FULLSCAPE
PAGE NO.: DATE

Software Engineering

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

V-2

* Defination:

→ It is a systematic, disciplined, cost-effective technique for software development.

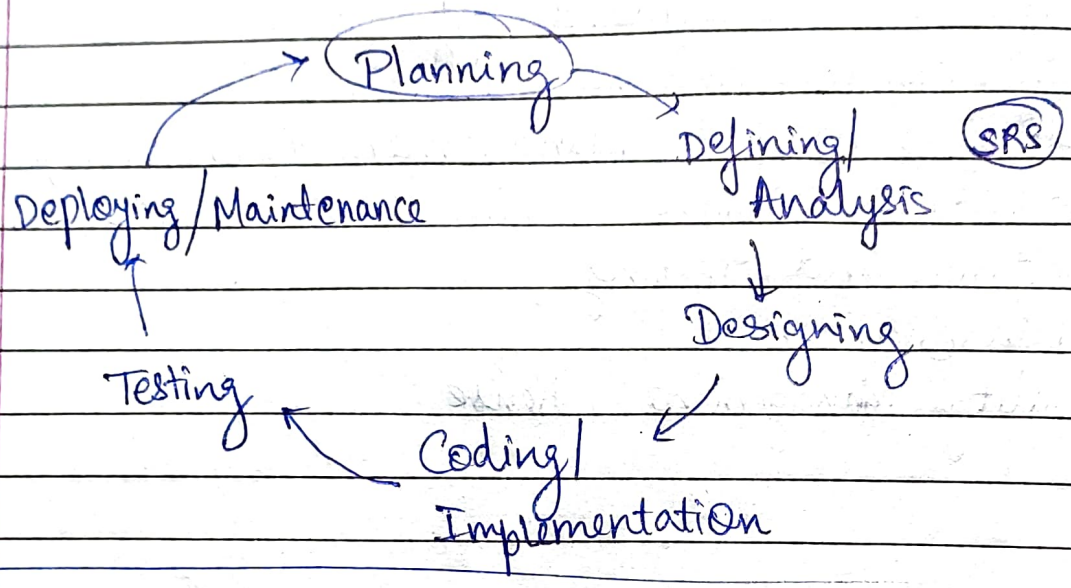
→ Engineering approach to develop a software.

* Evolution:

1945-65 → Origin
1965-85 → Crisis *
1990-2000 → Internet
2000-10 → Light weight
2010-Till → ML, AI, DL

FULLSCAPE
PAGE NO.: DATE

* Software Development Life cycle (SDLC)



V-4 "Classical Waterfall Model":

Feasibility study

Requirement Analysis and Specification

Design

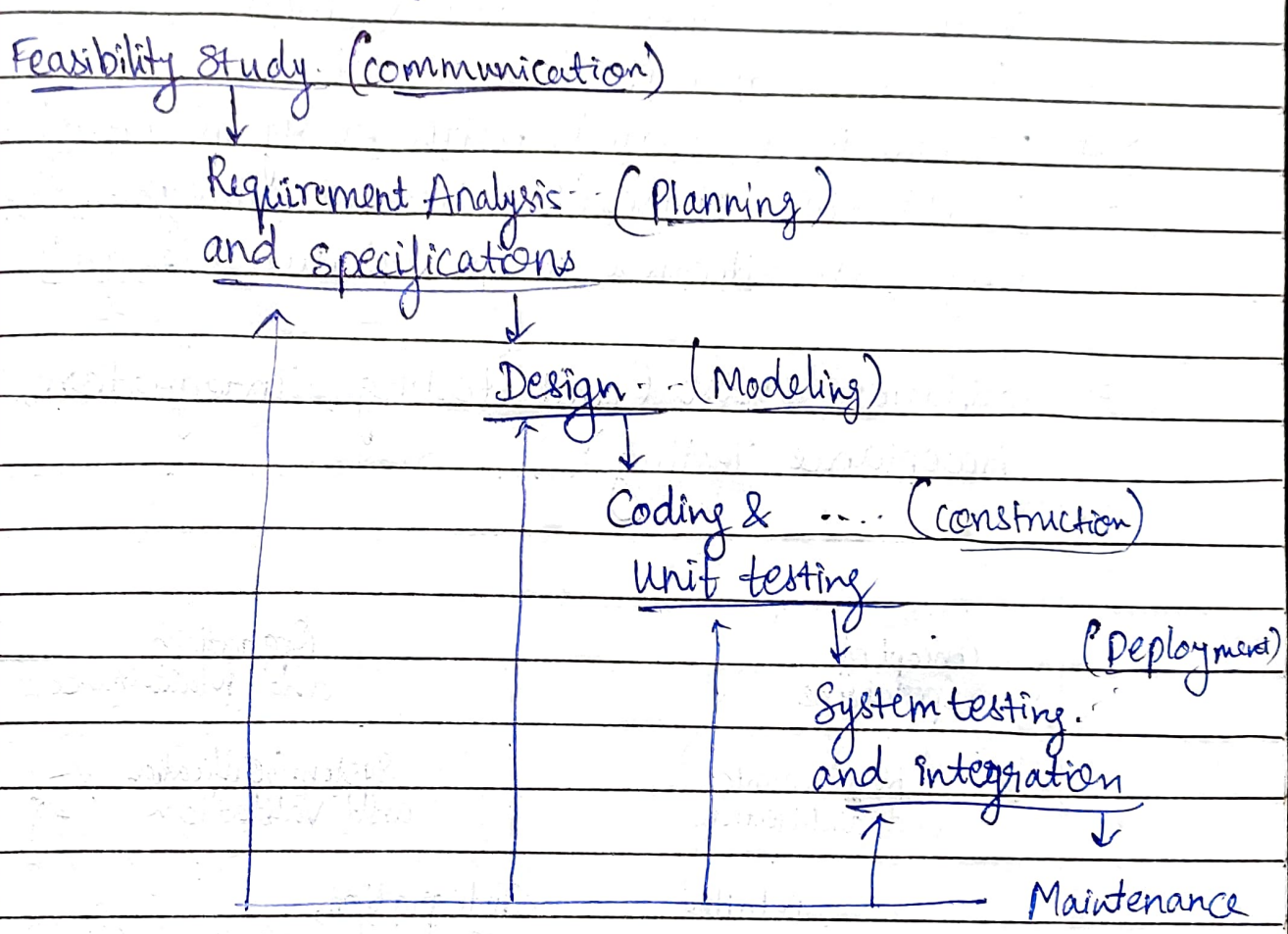
Coding and unit testing

System testing and integration

Maintenance

* Advantages	* Disadvantages
⇒ Base Model	⇒ No feedbacks
⇒ Simple & Easy	⇒ No experiment
⇒ Small projects	⇒ No parallelism
	⇒ High Risk
	⇒ 60% efforts Maintenance

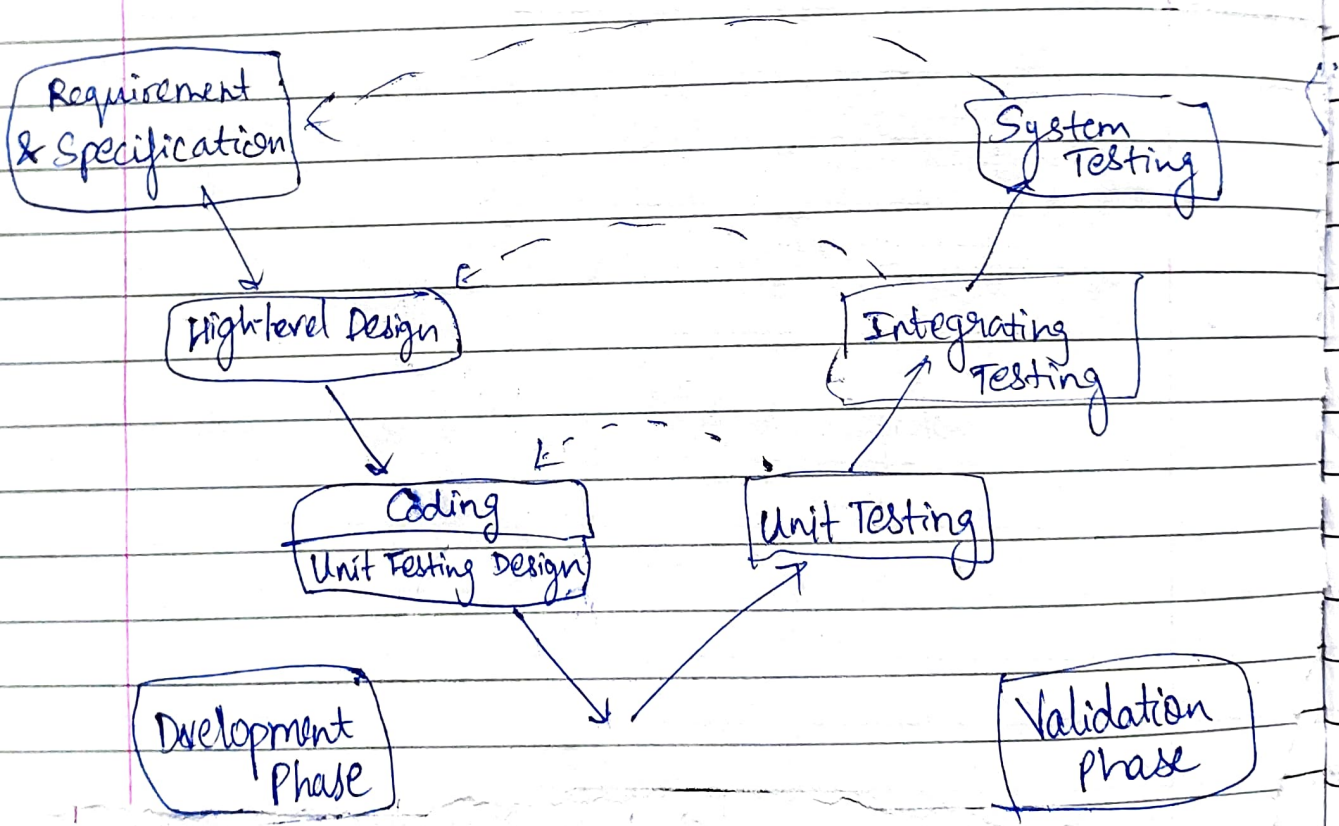
V-S Iterative Waterfall Model:



Advantages	Disadvantages
⇒ Base Model	⇒ No Phase overlapping
⇒ Simple and Easy	⇒ No intermidiate Delivery
⇒ Small Project	⇒ Rigid (No changes)
⇒ Feedbacks	⇒ 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).

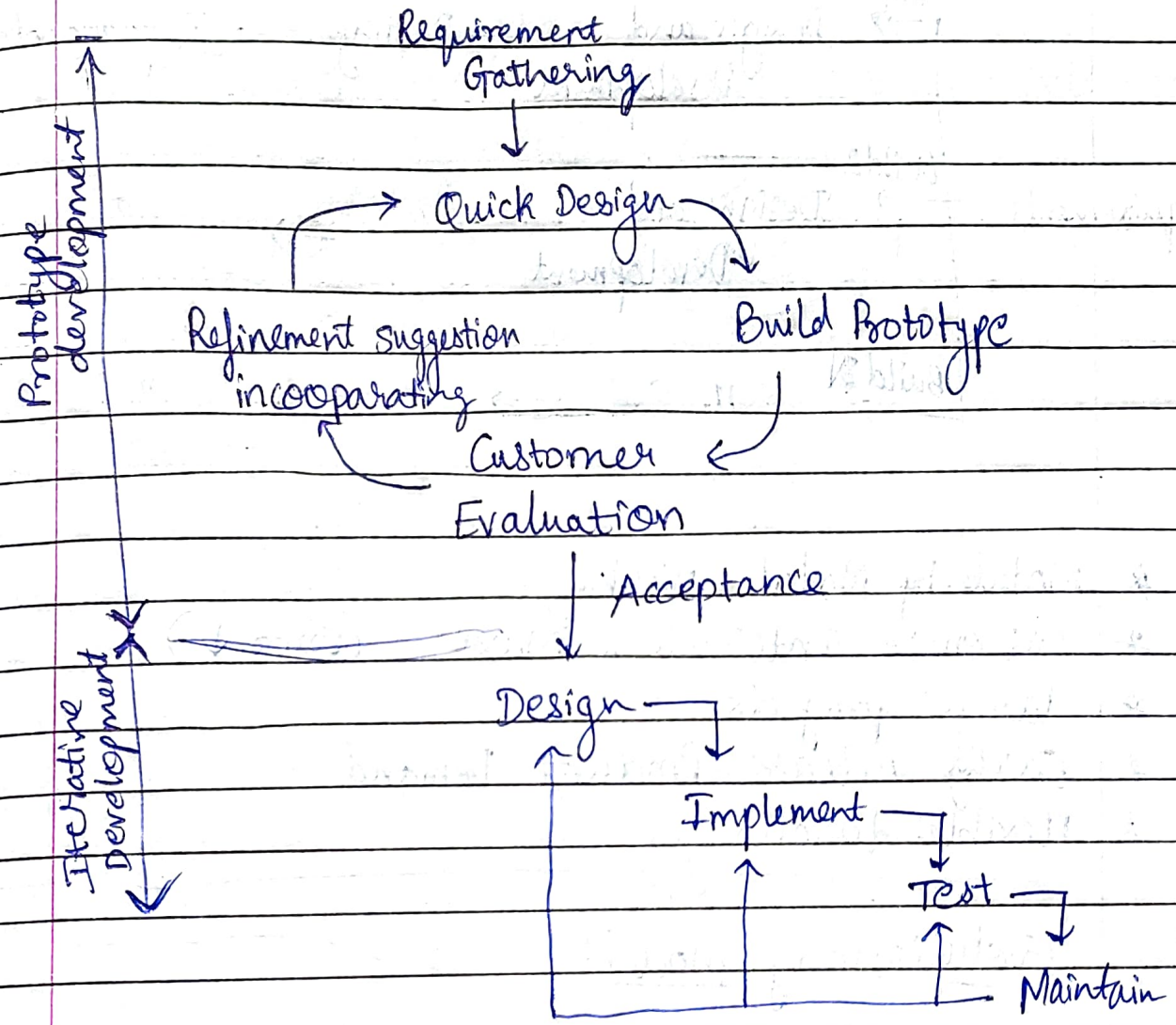


* Advantage:
- Time saving
- Every component must be testable
- Progress tracked easily
- Proactive detect tracking
- Good understanding of project.

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

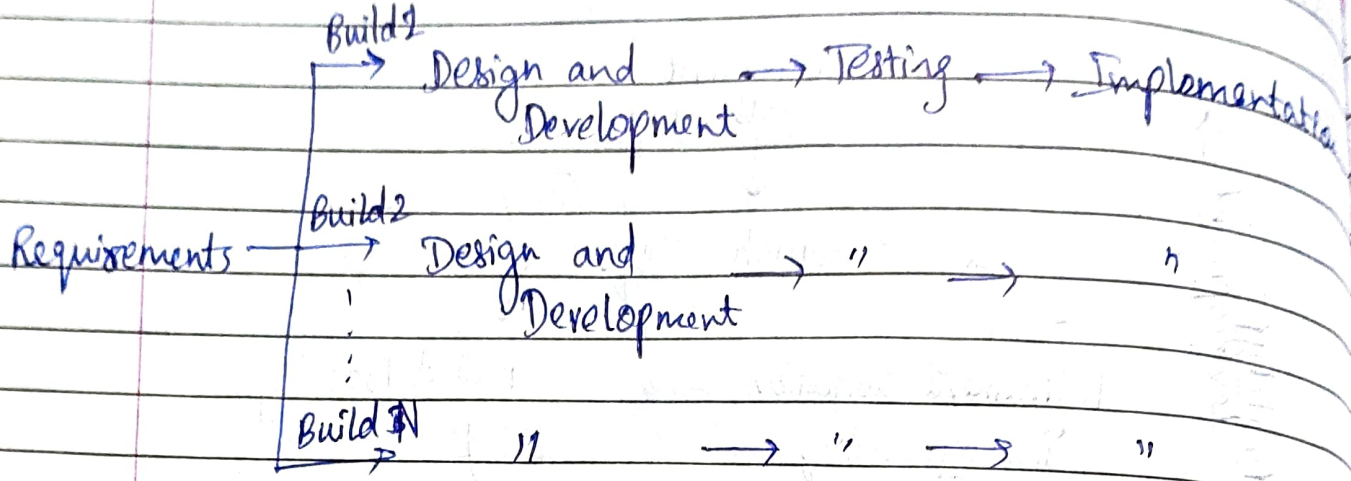
FULLSCAPE
PAGE NO.: DATE

V-7 Prototyping Model in S.E. :-



* Customer NOT clear with idea.
* Throw away Model
* Good for technical and requirement risks.
* Increased in Cost of Development

V-8 Incremental Model:

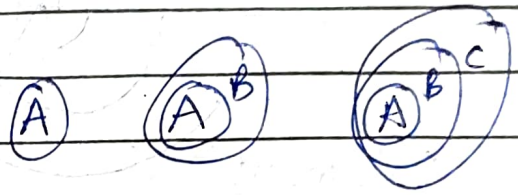


* Module by Module working
* Customer Interaction Maxm. (Error ↓)
* large projects
* Early Release Product Demand
* Flexible to changes.

V-9 Evolutionary model:

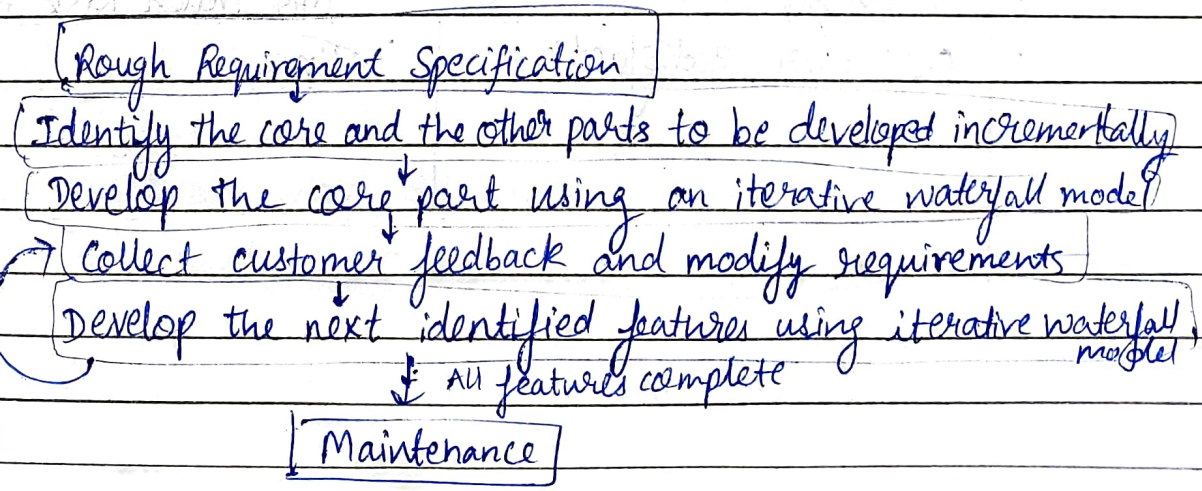
- Evolutionary model is a combination of iterative & incremental model of software development life cycle.
- Incremental model first implement a few basic features and deliver to the customer. Then build the next part and deliver it again and repeat this step until the desired system is fully realized. 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.

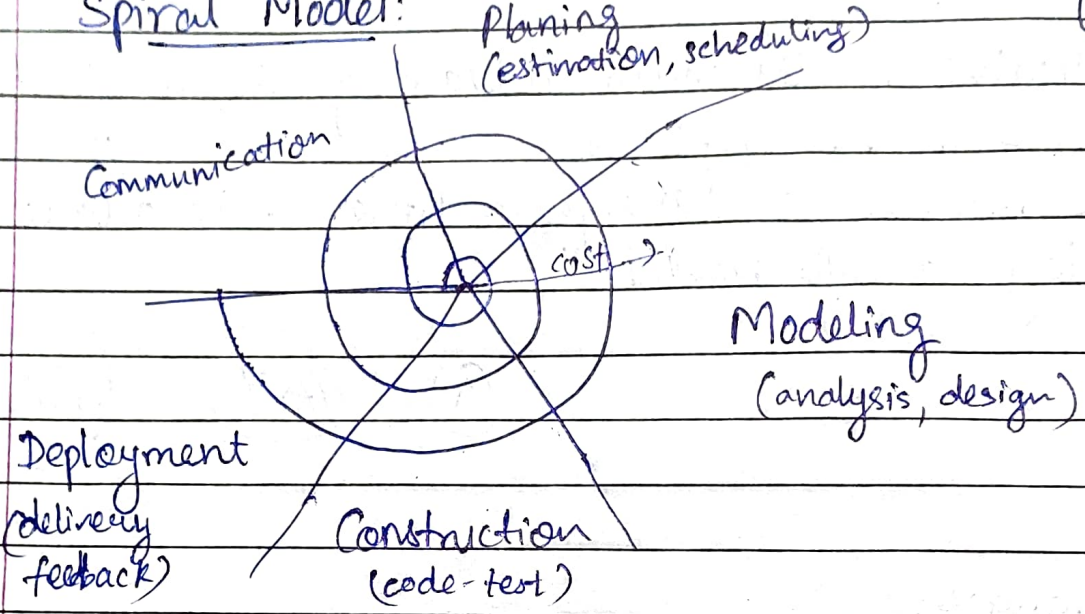


FULLSCAPE
PAGE NO.: DATE

Risk

V-IP

Spiral Model:



* Risk handling
* Radius of spiral = cost
* Angular Dimension = Progress
* "Meta-Model" called.

# Advantages:	# Disadvantage:
• Risk Handling	• Complex
• Large projects	• Expensive
• Flexible	• Too much Risk Analysis
• Customer satisfaction	• Time

V-11

Agile Model: (move quickly)

Large Projects

Small chuncks (Iterations*)

↓

Release

↓

feedback

↓

Enhance

Agile model implemented in
- SCRUM framework
- KANBAN framework
- LEAN
- CRYSTAL

* Advantages:
1. Frequent Delievery
2. Face to face communication with client.
3. Changes fast↑
4. Time ↓

* Disadvantages:
1. less documentation
2. Maintanance 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 (2week - 1month)

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

* Advantages:
- Freedom & Adaption
- 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

(*) 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 Requirment Specification)
  4. SRV (Soft. Req. Validation)

Types of Requirement Engineering

(I) Functional Req.

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

(II) Non-functional Req.

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

(III) User Req.

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

(IV) System Req.

FULLSCAPE
PAGE NO.: DATE

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. devel.
- 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 Definations

2 ~~User Interface~~ Overall Description
- 2.1 User interfaces
- 2.2 System interfaces
- 2.3 Constraints.

3 System Feature & Requirements
- 3.1 Fn. Req.
- 3.2 Use Cases
- 3.3 3rd Party interface
- 3.4 Non-fn. req.

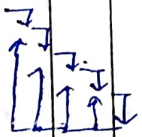
4 Deliver for Approval.

Classical Waterfall

- Basic
- Rigid
- Inflexible
- Not for Real Project

Iterative Waterfall

- Basic
- Problem is well understood
- feedback

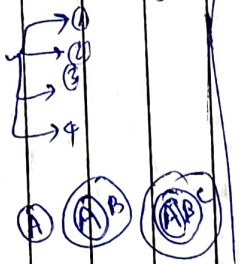


Prototype Model

- User Reqm. not clear.
- Costly
- No early lock on Req.
- High user involvement
- Reusability.

Incremental Model

- Module by module Delivery.
- Easy to test & Debug



Req. locked

Evolutionary Model

- large Projects
- Requirement not locked (Incremental model + Flexible)

RAD Model

- Time & Cost constraint
- User at all levels
- Reusability

Spiral Model

- R&R
- Not for small Project
- No early lock on req.
- Less Expirence Can rework

Agile Model

- Flexible
- Advanced
- Parallel
- Process defined into sprints

5-Min. Eng.

FULLSCAPE PAGE NO.: DATE

UML (Unified Memory Language)

→ Visual language

(I) Things	(II) Relationship	(III) Diagrams
✱ Modelling elements	✱ Represent Relation among Things.	✱ Pictorial Representation of system
✱ → Structural things ↳ static components like nouns, eg: class, usecase interface	① Association → Link both Thing Obj (⟶)	① Dynamic model: • Object • usecase • Sequence • Activity • Collaboration • State chart
✱ → Behavioural Thing: (Activity) eg: Interaction	② Dependancy "x → y"	
→ Grouping Things (logic similar thing → group) eg: Packages of group use case	③ Generalization: (⟶▷)	② Static Model - • class • Component • Deployment
→ Annotation Things: • Represent extra info • Sticky notes in diagram	④ Realization (- - -▷)	

FULLSCAPE
PAGE NO.: DATE

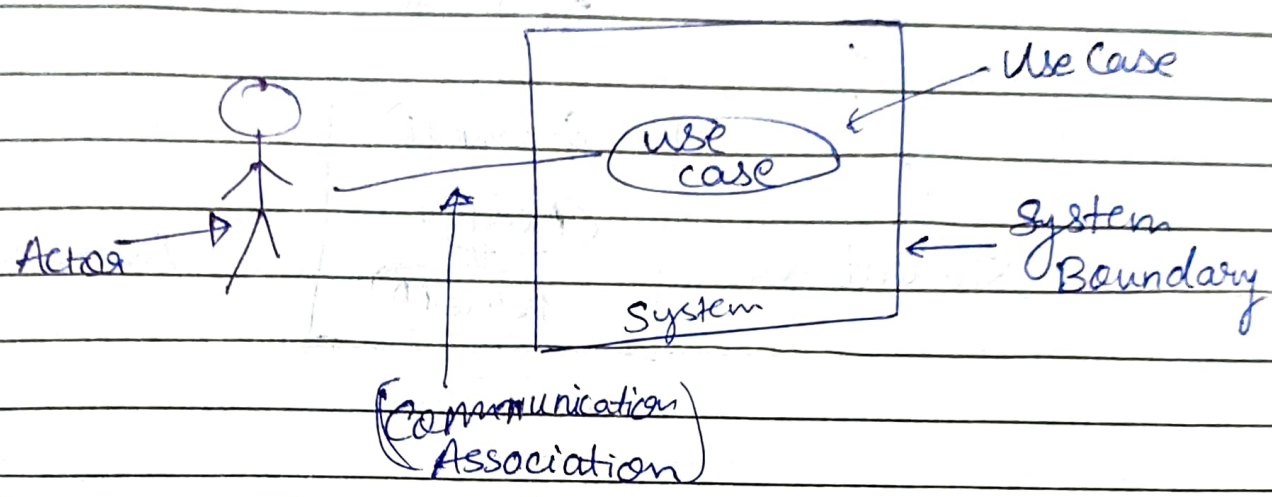
V-2 Concurrent Object Modeling and Architecture Design Method [COMET] (UML base)

(1) Requirement Phase	(2) Analysis Phase	(3) Design Phase
→ functional Requirements	→ Development of Static & Dynamic model	→ Software architecture of system is designed
→ Usecase & actors are considered	→ Static Structural Relation b/w Problem domain class [class diagram]	→ Mapping of Analysis model to design model
→ Description of use case	→ Dynamic Interaction of objects with use-case [communication diagram]	→ Subsystem Integration and Communication through message
→ Developing of Requirement model		
→ There must be a clarity in Requirement		

Problem Domain → Solution Domain

V:: Use-case:

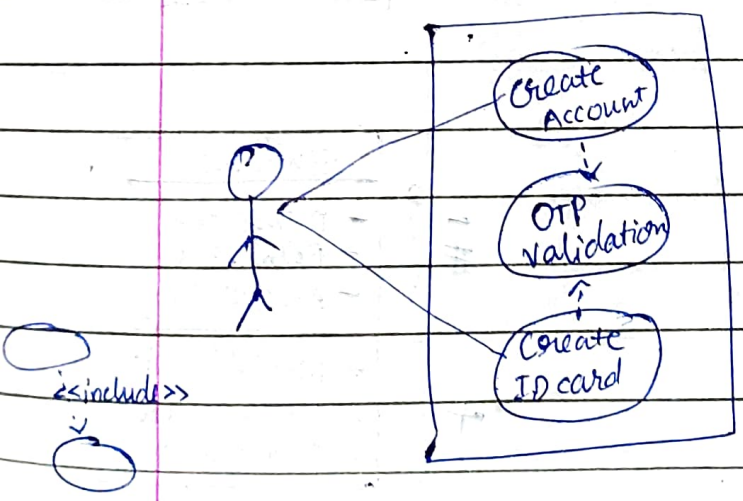
→ Captures the functionality of system from user perspective



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

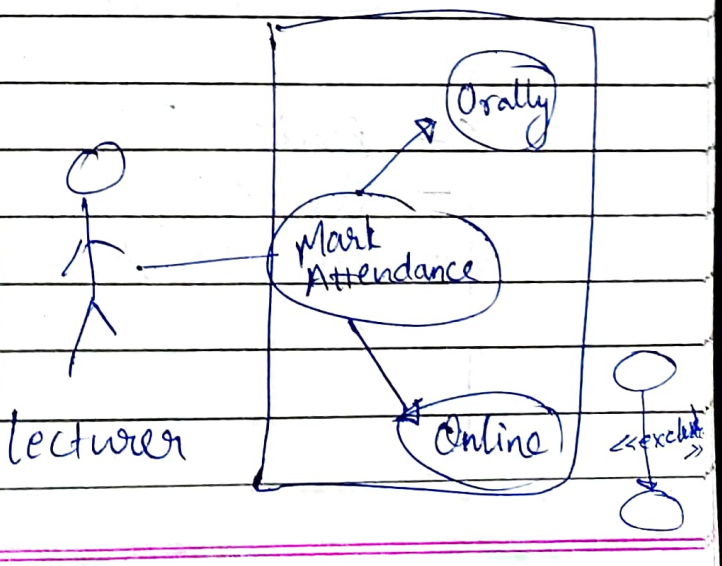
(I) Include Relationship -

→ Inclusive Use Case:



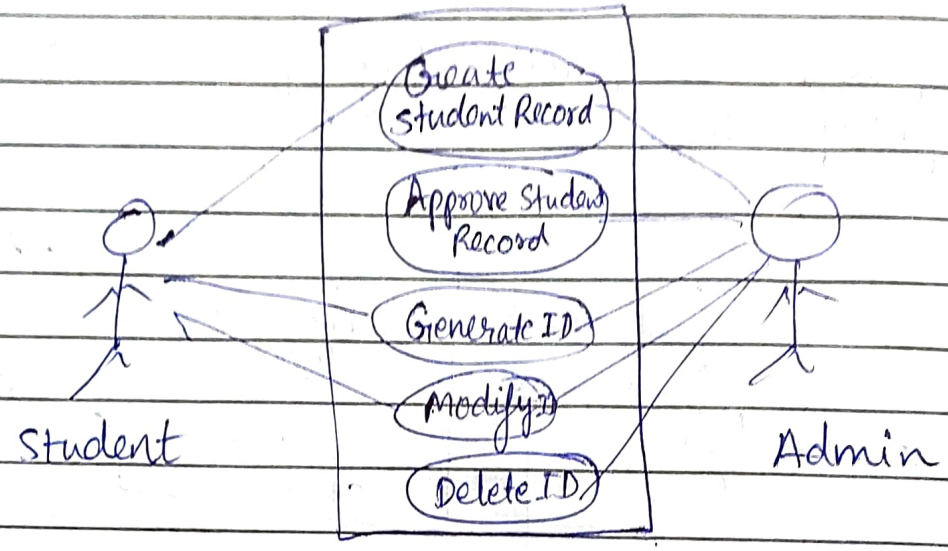
(II) Extend Relationship -

→ Extension Use Case:



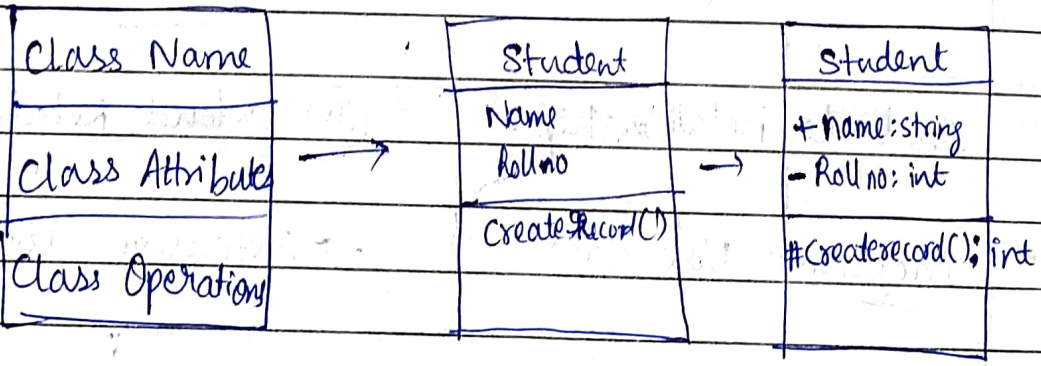
FULLSCAPE
PAGE NO.: DATE

Use case Packages:

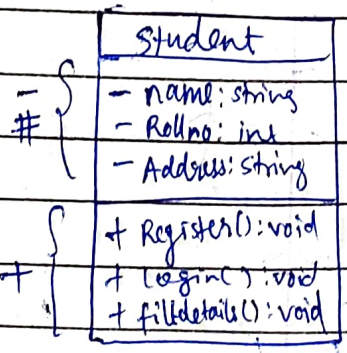


V-8

Class Diagram:



+ Public
- Private
Protected

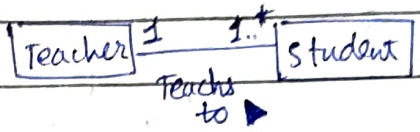


FULLSCAPE
PAGE NO.: DATE

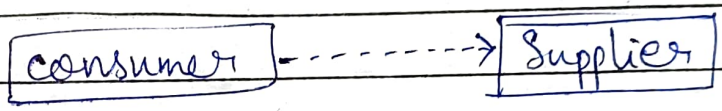
1 : Only one
0..1 : 1 or 0
0..*/* : 0 or more
1..* : 1 or more
Exact Number (3, 4 or 5)

Relationships.

① Association : (—)

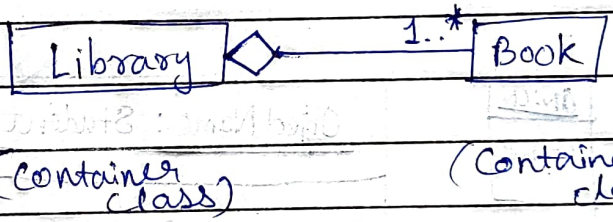


② Dependancy : (- - - - ->)



⇒ Depends on Supplier.

③ Aggregation : (◇—)



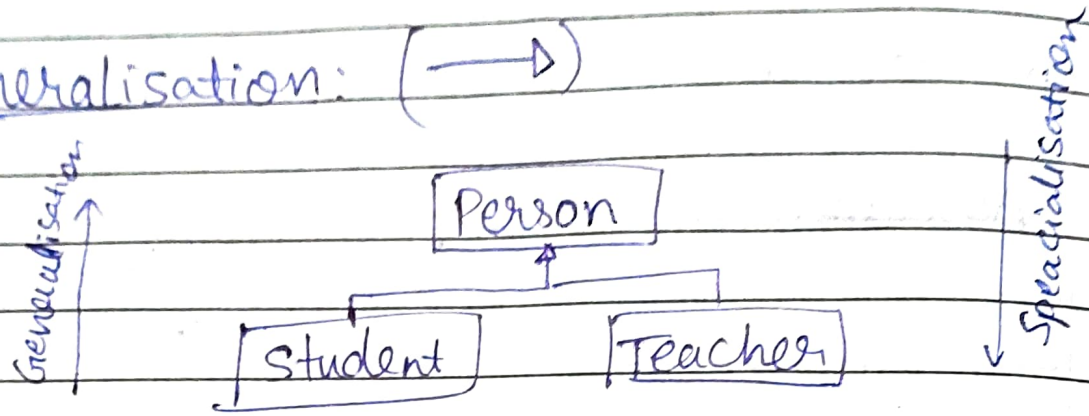
⇒ No strong dependancy. If library is not there but books will still be there.

(Association)

④ Composition : (◆—)

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

(5) Generalisation: (→)

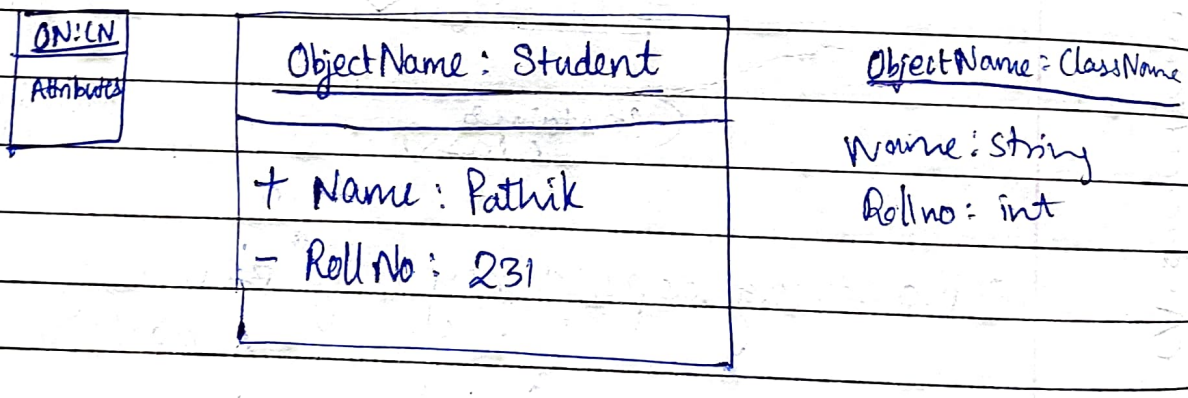


"is a" relationship.

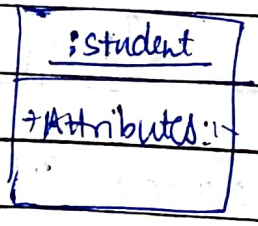
V-II # Object Diagram:

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

eg:



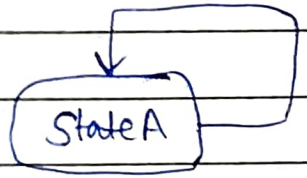
eg: Anonimous Objects:



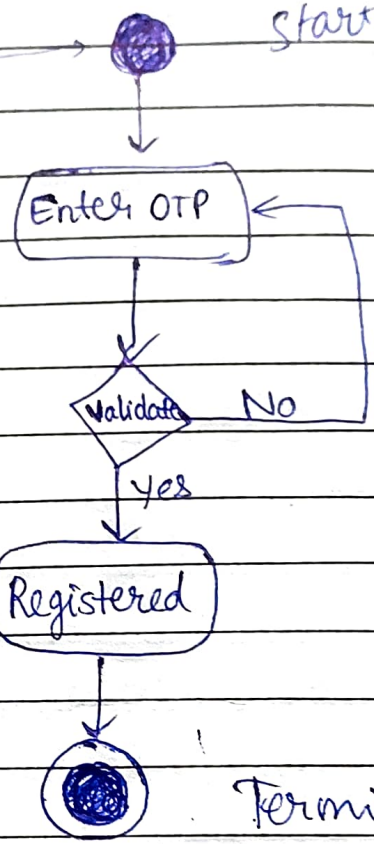
V-15

State Machine Diagram

- State
- Event
- Transition
- Initial State
- Final / End State
- Decision Box
- Merge Box

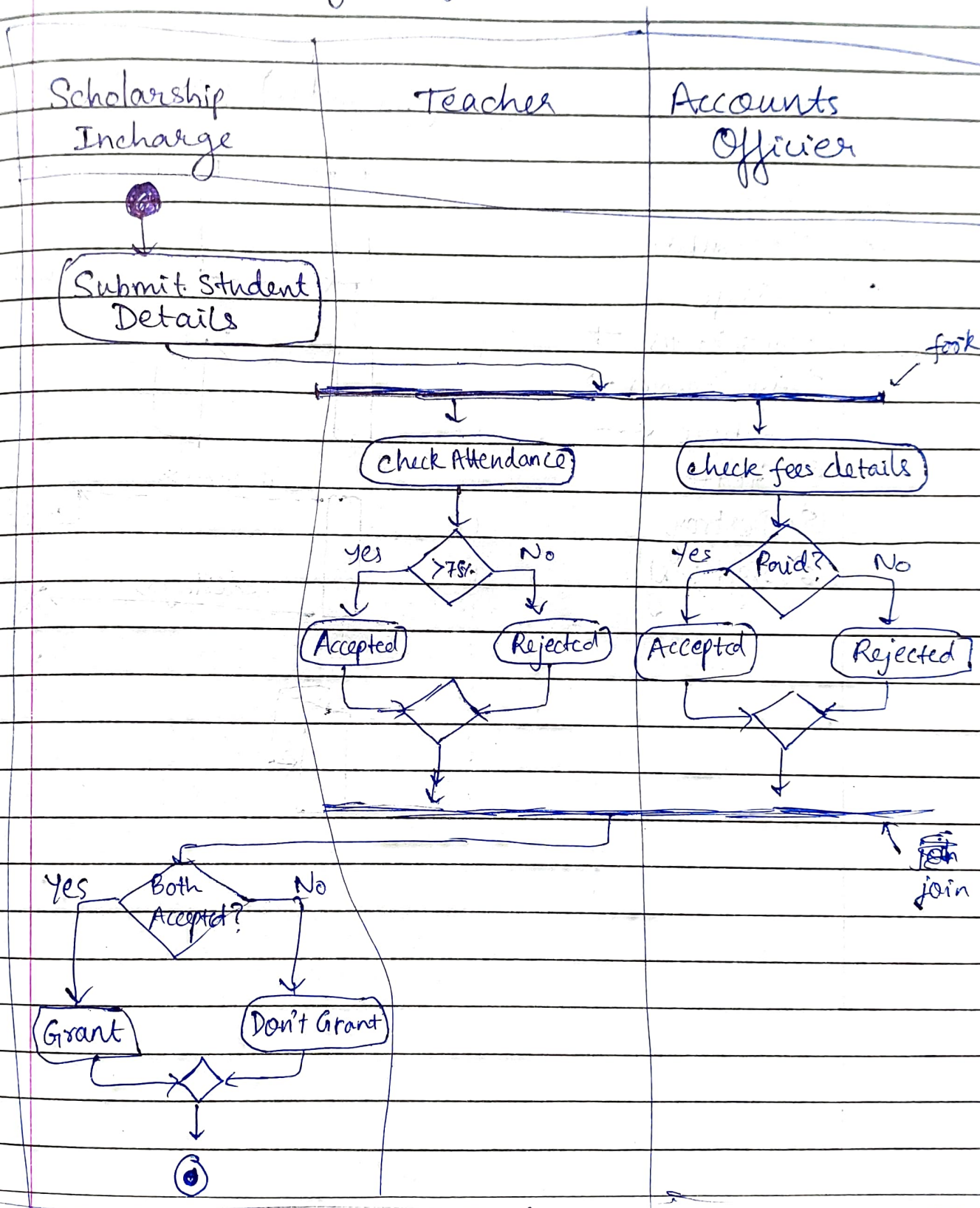


self-transition



V-17

Activity diagram:



↳ Making Table like this is called Swimlane.

V-IT (A) Sequence Diagrams:

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