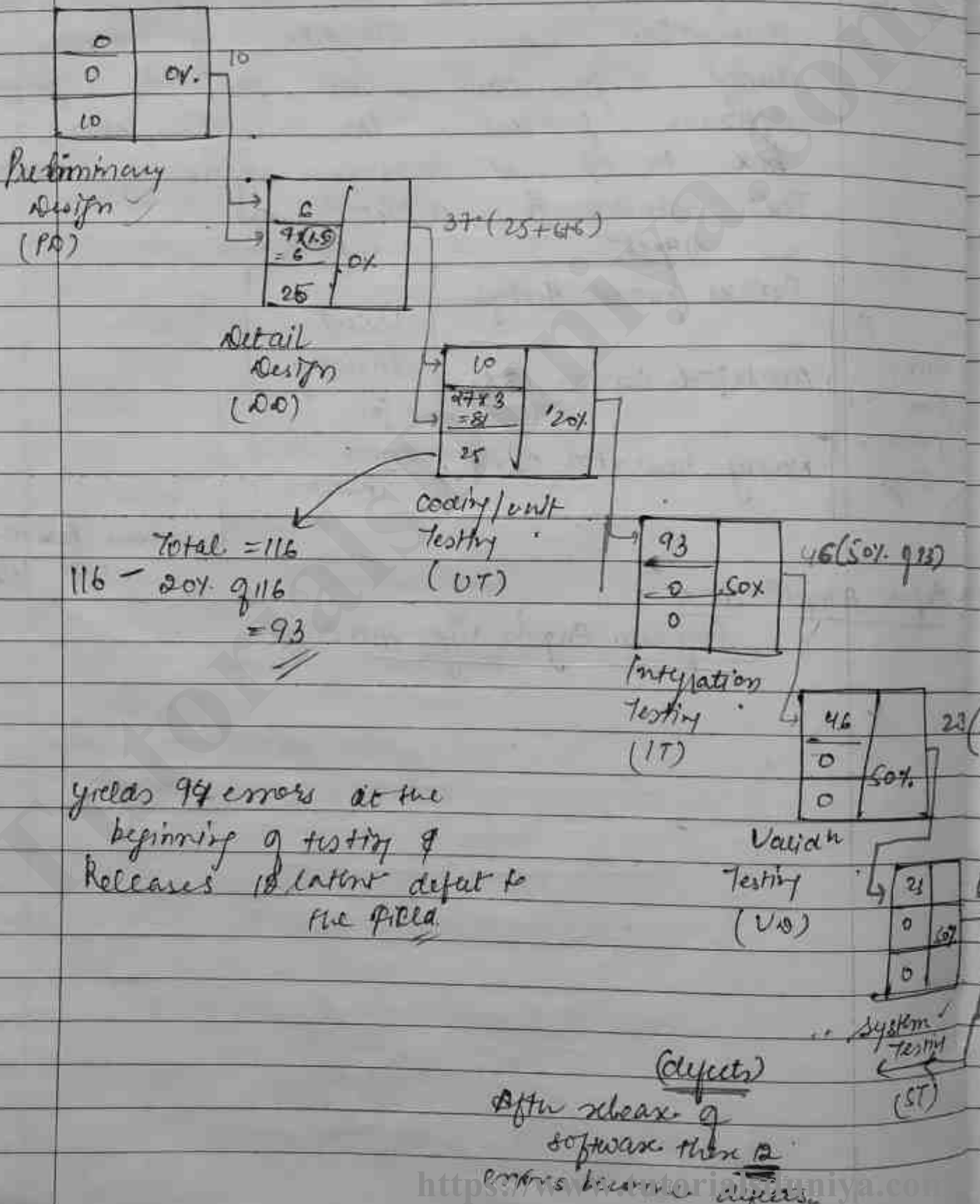


Defect Ampl^{n} box :

Software Engineering Action

Page No.
Date

⇒ Example for Defect Amplification Model Without reviews.



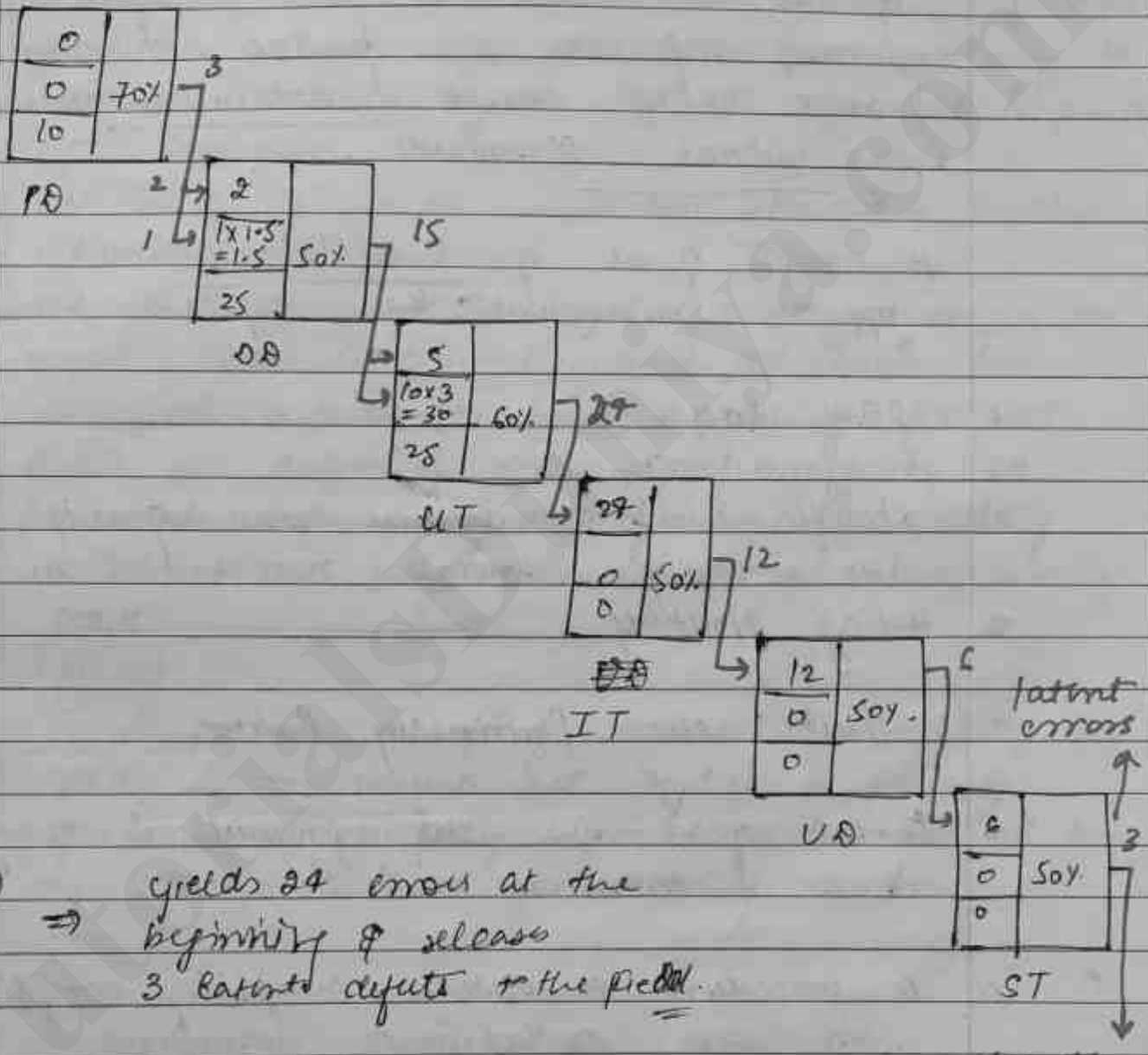
yields 99 errors at the beginning of testing & Releases 18 latent defect to the field.

(defects)

After release of software there 2 errors become defects.

Page No.
Date

⇒ Example for defect Amplification model with review of each step.



(56% of 4%) ⇒ yields 24 errors at the beginning of release
3 latent defects in the field.

latent errors (12)

⇒ A cost analysis indicates that the process with no reviews produce 3 times (approx) more than process with reviews, taking the cost of correcting latent defects into account

Page No.
Date

⇒ Software Quality Assurance (SQA)

SQA consists of set of monitoring activities applied to each & every software engineering proces to provide a quality product which ensures certain software & organisational standards.

It consists of set of Umbella activities that is applied throughout the software process.

1. SQA Process
2. Specific Quality assurance and quality controlled tasks including technical reviews & multi-tired testing strategy.
3. Effective s/w Engineering Proces
4. Control of all s/w work products & changes made to them.
5. A procedure to ensure compliance with s/w development standards
6. Measurement & reporting mechanism.

12-03-18

Page No.
Date

Elements of SQA

1. Standards → IEEE, ISO and other standard org^n have produced software e.g. standards that have/may be adopted voluntarily by the Org^n or may be imposed by the customer.
→ The job of SAQ is to ensure that standards have been adopted and followed.

2. Review & Audits → The intent of review activity is to uncover errors. The limitations and deficiencies are identified in order to build a quality product.
→ Audits are a type of review performed by SQA personnel with the intent of ensuring quality guidelines are being followed for S/w engineering work.

3. Testing
is a process to find errors. The job of SQA is to ensure that testing is properly planned & efficiently conducted so that it has the highest likelihood for detecting errors.

4. Errors / defects collection and Analysis.
SQA collect and analyse errors to better understand how errors are introduced and what actions should be taken to eliminate them with the intent to restrict error to become a defect.

Page No.
Date

5. Change Management
Change is one of the most disruptive aspect of software process. If it is not properly managed it can lead to confusion and may lead to poor quality. SQA ensures that adequate change Management practices have been adopted.

6. Education
For software improvement, education of Software engineers, managers & other stake holders is an integral software engineering practice. The SQA is the key member to initiate and develop such kind of educational program to enhance technical skills of the project staff.

7. Vendor Management.
The job of SQA is to ensure that high quality s/w results by suggesting specific quality practices that the vendor should follow & incorporating quality mandates as part of any contract with an external vendor. Based upon this 3 categories of softwares are identified.
(1) Shrink wrapped Packages { ex → ms office, computer project }
(2) Tailored project { component of above category }
(3) Customized project { Specifically tailored/engineered as per the requirement }

8. Security Management
With the increase in cyber crime & several security threats every s/w organisation should adopt policies

that protect data at all levels. SOA ensures that appropriate process and technology is used to achieve security.

9. Safety & Risk Management

SOA is responsible to assess the impact of software failure and to initiate the steps required to reduce the risk. SOA ensures that the Risk Mang. activities are properly conducted and risk related contingency plan have been established.

19-03-18

Page No.
Date

→ Product Metrics

- Measurement is the process by which numbers or symbols are assigned to the attributes of entities in the real world according to the clearly defined rules.
- Product Metrics helps s/w eng. to gain insight into the design & constn of s/w by focusing on specific measurable attributes of s/w eng. work products.
- To carry out qualitative assessment is not enough product metrics provides a basis from which the development steps are conducted more objectively and accessed more quantitatively.

→ Difference between Measures, Metrics & Indicators

(Individual quantity ↑ ; (degree of affinity. ↑ measure)

(relation b/w 2 measures)

- Within the context of software engineering, a measure provides a quantitative indication of the extent, amount, dimension, capacity or size of some attribute of a product or process.
- Measurement is the act of determining a measure.
- Metric is a quantitative measure of the degree to which a system, component or process possesses a given attribute. Metric establishes relation b/w 2 measures. Or it is a degree of defining measures.

⇒ Sin when a single datapoint has been collected.

Page No
Date

ef- No of errors encovered, a measure has been established. Measurement is the collection of one or more these data points and software metric relates these individual data points by establishing some formula for example → average no. of errors per review.

Significance or assessment of a measurement

- Indicator is a metric or combination of metrics that provides insight into s/w projects process, a s/w project or the product itself. An indicator ~~provides~~ enables s/w engineers to adjust s/w process and to improve s/w products. for example, measurement dashboards are used to monitor process and initiate change when required.

⇒ Metrics for Requirement Model

Read Pg → 619 to 623

⇒ Process and Project metrics

S/w process and project metrics are quantitative measures to gain insight the efficiency of s/w process and projects that are conducted using the process as a framework. The first

first step is to collect data for quality and productivity second These data are analysed, compared against past averages and assessed to determine whether improvements have occurred. third Metrics are used to pin-point problem areas, so that remedies can be developed and s/w process can be improved.

in order to find shortcomings

for improving

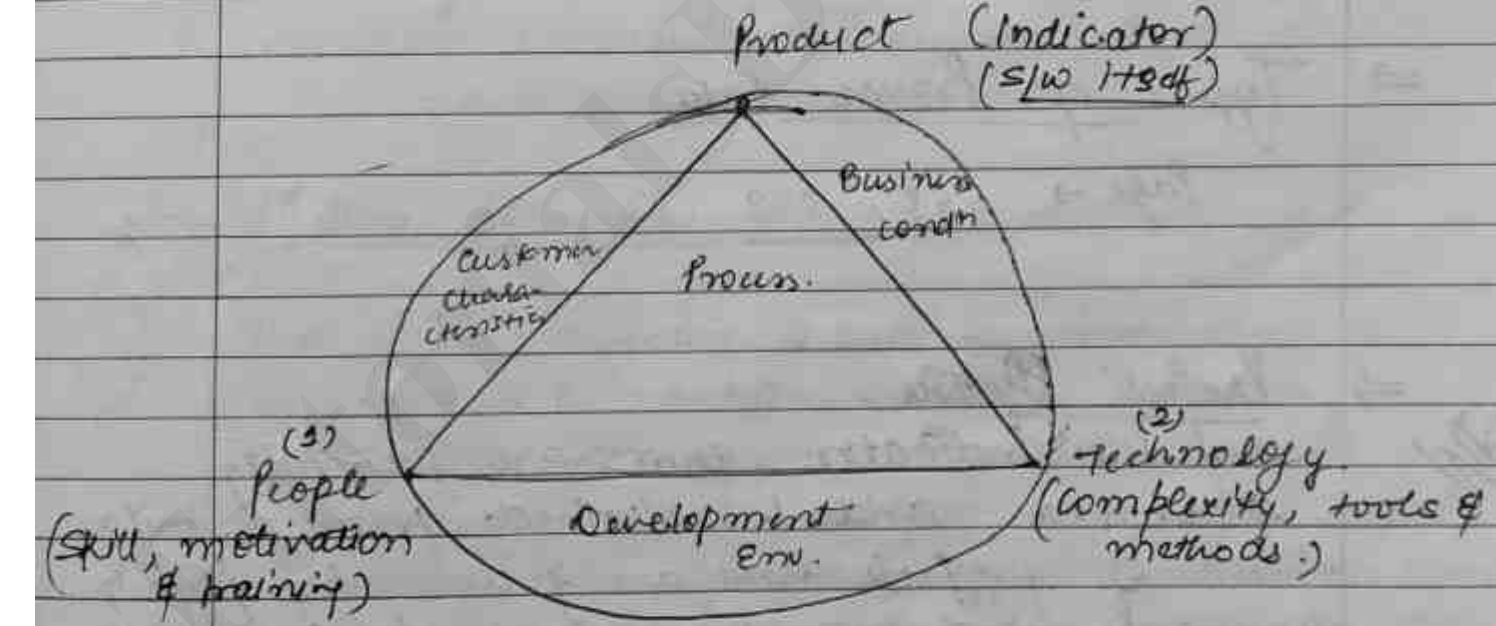
⇒ Metrics in process and project domain

- Process metrics are collected across all the projects and over long periods of time. The intent is to provide ~~a~~ process indicators for s/w process improvement.
- Project metric enables a s/w project manager to →
  1) Assess the status of ongoing ~~process~~/project
  2) Track potential risk
  3) Uncover problem areas.
  4) adjust work flow or task. {adjust timeline chart}
  5) Evaluate the project team availability to control work product quality {continous evaluation}

⇒ factors affecting process Metric. & milestones &

- To improve the measurement process; some specific attributes of the process are identified, a set of meaningful metrics based upon these attributes is developed and then indicators are provided to assess the quality and productivity and are analysed for improvement.

⇒ Determenants / factors.



- The factors that have a performed influence on s/w quality and organisational performance are people, technology and product.
- The skill and motivation of people is the key factor that influence quality & performance

Page No.
Date

- The complexity of the product can have a substantial impact on quality and team performance.
- The technology which includes methods and tools that populates the process has an impact on productivity and quality.
- In addition, a given process exists within a circle of environmental conditions. ex → Integrated s/w tools, business condition (deadlines) and customer characteristics (ease of communication.)

⇒ Types of Process Data

Page → 668, 669

⇒ Project Metrics.

are the indicators that are carefully planned and is derived from similar kind of projects and are used by project managers and s/w team to adopt project work flow and technical activities.

- Metrics collected from past project are used as a basis from which effort & time estimate are made for current s/w work. As the project proceeds, measures of effort & time are

Page No.
Date

compared to original estimates, and the project manager uses this analysis to monitor & control progress. These project metrics are used to minimize development schedule by making adjustments to avoid delay and mitigate potential problems and risks. These are used to assess product quality on an ongoing basis and activities may be modified when necessary.

⇒ Software Management

1. Size Oriented Metrics. {LOC}
   Page → 674 LOC per FP.
2. Function Oriented Metrics {FP}

⇒ Metrics for S/W Quality

High quality in time frame to meet market need

The prime objective of S/W engineering is to produce a high quality system, application or work product within a time frame that satisfies a market need. To achieve this objective, effective measures must be covered with modern tools within the context of mature S/W process, that results into a high quality software.

The quality of a system is as good as the requirements that describe the problem, the design that models the soln, the code that

approp.

→ The module must be loosely coupled & highly cohesive

leads to executable programs and test cases that exercise the software to uncover the errors.

Attributes for Measuring quality

Q. →

(1) Correctness –
It is a degree to which the s/w performs its required functions. The most common measure is defects per KLOC, where defect are those problems reported by the user after the program has been released for use.

(2) Maintainability –
It is the ease with which a program can be corrected if an error is encountered, adapted if environment changes or enhanced if requirements change. There is no direct way to measure maintainability but some indirect measures may be used.
for ex →
Mean Time to change (MTTC). It is the time to analyse the change request, design an appropriate modification, to implement the change, ~~testing~~ & test it & distribute the change to all the users

06-03-18

Page No.
Date

⇒ Function Oriented Design

The design activity begins when the requirement document for the s/w is available and its architecture as we design & define.
It gives the blueprint or plan for a solution in terms of components that are combined to become a system. That is, the design exercise determines the module structure of each components in a architecture.

The design process is divided into two levels. At the • first level the focus is to determine which modules are needed for the system, the specification of these modules & how the modules should be interconnected. This level is called system design or top-level design. In the • second level, the internal design of the modules and how the specifications of the module can be satisfied is decided. This is called as Detailed Design or logic Design.

In function Oriented design approach, a system is viewed as a transformation function, transforming the inputs to the desired output.

Page No.
Date

⇒ Design Principles { what are the basic principles of system designing phase }

1. Correctness.
2. verifiable
3. complete
4. traceable
5. efficient { use least resources }
6. simple and understandable.
7. Maintanable
8. Effective

⇒ Problem Partioning and hierarchy

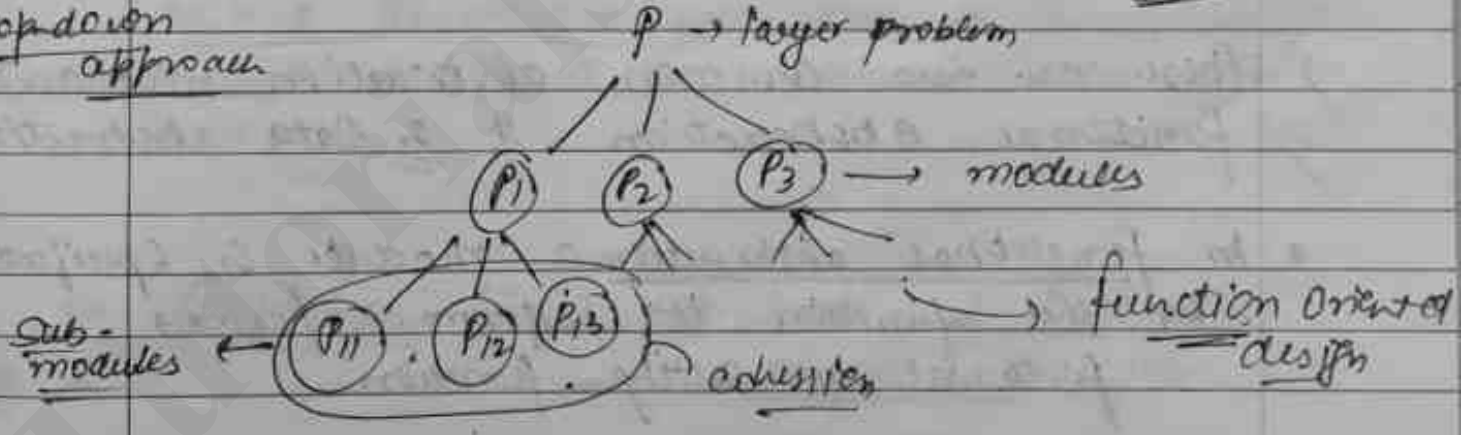
The complexity of s/w & ~~et~~ the implementation of large problems is avoided bcoz it results into a complicated design which is poor in maintainability. for solving large problems, the basic principle of divide and conquer is adopted, where the complete problem is divided into smaller pieces and each piece can be conquered separately.

for s/w design, the goal is to divide the problem into manageably small pieces that can be solved separately. The Basic idea is to reduce the cost and complexity of solving the entire problem as compared with the pieces when it is divided into pieces. The different pieces may not be entirely independent of each other & they have to coperate & communicate to solve larger problem.

This comp communication adds complexity which increases as no. of components and their dependency increases. So the design is aimed to identify the partitioning point where further partitions have to be stopped.

If a module can be modified separately, it is called as independent module and it is desired that we must have more Independent modules in our System. Total Independence of modules is not possible but the design process should support as much independence as possible.

The design produced by problem partitioning can be represented as hierarchy of components where relationship between these elements can be depicted.



⇒ Abstraction

An abstraction of a component describes the external behaviour of that component without bothering with the internal details that produce the behaviour.

Page No.
Date

Abstraction is an indispensible part of the design process ie essential for problem partitioning. Problem partitioning result into components which may interact with other components. If the external behaviour of other components is known to the designer, it reduces the complexity to understand the details of these components, which allows the designer to concentrate on one component and abstraction of other components.

During the design process, abstractions are used in reverse manner & the components that do not exist, their design is carried out. & for other components abstract specification are used.

⇒ There are two common abstraction mechanism
1. Functional Abstraction & 2. Data abstraction

- In functional abstraction a module is specified by the function it performs. for ex→ factorial or sorting function.
- The second unit of abstraction is Data Abstraction. The s/w is modelled upon the real world entities which is represented as by certain data structures and set of operations performed on these data structures. From outside an object the internals of the object are hidden, only

Page No.
Date

Operations on the object are visible. Data abst^{n} forms the basis for object oriented design. for example → banking transactions.

⇒ Modularity

A system is considered modular, if it consists of discrete components so that each component can be implemented separately and are changed to one component as minimal impact on other components. • Modularity helps in system debugging, in system repair & in system building. A S/w system cannot be made modular by chopping it into pieces for modularity, each module needs to support a well defined abstraction and have a clear interface through which it can interact with other modules.

Q → ⇒ Difference b/w Top-down & Bottom-up design strategies

Top-down	Bottom-up
Top-down approach starts from highest level component and proceeds through to lowest levels.	Approach starts with a lowest level component and proceeds through progressively higher level to the top level component.

Top-down	Bottom-up
- Identification of major components & decomposing into lower level components is done until desired partitioning level is achieved.	- It starts with designing the most basic or primitive components & proceeds to higher level.
- Stepwise refinement is adopted.	- Bottom-up method works with layers of abstraction.
- Top-down approach is suitable if specification of systems are clearly known and the system development is started from scratch.	- If a system is to be built from an existing system, a bottom-up approach is used.

3-04-18

⇒ Module Level Concept -

Coupling & Cohession → { max }

(must be min)

→ A module is a logically separable part of the program. It is a program unit that is discrete & identifiable with respect to compilation & loading.

Ex → A macro, function, process, or a package.

Page No.
Date

⇒ Coupling & cohesion are used to identify degree of modularity & level of partitioning. and it is expected that the coupling is minimum and cohesion is maximum.

⇒ Coupling (minimum)

- coupling between modules is the strength of interconnection b/w modules. It is considered as measure of independence among modules. Two modules are independent if they are modifiable separately. Highly coupled modules are joined by strong interconnections while loosely coupled modules have weak interconnections. The choice of module decides the coupling between modules. It is decided during the system design phase.

The coupling is affected by several factors which includes interface complexity, types of connection & type of communication.

type of interaction

Coupling increases with complexity & uncertainty of interface b/w modules.

To obtain low coupling, we could minimise the number of interfaces per module & complexity of each interface within a given module. Interface of a module is used to pass information to and from other modules. Coupling is reduced by passing information

through parameters & it will increase by the use of shared variables.

→ Complexity of the interface affects the degree of coupling & which increases with increase in complexity. Interfaces uses communication of objects b/w different modules & coupling is reduced if we transmit data selectively as required.
for ex→ A field of a record is needed by the procedure & then to reduce coupling just pass on the required field.

→ The type of Inf^n flow is the next factor that affects coupling. Information is divided into 3 categories.
1. control Inf^n
2. data Inf^n
3. Hybrid Inf^n {control + data}

passing on security control inform^n means that the action of the module will depend on the control information which makes it difficult to understand and to provide abstraction. Transfer of data means that a module passes some data as input & gets some data as output. Interface with only data communication result in lower level of coupling followed by the control data but it is considered highest if the inf^n is hybrid. i.e. both data & control.

Page No.
Date

Imp

⇒ Cohession

I F/P

Cohession of a module represents how tightly bound the internal elements of the module are to one another. Cohesion is the concept that establishes the intramodule dependency. With cohesion it is evaluated that how closely the elements of a module are related to each other. The greater the cohesion of each module in the system the lower is coupling b/w different modules. There are several levels of cohesion

→ 1. Co-Incidental Cohesion (Weakest

- [It occurs when there is no meaningful relationship among the elements of a module].
- [It occurs if an existing program is modularized by chopping it into pieces & by making each piece as a module.]

The modules contain a random collection of function

for ex⇒

- [• If a module is created to save duplicate code by combining some part of the code that occurs at many different places. The module is likely to have co-incidental cohesion.]

5-09-18

- The resulting modules may contain random collection of functions.

Note: Co-incidental is the weakest level & functional is the strongest level.

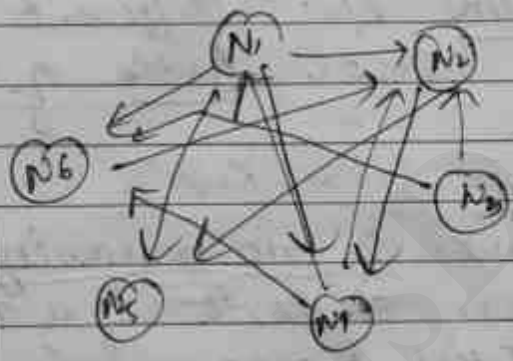
Page No.
Date

2. Logical Cohession

a module has logical cohession if there is some logical relationship b/w elements of a module & the module perform functions that fall in the same logical class.

eg → Datahandling, data I/P, data O/P etc.

A set of print functions generating an output report is arranged in a single module.



highly coupled network

Ideally it must have been

3. Temporal Cohesion

elements must be executed in same time

same as logical cohesion except that the elements also related in time & are executed together. Modules that perform activities like initialisation, clean up & termination are usually temporal bound. The degree of temporal cohesion is higher than logical cohesion as the elements are logically related and ordinarily execute together. This avoids the problem of passing the flag.

Page No.
Date

4. Procedural cohesion
It contains the elements that belongs to a common procedural unit. In which a certain sequence of steps has to be carried out in a predefined order for achieving an objective.
for eg → A loop or a sequence of decisive statements in a module may be combine to common separate module
• The algorithm for decoding a module

5. Communicational cohesion
a module with communicational cohesion has elements that are related by a reference to the same I/p and o/p data. In a comm. bound module the elements are together because it refers to adopt operations on same data structures.
for eg → • print & punch record or
• set of defined operations on a stack.

If all functions of module refers to an update of the same data structure

Module
Push
Pop
Here coupling increases as both push & pop depend on data in module

Hence, communication overhead increases and cohesion reduces

(6) Sequential Cohesion

When the elements are together in a module because the O/P of 1 forms the I/P to other, We get sequential cohesion.

If the Elements of the module from different parts are executed in sequence. A sequence bounded module may contain several functions or part of different functions.

Ex

(7) Functional Cohesion

It is the strongest cohesion. In a functional bound module all the elements of the module are related by performing a single function.

for eg → Computing the square root or Sorting the array.

→ By single function we not mean simple mathematical function but those modules which have single goal function like

Page No.
Date

Coincidental ← Logical → Temporal → Procedural → Communicational → Sequential → Functional

low Cohesion ↓ high Cohesion

weakest ↑ strongest

⇒ Coupling vs Cohesion

- While designing you should strive for low coupling i.e. Dependency between modules should be less
- Is Inter-module concept

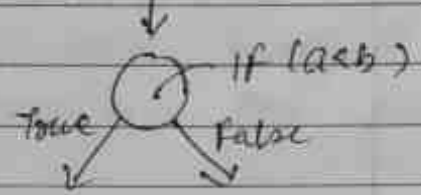
- While designing you should strive for high Cohesion i.e. a cohesive component / module focus on a single task with little interaction with other modules of the system
- is Intra-module concept

4-04-18

⇒ Basis Path Testing

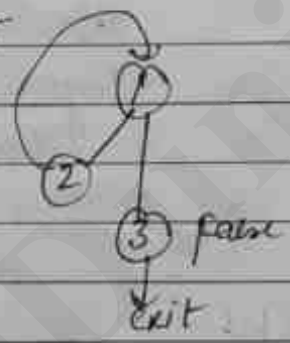
1. To design Program graph / Program flow graph

⇒ IF construct

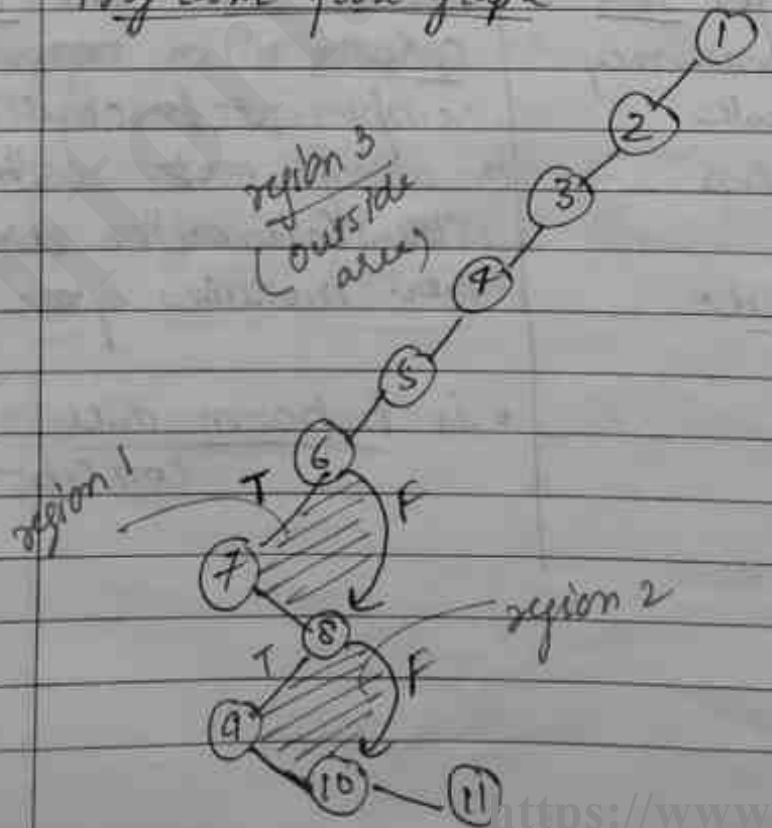



```
1  int main ()
2  { int a, b, c, x=0, y=
4  cout << "enter 3 no";
5  cin >> a >> b >> c;
6  if (a>b && a>c)
7     x = a*a + b*b;
8  if (b>c)
9     y = a*b - a*c;
10 cout << x << y;
11 }
```

⇒ While construct (if it evaluates true, loop back to initial node)



Program flow graph



Page No.
Date

2. Cyclomatic Complexity (VG)

We have three methods.

(a) $C = e - n + 2$ where e = Total no of edges,

(b) C = no. of regions n = no. of nodes

(c) $C = P + 1$ ↳ (can be cyclic)
↳ (Predicate nodes)

Here $e = 12$, $n = 11$

$C = e - n + 2 = 12 - 11 + 2$

$= 3$

no. of regions $= 3 \Rightarrow C = 3$

Predicates nodes are those nodes which have atleast 2 outgoing edges.

$P = 2$ (node 6 & 8)

$C = P + 1 = 2 + 1 = 3$

3. To find no. of Independent Paths

count of (Independent path) is cyclomatic complexity

∴ It is 3.

4. To determine Independent path.

(a) 1 - 6 → 7 → 8 → 9 → 10 → 11 ⇒ 1 - 11 {Linear path}

(b) 1 - 6 → 8 → 9 → 10 → 11 → 1 - 6, 8 - 11

(c) 1 - 6 → 8 → 10 → 11 → 1 - 6, 8, 10, 11.

(d) 1 - 8, 10, 11

computed from it is not Independent path

Page No.
Date

NOTE

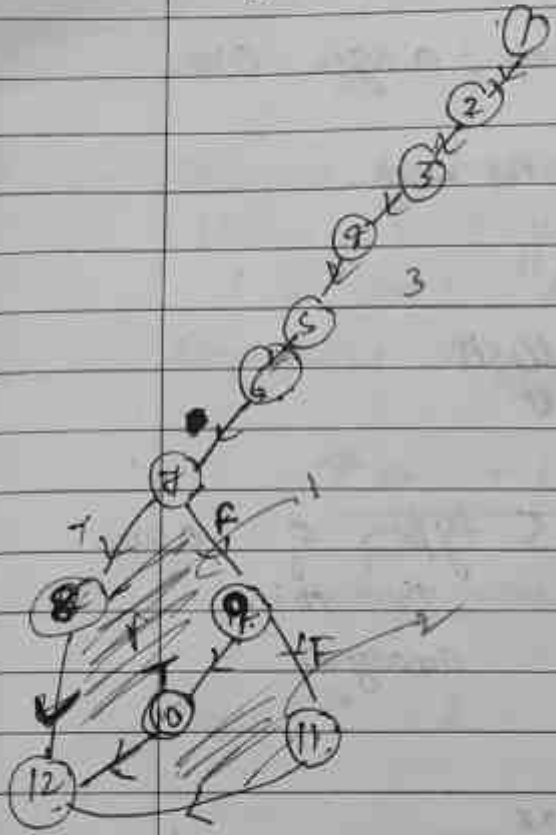
Minimum test cases must be equal to no. of independent paths for overall & efficient testing of project

Q→ Consider a program for determination of a division of a student based upon the marks in 3 subjects. write the program, draw the program graph, cyclomatic complexity & find the independent paths.

```
1.  int main()
2.  {
3.      int a, b, c;
4.      cout << "Enter the marks";
5.      cin >> a >> b >> c;
6.      int avg = (a+b+c)/3;
7.      IF (avg > 60)
8.       cout << "First division";
9.      else if (avg <= 60 && avg >= 40)
10.      cout << "Second division";
11.     else
          cout << "third division";
12. }
```

Page No.
Date

⇒ Program flow graph



(1) ∴ V(G) = 3 (regions)
cyclomatic

(2)
e = 13, n = 12
c = e − n + 2
13 − 12 + 2
= 3

(3) C = P + 1
2 + 1 = 3

No. of Independent Paths → 3 = C

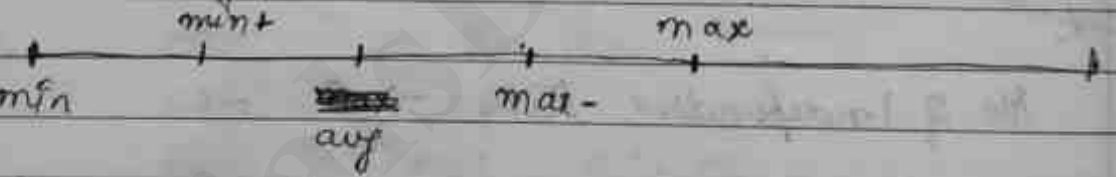
Independent Paths ⇒
(a) 1-7, 8, 12
(b) 1-7, 9, 11, 12
(c) 1-7, 9, 10, 12

Page No.
Date

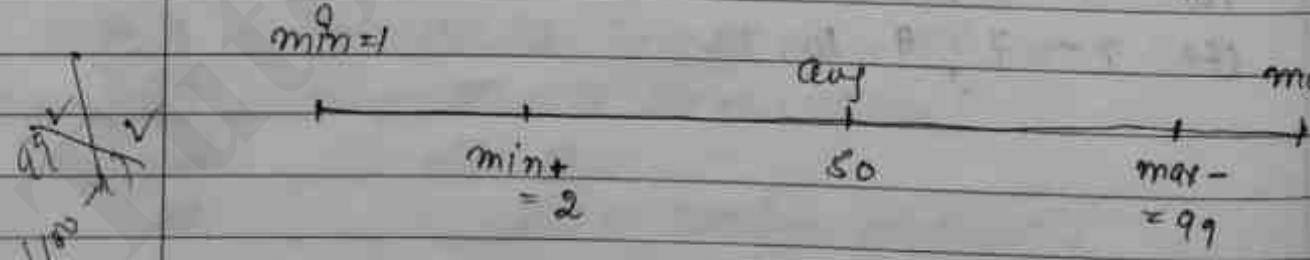
Q→ Perform basis path Testing.
(a) type of triangle
(b) in which quadrant a given angle lies.
(c) largest of 3 nos.
(d) determination of day of the week.

⇒ Boundary Value Analysis

(min)	1. Minimum	5 types of values in this analysis
(min+)	2. Just above min.	
(max)	3. Maximum	
(max-)	4. just below max.	
(avg)	5. average value	



Q→ determine test cases to find square of a given number x. within the range 1 to 100



	test cases	Expected output	Received o/p
1.	1	1	
2	2	4	
3.	50	2500	
4.	99	9801	
5.	100	10000	

NOTE

If there is a deviation b/w expected & required o/p, this means that there is some type of error in program.

Q- Consider a program for addition of 2 input values x & y with a range of x is (100, 300) & of y is (200, 400). Determine the test cases for this program.

$\frac{4 * n + 1}{\text{Input variable}}$ → Total no. of test cases for program

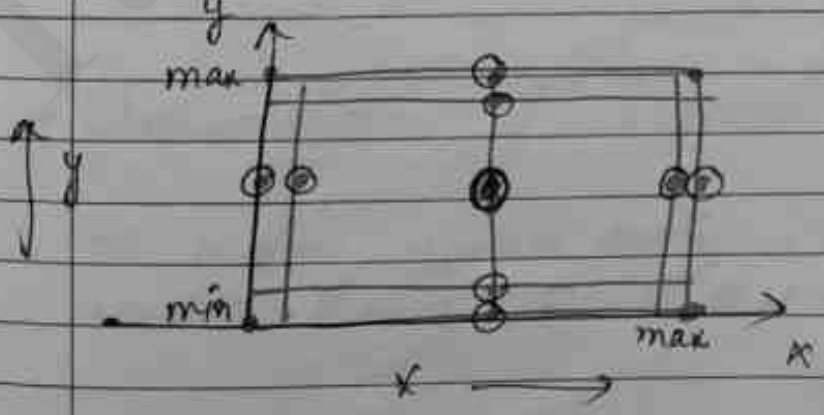
Here $n = 2$

$\therefore$ Test cases $= 4 * 2 + 1$

$= 9$

x →	min = 100	min + 101	avg 200	max - 299	max 300
y →	= 200	201	300	399	400

For this, we have 25 test cases but we won't explore all of them



5-09-18

Page No.
Date

Q→ Perform basis path testing

1. Prime no
2. factorial
3. Leap year
4. GCD (division

Page No.
Date

9-04-18

Unit-7 Testing

→ Testing is a set of activities that can be planned in advance and conduct systematically.

→ Testing is a process to uncover errors that were made inadvertently when it was defined and ~~cost~~ constructed.

→ It includes a set of steps into which specific test case design techniques and testing methods should be defined for a complete software process.

→ Test begins with a focus on single component or set of related components and applies test cases to uncover errors after components are tested they must be integrated until complete system is constructed.

→ A series of high order test are executed to meet customer requirements.

→ Test specification plan documents is constructed by project manager, software engineer or testing specialist that describes an overall strategy and procedure for specific testing steps and types of test that will be conducted.

Same as SRS, RMMM

→ Characteristics of Software Testing Process

1. To perform effective testing, effective technical reviews should be conducted.

Testing Independent Group → (TIG)
↳ has set of universal methods

Page No.
Date

2. Testing begins at the component levels and work outward towards the integration of entire computer based system or software

3. Different testing techniques are adopted for different software engineering approaches & are applied at different point of time.

4. Testing is conducted by the developer, project manager or an Independent test group (ITG)

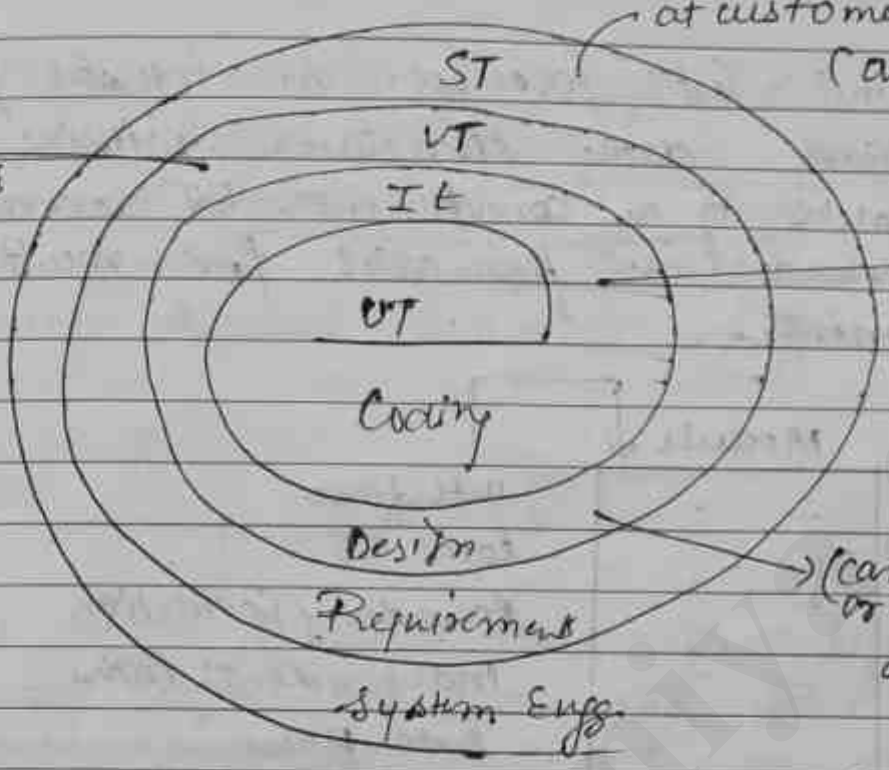
5. Testing and debugging are different activity but debugging must be accomodated in any testing strategy.
(discovering compile time error / run time error)

⇒ Verification & Validation

Page no → 951

⇒ Software testing Strategies

UT - Unit Testing
IT - Integration Testing
VT - Validation testing
ST - System testing



→ Integration Testing is depend on design.
i.e. for Integrating Testing, we need to know the design of the system properly.

→ System Testing is done at the end of customer with customer's perspective. It is testing of whole process.

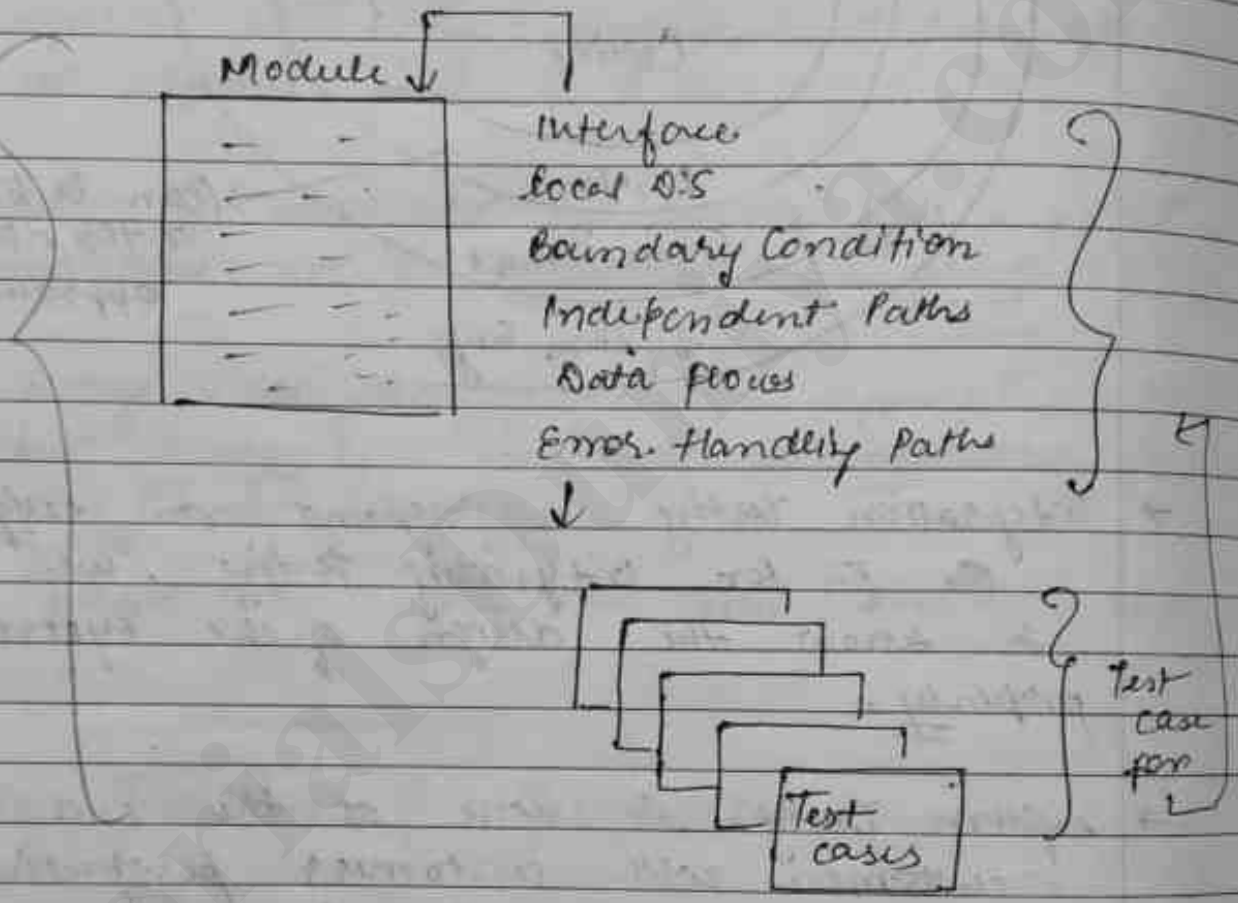
Unit Testing

(1) Unit Testing → main white box testing approach

[focusses on verification effort on smallest unit of software design i.e. the software component or module.] [Using the component level design, control paths are tested to uncover error within the boundary of the module.]

Unit or Integration → Dues

The unit Test focusses on internal processing logic and data structures within the boundaries of a component. UT can be performed in parallel for multiple components.



The module Interface is tested to ensure Input/output information flow.
Local Data Structures are examined for integrity & correctness during Program execution. All Independent paths are executed atleast once to provide completeness of all statements in a module. Boundary conditions are tested as it is assumed that Software often fails at its boundary

Page No.
Date

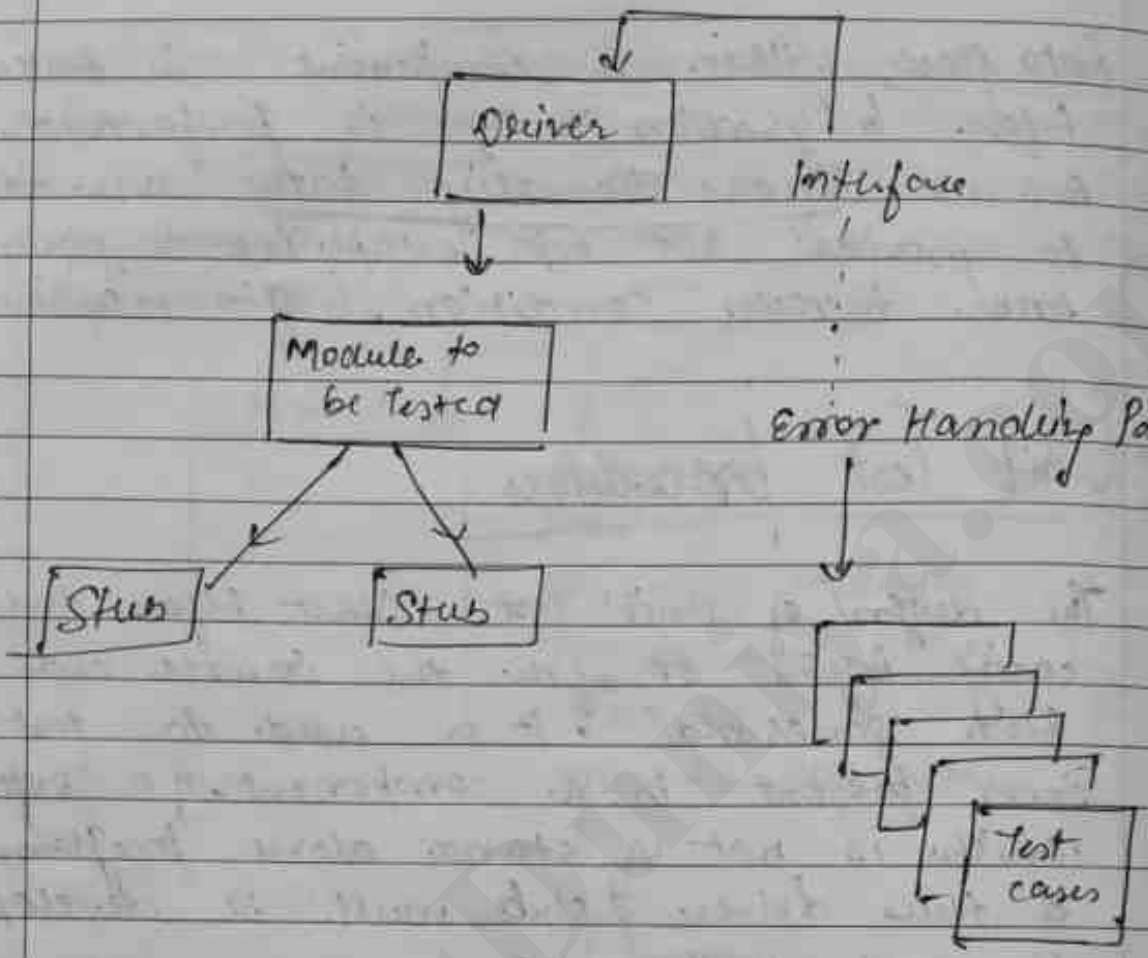
Data flow within a component is tested before Integration Testing is performed.. Etc or Error Handling paths are tested to provide ~~see~~ esp escape is a given error thrown condition. (Eg:→ exception handling)

⇒ Unit Test procedures

The design of unit test start before the coding begins or after the source code has been generated. It is used to test error present in a component of a software which is not a stand alone program & thus driver & stub must be developed for each unit test.

Driver is a main program that accepts test case data passes this data to the component to be tested and prints the relevant results.

Stub serve to replace the module that are invoked by the component to be tested. It is also called the dummy sub program which uses the subordinate module. Interface may do minimal manipulation, prints verification of entries & returns control to the module undergoing testing.



INTEGRATION TESTING :-

- problem with UT is that it will not take into consideration interface b/w diff. modules.
  Interfacing results in loss of data & many potential problems
- → So, Even if the units of S/w are working fine individually, there is a need to find out if units integrated together will also work or not.
  This type of testing is known as Integration testing.

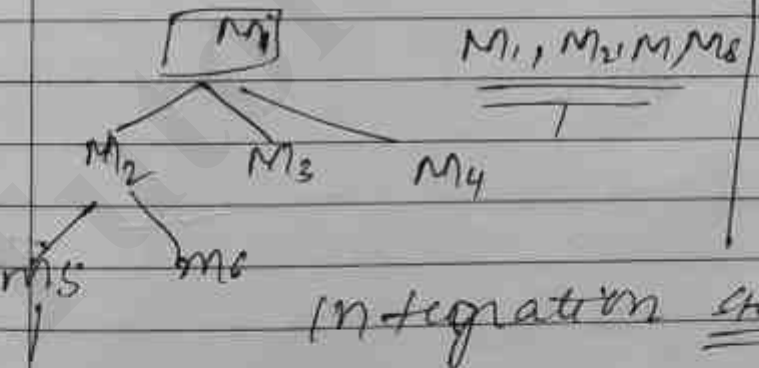
Page No.
Date

⇒ Top down Integration

- is an incremental approach to construction of software architecture.
- Modules are integrated by moving downwards through the control hierarchy, beginning with the main control module to subordinate module upto the last module.
- uses 2 common havers of

DF Integration

(1) integrates all components on a major control path of the program structure from main module to last module until a given depth.



M1, M2 M3 M6

B.F Integration

(1) It encorporates all components directly subordinate at each level moving along a horizontal structure

[M2 M3 M4] first.

- Only those modules can be collected together who have same parents

Integration steps

1. the main control Module is used as a test driver & subordinate modules as stubs.

Page No.
Date

⇒ Bottom-up Integration

- this testing begins construction & testing with components at lowest level in program structure

Implemented with following steps

(1) Low level components are combined into clusters that form a specific software subfunction.

(2) A driver is written to coordinate test case & output
(3) A cluster is tested.
(4) Drivers are removed & combined moving upward in the program structure.

[M8 - M5 - M6] Cluster