

Software Engineering

- ① Introduction
- ② Software Development Life cycle
- ③ Requirement analysis (SRS)
- ④ Software Project Management (cocomo)
- ⑤ Software Design (coupling, cohesion, UML, DFD, class Diagram)
- ⑥ Coding and Testing
- ⑦ Maintenance
- ⑧ Quality Management, Rense

v-2

* Definition:

- It is a systematic, disciplined, cost-effective technique for software development.
- Engineering approach to develop a software

* Evolution:

1945 - 65 → Origin

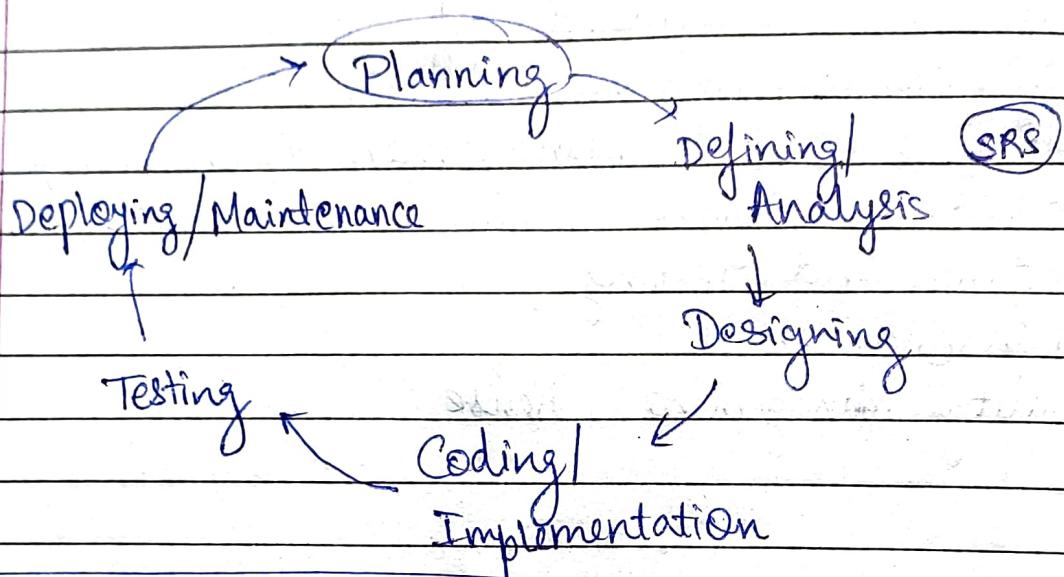
1965 - 85 → Crisis *

1990 - 2000 → Internet

2000 - 10 → Light weight

2010 - Till → ML, AI, DL

* Software Development Life cycle (SDLC)



V-4

"Classical Waterfall Model":

Feasibility study

Requirement Analysis and Specification

Design

Coding and unit testing

System testing and integration

Maintenance

* Advantages * Disadvantages

- | | |
|------------------|------------------|
| ⇒ Base Model | ⇒ No feedbacks |
| ⇒ Simple & Easy | ⇒ No experiment |
| ⇒ Small projects | ⇒ No parallelism |
| | ⇒ High Risk |
| | ⇒ 60% efforts |

Maintenance

V-5. Iterative Waterfall Model:

Feasibility Study (communication)

Requirement Analysis (Planning)
and specifications

Design (Modeling)

Coding & ... (Construction)

Unit testing

(Deployment)

System testing
and integration

Maintenance

Advantages

- # Base Model
- # Simple and Easy
- # Small Project
- * # Feedbacks

Disadvantages

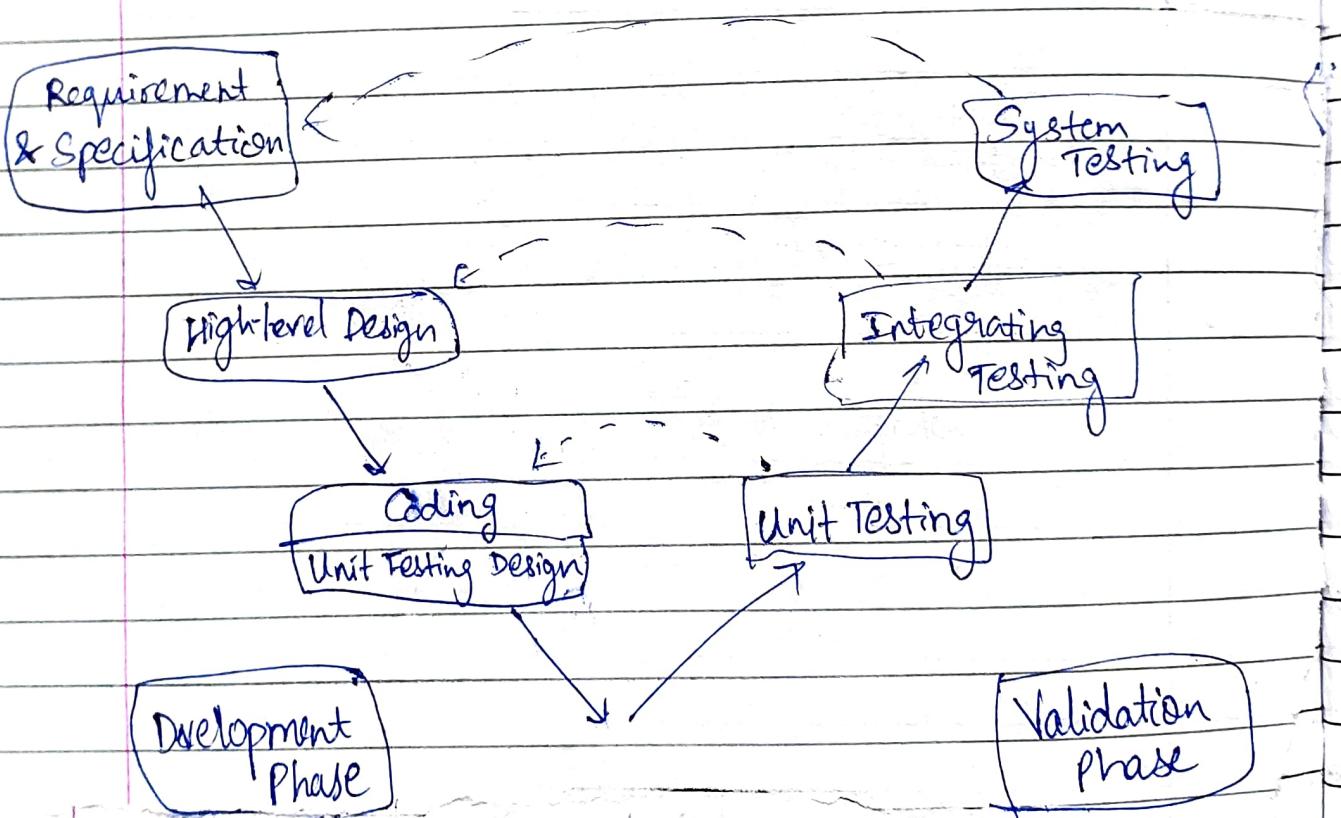
- ⇒ No Phase overlapping
- ⇒ No intermediate delivery
- ⇒ Rigid (No changes) ☹
- ⇒ Less customer interactions ☹

v.6

V-Model (V-shaped Model)

Testing (Imp)

- Verification and Validation Model.
- Extension of Waterfall Model.
- Testing is associated with every phase of lifecycle.
- Verification Phase (Requirement analysis, System design, Architecture design, Module design.)
- Validation Phase (Unit testing, Integration, System, Acceptance Testing).



* Advantage:

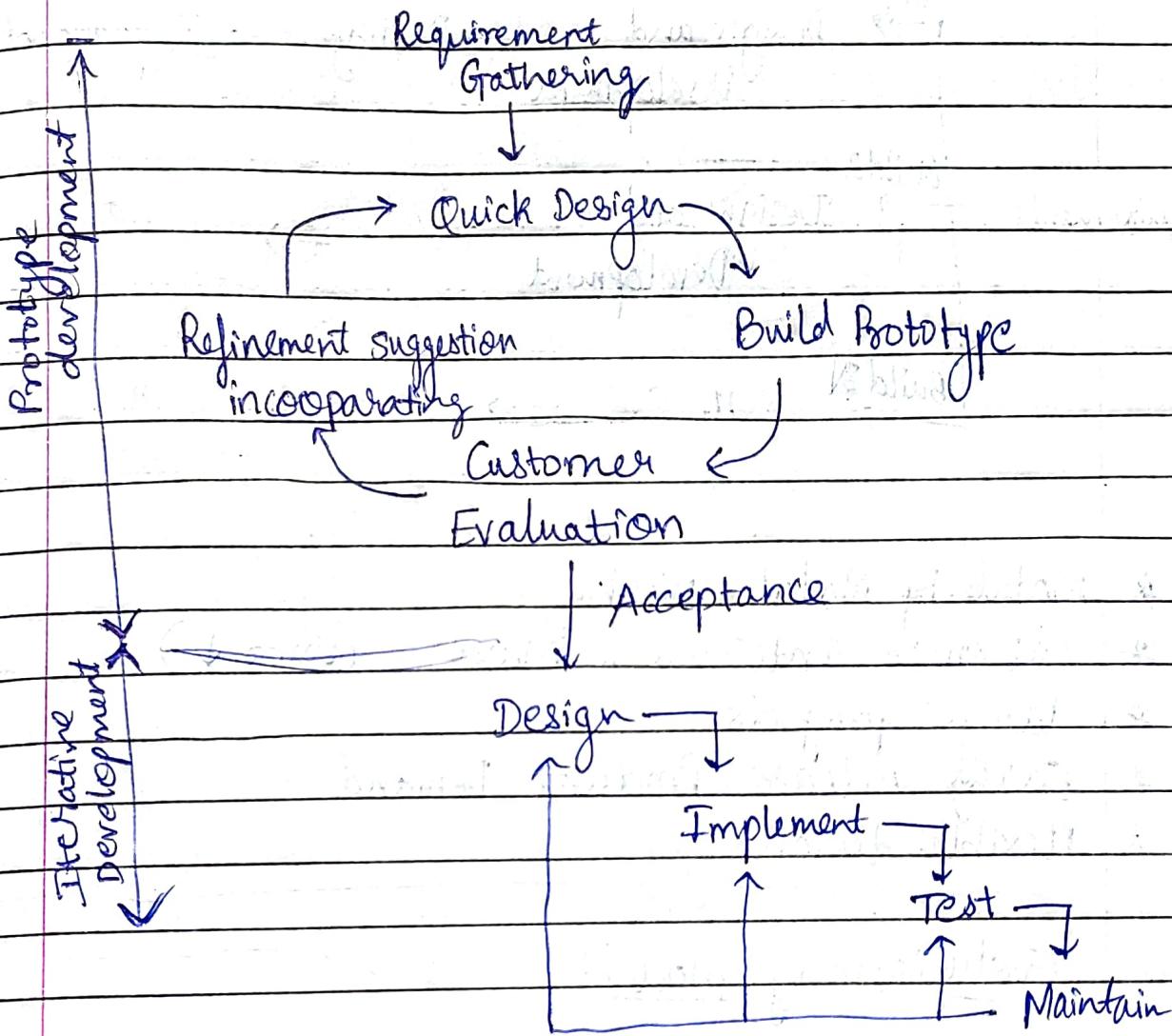
- Time Saving
- Every component must be testable
- Progress tracked easily
- Proactive defect tracking
- Good understanding of project.

* Disadvantage:

- No feedback
- Risk analysis not done
- Not good for big project.

Y-7

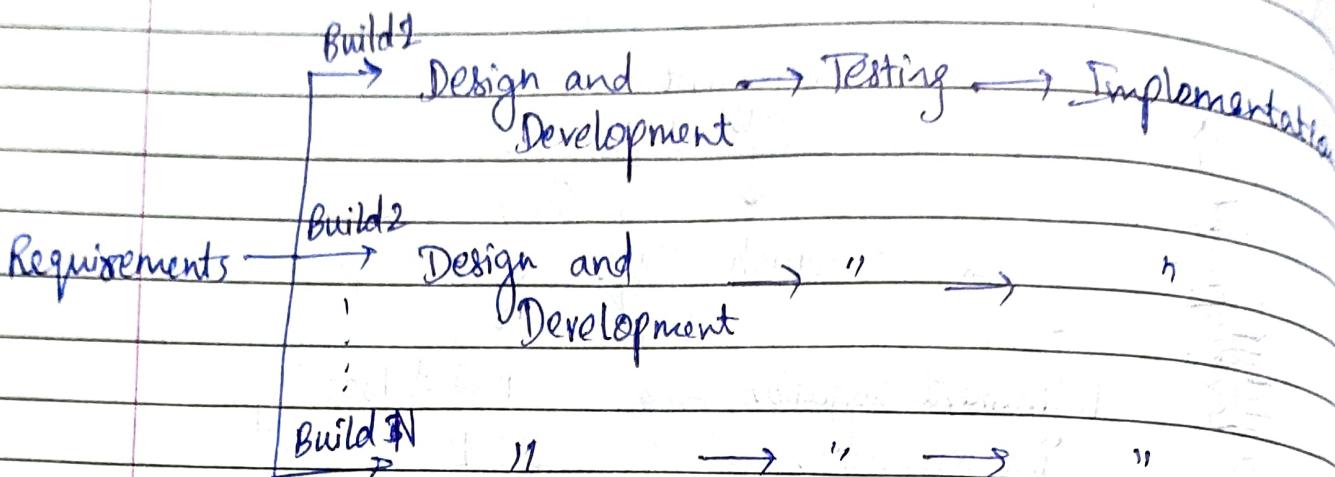
Prototyping Model in S.E. :-



- * Customer Not clear with idea.
- * Throw away Model
- * Good for technical and requirement risks.
- * Increases in Cost of Development.

V-8

Incremental Model:



- * Module by Module Working

- * Customer Interaction Maxm. (Error ↓)

- * Large projects

- * Early Release Product Demand

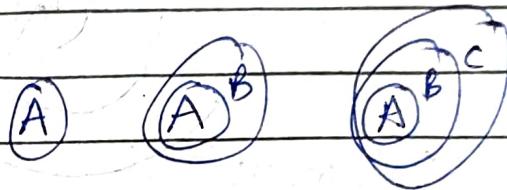
- * Flexible to changes.

V-9

Evolutionary model:

- Evolutionary model is a combination of iterative & incremental model of software development life cycle.
- Incremental model first implement a few basic features and deliver to the customer. Then build the next part and deliver it again and repeat this step until the desired system is fully realized. No long term plans are made.

- Iterative model main advantage is its feedback process in every phase.
- Also known as "Design a little, build a little, test a little, deploy a little model."



* Advantages:

- Customer requirements are clearly specified.
- Risk analysis is better.
- It supports changing environment.
- Initial operating time is less.
- Better suited for large mission-critical projects.

* Disadvantages:

- Not suitable for smaller projects
- Cost
- Highly skilled resources are required.

(Rough Requirement Specification)

Identify the core and the other parts to be developed incrementally

Develop the core part using an iterative waterfall model

Collect customer feedback and modify requirements

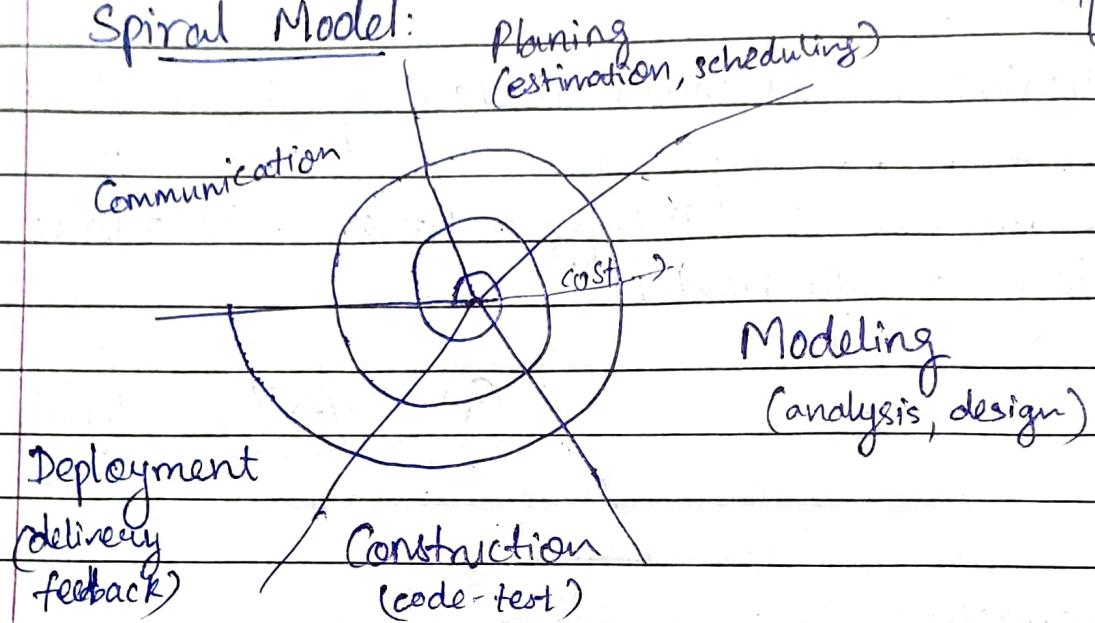
Develop the next identified features using iterative waterfall model
↓ All features complete

Maintenance

Delivery
next version
to the
customer

V-10

Spiral Model:



Risk

- * Risk handling
- * Radius of Spiral = Cost
- * Angular Dimension = Progress
- * "Meta-Model" called

Advantages :

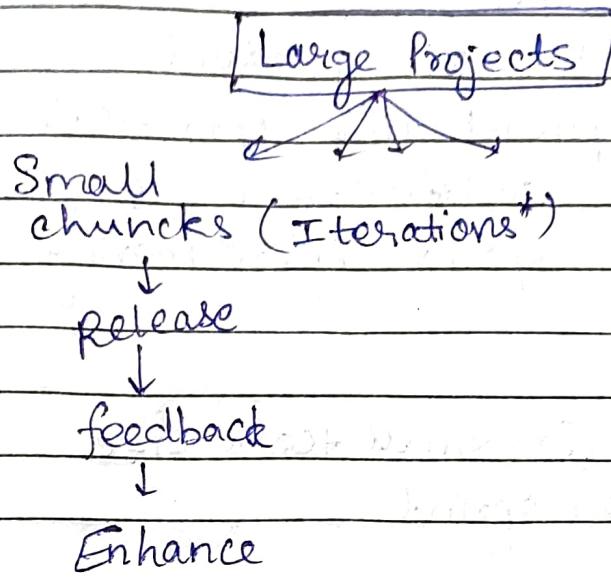
- Risk Handling
- Large projects
- Flexible
- Customer Satisfaction

Disadvantage :

- Complex
- Expensive
- Too much Risk Analysis
- Time

V-11)

* Agile Model: (move quickly)



* Advantages:

- ① Frequent delivery
- ② Face to face communication with client.
- ③ Changes fast
- ④ Time ↓

Agile model implemented in

- SCRUM framework
- KANBAN framework
- LEAN
- CRYSTAL

* Disadvantages:

- ① less documentation
- ② Maintenance Problem .

V-12)

SCRUM: (framework)

- Most agile methodology
- Lightweight, iterative and incremental framework
- "Sprints": stages / cycles which are the development phases breakdown by SCRUM.
- Scrum Team has Scrum Master & Product Owner
- One sprint at a time (2 week - 1 month)

* Keywords:

- Backlog (Basic Requirement / what to make?, in a document)
- Sprint, Daily Scrum
- Scrum master, Product Owner

* Advantages:

- Freedom & Adaptation
- High-Quality, low-risk product.
- Reduce Development time upto 40%.
- Scrum customer satisfaction is very important
- Review current sprint before moving to new one.

* Disadvantage -

- More efficient for small team size
- No change in the sprint

②

Software Requirement:

- Process of defining, documenting and maintaining requirements in the engineering design process

* Process : (Software Management)

- ① Feasibility study
- ② Requirement Gathering / Analysis
- ③ SRS (software Requirement Specification)
- ④ SRV (soft. Req. Validation)

Types

Requirement Engineering

(I)

Functional
Req.

(II)

Non-functional
Req.

- Administrative functions
- Authentication
- Data collection
- Business Rules
- Legal /Regulatory requ.

FUNCTIONS

- Response Time, Scalability
- Security, Capacity
- Usability, Regulatory
- Recoverability
- cost, Flexibility

(III)

User Req.

(IV)

System Req.

- Easy & Simple to Operate.
- Quick Response.
- Effectively handling operational errors.
- Customer support.

① SRS - (Software Requirement Specification)

- SRS is a description of a software system to be developed.
- It lays out fn. & non-fn. req. of the soft-devl.
- It may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

→ SRS structure -

1 Introduction

1.1 Purpose

1.2 Audience Target

1.3 Scope

1.4 Definitions

2 ~~User Interface~~ Overall Description

2.1 User interfaces

2.2 System interfaces

2.3 Constraints

3 System Feature & Requirements

3.1 F1. Req.

3.2 Use Cases

3.3 3rd Party interface

3.4 Non-fn. req.

4 Deliver for Approval

Agile Model

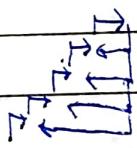
- Incremental / Evolutionary
- RAP
- Spiral
- Model

<u>Risk</u>	<u>Time & Cost</u>	<u>Flexibility</u>
• Not for small project	• User constraint	• Advanced parallel process
• User at all levels	• User at all levels	• No early defined req.
• Reusability	• Requirements not locked (Incremental model + Flexible)	• lock on req.
		• Less experience
		• Can integrate

Waterfall Model

- Iterative
- Prototypic

<u>User Regn.</u>	<u>Module by module delivery</u>	<u>Easy to test & Debug</u>	<u>High user involvement</u>
• not clear.	• costly	• No early lock on req.	• Reusability
• Problem is well understood	• No early lock on req.		
• Inflexible	• feedback & feedback		
• Not for real project			



UML

(Unified Memory Language)

→ Visual language

13

① Things

② Relationship

③ Diagrams

④ Modelling elements

⑤ Represent Relation among Things.

⑥ Pictorial Representation of system

⑦ → Structural things

↳ static components
like nouns,
eg: class, use case
interface① Association
→ Link both Thing Obj
(\rightarrow)

① Dynamic model:

• Object

• use case

• Sequence

• Activity

• Collaboration

• State chart

⑧ → Behavioural Thing:

(Activity)
eg: Interaction② Dependancy
 $x \rightarrow y$

③ Generalization

② Static Model -

• class

• Component

• Deployment

→ Grouping Things
(logic similarity
thing \rightarrow group)(\rightarrow)eg: Packages of group
use case

④ Realization

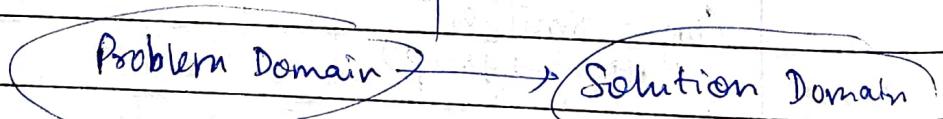
→ Annotation Thing:

(- - - \rightarrow)

- Represent extra info
- sticky notes in diagram

V-2 Concurrent Object Modeling and Architecture Design (COMET) (UML base) method

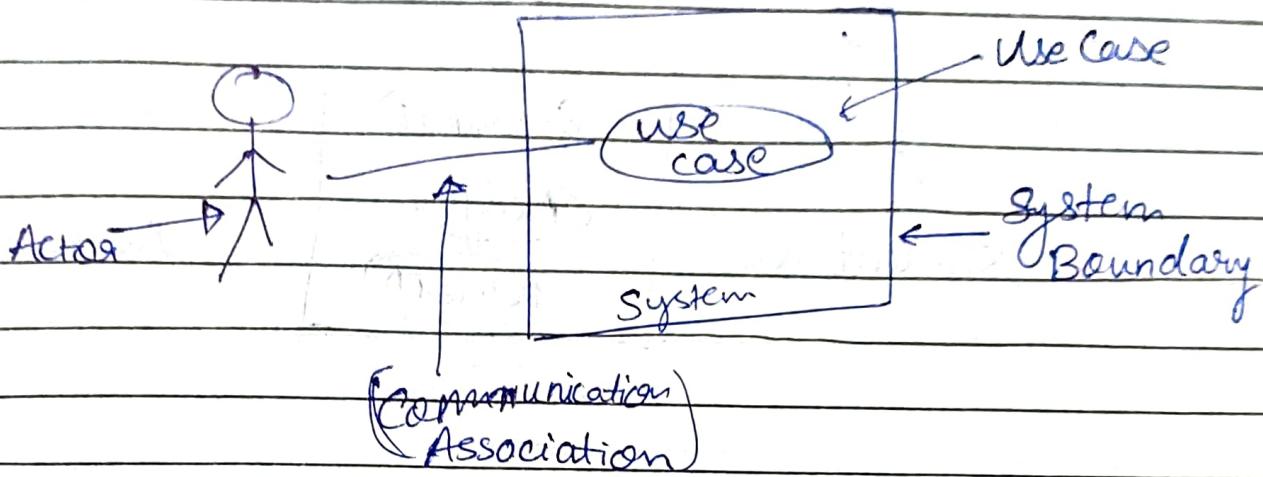
① Requirement Phase	② Analysis Phase	③ Design Phase
<ul style="list-style-type: none"> → functional Requirements → Usecase & actors are considered → Description of use case → Developing of Requirement model → There must be a clarity in Requirement 	<ul style="list-style-type: none"> → Development of Static & Dynamic model → Static Structural Relation b/w problem domain class [class diagram] → Dynamic Interaction of objects with use-case → Communication diagram 	<ul style="list-style-type: none"> → Software architecture of system is designed → Mapping of Analysis model to design model → Subsystem Integration and Communication through message



V.S.

Use-case:

- Captures the functionality of system from user perspective



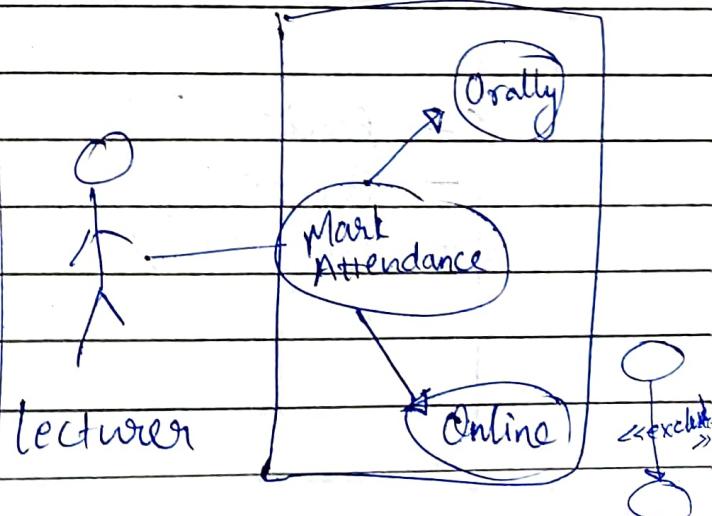
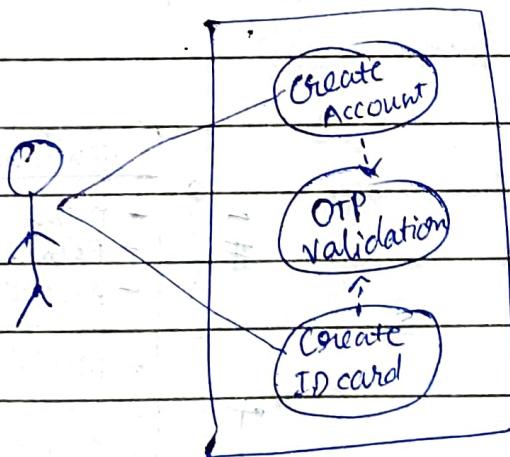
- (I) Primary Actor - (first user who activates the system)
- (II) Secondary Actor - (second or next actor who interact with system)

(I) Include Relationship-

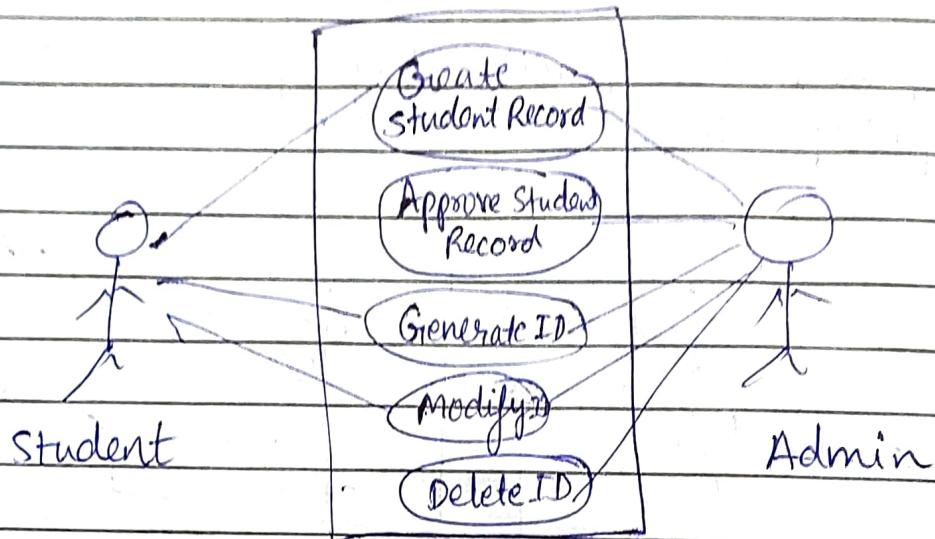
- Inclusive Use Case:

(II) Extend Relationship-

- Extension Use Case:



Use Case Packages:



V-8

Class Diagram:

Class Name	Student	Student
Class Attribute	Name Rollno	+ name: string - Roll no: int
Class Operation	CreateRecord()	# CreateRecord(): int

+ Public
 - Private
 # Protected

Student	
-	name: string
-	Rollno: int
#	Address: string
+ {	
+ Register(): void	
+ Login(): void	
+ filldetails(): void	

Relationships.

1 = Only one

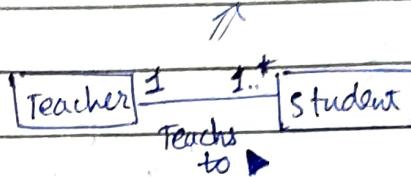
0..1 = 1 or 0

0..*/* = 0 or more

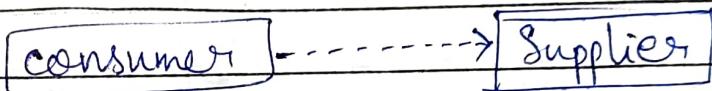
1..* = 1 or more

Exact Number (3, 4 or 5)

① Association : (—)

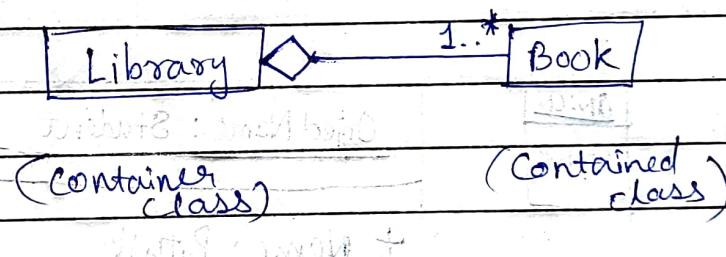


② Dependancy: (---->)



⇒ Depends on supplier.

③ Aggregation: (○—)



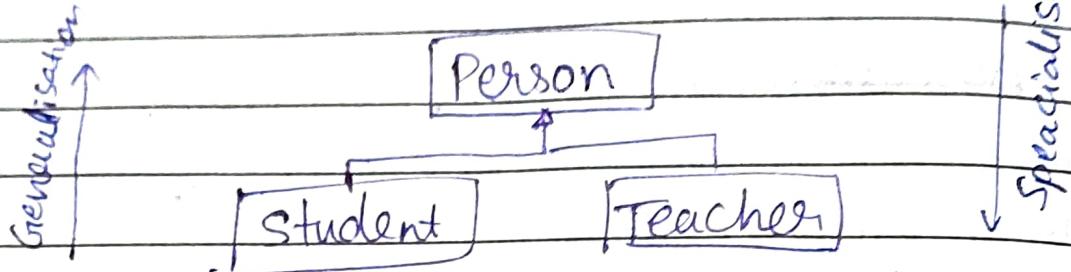
⇒ No strong dependency. If library is not there but books will still be there.

④ Composition: (◆—)



⇒ Strong Dependancy. If Bag burned, Pocket burned / deleted.

⑤ Generalisation: (\rightarrow)



"is a" relationship.

V-II # Object Diagram:

Provides a

→ A snapshot / static view at a given particular time.

eg:

ON:CN
Attributes

Object Name : Student
+ Name : Pathik
- Roll No : 231

Object Name = ClassName

Name : String

RollNo : int

eg = Anonymous Objects:

:student
+Attributes: +

✓-15

State Machine Diagram

→ State

→ Event

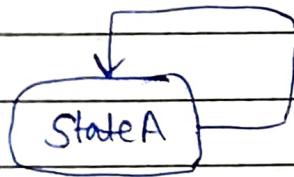
→ Transition

→ Initial state

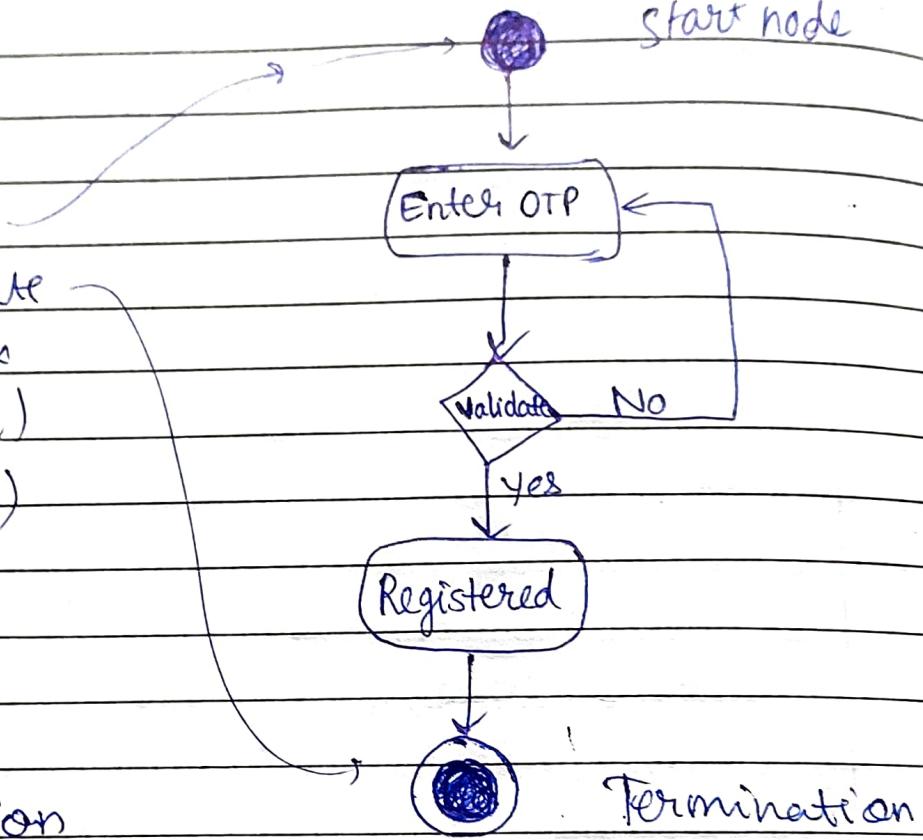
→ Final / End state

→ Decision Box
()

→ Merge Box ()



sky-transition



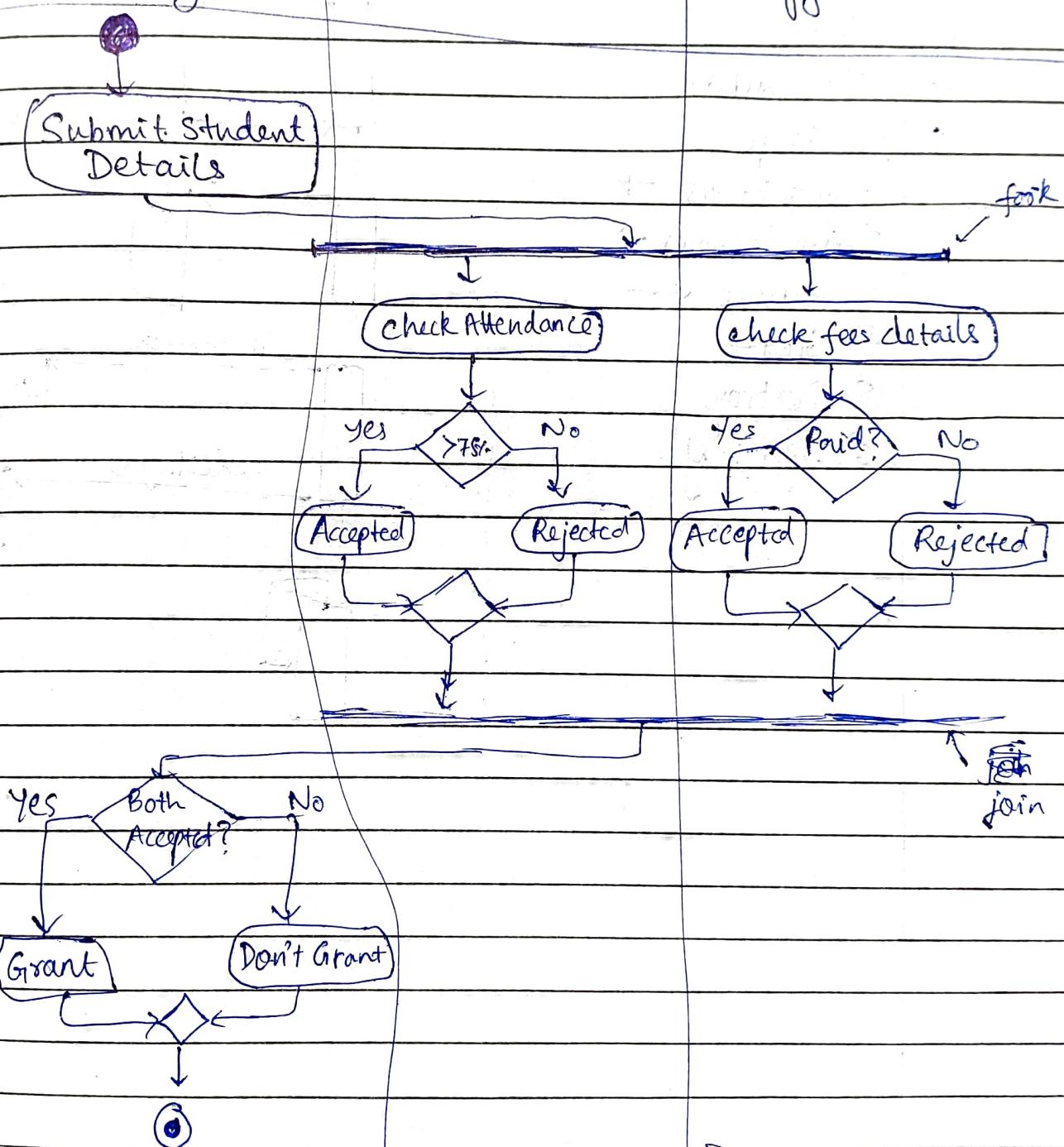
V-17

Activity diagram:

Scholarship
Incharge

Teacher

Accounts
Officer



↳ Making Table like this
is called Swimlane.

V-19 A

Sequence Diagram:

- ① Object
- ② Lifeline
- ③ Actor
- ④ Activation Bar
- ⑤ Message
 - ① Synchronous →
 - ② Asynchronous →
 - ③ Return ←--
- ④ Create
- ⑤ Destroy
- ⑥ self

