Software Engineering

CSE 327

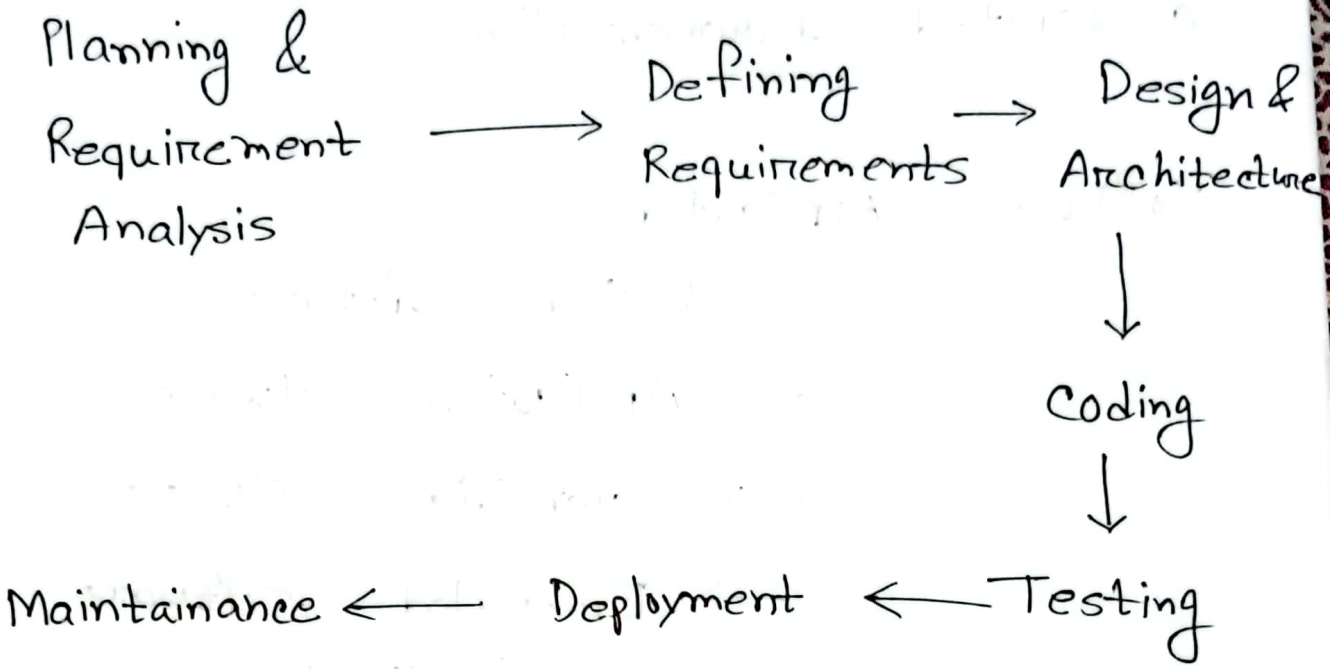
CSE 328

Software : It is computer programs and associated documentation.

Software Applications:

- → System software
- → Application software
- → Scientific software
- → Embedded software
- → Web Applications
- → AI software

SDLC



1. Planning & requirement Analysis : In this phase, the project team gathers needs from stakeholders and analyzes them to understand the requirements.

2. Defining Requirements : All software requirements been documented.

3. Design: Creating the architecture and design the system.

4. Coding: Code is developed by programmer developers. Code into programming language.

5. Testing: After development, it is send to testers. In this phase, the software is checked for bugs and error.

6. Deployment: After testing, if the software is bug free then it is launched and available for users.

7. Maintainance: Look over the system.

A software process is a set of related activities that leads to the production of a software product.

Framework Activities:

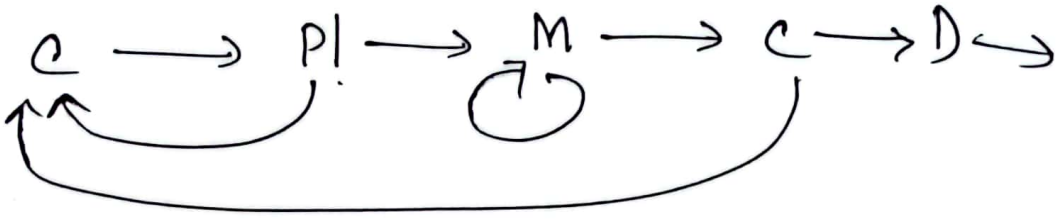
1. Communication : Requirement gathering and Analysis.
2. Planning :
3. Modeling : Design
4. Construction : code
5. Deployment

Process flow

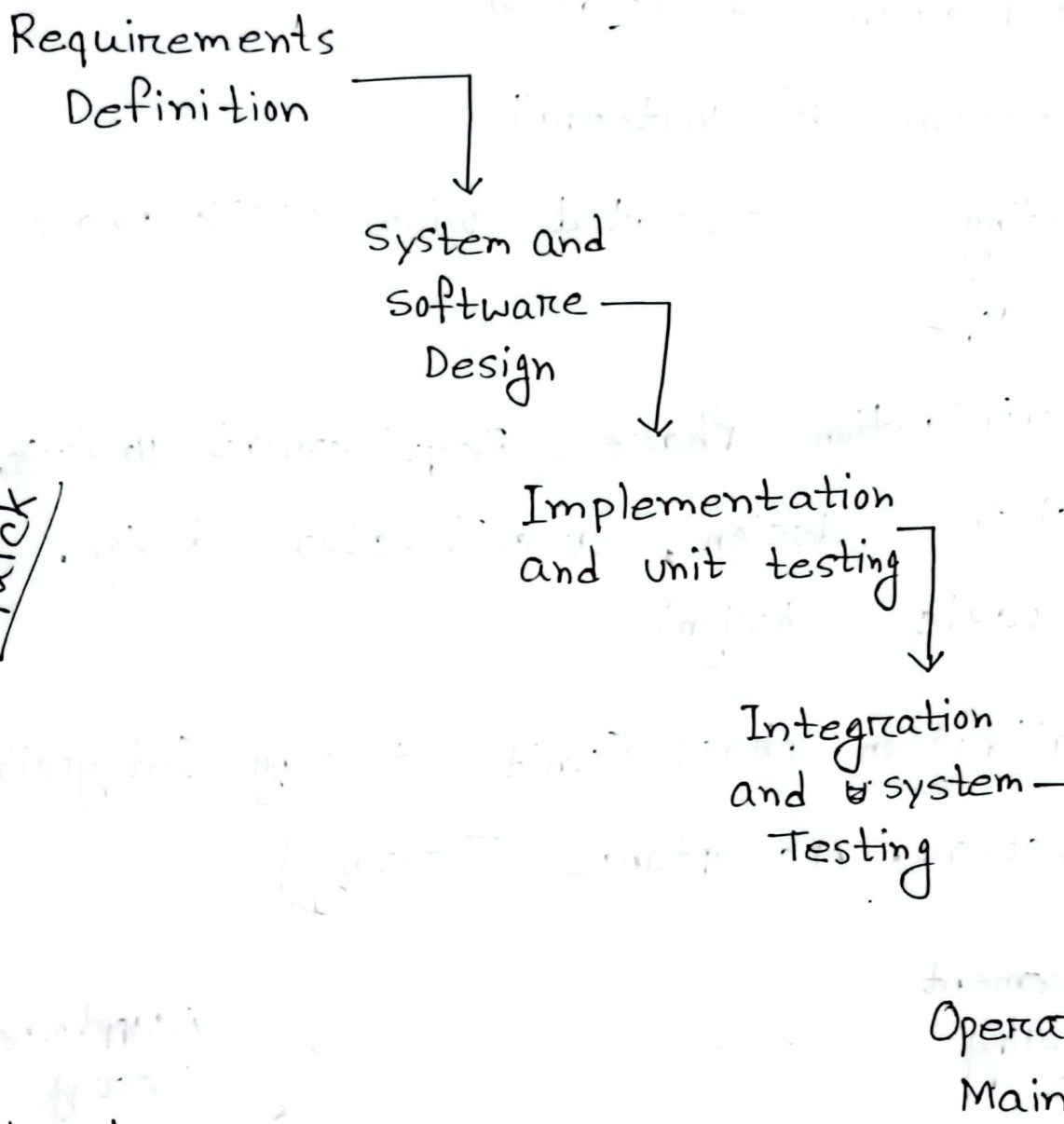
Linear process flow:

C → P → M → C → D →

Iterative process flow:



The waterfall Model



Quick

Advantages:

1. Base Model:
2. Simple and Easy
3. Small project

Disadvantages:

1. No feedbacks
2. No experiments
3. No Parallelism
4. High risk
5. 60% efforts Maintainance.

The V model

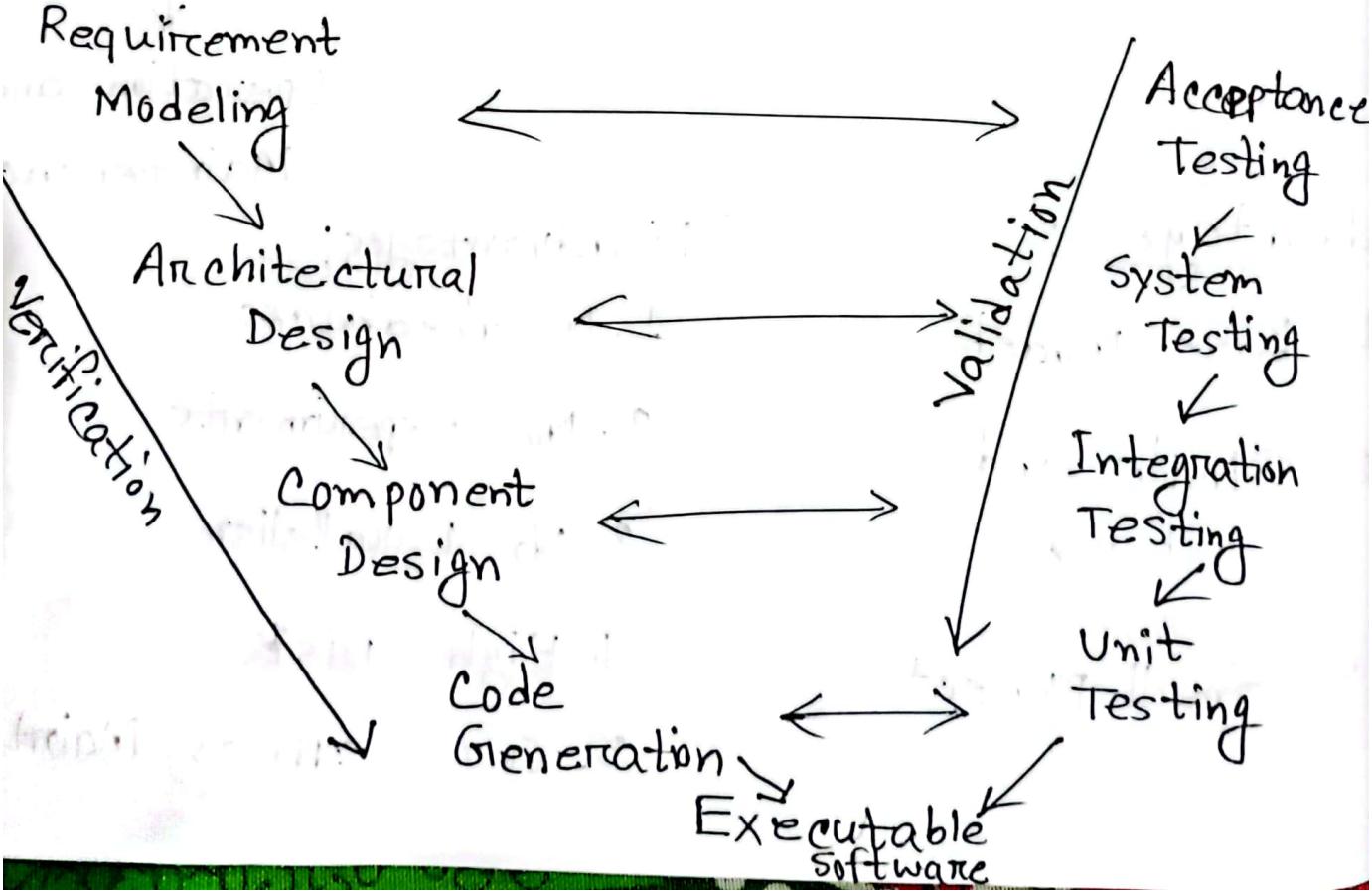
→ Verification & validation model

→ Extension of waterfall

→ Testing is assosiated with every phase of life

→ Verification Phase (Requirement analysis system design, architecture design, module design)

→ Validation phase (Unit testing, Integrati system, Acceptance Testing)



Advantages:

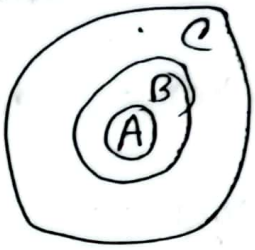
1. Time saving
2. Good understanding of project in the beginning
3. Every component must be testable.
4. Progress can be tracked easily.
5. Defect Tracking

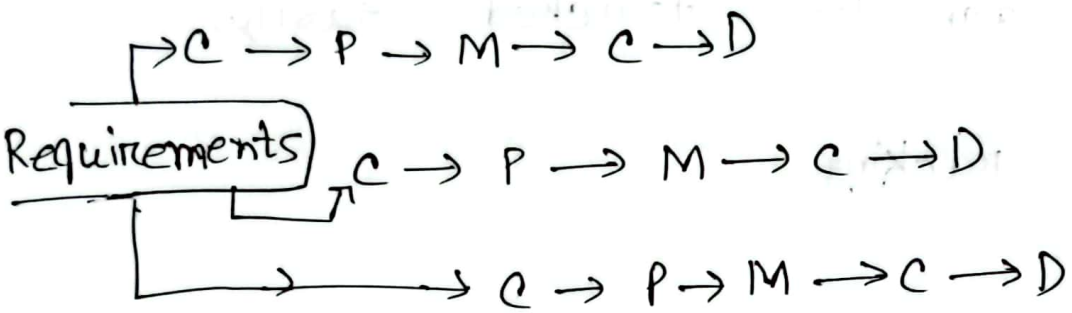
Disadvantages

1. No feedback so less scope of change
2. Risk analysis not done
3. Not good for big or object - oriented projects.

The Incremental Model

Communication → planning → Modeling → Construction → Deployment





Advantages:

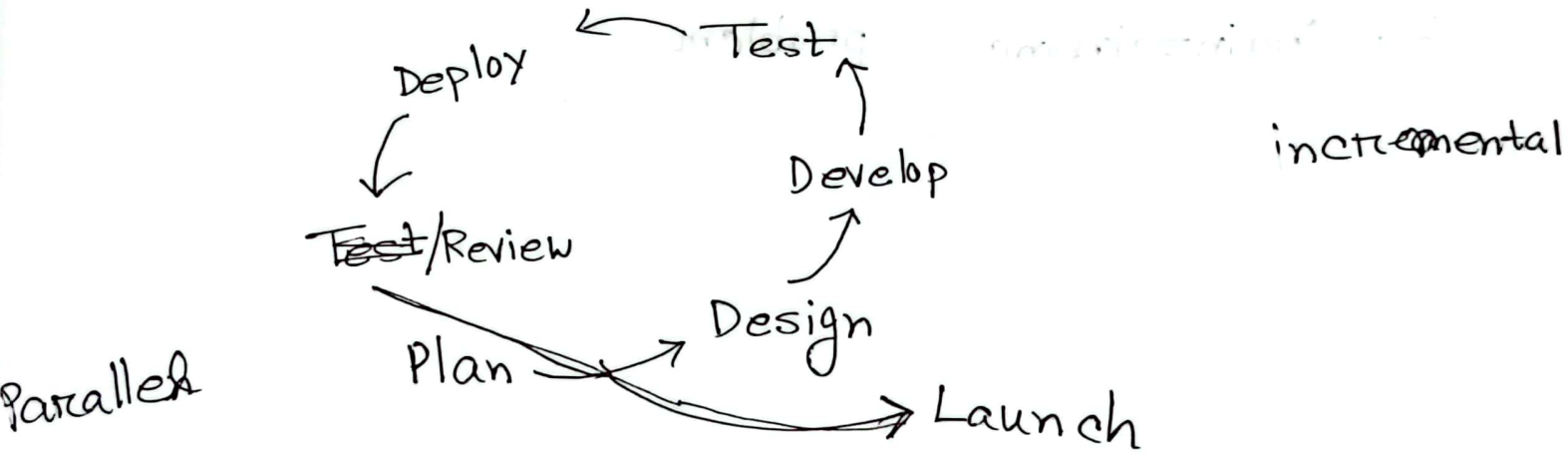
Large software

1. Important modules are developed first and then the rest are added.
2. Working software is prepared quickly and early during the software development.
3. Flexible.
4. Customer interaction maximum.
5. Project progress can be measured.

Disadvantages:

1. Management is a continuous activity that must be handled.
2. Total cost of this model is higher.

Agile Model (moves quickly)



→ Breaks the large projects into small Chunks / iterations

→ Work on the iterations parallely

→ ~~Each iteration~~

Advantages:

1. Frequent Delivery
2. Face to Face communication with client
3. Changes
4. Time

Disadvantages:

1. Less Documentation
2. Maintainance problem

Extreme Programming (XP)

→ is an approach / methodologies for agile

→ is a software development methodology which is intended to improve software quality and responsiveness to changing user requirements.

XP values

Five ~~component~~ values

Communication: Taking regularly is very important in XP. Instead of writing of long documents, people on the project talk often to make sure everyone understands what's going on.

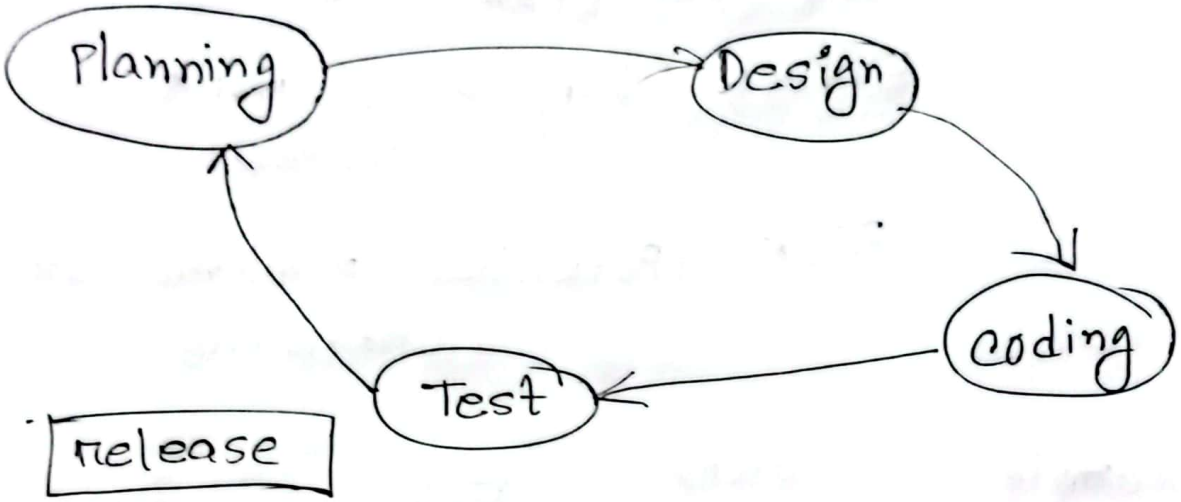
Simplicity: Keep things simple. Build only what's needed right now. Don't add extra features unless they are necessary. If things change later, you can update the code.

3. Respect : Everyone in the team should respect each other. Every person's ideas and contributions matter, and this helps create a good work environment.

4. Feedback : Get feedback from three places :- the software it-self, the customer, and your teammates. Testing helps ensure the software works as expected.

5. Courage : Have the courage to make decisions based on what's needed now, ~~or~~ even if it means making big changes later. Teams should be willing to adapt and improve the code as things evolve.

XP Process



1. Planning : Begins with the creation of user stories, which are short, simple descriptions of ~~the~~ what the user wants the system to do.

   Structure :

   As a (user type) I want to (action) so that (reason).

   Example : As a library member, I want to search for books by title so that I can find the books I need ~~que~~ quickly.

Each story given a score (story points) that shows how much effort on

P.T.O.

time it will take to finish it.

Factors: complexity (how hard it is)

Effort (how much work it takes)

Risk (Potential problems or unknowns)

Example	Story Points	Efforts Required	Deadline
Implementing login screen	1	Low	5 hours

2. Design: The design is kept as simple as possible, focusing only on what is needed right now ("keep it simple" principle

Class responsibility collaborator

↖ CRC cards are used to plan classes:

Class: Name of the class

Responsibilities: What does it do?

Collaborators: What other classes does it work with?

3. Coding: Promotes pair programming. (Two developers work together at one workstation, improving code quality and sharing knowledge.

Code refactoring involves constantly improves the structure of the code without changing what it does, to make it cleaner and easier to maintain.

4. Testing: Unit testing: All unit tests are run everyday to catch issues early.

Acceptance tests are defined by the customer to check if the software works as they expect.

Advantages:

1. Features are built and delivered quickly and often, so there's steady progress.
2. Having customer on the team helps avoid confusion about what the software should do.
3. The customer works closely with the team, giving feedback and making sure the software meets their needs.
4. XP promotes teamwork and sharing knowledge, so team members can easily pick up each other's work. This helps keep the team stable and reduces the chance of people leaving.

Disadvantages:

1. There's lot of pressure to work fast because code has to be written and tested right away, which can make it stressful.

2. XP requires the customers to be available for regular meetings, which can be tough if they are far away or busy.

Industrial Extreme P (IXP)

IXP is an improved version of XP, made for bigger projects and larger teams, often working in different places. It still focuses on simplicity and customer involvement but adds new steps to manage complex projects.

Here are six new steps in IXP:

1. Readliness check: This step makes sure the team and company are ready for IXP. It checks if everything is in place, like the right environment and people, before starting the project.

2. Project Community: In IXP, the "team" includes more people, like legal experts, quality checkers, and others who help with the project.

3. Project planning: This step checks if the project is worth doing and how it fits with the company's goals, so, everyone knows why the project is important.

4. Goal Tracking: IxP sets clear goals and regularly checks progress to make sure the team is meeting its targets!

5. Look Back: After each part of the project, the team reviews what went well and what could be better.

6. Keep Learning: Team members are encouraged to keep learning new skills to improve the product and how they work.

Requirement Analysis

Requirement Analysis helps make sure a software project succeeds by understanding what it needs to do. There are two main types of requirements:

1. Functional
2. Non - Functional

Functional Requirements:

It describe what the product should do. They focus on features that allows users to achieve their goals. It needs to be clear, simple.

Example:

The system must "send a ~~notifie~~ confirmation email when an orders is placed."

User story Example:

As a user, I want to log into my account.

Feature: Login

Functional Requirements:

The system must allow users to log in with their email and password.

The system must allow users to reset their password by email.

Non Functional Requirements (NFR):

These describes how the system operates and it's quality (speed, security, usability)

Example:

"The system must load a dashboard within 2 seconds"

The system must support up to 40000000 users without slowing down.

1. Performance : How fast the system responds and processes data.

Ex : A web app should load the user's dashboard in 2 seconds after login to provide a good experience.

2. Portability : The ~~abati~~ ability of the system to run smoothly on different platforms.

Ex : The web app should work equally well on windows and macOs.

3. Scalability : The system's ability to handle more users or data by adding resources or upgrading existing resources.

Ex : The system should support 4000000 users without slowing down.

9. Reliability: How dependable the system is, even in different situations.

Key factors

- Accuracy: Does the system provide correct results?

ex; GPS shows the correct location within a few meters.

- Error Tolerance: Can the system handle errors without crashing?

Ex: If Facebook fails to load, it shows a helpful message instead of crashing.

- Consistency: Does the system work the same everytime?

Ex: Google always gives accurate weather info.

- Simplicity: Is the system easy for beginners?

Ex: Google search bar.

5. Usability: How easy and pleasant the system is to use?

Key factors: Learnability

Efficiency

Satisfaction

6. Reusability: Whether parts of the system can be reuse in other projects

Ex: A library of functions for handling dates

7. Maintainability: How easy it is to fix, update, or add features to the system

Ex A well-designed web page allows developers to add new features.

8. Security: The system's ability to protect against unauthorized access and threats.

5. Usability; How easy and pleasant the system is to use?

Key factors: Learnability

Efficiency

Satisfaction

6. Reusability: Whether parts of the system can be reuse in other projects.

Ex: A library of functions for handling dates

7. Maintainability: How easy it is to fix, update, or add features to the system.

Ex: A well-designed web page allows developers to add new features.

8. Security: The system's ability to protect against unauthorized access and threats.

Ex: A banking app uses encryption.

9. Availability: How often the system is up and ~~down~~ running.

Ex: An e-commerce should be available 99.9%.

10. Capacity: How much data or how many users the system can handle at once.

Ex: A cloud storage system should handle millions of files without slowing down.

Difference between FR and NFR

FR	NFR
→ describes the behaviors (functions or services) of system	→ are system properties
→ support user goals, tasks or activities	→ supports user expectations
→ answer what a software will do	→ answer how the software will do
→ FR testing can be API testing, integration testing.	→ NFR can be stress testing, security testing

Use Case Diagram

→ The behavior of a system.

→ interaction between actors and a system.

→ Two type of actors

Primary, secondary

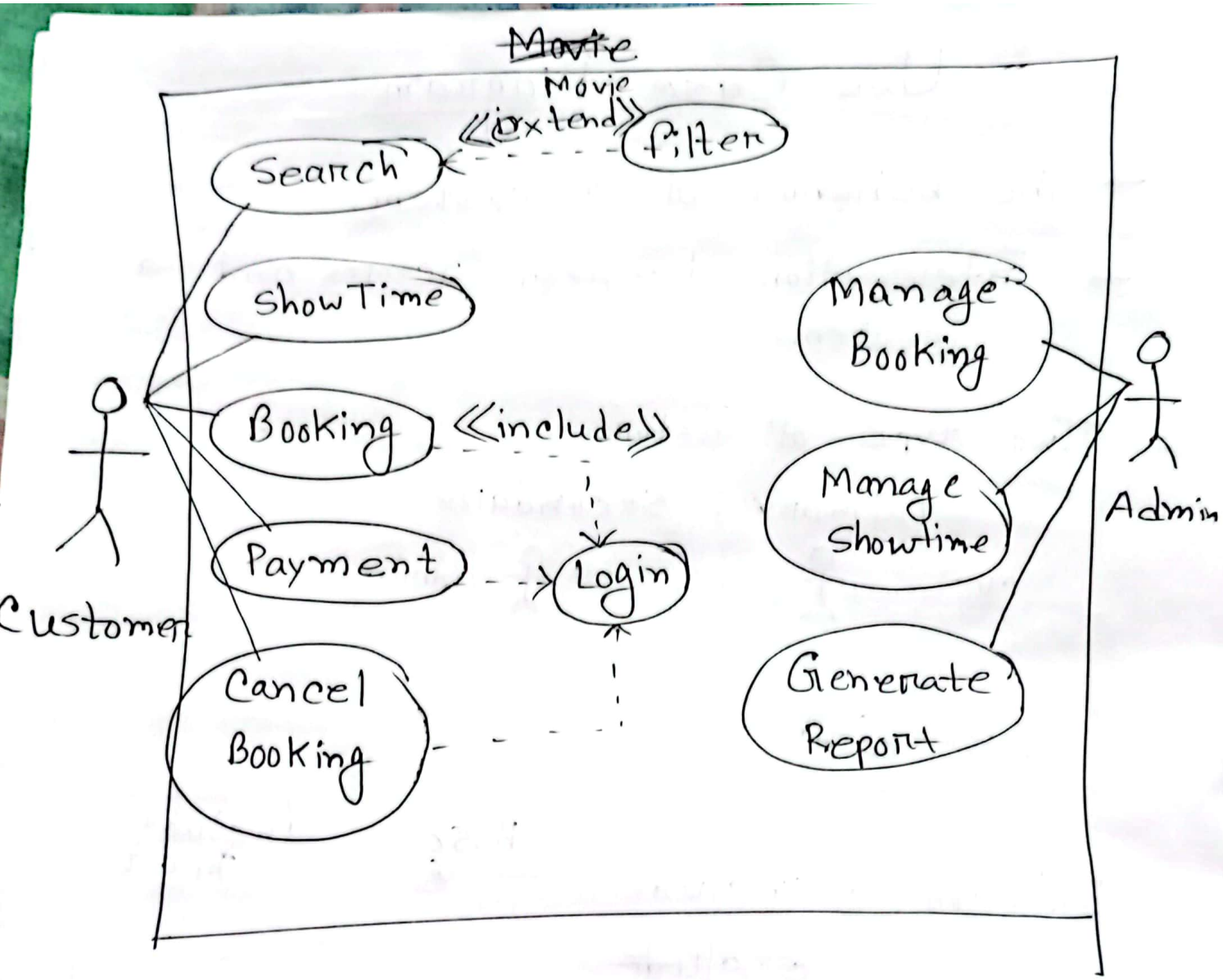
customer   admin

→ 

→ Relation ⇒ include ——— Base - - - ≫ include must

~~exclude~~

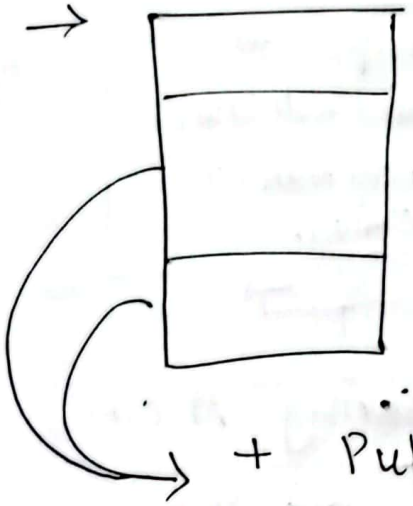
extend Base - - — — — optional extend


Movie
Movie
«extend»
Filter
Search
ShowTime
Booking
«include»
Payment
Login
Cancel Booking
Customer
Manage Booking
Manage Showtime
Generate Report
Admin

Class Diagram

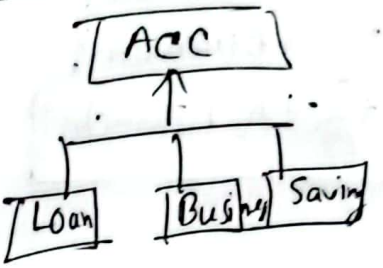
→ represent the structure and relationship of classes in a system

→ Follows OO concept

→ 

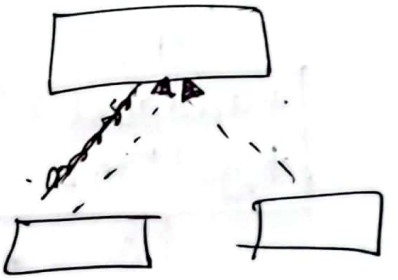
+ Public
- Private
Protected

inheritence



Dependency

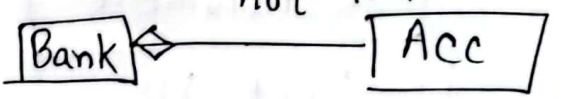
Realization :



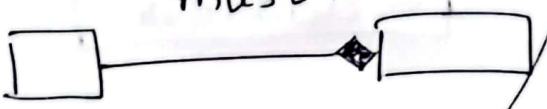
→ Relations :

Association :

Aggregation :

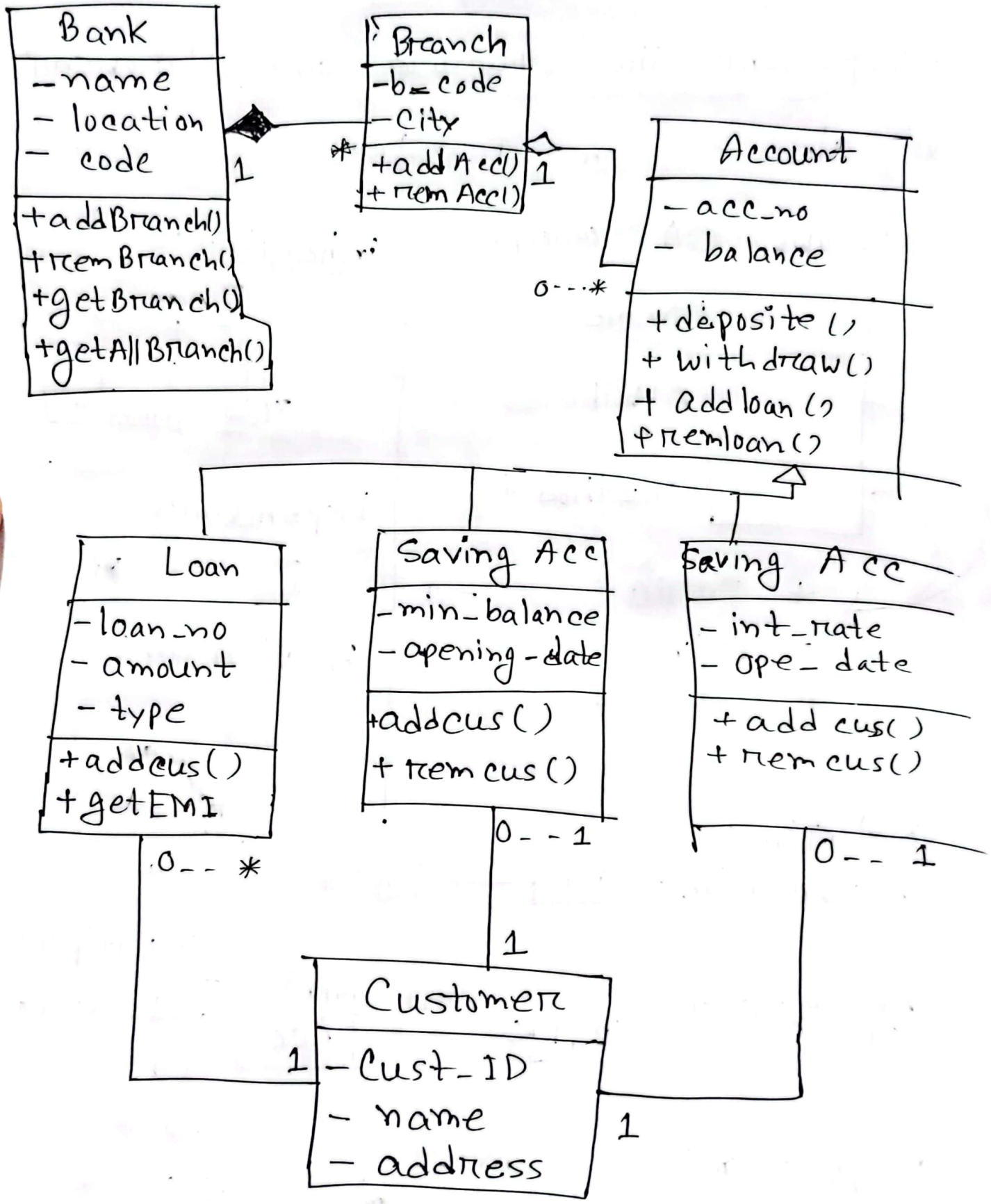


Composition :



Multiplicity

→ 1 exact 1
→ 0 .. 1
→ *
→ 0 . . . *
→ 1 . . . *


Bank
-name
- location
- code
+addBranch()
+remBranch()
+getBranch()
+getAllBranch()
Branch
-b_code
-city
+addAcc()
+remAcc()
Account
- acc_no
- balance
+deposite()
+withdraw()
+addloan()
+remloan()
Loan
-loan_no
- amount
- type
+addcus()
+getEMI
Saving Acc
-min_balance
-opening-date
+addcus()
+remcus()
Saving Acc
- int_rate
- ope_date
+addcus()
+remcus()
Customer
-Cust_ID
- name
- address
1
*
0..*
0..1
0..*
0..1
1