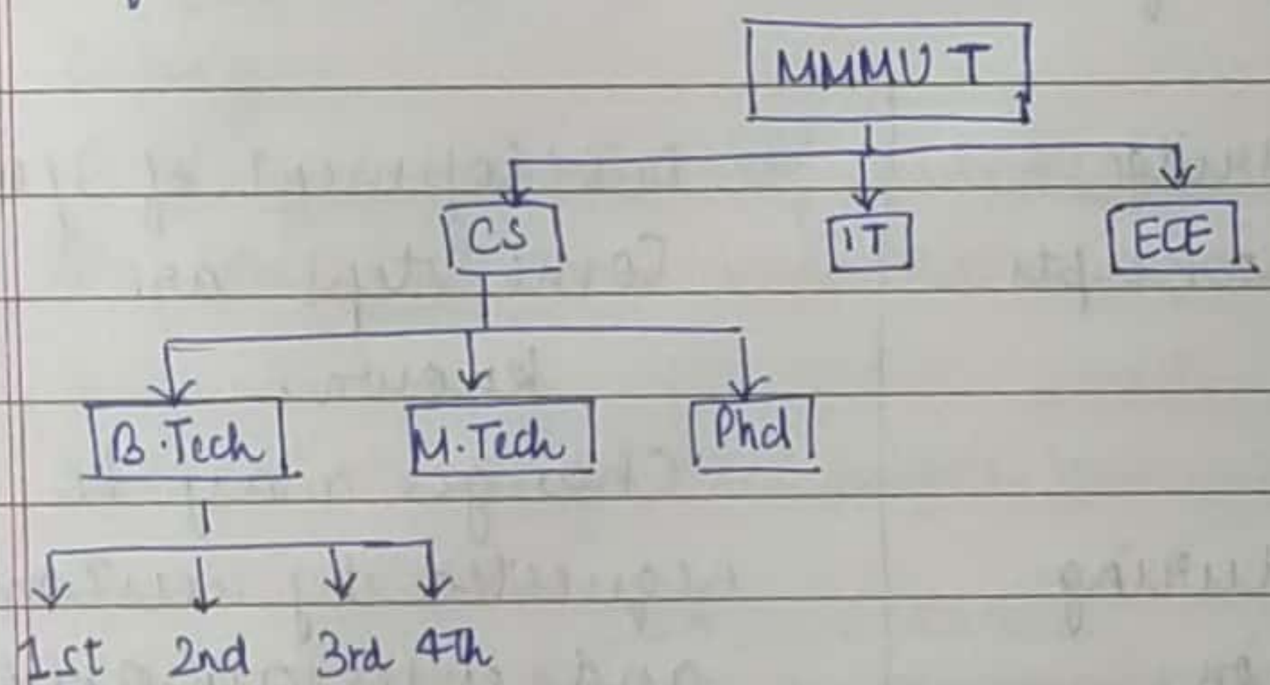
DATE / / PAGE
NOTEBOOK

Software engineering

Discipline which aims
→ to produce quality software or product.
→ to deliver in time.
→ to develop the s/w within budget → Tangible (direct)
→ to satisfy the customer's requirement. → Intangible (hidden)

Product development

Soft source of project
→ Inside
→ Outside
→ self - initiated



1. Always start from bottom - up approach.
2. Divide it in small - small modules MODULAR APPROACH

AGILE MODEL → forms a part & its running effectively while working side - by - side on other parts.

Qualities of SWE

1. Good analytical bent of mind.
Eg of creativity : You have been given 2 parallel lines
→ These 2 parallel lines draw max possible products
→ You draw such a product that is much more creative.

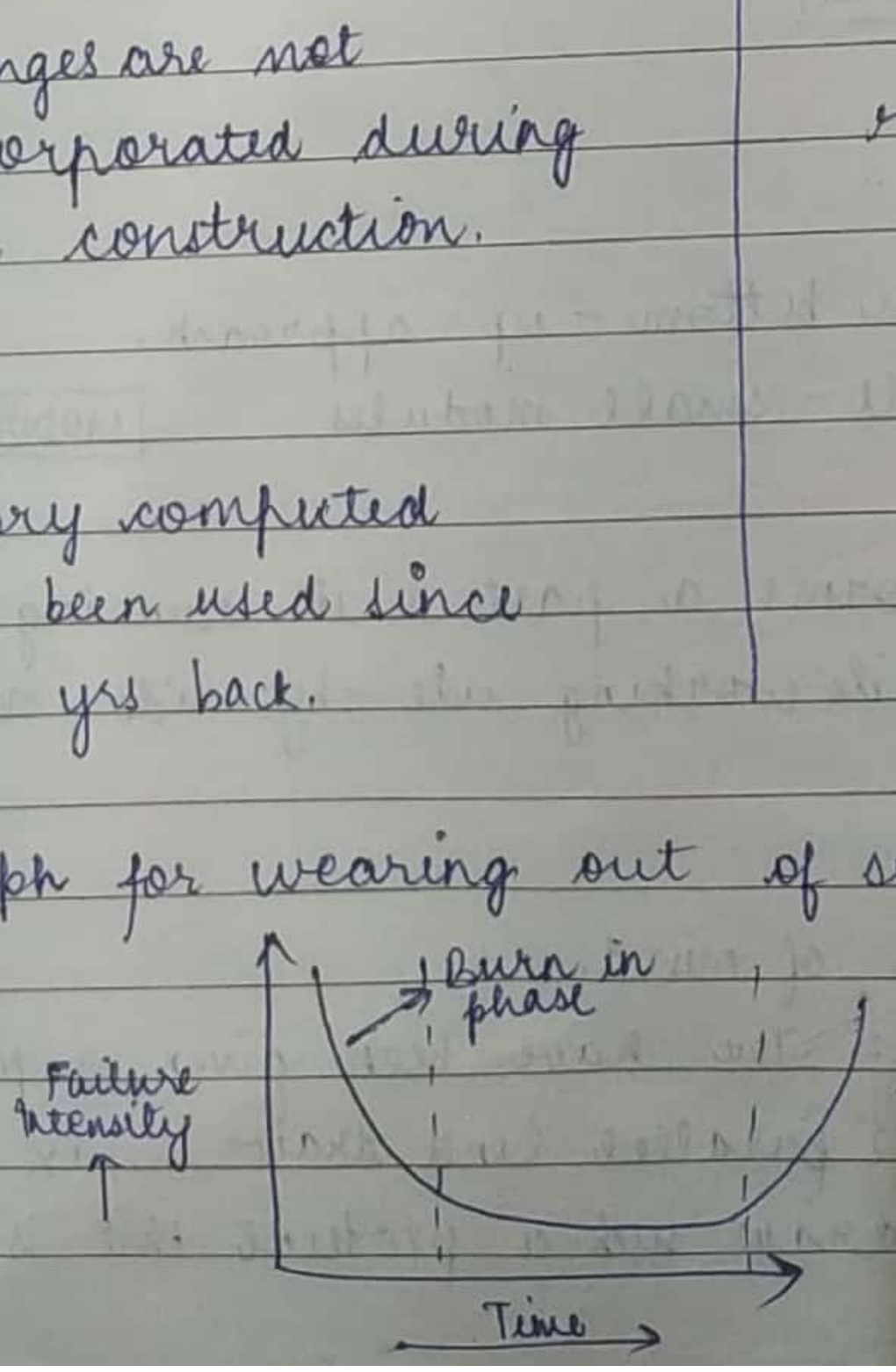
1. Good analytical bent of mind
2. Good communication skill
3. Motivator
4. Conflict resolver
5. Politician
6. Problem solving skill.

S/W characteristics.

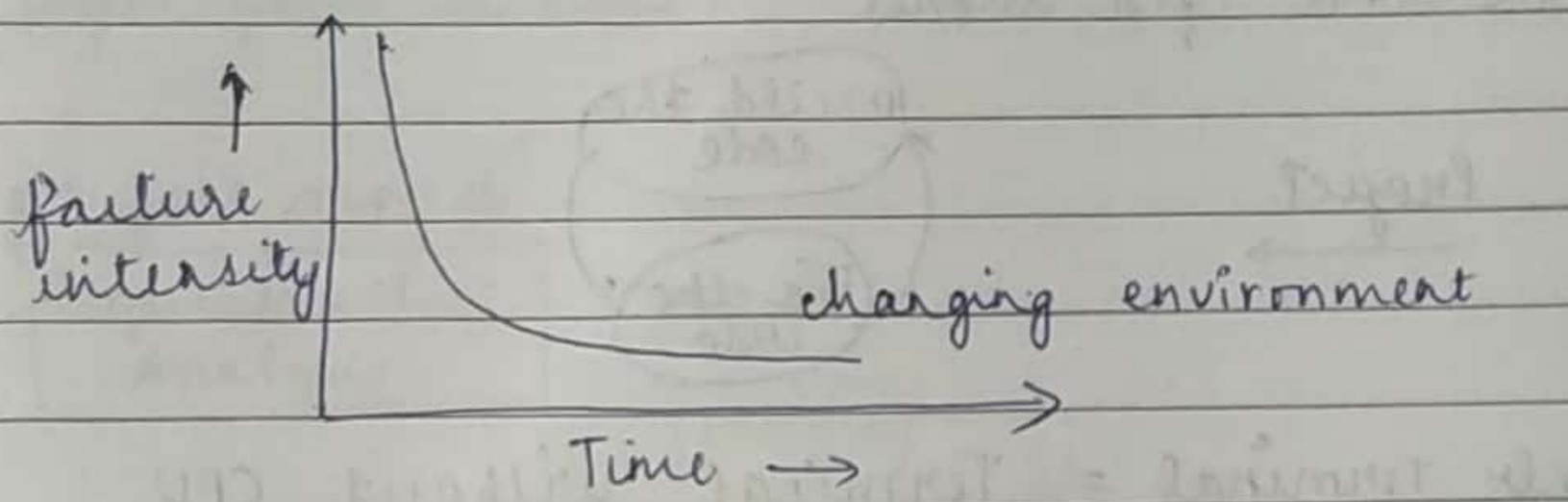
1. S/W does not wear out.
2. software flexibility
3. Reusability of component
4. S/W is not manufactured.

construction of Bridge	Development of S/W.
1. Entire steps or concepts are known.	Some steps are known.
2. Changes are not incorporated during the construction.	Changes may be requested by customers and entertained by developers at any instant.
3. Theory computed has been used since 1000 yrs back.	Theory computed has been used since 50 yrs back.

Graph for wearing out of software :-



a) Starting of project - Burn in phase → max faults
b) Useful life phase - No faults
c) wear out phase - failure



Software development Life cycle (SDLC)

a) Analysis ⟵ System Analysis → Requirement elicitation - Gathering of data.
b) Design → Data Flow diagram (DFD), E-R diagram, flowchart
c) Coding
d) Testing (process of evaluating software with intend of finding errors).
e) Implementation, Maintenance, Review

Data Gathering techniques:-

(i) Questionnaires
(ii) Interviews → Structured / Unstructured
(iii) On-site observations.

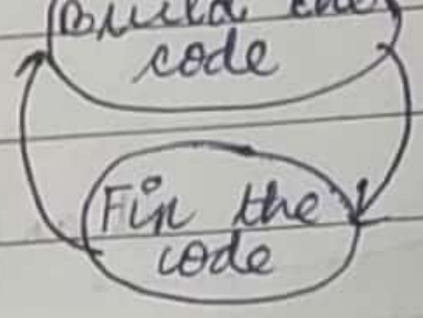
Models used for building software :-
(Next pg)

DATE / /
NOTEBOOK

Models by which we can develop a software

10 Build and fix model

Project →

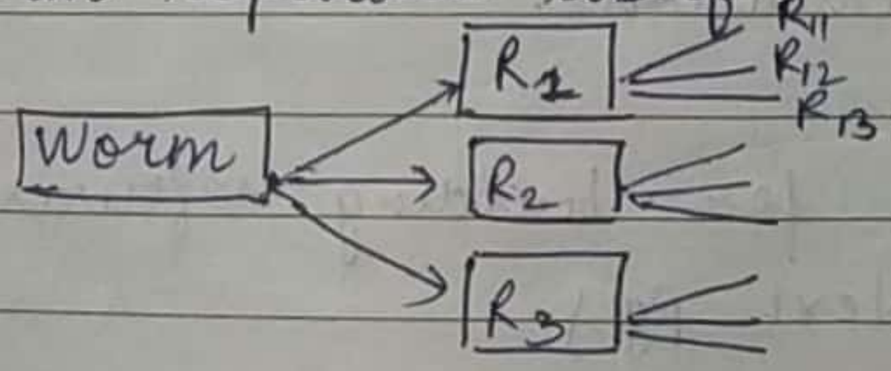


→ Dumb Terminal = Terminal without CPU
→ Intelligent = Terminal with CPU

Good if size is small
Bad as only fixed a small portion of error not whole.

Virus: Piece of code which attaches itself with legitimate program & when l.p. execute, it also execute thats why it affects the files and file allocation table.

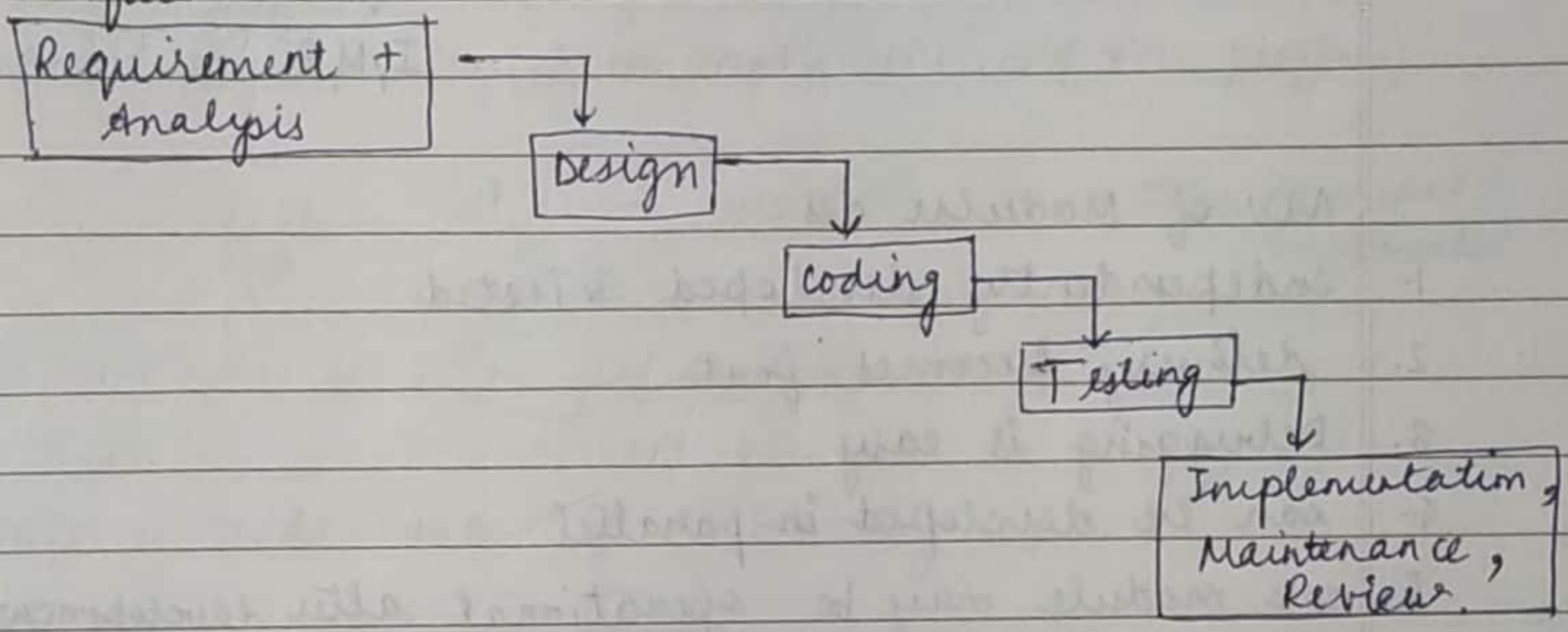
Worm: → does not perform any destruction, worm replicates itself



→ After certain interval of time, creates too much replica and forces the system to go into heart stage (shuts down)
→ After worm gets deleted, system gets functional.

Trojan horse: Works as spy & steals data from our comp to owner's computer.
Sits silently in login and gets its information as the login port executes.

Waterfall model



Feasibility study

1. Economic feasibility
   (cost / benefit analysis) → Tangible & Intangible costs

2. Technical feasibility
   →
   Eg: Java / C++ expertise

3. Operational feasibility
   → is it feasible to incorporate or not.

4. Social feasibility
   →

5. Behavioural feasibility
   → Blue whale, Pokemon GO

6. Legal feasibility
   →

Soft. Engineering by KK Agarwal & Yogesh Singh
Pressman.

DATE 21/9/21 PAGE
NOTEBOOK

Analysis → Design → Modular design
Design → Coding → Testing → I, M, R

Adv of Modular des.

1. Independently developed & Tested
2. delivery becomes fast
3. Debugging is easy.
4. can be developed in parallel
5. one module may be operational after development
6. development time can be reduced.

Draw the ER diagram of Library management system.
(Next pg)

Prototype model

Advantages:

a) Accomodate the changes / new requirements.
b) Greater level of satisfaction of customer.
c) Better - quality product due to involvement of customer.
d) Customer & developer both will understand system better.

Drawbacks:

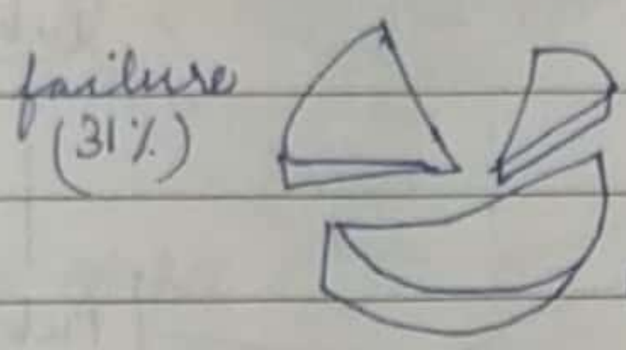
a) Time - consuming process.
b) Cost is increased.
c) Poor documentation
d) Complex design.
e) conflict may arise b/w developers & customer
f) customers may demand a large no of projects in a short interval of time

Why software engineering ?

1. Change in nature & complexity of software
2. Concept of 1 "guru" is over.
3. We all want improvement. ⟶ Ready for change

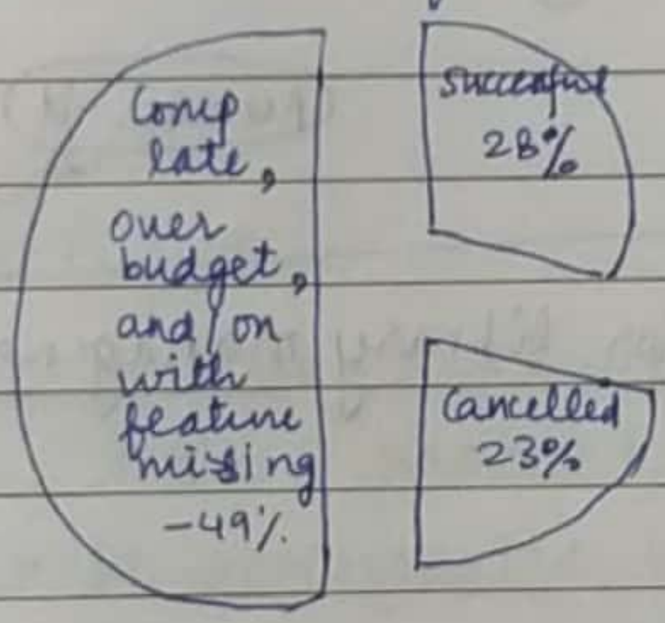
The evolving role of software

Software industry is in crisis (Y2K problem)

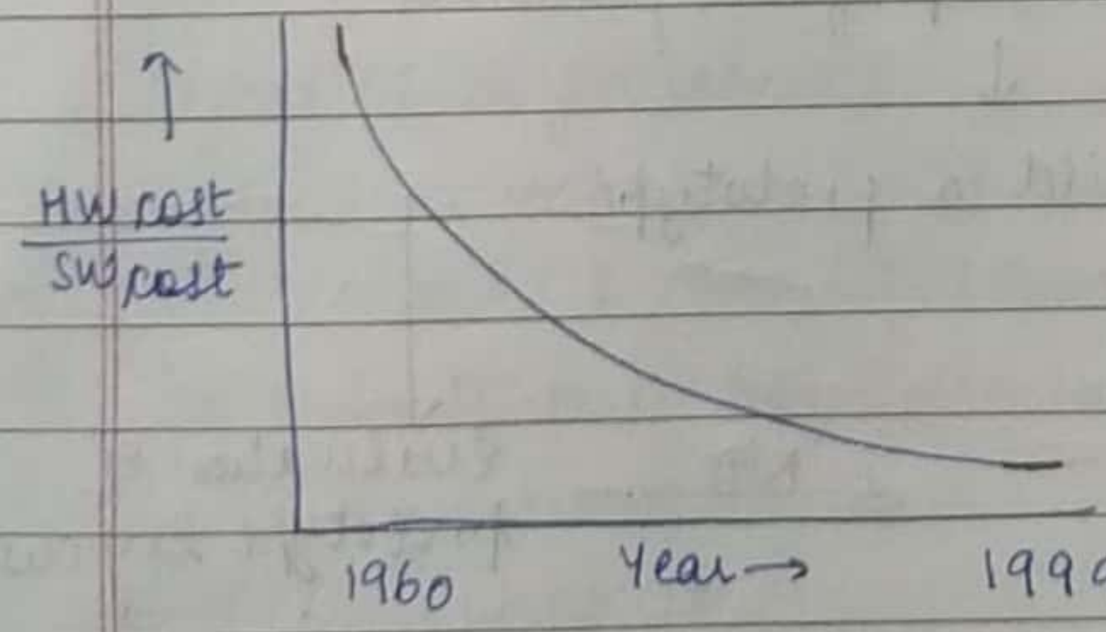


→ in time developed & implemented.

This is SORRY state of SWE today!



Data on 28,000 projects in 2000.

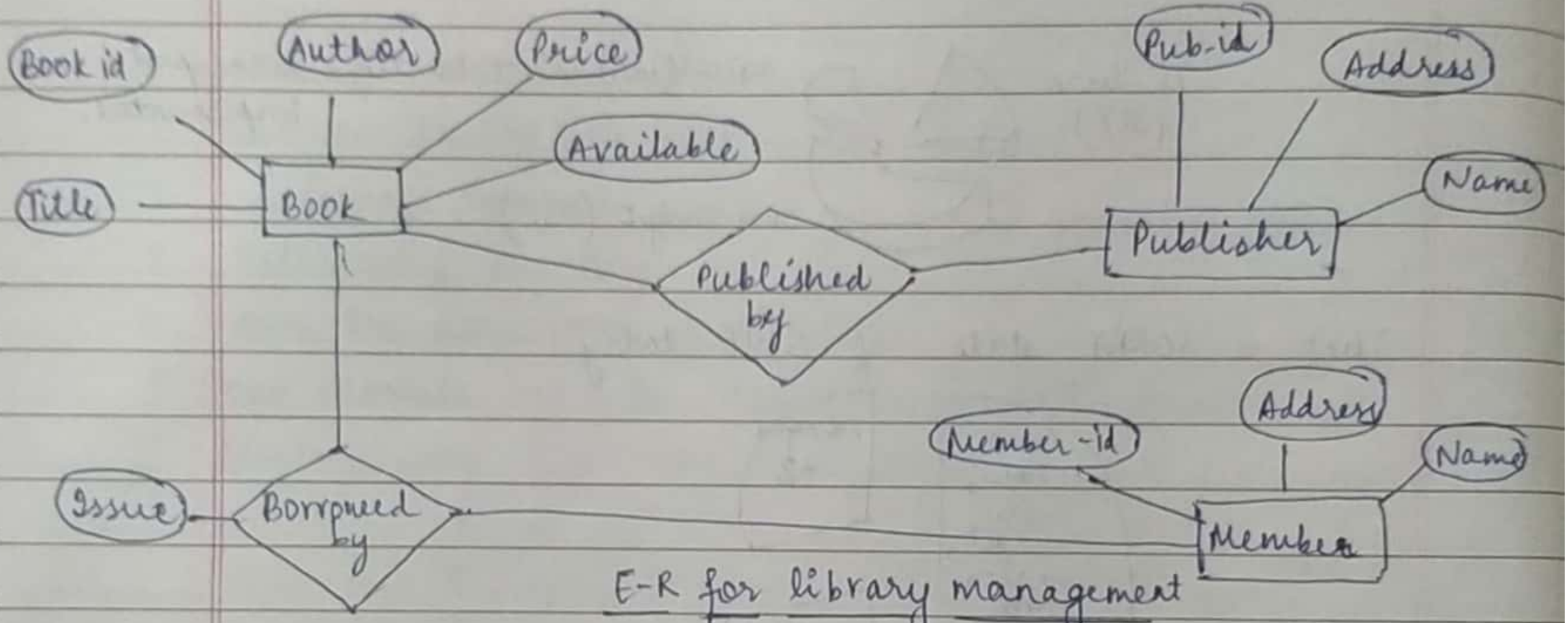


Relative cost of Hardware & Software

Advantage of waterfall model : a) suitable for small projects where requirements can be gathered early in the phases.

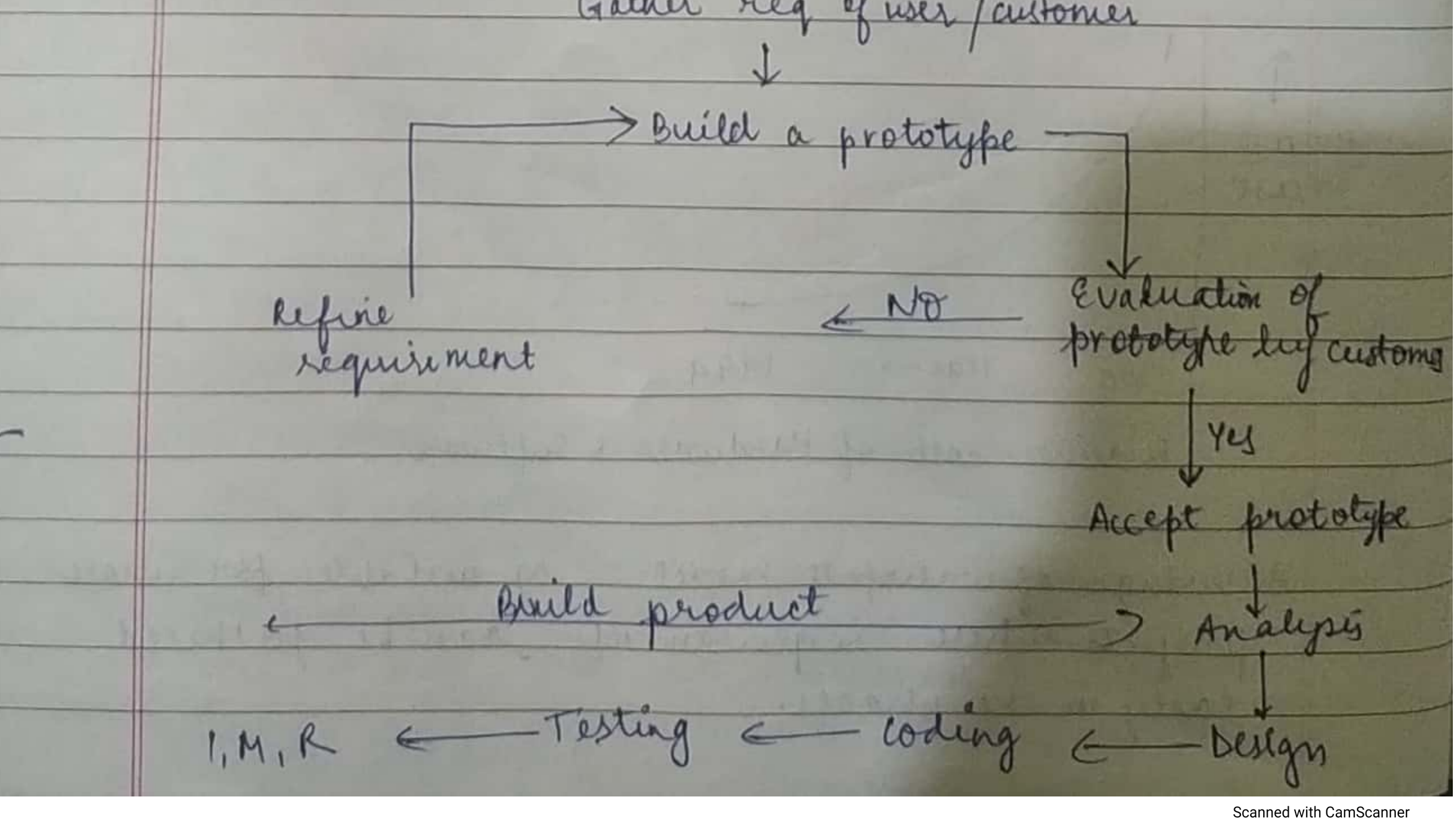
Drawbacks:

a) It is very difficult to gather entire requirements in beginning of development.
b) changes cannot be incorporated in b/w development phase.
c) does not scale up the system.
d) Projects are rare to develop in sequential model.



E-R for library management

3. PROTOTYPE MODEL

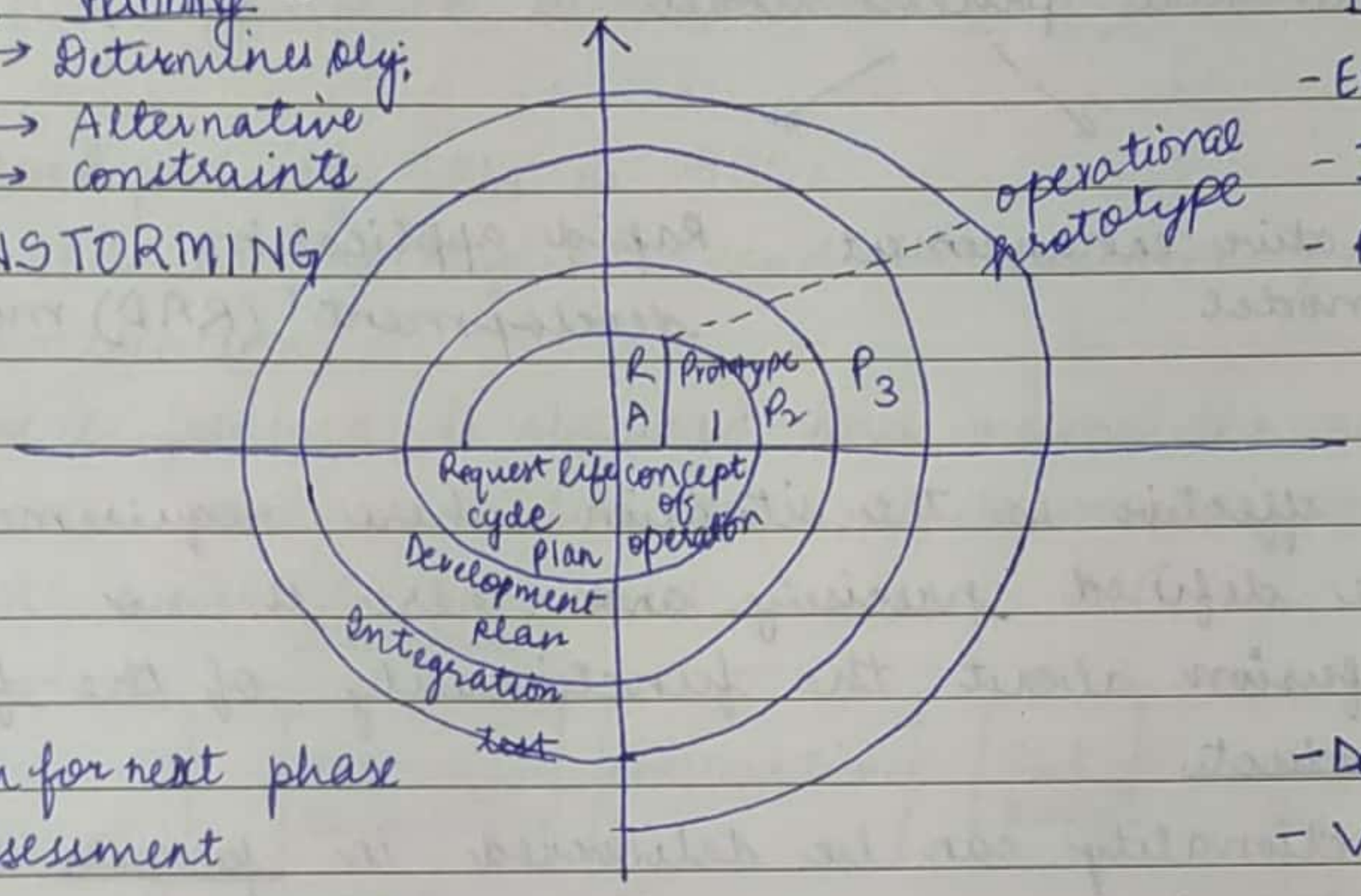


DATE / / PAGE
NOTEBOOK

SPIRAL MODEL

I Planning -
→ Determines obj.
→ Alternative
→ constraints

BRAINSTORMING



II Risk analysis
- Evaluate alternati
- Identifying -ve risks
- Resolve risks.

III
- Develop
- Verify next level product.
- Design (DFD, FC, ER)

IV
→ Plan for next phase
Assessment
→ Customer evaluation

Adv + Disadv.
customer satisfaction
High amt. of risk analysis
Mutual for large products
Changes can be accommodated in any phase.

Time consuming

Types of requirement
→ Known, Unknown and undreamt requirements.
→ Functional & Non-functional requirement
→ Users & System requirement.

FUNCTIONAL
↳ what user wants to do?
eg: → generating names & roll no.
→ product feature

USER
what are the requirements of user.

↳ well
how it does, op" performed inside the computer.
eg: retrieving & gathering data.
→ Quality.

SYSTEM
Generated with help of user

DATE / / PAGE
NOTEBOOK

☆ Incremental process model

Iterative enhancement model

Rapid application development (RAD) model.

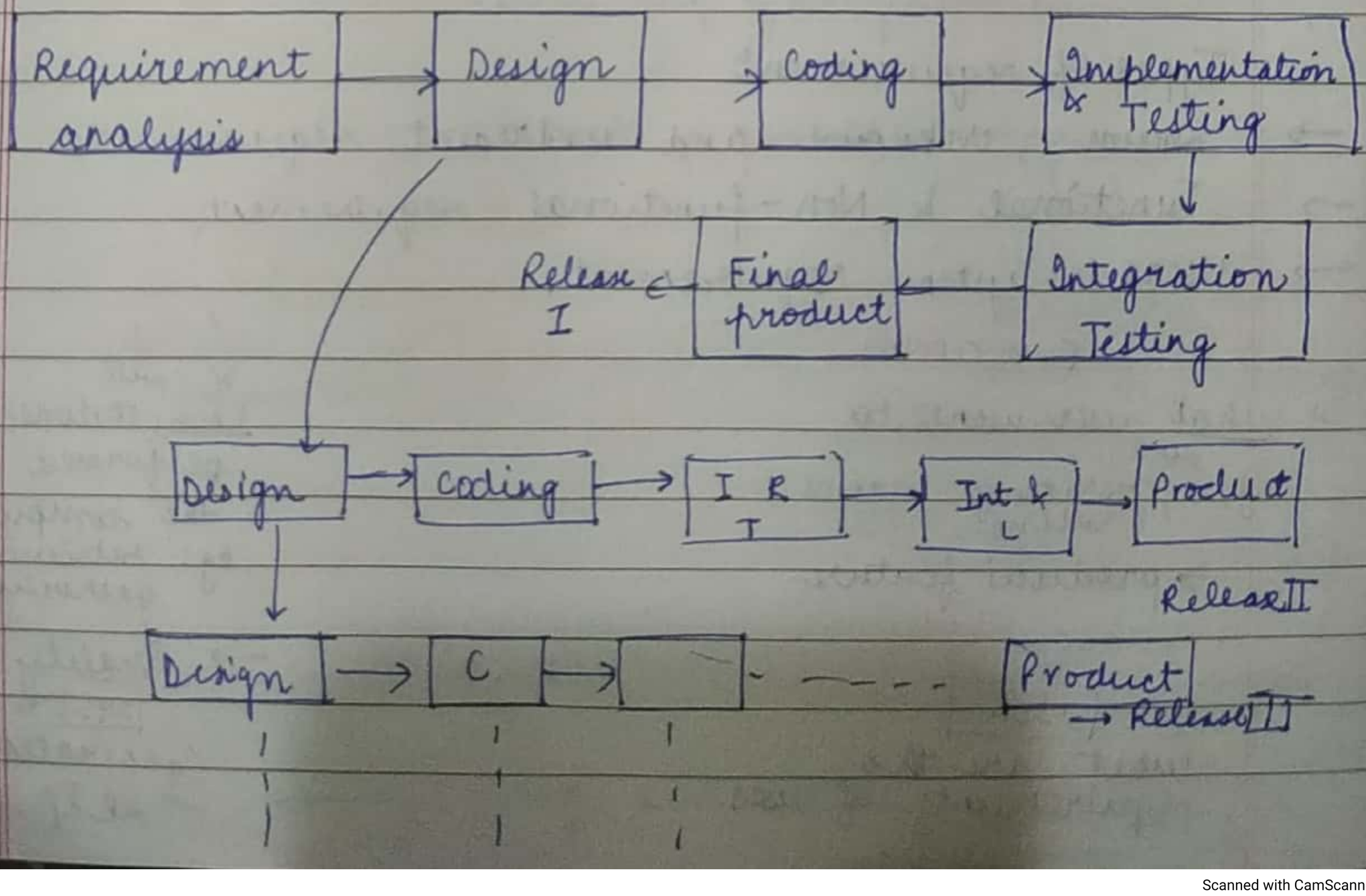
→ are effective in the situation where requirements are defined precisely and there is no confusion about the functionality of the final product.

→ Functionality can be delievered in phases as per desired priorities.

→ After every cycle a usable product is given to customer.

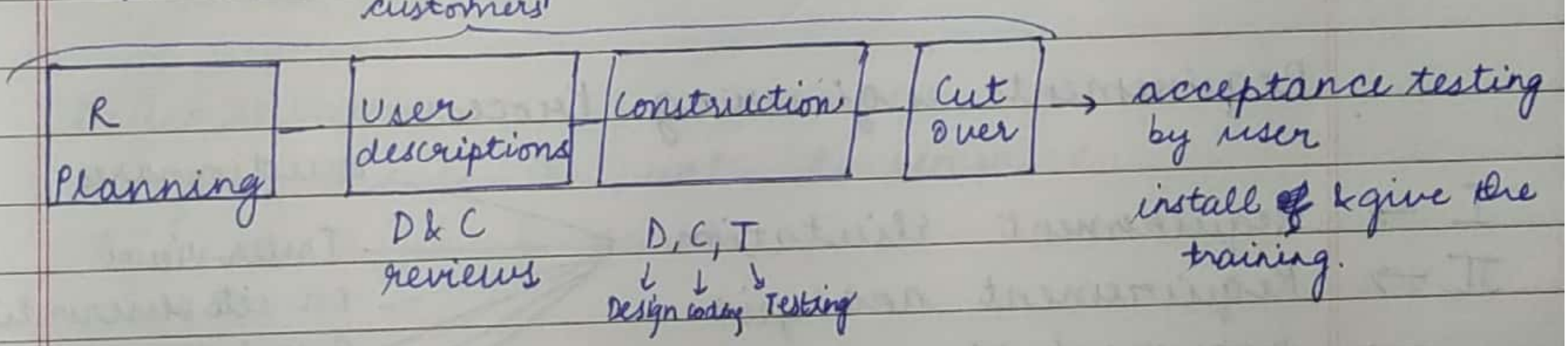
→ More popular when we have to quickly deliver a limited functionality system.

A) Iterative enhancement model



B) Rapid Application Development (RAD) model.

— Developed by IBM in 1980s
— Build a prototype & send to customer/user for evaluation.
— User's feedback is obtained and refine the requirement.
— Process continues till the requirements are finalized.
— SRS will be prepared.



Diff b/w RAD & Prototype

RAD	Prototype.
Customer is involved throughout the whole process until the product is delivered. ∴ less time consuming.	in requirement analysis. ∴ More Time consuming

DATE / / PAGE
NOTEBOOK

☆ Evolutionary Process Model

Prototype | Spiral

Incremental	Evolutionary
Can be delivered in phases	Delivered as a whole.

Requirement engineering Process

I → Requirement Elicitation
II → Requirement analysis
→ Documentation
→ E-R diagram
→ DFD
→ SRS
→ IEEE format of SRS

- questionnares
- Interviews
- On site observation
- Brainstorming sessions
- FAST
- Use case **technique**

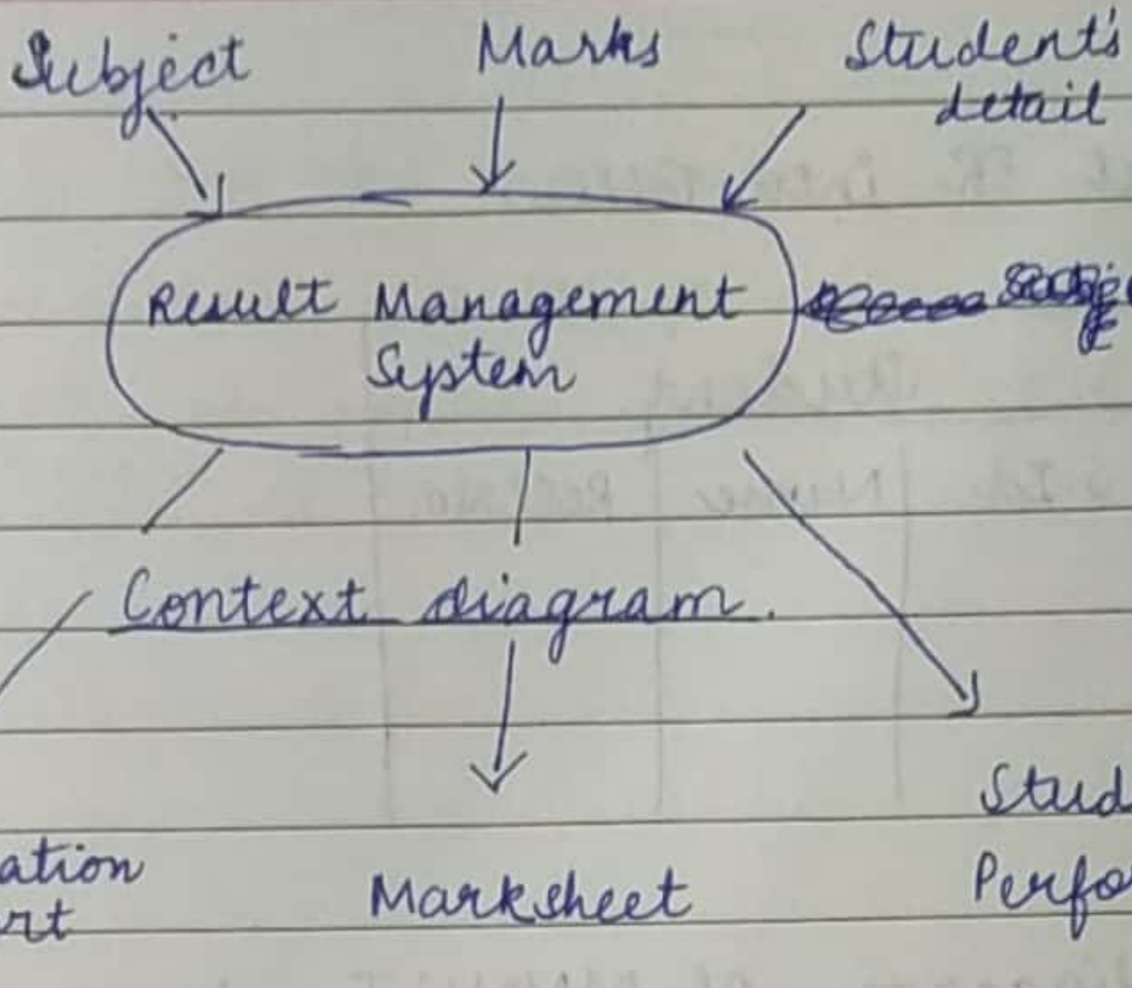
No criticism is allowed in FAST.

II Requirement → Requirement Analysis

- Draw the context diagram
- Develop the prototype (optional)
- Model the prototype
- Finalize the requirement

: Analysis is the phase which requires 60% time as once you've analyzed fully, designing & coding becomes easier.

→ You can't undo anything wrong.



III Documentation

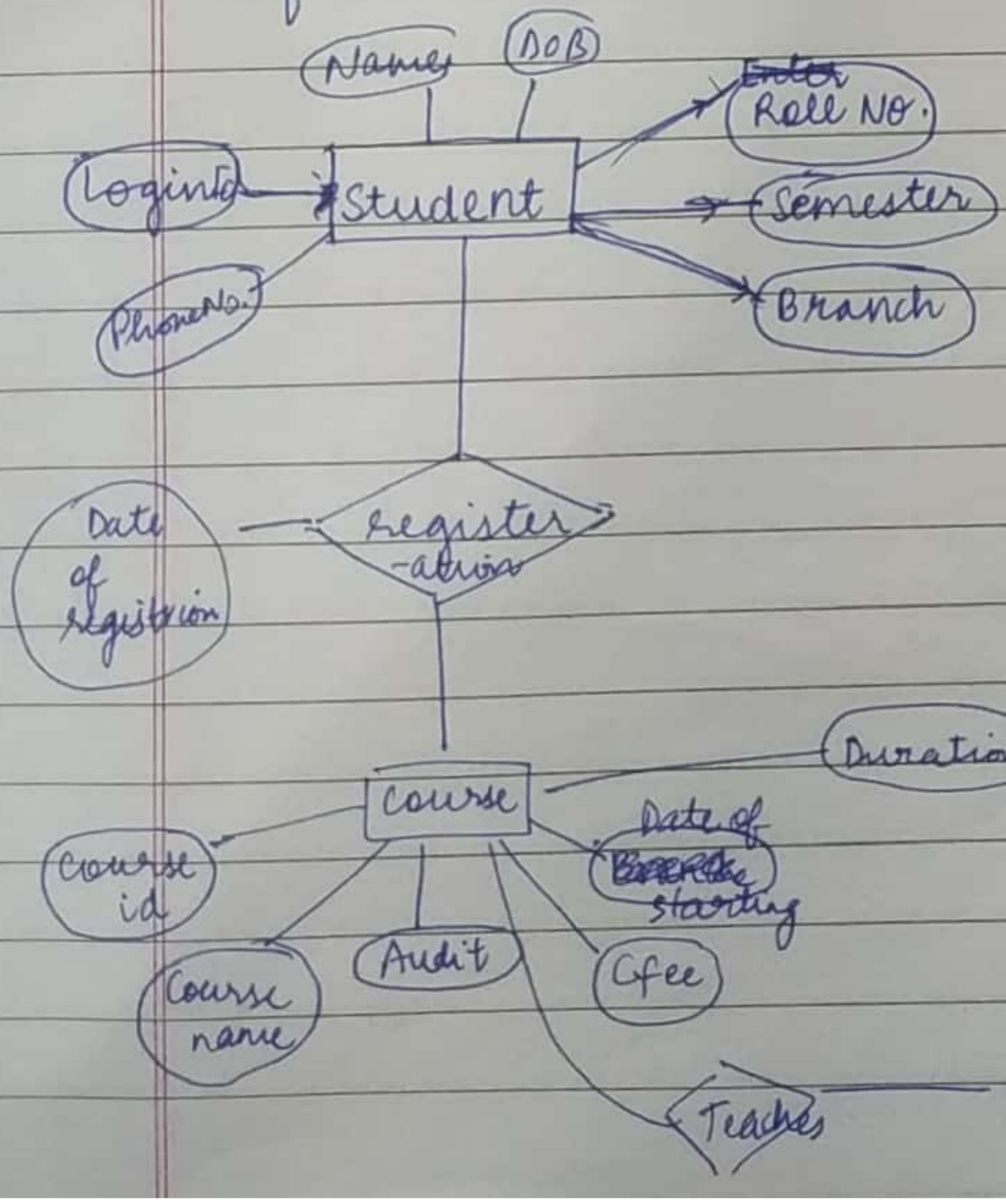
Everything should be well documented.

Eg : Mobile phone manual.

IV ER diagram

Q1) Draw ER diagram for the course registration process of student.

1) Entities → Stud, course
2) relationship → register
3) Attributes with entities



DATE 8/10/21 PAGE
NOTEBOOK

Requirement Documentation

→ Very important comes after elicitation & Analysis.

→ Way of representing the requirements in specific format.

→ SRS – Software Requirement Specification

→ SRS written by Customer → Need & Expectations of customers.

→ SRS written by Developer. → acts as a contract b/w C & D.

Nature of SRS

→ Functionality → what s/w is supposed to do.

→ External interface → how s/w interact with hardware/people.

→ Performance → speed, availability (if you're authentic users, resources of server must be present for you encryption), Response Time

→ Attributes → Quality (correctness, Portability, Reusability, Maintainability, Security)

→ Design considerations for implementations.

(Integrity constraints, Operating environment, Implementation language)

Characteristics of good SRS

→ Correct – s/w meets customer requirement.

→ Unambiguous – must be single interpretation.

→ Complete – Functionality, EI, I, A, DC must be present.

→ Consistent – If a data is changed, it must be modified everywhere.

→ Modifiable

→ Traceable → Backward Traceable, Forward Traceable } For this we must have reference no.

→ Ranked for Importance – Stability

→ Maintainable

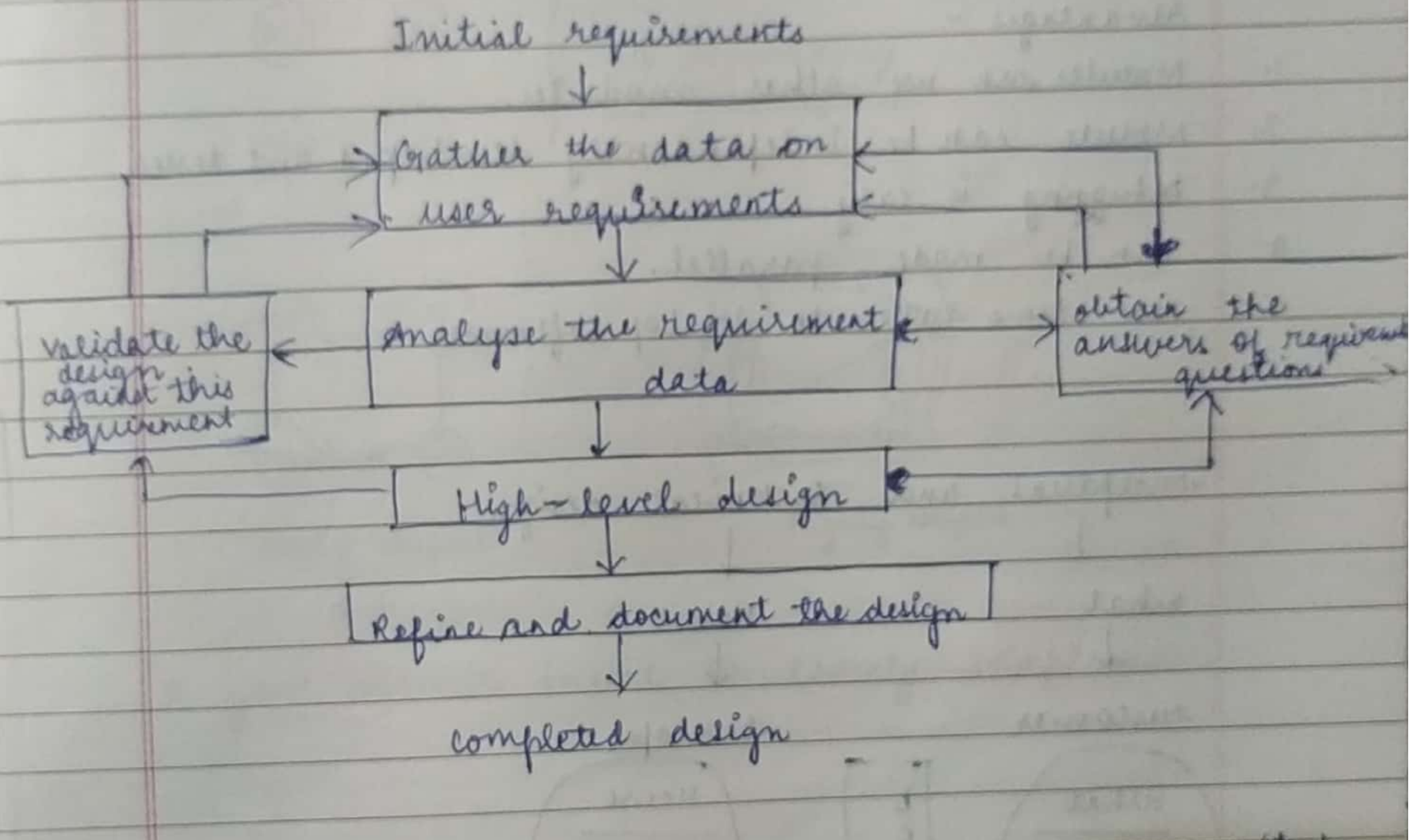
CUCRT MM
Cricket Ring Tube MM

DATE 27/8/24 PAGE
NOTEBOOK

Basic issues in design

SRS = what , design = how to make
SRS becomes the input of design phase.

Design framework →

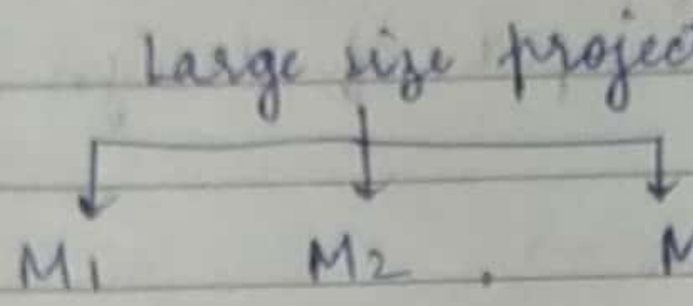


Design needs to be → correct and complete.
→ Understandable
→ At the right level
→ Maintainable

(can be only correct but incomplete)

(User must be able to understand)

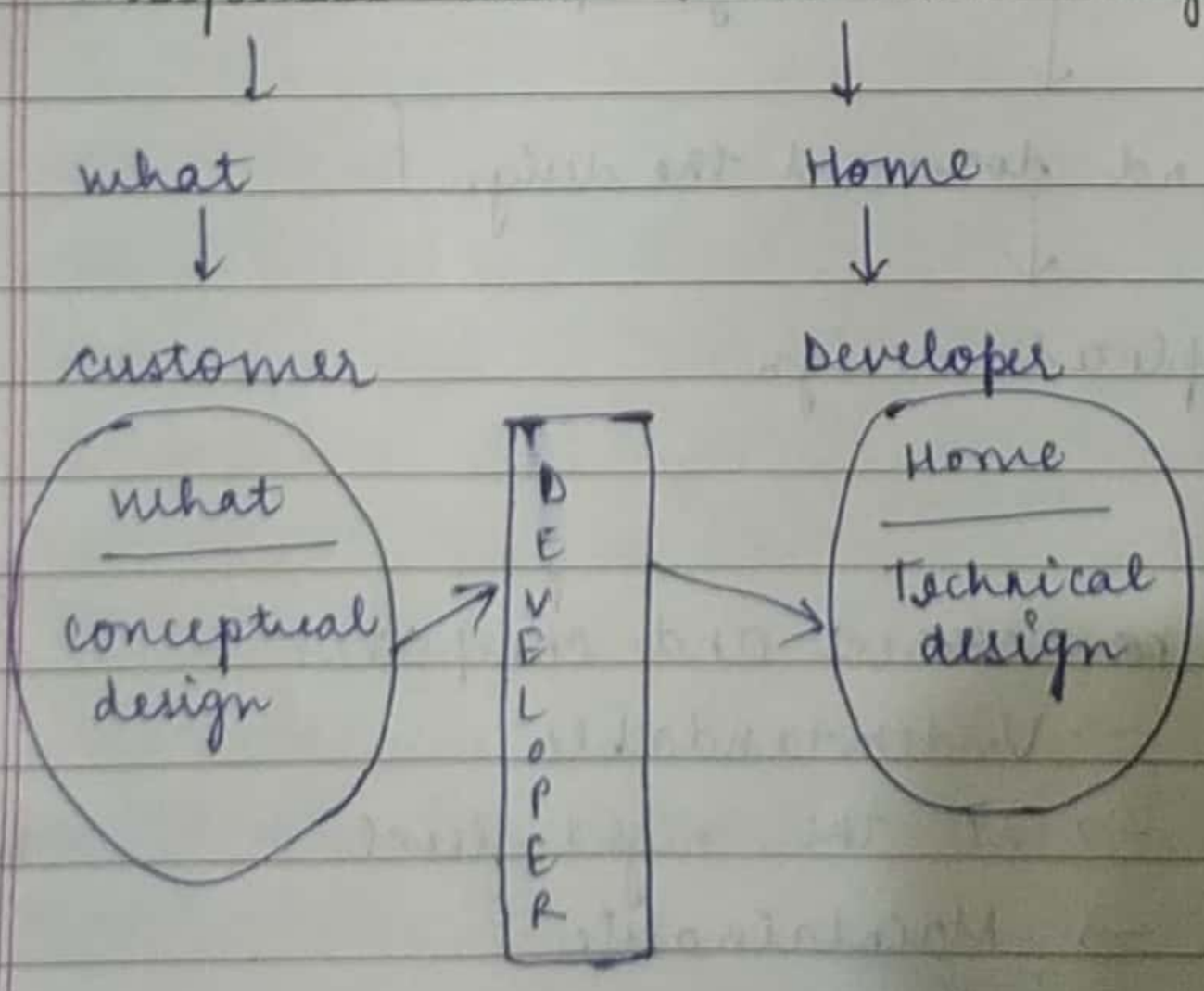
Modularity
Process of dividing projects into small-small modules.



Advantages -

1. Modules can use other modules.
2. Module can be independently developed and tested.
3. Debugging is easy.
4. Can be made parallel.
5. Less time taken to develop software.

Conceptual and technical design.

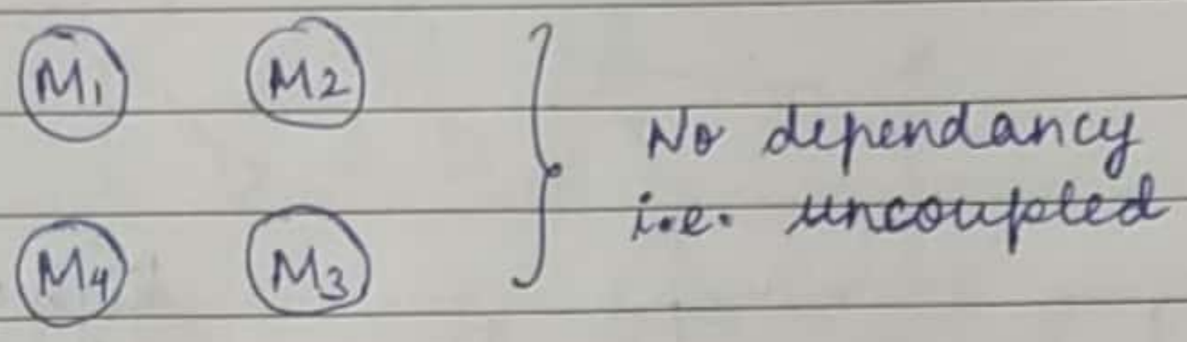


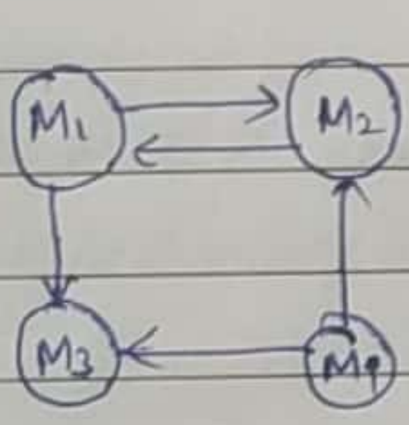
DATE / / PAGE
NOTEBOOK

Module coupling (Inter-dependance of/bet" 2 modules)

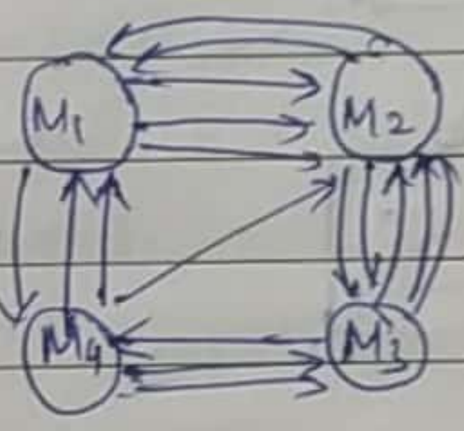
M_1, M_2 — Tightly or Highly coupled. (If highly inter dependant)

M_1, M_2 — less dependant (If loosely coupled design)





Loosely coupled



Highly coupled

A good software must be loosely coupled.

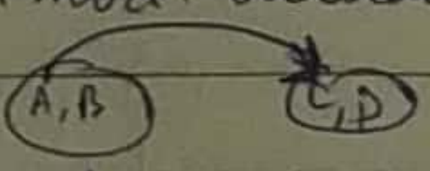
Types of coupling

- Data coupling
- Stamp coupling
- Control coupling
- External coupling
- Common coupling
- Content coupling

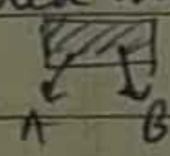
↑ Best

Worst (as dependancy is increasing)

- Data coupling → A & B are said to be data coupled if they only pass data from 1 mod. to another.
- Stamp coupling → If they are only passing data structure.
- Control coupling → Control structure passing
- External coupling → depends on mod. outside project



- Common coupling → If 2 mod. are working in shared mode.



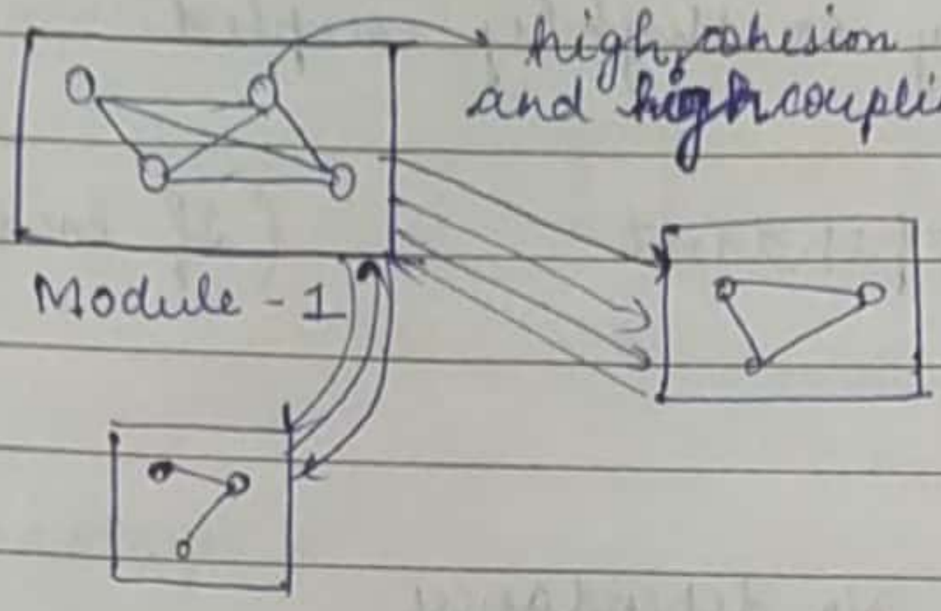
- Content coupling → entire content sharing (data, d.s., logic ...)

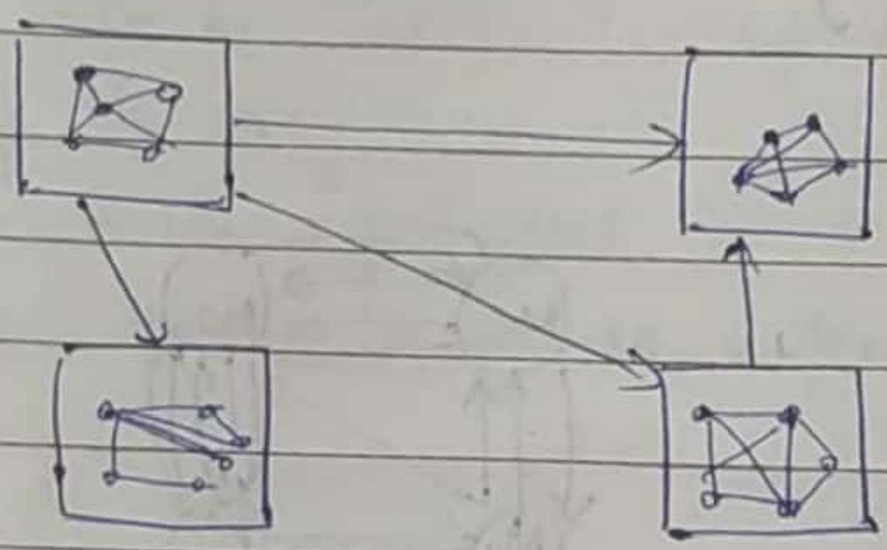
BEST : Low coupling & high cohesion

DATE / / PAGE NOTEBOOK

Cohesion

Interdependancy inside the module.





High cohesion & low coupling

Types

Best ↑

- Functional cohesion — There is single input & single output
  A → B → C
- Sequential cohesion
- Communicational cohesion
- Procedural cohesion — All elements must apply same procedure x, y → A
- Temporal cohesion — Temporal - Geographical / Spatial
- Logical cohesion — All modules must implement same logic.

Worst ↓

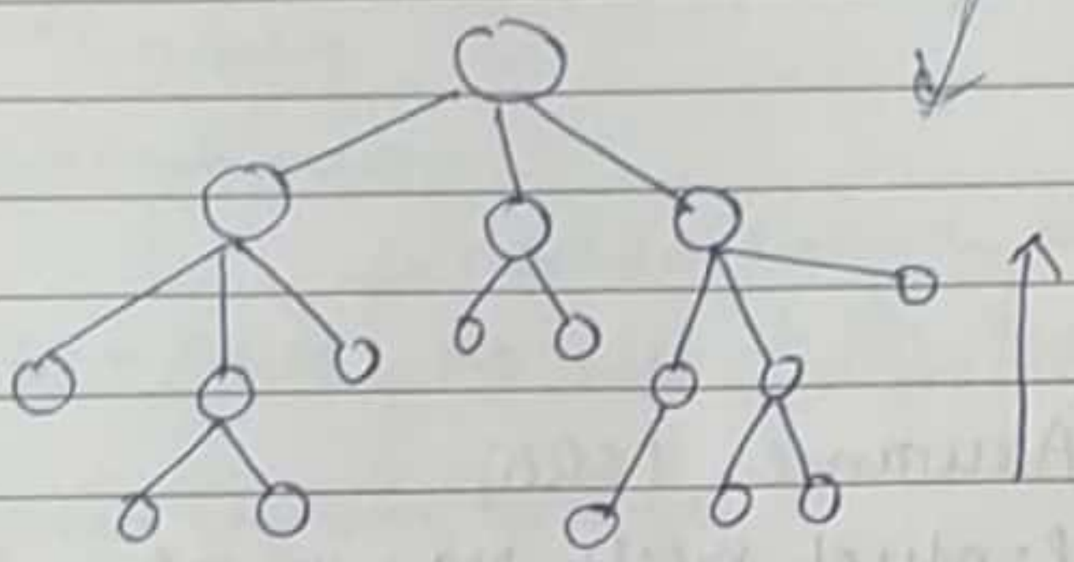
- Coincidential cohesion — Very less dependency bet^n elements of modules.

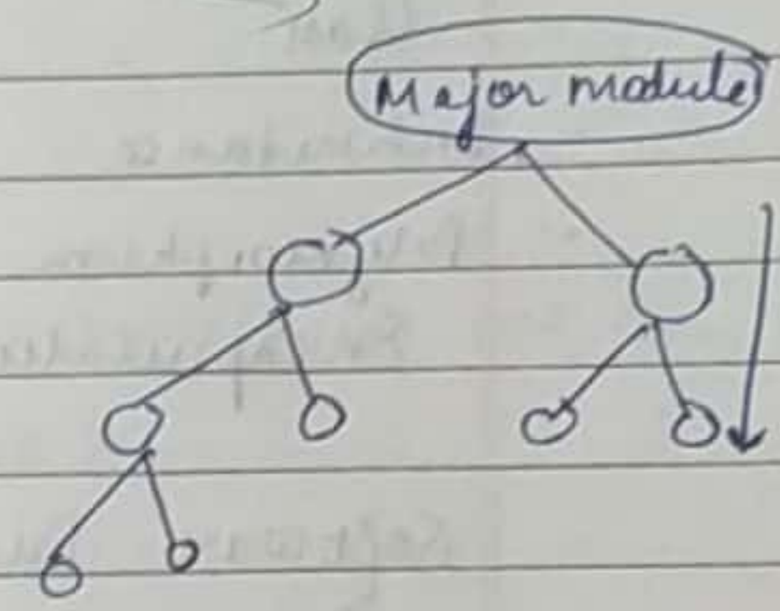
DATE / / PAGE
NOTEBOOK

Strategy of design

→ Bottom-up designing

→ Top-down designing

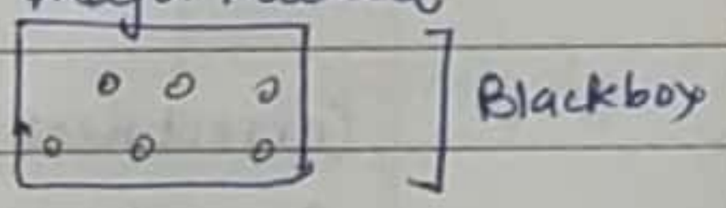




adv:
→ module is small, it can be independently developed / tested.

dis:
→ fault in one module makes whole project faulty.

adv:
If you're dividing the major module there are refinements



dis:
- deals with all comp together.
- Testing can't be performed individually.

Hybrid approach of s/w desig" : depending upon situation we use Top-down + Bottom-up

DATE / / PAGE
NOTEBOOK

Object Oriented approach designing

- Object
- Message
- Abstraction
- Class
- Inheritance
- Polymorphism
- Encapsulation

Software Quality Assurance (SQA)

Quality → Product meets requirements
Conformity of requirement
Level of satisfaction

20 software quality attributes form software quality attributes

Correctness	✓
Consistensy	✓
Robustness	
Simplicity	✓
Traceability	✓
Usability	✓
Accuracy	
Clarity	
Conformity	
Completeness	✓
Efficiency	✓
Testability	
Maintainability	✓

CDC CBT MKM
FERUMP
RAT MACCE
Modularity
Adaptability
Testability
Robustness
Accuracy

C = Clarity
C = Conformity
E = Expandability

DATE / / PAGE
NOTEBOOK

Modularity
Readability ✓
Adaptability
Modifiability ✓
Expandibility
Portability ✓
Unambiguous ✓

SEI - CMM → Capability Maturity Model, developed by SEI, USA, 1986
module & ISO 9000 model
↓
Software Engineering Institute

- CMM improves s/w process & improves quality of software
- Five level of CMM

Level 5	- Process change management - Technology change management - Defect prevention	Optimizing
Level 4	- Software Quality Management - Quantitative Management	Managed
Level 3	- Peer Reviews - Inter-group co-ordination - Organization processdef" - Organization process focus - Training programs.	Defined
Level 2	- Project Planning ~~No KPA's~~ - Configuration Management - Requirement management	

DATE / / PAGE
NOTEBOOK

Level 2 | - Sub-contract Management
- Software Quality Assurance | Repeatable

Level 1 | No KPA's | Initial

Level 1 : Initial

1. No KPA's defined
2. Processes followed are Adhoc & immature & not well defined.
3. Unstable environment for software development.
4. No basis of predicting product quality, time for compl" etc.

Level 2 : Repeatable

(basic project management policies).

Project planning - defining goals, ~~no~~ resources, constraints etc.
Config" management - maintaining performance.
Requirement management - management of customer reviews & feedback
Subcontract ma" - effective management of qualified contr
Software Quality Assurance - guarantees good quality software by following certain rules & quality stand

KPA

Level 3: Defined

(documentation of standard guidelines).

Peer reviews - defect removed by review method like - walkthrough, inspection, buddy-check.
Intergroup co-ordination - planned interaction bet" diff. dev. teams
Organization process def" - development & maintenance of standard dev. process
Organization process focus - activities & practices ~~that~~ to improve process capability.
Training programs - enhancement of knowledge and skill of team members.

DATE / / PAGE
NOTEBOOK

Level 4 : Managed. (setting quantitative goals)

Software Quality Management - establishment of plans & strategies to develop quantitative analysis.

Quantitative management - controlling project performance.

Level 5 : Optimizing. (continuous process improvement using quantitative feedback).

Process change management - improvement of organization software procedures.

Technology Change management - identifⁿ & use of new technology

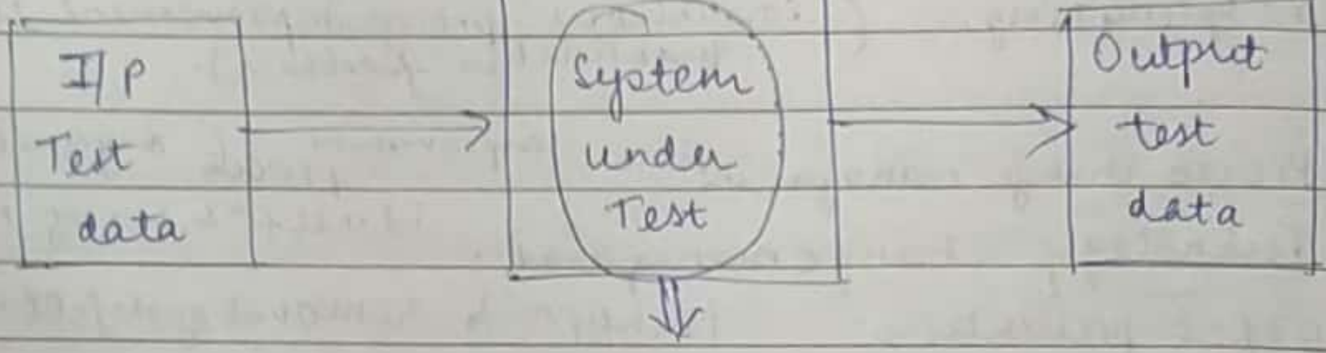
Defect prevention - identifⁿ & removal of defect.

|RDMO

DATE / / PAGE
NOTEBOOK

Software Testing

1) Black Box Testing (I/O) (functional testing) also known
2) White Box Testing (component level testing).
3) Robustness Testing (Extreme condition)
4) Sample Testing (No fix)
5) Random Testing



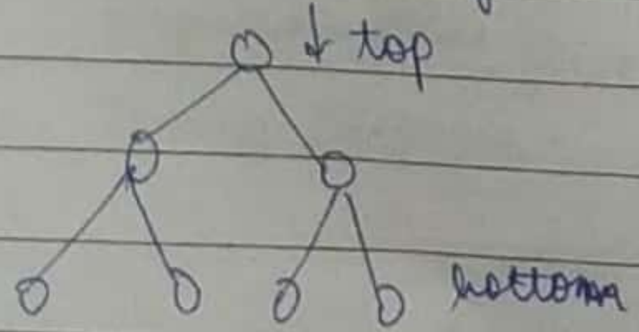
Various functionality ~~testing~~ taking place inside the system.

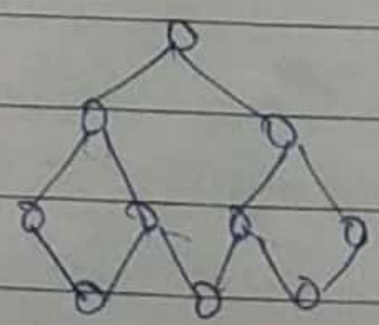
Every parts will be tested independently.

Level of Testing

→) Unit Testing (Independently developed)
→ Integration Testing (check any error occur during)
→ System Testing (& → more things)

→ top-down integration testing
→ bottom-up integration testing
→ Sandwich integration testing





DATE 29/11/21 PAGE
NOTEBOOK

Test, Test case and test suite

Test case: It describes the input and expected output description

Test Case ID

Section - I Before execution	Section - II After execution
Purpose: Pre-conditioning: Inputs: Expected o/p: Past condition: Written bug: Date:	Execution History: Result: if fails, any possibility reason (if any) Any other observation Any suggestion Run By Date:

Test Case Template

SOFTWARE MAINTENANCE

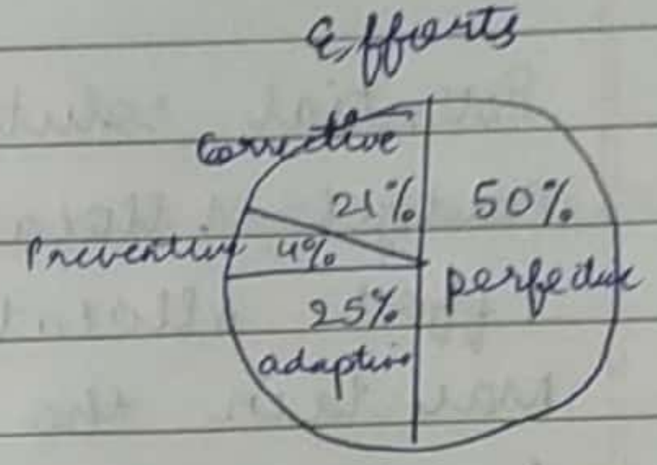
1. Process of detecting errors in software & removal. (error correction)
2. Enhancing the capabilities of software.
3. Deletion of obsolete capabilities from s/w.
4. Changing the Environment. / optimization (max output in min effort)

LOC : line of code (less is better)

DATE 30/11/24 PAGE
NOTEBOOK

Type of maintenance

→ Corrective maintenance 21%.
→ Adaptive maintenance 25%.
→ Perfective maintenance 5%.
→ Preventive maintenance 4%.



Effort: no. of person required per month to develop a particular software.

Corrective:

Adaptive: Software is modified to adapt to the changes of ever-changing environment.

Perfective: We're gonna improve the efficiency, performance as well as modify something in existing software so as to increase its efficiency.

Preventive: To reduce complexity of software.

Problems associated with S/W maintenance

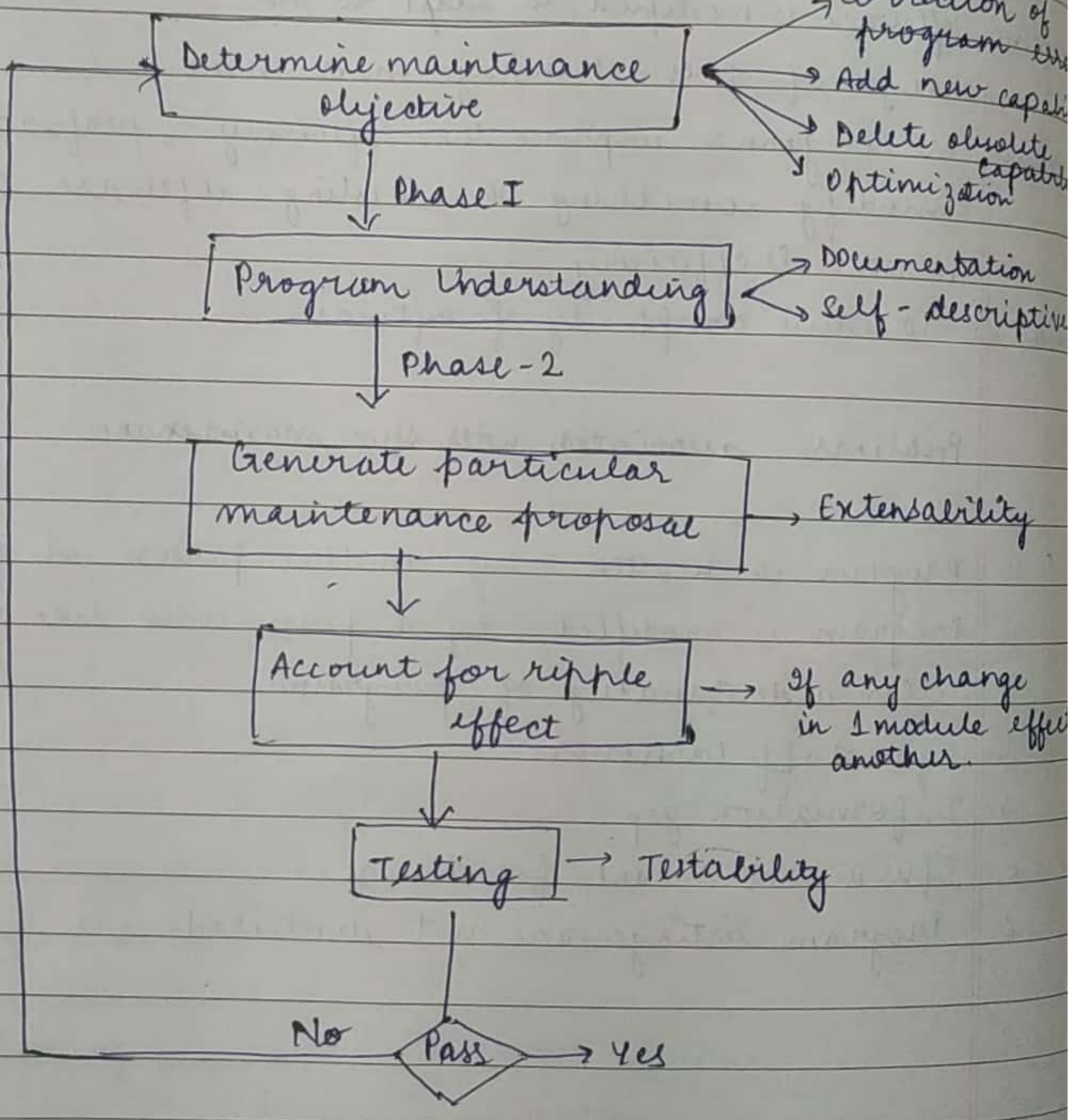
1. Program is written by another person or group of person.
2. Program is modified by a person who does not have clear understanding of program.
3. High staff turnover
4. Information gap.
5. S/W is not meant for change.
6. Program listings are not structured.

DATE 24/12/24 PAGE NOTEBOOK

Potential solution of S/W.

1. Budget Allocation
2. Effort Allocation
3. Maintain the existing system
4. Complete replacement of software

Maintenance Process



Case study

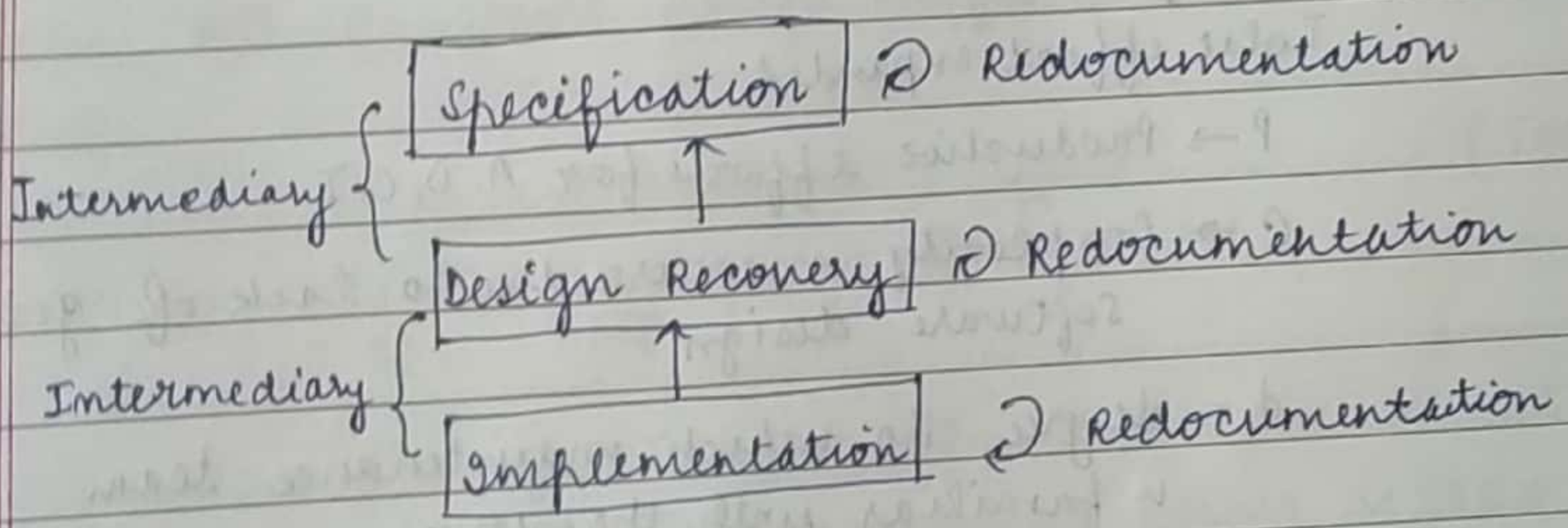
2) You are given a problem - Generate maintenance proposal model for a software which is being used in examination section of MMMUT.

DATE / / PAGE
NOTEBOOK

can do in web d. (analyze code)

Reverse Engineering

Process of finding difficult, unknown and hidden information in the software.



DATE 6/12/21 PAGE
NOTEBOOK

$pc = P + Ke^{(c-d)}$
$M = P + Ke^{(c-d)}$
$M = P + Ke^{(c-d)}$

Cost of maintenance

Belady Model

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

Total effort expanded

P → Productive efforts for A, D, C, I ----

C → Complexity measures due to lack of good Software design

d = degree to which maintenance team is familiar with the software.

Q) The development effort for a software project is 500 pm (person month). ~~Infinitetta~~ We empirically Define constant & K = 0.3. The complexity of the code is quite high & equal to 8. Calc. the total effort and if

i) maintenance team has put good level of understanding of project (d = 0.9)

ii) maintenance team has poor level of understanding of project (d = 0.1).

A) i) K = 0.3, C = 8, d = 0.9, P = 500

$M = 500 + 0.3 \times e^{(8-0.9)}$

$= 863.5901$ PM

ii) $M = 500 + 0.3 \times e^{(8-0.1)}$

$= 1309$ PM

DATE / / PAGE
NOTEBOOK

Boehm Model

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

ACT → Annual Change Traffic

AME = ACT x SDE

AME → Annual Maintenance Effort (For 1 year)

SDE → s/w development effort in PM.

TE = SDE + AME

Q2) Annual change Traffic for software is 15 % per year. If the development effort is 600 per month. compute & estimate for annual maintenance efforts. If life-time of project is 10 years. What is total effort of the project.

ACT = 15%.

SE = 600 pm

AME = ?

AME = ACT X SDE

$AME = \frac{15}{100} \times 600$

= 90

TE = SDE + AME → For 10 years)

= 600 + (90 X 10)

= 600 + 900

= 1500

DATE / / PAGE
NOTEBOOK

S/W Re-engineering

- Redocumentation
- Re analysis
- Source code can be modified as per latest demand, programming languages etc.

DATE 7/12/21 PAGE
NOTEBOOK

Software Risk Management

Tomorrow's problem is today's risk.

Dealing with concern before it becomes crisis: Risk Management

Typical s/w risk

1) Dependencies

2) Requirement Issues (Unrealistic issues)
   - ↳ Lack of clear product vision.
   - ↳ Lack of agreement on product requirements.
   - ↳ Unprioritize requirements.
   - ↳ New Markets with certain need.
   - ↳ Rapidly changing requirements.

3) Management Issues
   - ↳ Inadequate planning
   - ↳ Inadequate visibility into actual project status.
   - ↳ Unclear project ownership and decision making.
   - ↳ Unrealistic commitment made, sometimes for wrong reasons
   - ↳ Managers or customers with unrealistic expectations
   - ↳ Staff personality conflicts
   - ↳ Poor communications.

4) Lack of knowledge
   - ↳ Inadequate training
   - ↳ Poor understanding of methods, tools & techniques.
   - ↳ Inadequate application domain experience.
   - ↳ New Technology
   - ↳ Poor documentation

5) Other risk categories

BIT-28

Roll No. 2 0 1 9 0 7 1 0 2 7

B. Tech

ODD SEMESTER

Minor Exam (Vth Semester)

Session: 2021-2022

Software Engineering

Time: 02 Hrs

Max. Marks: 30

Note: Be precise in your answer.

Q.1 Attempt any Three parts of the following. Q. 1(a) is compulsory.

a. What is Software Engineering? Describe the various software characteristics in detail. List and elaborate the various similarities and differences between Software engineering process and conventional engineering process. 4

b. Draw the block diagram of Waterfall Model and explain its various steps in detail. List its various advantages and disadvantages also. 3

c. What is Module Coupling? Describe the various types of module couplings in detail. 3

d. What is modularity? Describe the important properties of a modular system. List its various advantages. 3

Q.2 Attempt any Three parts of the following. Q. 2(a) is compulsory.

a. Draw the block diagram of Spiral model and explain its working in detail. What are the limitations of such a model? 4

b. What is the significance of feasibility study in software engineering? Illustrate the various types of feasibility studies with a suitable example. 3

c. Describe the working principle of Prototype model along with its advantages and disadvantages. 3

d. Describe the working of Iterative enhancement model along with its advantages and disadvantages. 3

Q.3 Attempt any Three parts of the following. Q. 3(a) is compulsory.

a. What is Software Design? Draw the block diagram of Software Design Framework and explain its working in detail. Differentiate between Conceptual and Technical design also. 4

b. What is Software Quality? Discuss the various software quality attributes in brief. 3

c. Describe the various steps involved in analysis and design of object-oriented system. 3

d. Explain the various key process areas of CMM at various maturity levels. 3