OOSE

Unit-II :- Object Oriented Methodologies: Rumbaugh Methodology, Jacobson Methodology, Blooch Methodology, Patterns, Frameworks, The Unified approach, Unified Modeling Language (UML)

* **Rumbaugh Methodology** :- object -(object modeling technique) was developed in the late 1980s at the general electric research and development.

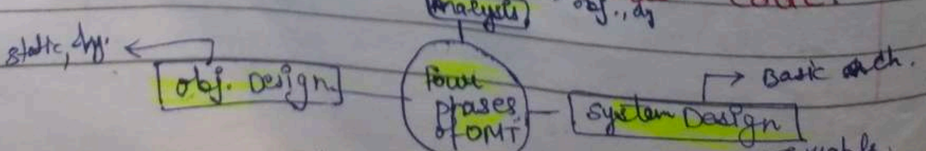
→ Rumbaugh's ~~Method~~ OMT describes the method for the analysis, design & implementation of system using an obj. oriented technique.

Rumbaugh's OMT :-

- ★ A method for analysis, design & implementation by an obj. oriented technique.
- ★ fast & intuitive approach for identifying & modeling all objects making up a system.
- ★ class attributes, methods, inheritance & association can be expressed easily.
- ★ Dynamic behaviour of objects can be described using the OMT dynamic model.
- ★ Detailed specification of state transitions and their descriptions within a system.

→ Four phases of OMT (can be performed iteratively)

- ⊛ Analysis :- obj's, dynamic & functional models.
- ⊛ System Design :- Basic architecture of the system.
- ⊛ object Design :- static, dynamic & functional models of objects.
- ⊛ Implementation :- reusable, extendible & robust code.



OMT models:-

1) ~~An~~ obj. model:- obj model & data dictionary.

2) ~~A~~ dynamic model:- state diagrams & event flow diagrams.

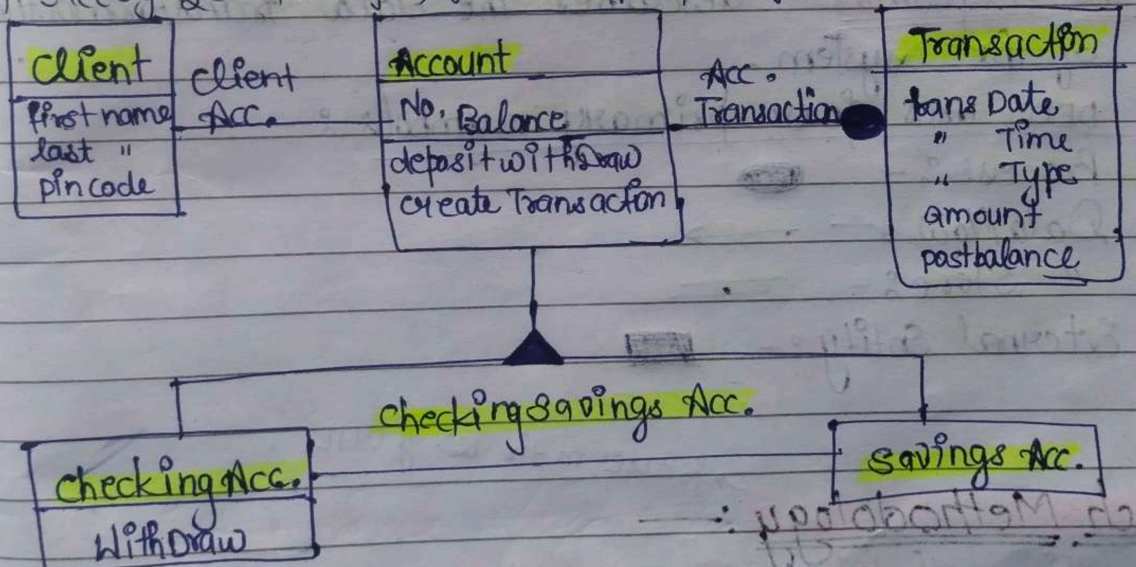
3) ~~A~~ functional model:- data flow & constraints.

OMT obj. Model :- ① Describes the static structure of the obj. in the system & their relationships.

② The obj. model is represented graphically with an obj. Diagrams.

③ The obj. diagrams contains classes interconnected by an association lines. # (∵ Acc. → Account)

④ It recognize the obj, relationships b/w obj, & classes.



→ OMT obj. model of a bank system. boxes represent classes & filled Δ represent specialization. Association b/w Acc. & Transaction is one to many; since one acc. can have many transactions, the filled circle represents many (zero or more). The relationship b/w client & Acc. classes is one to one. A client can have only one acc. & acc. can belong to only one person (joint acc. not allowed)

OMT Dynamic Model:— It represent those aspect of a system that are concerned with time & changes.

Event :— An event is something that happens at a pt. in time.

State :— A state is an abstraction of the attribute value & link of an obj.

State Diagram :— A state diagram is a n/w of states & events.

OMT Functional Model:— It describes those aspects of a system concerned with transformations of values.

→ It specifies the result of a computation without specifying how or when they are computed.

Data Flow Diagrams :— describes the data transformation of the system.

⊛ DFD uses four primary symbols :—

1. Process :— ⬬
2. Data Flow :— →
3. " Store :— ═
4. External Entity :— ■ △

⊛ Booch Methodology :— →focuses on the use of UML.

→ This methodology was developed by G. Booch in the early 1980's

→ It is a widely used obj. oriented method that helps you design your system using the obj. paradigm.

→ It consists of following diagrams :—

★ obj. Diagrams
★ class "
★ State Transition "
★ Module "
★ Process "

★ Interaction Diagrams.

DATE
PAGE

class diagram:—

(i) describe roles and responsibilities of obj.'s

(ii) obj. diag :— Describe the desired behavior of the system in terms of scenarios.

(iii) State of a class based on a stimulus.

(iv) Map out where each class & obj. should be declared.

(v) To determine to which processor to allocate a process.

(vi) Describes behavior of the system in terms of scenarios.

This methodology follows 2 processes :—

(i) Macro Development Process :— → mainly focus on techincal management of system.

→ steps for macro development process —

(i) Conceptualization:— Establish the case requirements & develop a prototype.

(ii) Analysis & development of the model:— uses class diagram to describe roles & responsibilities of objects.

uses the obj. diagram to describe the desired behavior of the system.

(iii) Design system Arch. :— → Uses class diagram to decide what classes exists & how they relate to each other.

→ " objects " " " " objects involved & their attributes with operations.

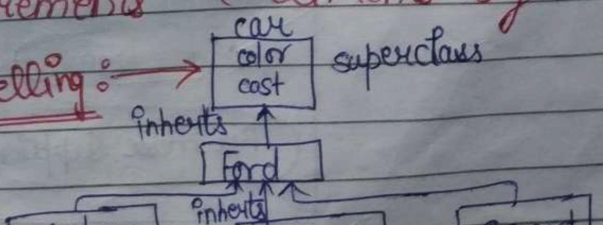
→ Process diagram to identify various processes & functions.

→ Module " " " " modules with each other.

(iv) Implementation:— Refine the system through rigorous iterations.

(v) Maintenance:— Make changes to the system for adding new requirements & eliments bugs.

obj. Modelling :— →



DATE
PAGE

(ii) Micro Development Process :— It describes the total detailed activities (day to day activities)

→ It involves following steps :

(i) Identify classes & objects

(ii) " " " " with ~~this~~ their semanties

(iii) " " " " relationships.

(iv) " " " " Interfaces & implementation.

⊛ Jacobson Methodologies :—

→ This methodology focuses on reuse of analysis & design work.

→ It also focuses on use ~~for understandin, case~~ concept for system designing.

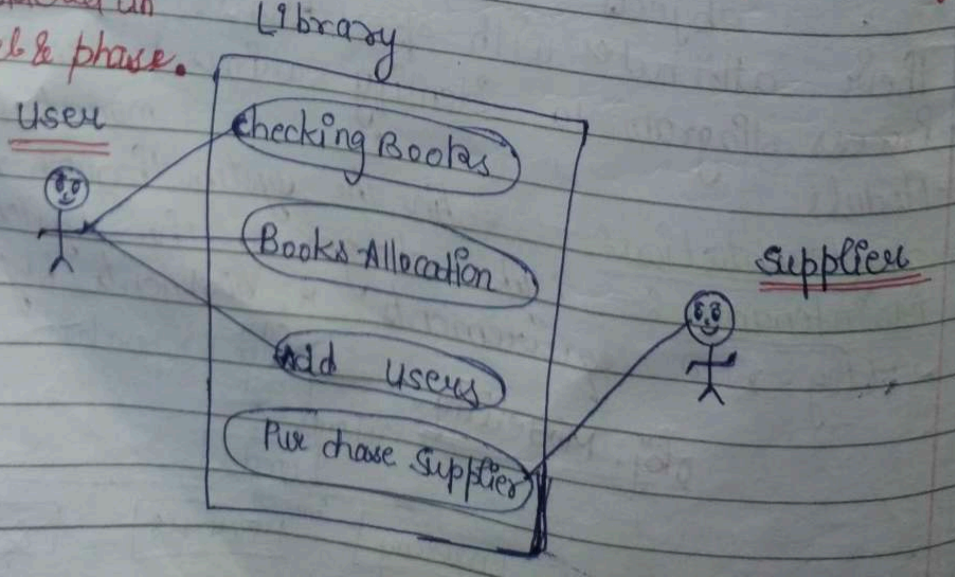
Use Cases :— ⊛ scenarios for understanding the system requirements.

⊛ Easy to read & understand.

⊛ Used for understanding requirement b/w the users & system.

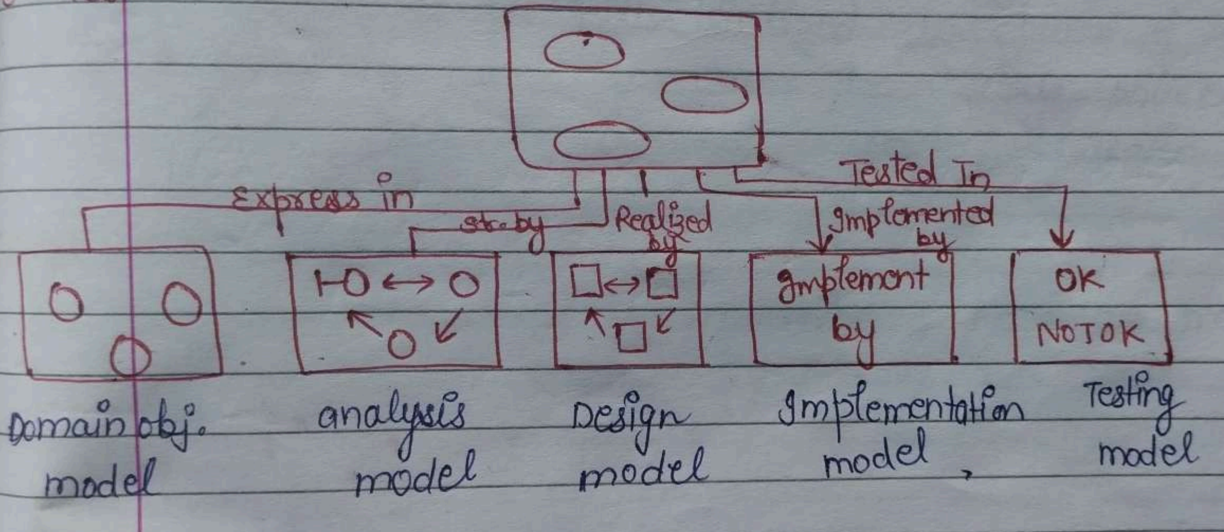
⊛ captures the goal of the user & responsibility of system to its users.

⊛ It is considered in every model & phase.



Objectory is build around several different models:—

- Use case model
- Domain object "
- Analysis " "
- Implementation "
- Test "



Booch was great in design
OMT & OOSE were great in analysis

DATE

PAGE

→ It ~~follows~~ follows 2 approache cases :—

→ Object Oriented Software Engineering (OOSE)

→ " Business " (OOBE)

OOSE :— ① ~~Sem~~ Stands on this

② Development process is also called as use case driven approach.

③ It aims to develop large and real time systems (Satelite, whether, Rocket Launch)

④ Its a process for industrialized development of s/w.

⑤ Focuses mainly on use cases for various phases of development such as Analysis, design, Validation & Testing.

OOBE :— ① Stands on ———

② Also follows use cases for modelling & providing traceabilty throughout the s/w engineering process.

③ It Involves Various phases.

(a) Analysis phase

(b) Design & Implementation "

(c) Testing "

Model :— Abstract represtantion of a system → static → It provides parameters to the system at a specific pt. in tym. eg:- class diag.
→ dynamic → represent a system behavior over tym eg:- Interact. & activity diag.

*UML :— → it stands on Unified Modeling Language

⇒ The UML is a graphical / standard Language for visualizing, specifying, constructing & documenting the software system & its components.

→ UML can be used to support your entire life cycle. Fic

→ UML supports diagrams.

→ Diagrams :- Graphical presentation of model elements, which is connected through vertices (things) & arcs (relationships)

things + relationships ⇒ diagrams

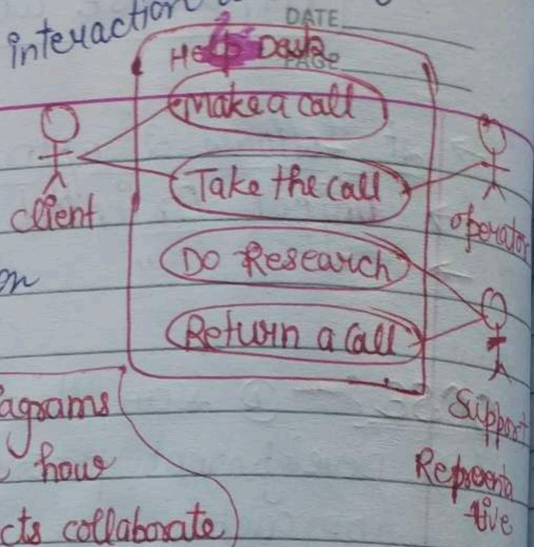
Diagrams

1. ~~Use case~~ diagram → represent user's interaction with the system.
2. Activity "
3. Sequence "
4. Collaboration "
5. Class "
6. State Chart "
7. Component "
8. Deployment "
9. obj. "

Sequence & Collaboration → Interaction diagram

↓ I.D. are diagrams that describe how groups of objects collaborate to get the job done.

Component & Deployment → Implementation Diagrams

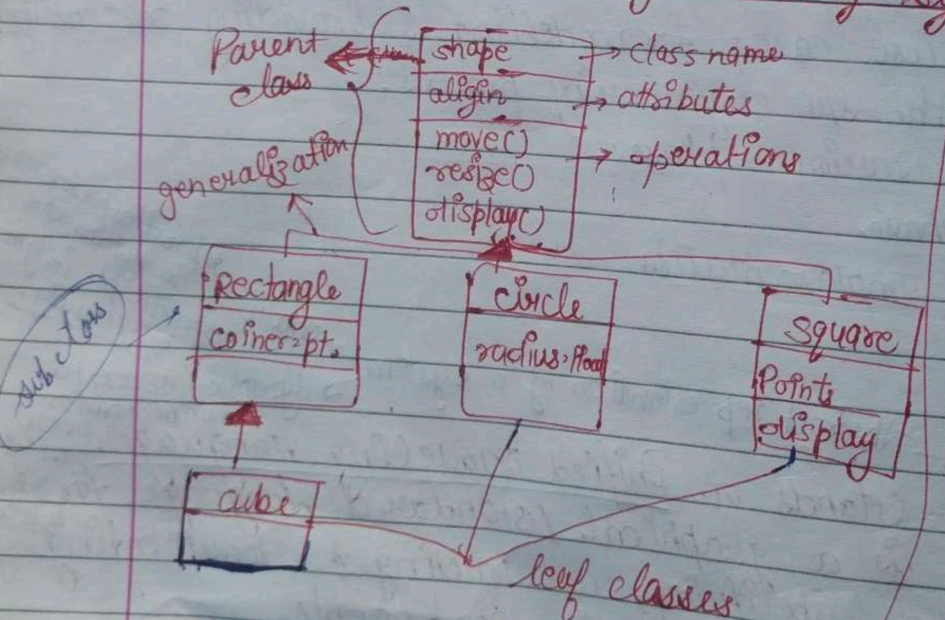


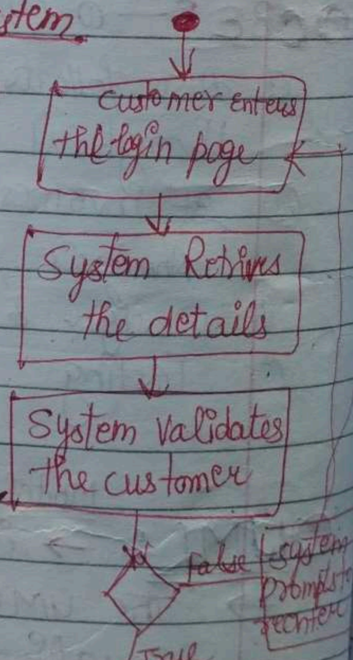
① class Diagram:- → class diag. shows a set of classes, interfaces i.e. collaborations & their relationships.

(group of classes)

→ Most commonly diag. used in OOS. → system

→ address the static design view of system.




customer enters the login page
System Retrives the details
System Validates the customer
False
system prompts to reenter
True
System Welcomes the Customer

Activity diagram

② Activity Diagram:- Illustrate the flow of control in a system & refer to the steps involved in the execution of a usercase.

It represent flow from one activity to the another activity.

→ We draw the activity diagram for each & every use case.

→ An activity is shown as a round box, containing the name of the operation.

→ An outgoing solid arrow attached to an activity symbol indicates a transition triggered by the completion of the activity.

③ Sequence Diagram:- represents interaction b/w objects in a sequential order. (focus on lifelines & follows sequential order)

④ Object Diagram:- encompasses objects & their relationship at a pt. in time. Instance of class diagram.

⑤ State Diagram:- represent the condition of system or part of the system at finite instance of time. (diff. states of components को show करता है)

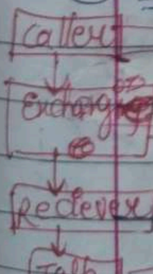
show dynamic behavior of system.

⑥ Component Diagram:- represents how components are wired together to form ~~larger~~ components or software systems. (complex software में use कर सकते हैं)

related

⑦ Deployment Diagram:- used to visualize topology of the physical components of a system. → It is a graph of nodes connected by comm.ⁿ association.
(how s/w components are deployed in the system)
arrange

⑧ Collaboration Diagram:- shows objects and relationships involved in an interaction, and the seq. of msgs. exchanged among the objects during the interaction.
→ seq. & collaboration Diagm. are same only diff. is seq. diag. focuses time & order of events, but colla Diag. focuses on msgs exchaged. & also known as Communication Diagram.



⑨ state chart Diagram:- It shows the seq. of states. A state is represented as a rounded box, which may contain one or more compartments.

By chychut

Name compartment holds the name of the State.

Idle
lift receiver &
get dial tone

* Patterns :- A pattern involves a general descripts on of a solⁿ to a recurring problem bundle with various goals and constraints.

A good patterns will do the following :-

→ It solves a problem.
→ It is a proven concept.
→ The solⁿ is not obvious.
→ It describes a relationship.
→ The pattern has a significant human component.

Types of Patterns :-

Generative
↳ are patterns that not only describe a recurring problem, they can tell us how to generate something & can be observed in resulting system architecture they help shape.

Non-Generative
↳ are static & Passive. They describe the recurring phenomena without saying how to reproduce them.

Patterns Template :- essential components for patterns template are:

(i) Name :- A meaning-ful name, good pattern name form a vocabulary for discusing conceptual abstration. A pattern may have more than one name in the literature.

(ii) Problem:— A statement of the problem that describe its items. Goals & objective it meants to reach within the given context & forces.

i) Context:— The pre-condition under which the problem & sol^n seen to recover & for which sol^n is desirable.

) Forces:— A description of the relevant forces & constraints & how they interact & conflict with one another & with the goals we wish to achieve with some indication of their priorities.

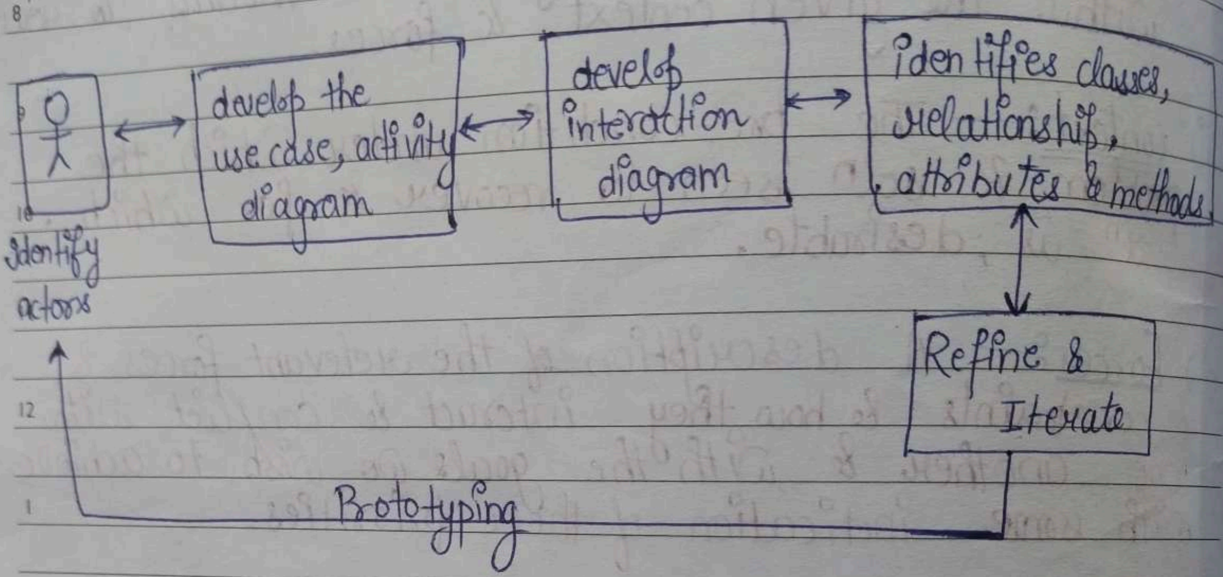
Solution:— sol^n should describe not only the static structure but also the dyamic behavior. static structure tells us the form & organization of pattern, but often the behavioreal dynamics is what makes the pattern "Come alive".

i) Example:— Eg. help the reader understand the pattern is use & applicatility. Visual eg. & analogien aftenly very useful.

Unified Approach:— (1) OOA unified Approach:—
→ The goal of OOA is to understand the problem domain & the system responsibilities by understanding how the user will use the system. So depending upon the users usage how the user will use the system the analysis has to be done & all the requirement are to be gathered & they are to be analyzed

in this obj.-oriented analysis.

step1 to step 6 → previous.

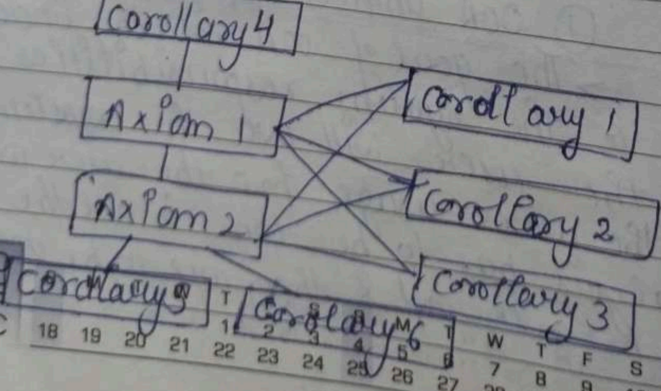

identify actors
develop the use case, activity diagram
develop interaction diagram
identifies classes, relationship, attributes & methods
Refine & Iterate
Prototyping

② OOD unified Approach:-

(i) Definition of OOD Axioms:- An axioms is a fundamental truth that always is observed to be valid & for which there is no counter eg. or exception.

(ii) Definition of OOD corollary:- A corollary is a proposition that follows from an axiom or from another proposition that has already been proven.

NOTES


corollary 4
Axiom 1
Corollary 1
Axiom 2
Corollary 2
Corollary 3
Corollary 5
Corollary 6