# OBJECT ORIENTED SOFTWARE ENGINEERING.

## • Software Development Life Cycle Model. (SDLC)

→

→ Object Oriented Software Life Cycle Models.
① Bases of OO methodology are objects.
② Focus is on objects.
→ Objective: Decrease maintenance.

(I) Object Oriented Requirement Analysis:
↳ Use case approach is used for capturing requirements from the customer.

(II) Object Oriented Analysis.
→ Ideal structure is created.

(III) Object Oriented Design.

(IV) OO Programming and Testing.

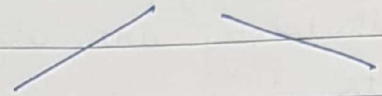
→ • Models:

① Fountain Model.
→ Emphasizes on reusability of source code.
↰ → Iteration.
○ → Overlapping phases.

② Rational Unified Process
→ Maintained by rational. Software.
→ Features:
(i) Iterative Software Development.
(ii) Effective Requirement Capturing.
(iii) Visual Modelling
(iv) Ensuring Quality
(v) Change Control Management
(vi) Automated Testing

→ Software Baseline → When we complete a phase and enter new phase.
After software baseline is crossed, changes are done through change control management.

→ RUP



Static

→ Describes roles, activities, artifacts, workflow, disciplines.

Dynamic

→ Inception, Elaboration, Construction, transition.

→ RUP consists of six major disciplines.

① Inception → Est. scope and boundary.
→ Determination of cost.
→ Dev. of initial iteration plan.
→ Identify high risk.

② Elaboration →

③ Construction.

④ Transition → Delivering, training users.

Assignment → Submit on Thursday.

① Compare and contrast waterfall model, prototyping, iterative enhancement, spiral and RUP.

# Chapter - 3

## Software Requirements Elicitation and Analysis

→ Combine use case approach with interview, brainstorming and FAST.

→ Every affected person in a system is a stakeholder.

(I) Internal people of customer's organization.
- → Customers
- → Users.

(II) External people of customer's organization.

(III) Internal people of developer's
- → Developers, programmers, testers.

(IV) External people of developer's.
- → Consultants, outside domain experts, third party testers.

→ ① Interviews
- → Selection of stakeholder
- → Distribution of questionnaire.
- → Preparation of list of req.
- → Preparation of consolidated list

Interview and brainstorming are most popular. Use case is used in combination of these.

→ ② Brainstorming
③ FAST.

④ Prototyping
→ It is made when unstable requirements.
→ Prototype is discarded after req. are final.

→ First outcome of req. gathering — IRD.

→ Use Case Approach.
- Addresses only the functional requirements.
- Use Case Diagram
  ↙
  Relation b/w actor and use cases
  → Gives top view.

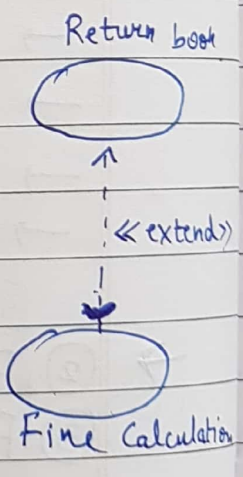
(1) Actor — Outsider to the system that interacts with system.

(2) Use Case - Dynamic relationship b/w actor and system.
- Every functionality should have use case.
- Unique name is assigned to a use case.
- Initiated by an actor with purpose.
- Role of actors must be defined clearly.

Eg In LMS,
① Maintain book details.
② Issue Book
③ Return Book etc.

→ Use Case — Type of relation b/w use cases.

(I) Extend.
→ Eg fare calculation is extra functionality to return book.




(II) Include → Common functionality to more than one use case.

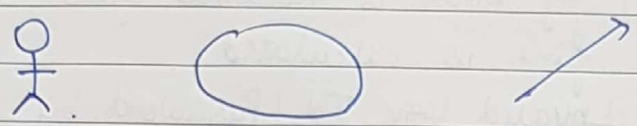
CAD * UML Use Case

→ Identify use case, actor, UML diagram, Use Case Template.

→ Types of flow.

① Basic Flow
→ Describes sequence of events that takes place most of time

② Alternate Flow

→ 

Actor Use Case. Relationship.

→ Use Case Template.

① Brief Description → Purpose is to login user/faculty into system

② Actors

③ Pre Condition → State of system before the use case begins (Eg for issue book → login).

④ Post Condition → State of system after use case ends.

⑤ Basic Flow and Alternative flow.
↳ (i) Invalid Id / Password
(ii) Blank Id / Password
(iii) User leaves in between.

⑥ Special Req.

⑦ Associated use cases → Eg for issue book → login

→ Consider return book use case. Write use case description of the same.

→ ① Brief Description → The purpose is to return book.
② Name of Actors → Admin, Library Staff,
③ Pre Condition → Login.
④ Post Condition → ~~Database is updated~~. Book is returned successfully ~~without~~
⑤ Basic Flow → (i) Library staff scans library card of the employee / student / faculty.
(ii) Employee / faculty / student returns book.

(iii) Availability of slots of books for student/employee is increased
(iv) No. of books is increased in library database
(v) Stamp is cancelled from the book.

Alternative Flow : ① If book is returned after due date, fine is calculated.
② Invalid login Id / Password by library staff
③ If book is not issued in name of student/employee/faculty,
④ User exits

⑥ Special Requirements — None.

⑦ Associated Use Case — Login, Fine Calculation.

Consider maintain book use case.
① Brief Description : The purpose is to add/update/delete/view book details in the system
② Name of Actors : Admin,, Data Entry Operator, Library Staff
③ Pre Condition : Login.
④ Post Condition : The database is updated successfully.
⑤ Basic Flow : ① Add a book
→ (i) The actor enters details of books in database. (BookId, Author, Name, ISBN)
(ii) Book info is successfully saved.
② View
③ Delete.
④ Update.

(5) Alternative Flow : ① Updation cancelled because updation not confirmed by actor. Use case ends there.
② Deletion not allowed if book is already issued to member.
③ Updation not allowed if book is issued.

(6) Special Req → None.

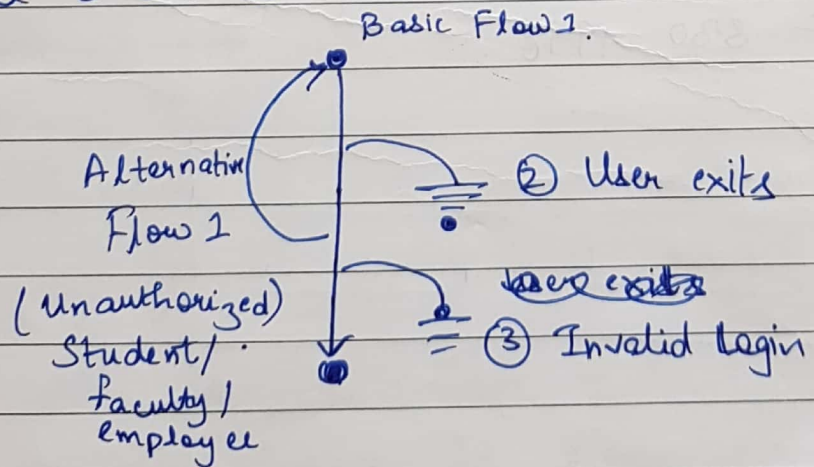
→ Application of use case description
↳ ① Extracting attributes.
↳ ② Generating test cases
↳ ③ Used for use case scenario diagram.

→ Use Case Scenario — Path in a system.

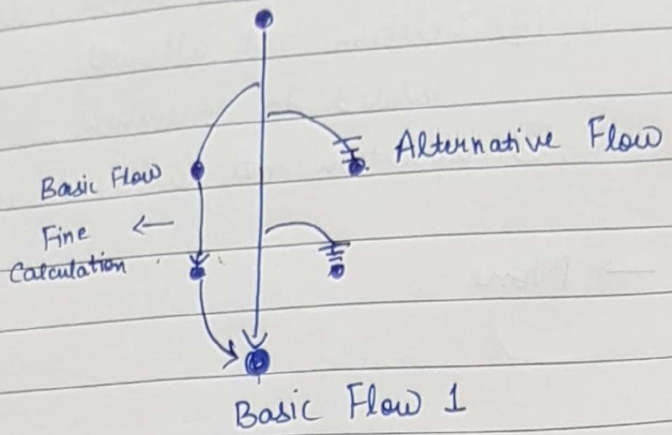
→ Assignment :
Consider fine calculation use case. Write use case description of the same.

→ Use case scenario is an instance of a use case.

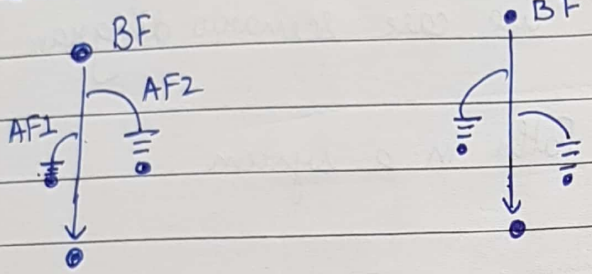
→ Issue Book Use Case Scenario.



→ Scenario Diagram for return book use case.




→ Scenario diagram for maintain book details.
① Add. ② Delete ③ Update ④ View




→ Use case Scenario Matrix.
↳ Used for generating test cases.

→ Ambiguous words like robust, safe, accurate etc. should be avoided in SRS.

→ SRS - IEEE 830 -1998.

LAB
→ Section 1

Date
Page

- SRS

→ Nature of SRS document.
→ ① Functionality
② External Interface.
③ Performance.
④ Quality Attributes → Maintainability
⑤ Design constraints Imposed.

(1) Introduction.
→ 1.1) Purpose
1.2) Scope.
→ Assign a name.
→ Do's and don'ts of the software.
→ Application of software.
→ Maintain consistency.
1.3) Definitions, Acronyms and abbreviations.
1.4) References.

2 → Requirements.

→ Mid-Sem Syllabus :

→ Section-3 →
(1) External Interface
→ Specify format of all the external interfaces, output forms, reports.
→ Eg
Barcode Reader → Accession No. / Book_Id.
Member_Id, Member_Name.
Date.
Issue | Cancel

→ If any of validation check does not hold, appropriate error message is shown.

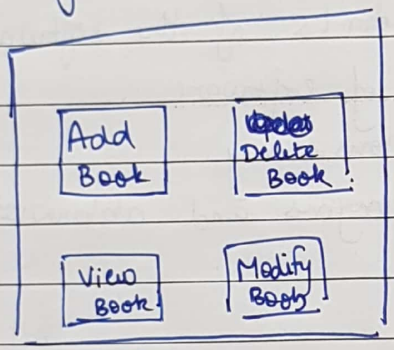
→ Describe each and every field used in the scree
Eg member_id → It is 11 digit number.

→ Validation Checks →
Member_Id → Its alphanumeric type and cannot be left blank.

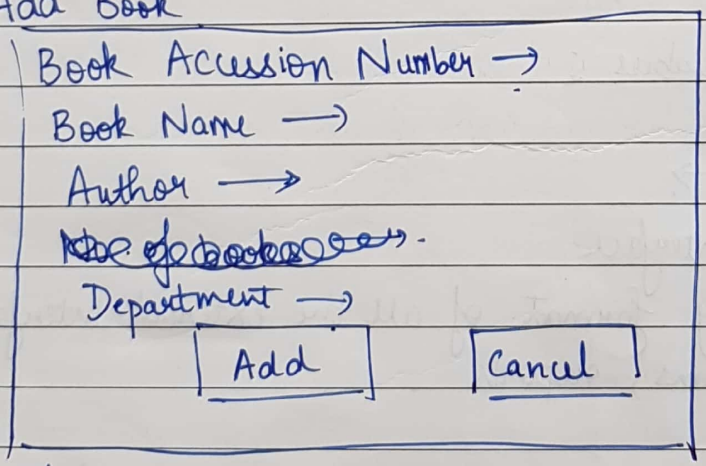
→ Use Case Description
→ Error Handling →

Q Consider maintain book details use case. Create external interface for it. Describe each item in the interface and design validity checks for it.

→ External Interface.

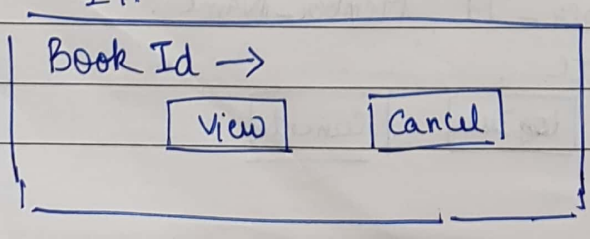



(1) Add Book




(2) View Book

I/P.




O/P

Book_Id →
Name →
Author →
Department →

Validity Check (Add a Book)
(1) Admin
(2) Accession Id should not be blank
(3) Each book should have unique accession id

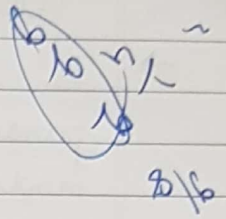
(3) Modify Book.

→ I/P.

Enter Book Id →

[Modify] [Cancel]

Book Id →
Name →
Author →
Department →
[Confirm] [Cancel]



(4) Delete Book.

→

Enter Book Id →

[Delete] [cancel]

→ Description of Field.

Section 3 → Performance Requirements.
Eg Simultaneously 3 users can work on the system.

→ Logical Database Design.
— Tables and their relationship.

→ Design Constraints

→ Requirement Change Management.
Every req. should be traceable to class
Change control board is official authority that decides whether change has to be made or not. Every change is officially notified in the formal document. The procedure is known as configuration management.

# Chapter - 5

## Object Oriented Analysis

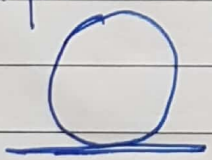
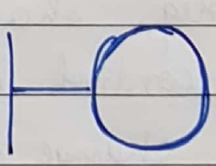
→ It is second most imp phase of object oriented paradigm. OOA constructs a robust and ideal model that is free from details of implementation environment.

→ It is ideal because we don't consider implementation. Advantage is that we can shift easily to another environment

→ Data – Attributes
Functions – Operations

→ Output of this phase → Class diagram.

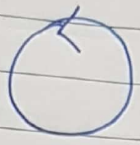
→ Class

- Entity
  - → Classes that are going to persist longer into system
  - Eg transaction, member, book
  - → Represented by
  - 
  - → Instances of entity classes are known as entity objects.
  - → Attributes and op of entity class may be identified by actor of use case.
- Interface
  - → Handle interaction in the system
  - → Interface b/w actor and system
  - → Also known as boundary classes
  - → Represented by
  - 
- Control.

Midsem — 1, 2, 3, 5

IEEE - Class Diagram (2017)

→ Control Classes.
- Are responsible for coordinating and managing entity and interface classes.
- Each use case has 1 control class.
- Represented by:



→ CLASS DIAGRAM — Depicts interaction between classes for a whole system.
→ Only one class diagram for whole system.

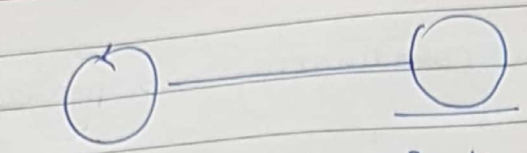
Eg In LMS.
=>
① Member - Entity
② Book - Entity.
③ Transaction - Entity
④ IssueBook Interface - Interface
⑤ Barcode Reader - Interface
⑥

→ Return Book (Identification of classes)
① Member
② Book
③ Transaction
] Entity.
④ ReturnBook Interface.
⑤ BarCode Reader.
⑥ ReturnBookController.
⑦ Fine Calculation Control
⑧ FineStatus → Entity.

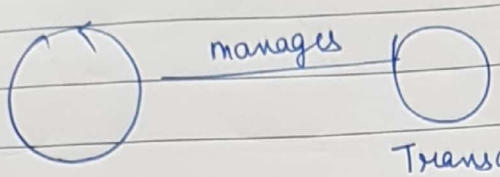
→ Identification of Relationship Among Classes.
- ① Generalization
② Association
③ Dependency
④ Composition
⑤ Aggregation

→ Association → Structural connection. Mostly bidirectional. Classes are aware of existence of other class.

Association relationship may be associated with a name.

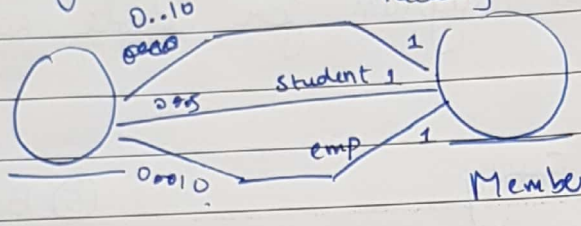


Issue Book Controller — Book




Transaction

→ Multiplicity → Represents no. of instances of classes related to one instance of another class.




Transaction — Member

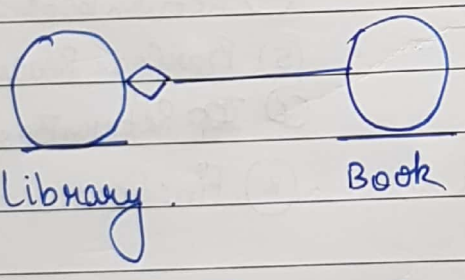
(0..*)

min_value .. max_value

(3) Aggregation.

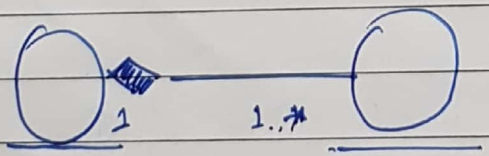
→ Whole part type of relationship.
Represents 'has-a' relationship.

→



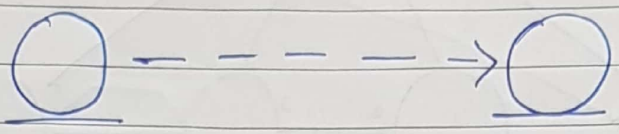
Library — Book

(3) Composition → Represents strong form of whole-part relationship.



④ Dependency

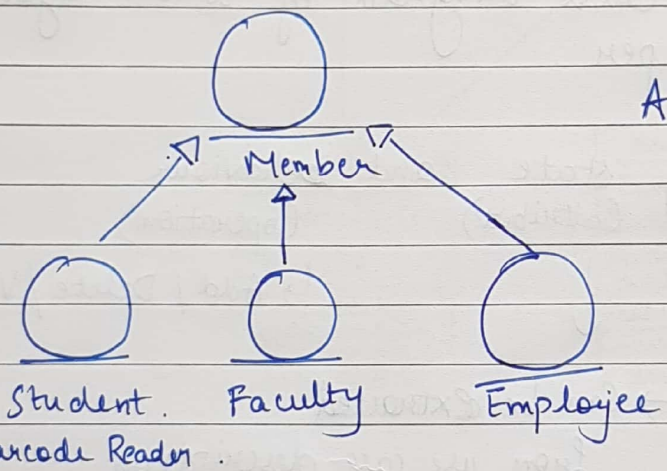
→ One class is used as attribute of another class. Change in the class being referenced affects class using it.



SchoolSchedule. Program.

add (prog: Programme)

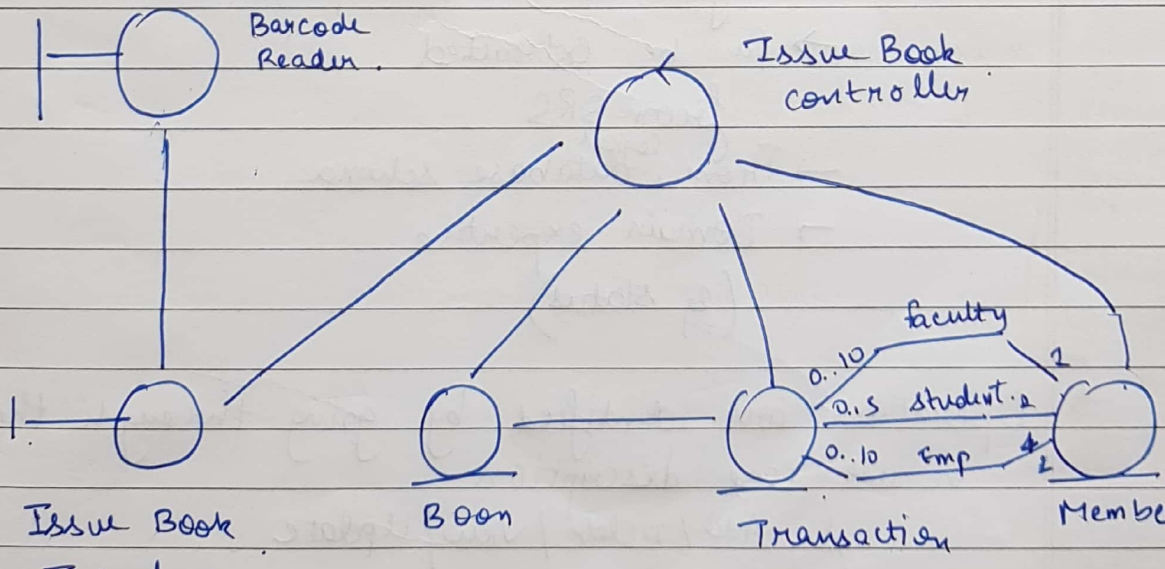
⑤ Generalization / Parent-Class Relationship.

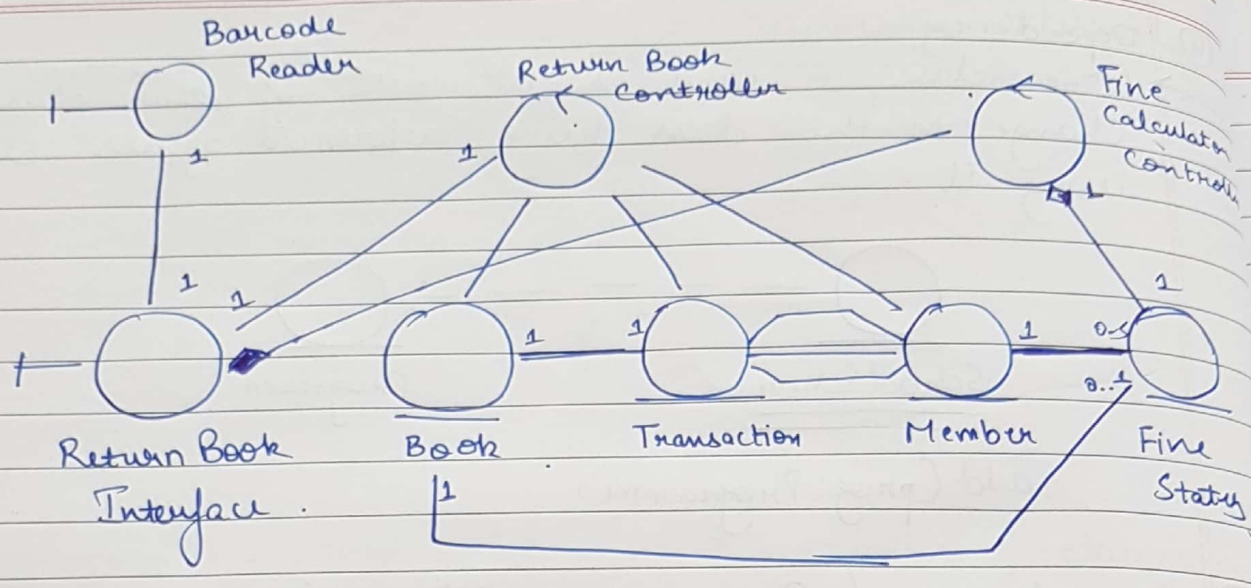



Arrows is towards parent class.

Barcode Reader.

→






LAB :

① Create class diagram of whole system on chart paper.

Identifying state (attribute) and behaviour (operation)

↓

↳ Add / Delete / View / Update

→ Can be extracted from use case description.

→ Can be extracted from SRS

→ From logical database schema

→ Domain expertise (eg status)

Operations are identified by going through the flow in use case description

↳ Add / Delete / View / Update

↳ ~~Issue Status~~ getBookStatus

→ Issue Book.

→ We need to store date of issue.

→ (Barcode Reader) sends message to (Issue Book Interface) This message sending is operation of receiver class

→ Issue Book Use Case with attribute and operation