Subject: OOAD **Semester: 8TH** **Course No: CSE-802**

Lecture Notes

On

UML

UNIT-II

IMPORTANCE OF MODELING, BRIEF OVERVIEW OF OBJECT MODELING TECHNOLOGY (OMT) BY RAMBAUGH, BOOCH METHODOLOGY, USE CASE DRIVE APPROACH (OOSE) BY JACKOBSON.

Importance of modeling:

Model: A model is a simplification of reality. We build models so we can better understand the system we are developing.

There are many elements that contribute to a successful software organization; one common thread is the use of modeling. Modeling is a proven and well-accepted engineering technique.

Aims achieved throughout modeling:

-Model helps us to visualize a system as it is or as we want it to be.

-Models permit us to specify the structure or behavior of a system.

-Model gives us a template that guides us in constructing a system.

-Models document the decision we have made.

Object modeling technology (OMT) by Rambaugh:

-A method for analysis,design and implementation by an object oriented technique.

-Fast and intuitive approach for identifying and modeling all objects making up a System. .

-Class attributes, methods, inheritance and association can be expressed easily.

-Dynamic behavior of objects can be described using the OMT dynamic model.

-Detailed specification of state transitions and their descriptions within a system.

The object modeling techniques is an methodology of object oriented analysis, design and implementation that focuses on creating a model of objects from the real world and then to use this model to develop object–oriented software. object modeling technique, OMT was developed around 1991 by James Rambaugh. Now-a-days, OMT is one of the most popular object oriented development techniques. It is primarily used by system and software developers to support full life cycle development while targeting object oriented implementations.

OMT has proven itself easy to understand, to draw and to use. It is very successful in many application domains: telecommunication, transportation, compilers etc. The popular object modeling technique are used in many real world problems. The object-oriented paradigm using the OMT spans the entire development cycle, so there is no need to transform one type of model to another.

Four phases of OMT:

- **Analysis**: objects, dynamic and functional models
- **System Design**: Basic architecture of the system.
- **Object Design**: static, dynamic and functional models of objects.
- **Implementation**: reusable, extendible and robust code.

The OMT methodology covers the full software development life cycle. The methodology has the following phase.

1. **Analysis** - Analysis is the first phase of OMT methodology. The aim of analysis phase is to build a model of the real world situation to show its important properties and domain. This phase is concerned with preparation of precise and correct modelling of the real world. The analysis phase starts with defining a problem statement which includes a set of goals. This problem statement is then expanded into three models; an object model, a dynamic model and a functional model. The **object model** shows the static data structure or skeleton of the real world system and divides the whole application into objects. In others words, this model represents the artifacts of the system. The **dynamic model** represents the interaction between artifacts above designed represented as events, states and transitions. The **functional model** represents the methods of the system from the data flow perspective. The analysis phase generates object model diagrams, state diagrams, event flow diagrams and data flow diagrams.

2. **System design** - The system design phase comes after the analysis phase. System design phase determines the overall system architecture using subsystems, concurrent tasks and data storage. During system design, the high level structure of the system is designed. The decisions made during system design are:
    - The system is organized in to sub-systems which are then allocated to processes and tasks, taking into account concurrency and collaboration.
    - Persistent data storage is established along with a strategy to manage shared or global information.
    - Boundary situations are checked to help guide trade off priorities.
3. **Object design** - The object design phase comes after the system design phase is over. Here the implementation plan is developed. Object design is concerned with fully classifying the existing and remaining classes, associations, attributes and operations necessary for implementing a solution to the problem. In object design:
    - Operations and data structures are fully defined along with any internal objects needed for implementation.
    - Class level associations are determined.
    - Issues of inheritance, aggregation, association and default values are checked.

4. **Implementation** - Implementation pahse of the OMT is a matter of translating the design in to a programming language constructs. It is important to have good software engineering practice so that the design phase is smoothly translated in to the implementation phase. Thus while selecting programming language all constructs should be kept in mind for following noteworthy points.

   -To increase flexibility.

   -To make amendments easily.

   -For the design traceability.

   -To increase efficiency.

Three different parts of OMT modeling:

Object Model

1. Object model & data dictionary
2. It represents the static structure of the application.
3. It specifies to whom it happens to.
4. Operations in an object model corresponds to events in dynamic model and functions in functional model.
5. It is represented using class diagrams.
6. Identity, relationships to other objects, attributes and operations.

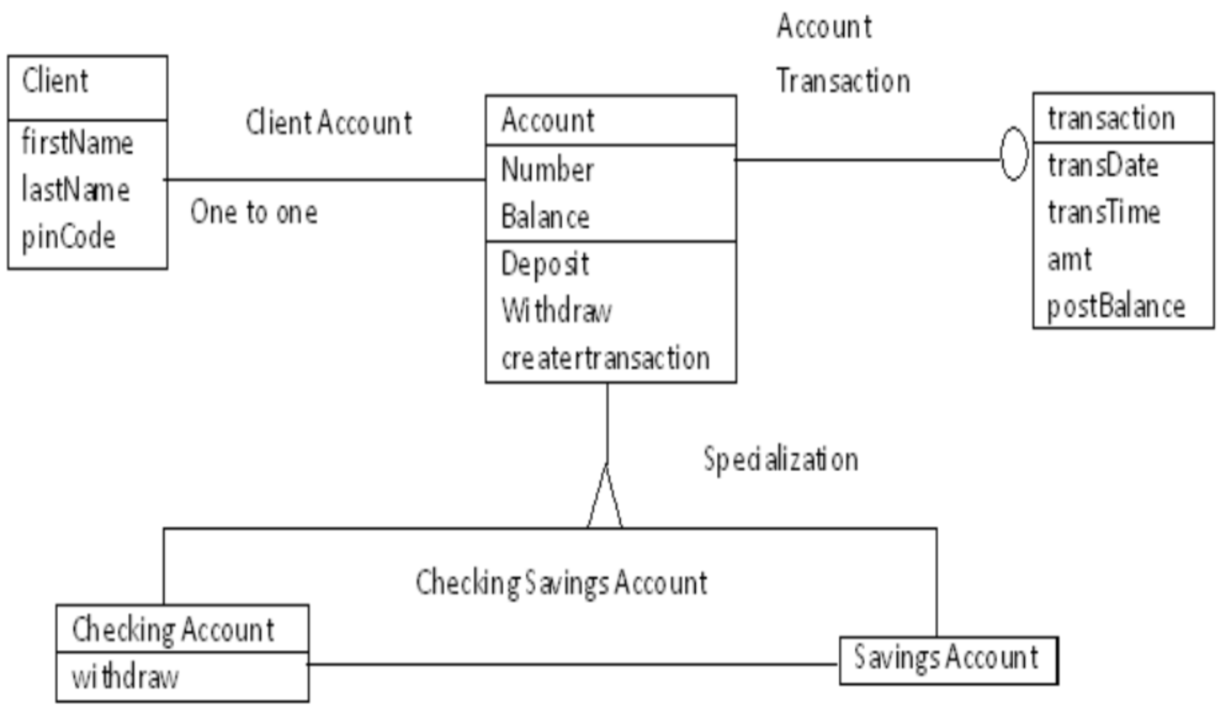
7. Object diagram:

   -Classes interconnected by association lines

   - Classes- a set of individual objects

   -Association lines- relationship among classes (i.e., objects of one class to objects of another class)

   Following diagram is an example for object diagram/ model

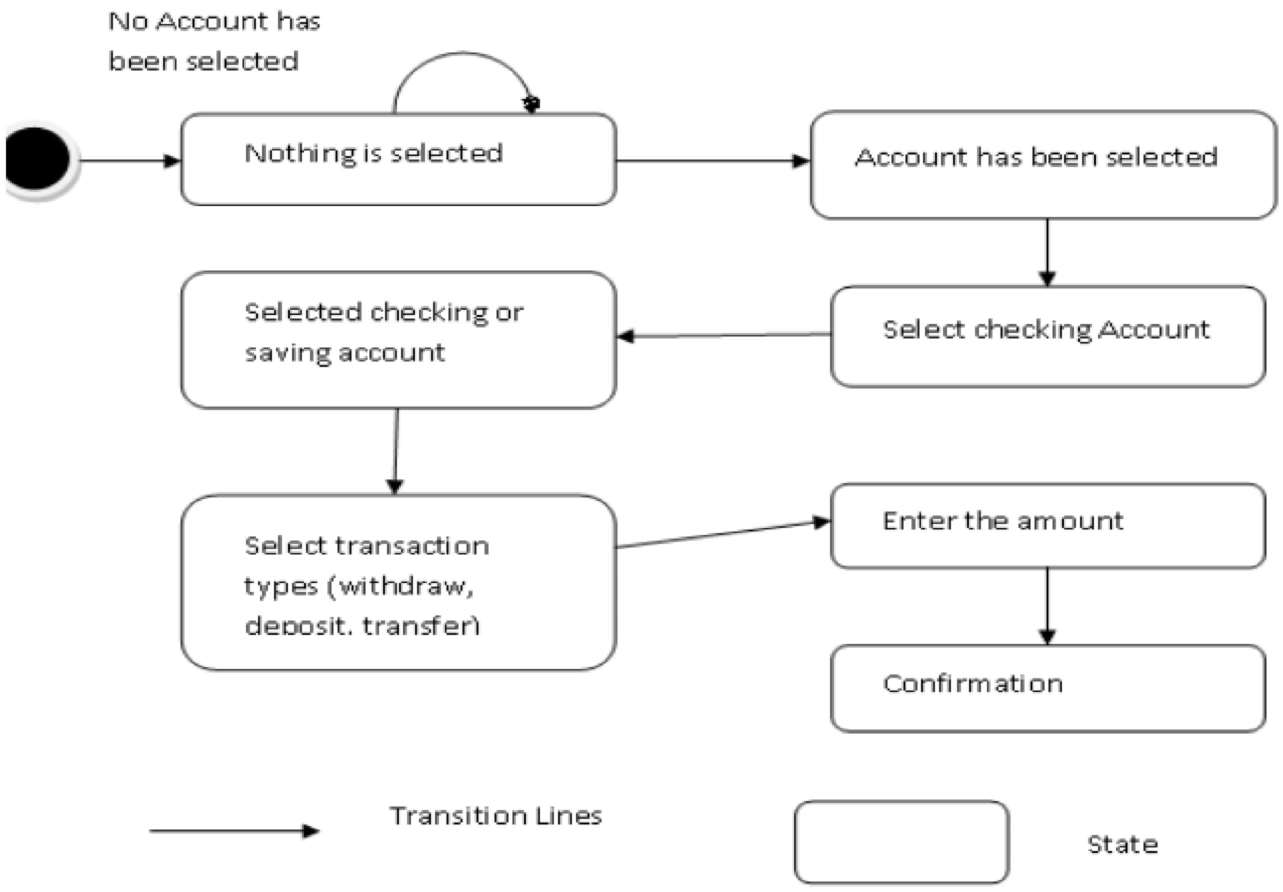


Dynamic Model

- State diagrams & event flow diagrams
- It represents the essential behavior of the application.
- It specifies when it happens.

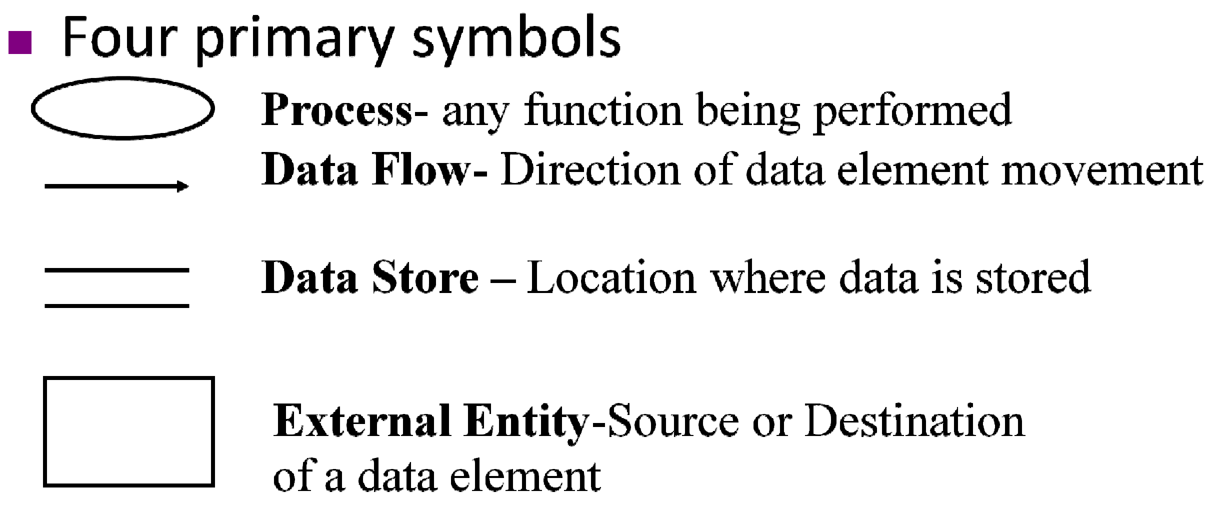
- Dynamic model describes the control structure of the objects. It defines decisions which are dependents of object values and which can cause action to change object values and invoke their functions.
- It is represented using state diagrams.
- States, transitions, events and actions
- OMT state transition diagram-network of states and events

Following diagram is an example for state diagram/ dynamic model


No Account has been selected
Nothing is selected
Account has been selected
Select checking Account
Selected checking or saving account
Select transaction types (withdraw, deposit, transfer)
Enter the amount
Confirmation
Transition Lines
State

Functional Model

- ✓ Data flow & constraints
- ✓ It represents what the application does and not how it does.
- ✓ It specifies what happens.
- ✓ It describes functions to be invoked by operations in object model and actions in dynamic models.
- ✓ It is represented using data flow diagrams.
- ✓ Shows flow of data between different processes in a business.
- ✓ Simple and intuitive method for describing business processes without focusing on the details of computer systems.
- ✓ DFD- (Data Flow Diagram)



- Four primary symbols

Process- any function being performed

Data Flow- Direction of data element movement

Data Store – Location where data is stored

External Entity-Source or Destination of a data element

Booch Methodology:

It is a widely used OO method that helps to design the system using object paradigm. Booch methodology was developed around 1986 by Grady Booch. This methodology Covers analysis and design phase of an OO System.

1. Lots of symbols and diagrams are in BOOCH methodology, but for designing only less symbols and diagram are used
2. It consists of six diagram for designing: 1.Class diagram 2.Object diagram 3.State Transition diagram 4.Module diagram 5.Process diagram 6.Interaction diagram

- **Class diagrams-**

  Describe roles and responsibilities of objects

- **Object diagrams**

  Describe the desired behavior of the system in terms of scenarios

- **State transition diagrams**

  State of a class based on a stimulus

- **Module diagrams**

  To map out where each class & object should be declared

- **Process diagrams**

  To determine to which processor to allocate a process

- **Interaction diagrams**

  Describes behavior of the system in terms of scenarios

BOOCH Methodology prescribes 2 different process:

1. Macro Development Process
2. Micro Development Process

1. Macro Development Process Serve as a controlling framework for the microprocessor and can take weeks or even months.

   - Primary concern is technical management of the system.
   - Consists of 5 different steps
     1. Conceptualization
     2. Analysis and development of the model
     3. Design or create the system architecture
     4. Evolution or implementation
     5. Maintenance

       ✓ Conceptualization:- Providing the basic needs for the S/W. Establish Core requirement of the system. Establish a set of goals and develop a prototype to prove the concept .

       ✓ Analysis and development of the model:- Developing behavior tendency model for the system.Use class diagram to describe the role and responsibility an object have to carry, interaction/ behavior diagram.

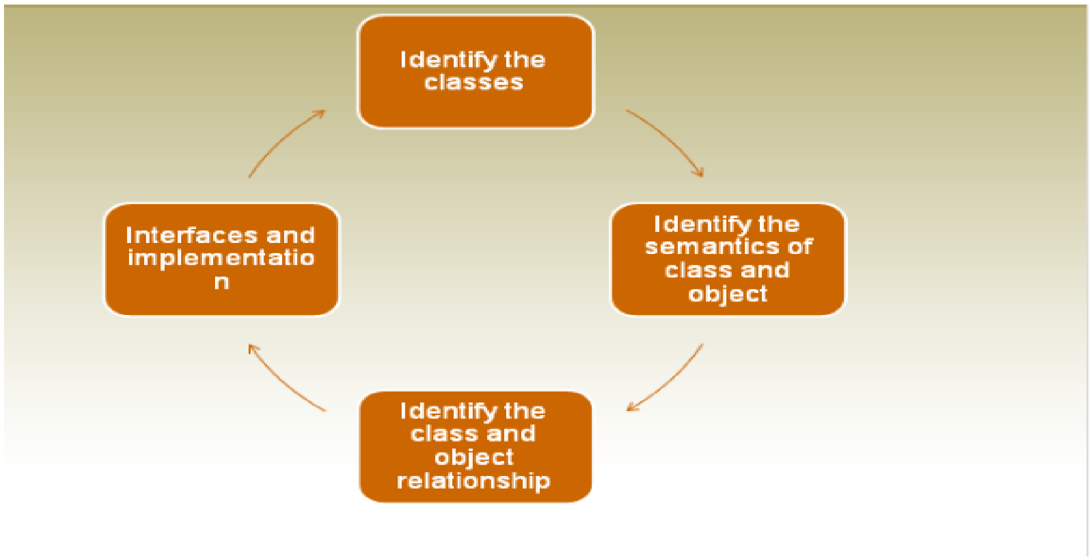
       ✓ Design or create the system architecture:-Forming the architect for development. In design phase use class diagram, object diagram, module diagram and process diagram.Class diagram decides what classes exists and how are they relate to each other. Object diagram decides about the different object collaboration.Module diagram describes where all classes and object to be declared. Process diagram determines which process to allocate .

- ✓ Evolution or implementation:- Refine the system by many iteration and produce a stream of software implementation.
- ✓ Maintenance:- Add new requirement and remove bugs.

2. Micro Development Process :

Each macro development process have its own micro development process. It consists of following steps

1. Identify classes and objects
2. Identify classes and objects semantics
3. Identify classes and objects relationships
4. Identify classes and objects interface and implementation


Identify the classes
Identify the semantics of class and object
Identify the class and object relationship
Interfaces and implementatio n

Micro improvement process is defined by scenario and structural products are successively refined by macro processes. Micro processes explain daily activities of an individual and small development team.

Use case drive approach (OOSE) by Jackobson:

- Jackobson methodology was developed around 1994 by Ivar Jacabson.
- It covers the entire life cycle and stress traceability between the different phases, both forward and backward, eg., OOSE, OOBE, Objectory(object factory for s/w development)
- Enables reuse of analysis and design, reduce development time than reuse of code
- At the heart of their methodologies is the usecase concept, which evolved objectory.
  1. Use cases:
     - ✓ Are scenarios for understanding system requirements
     - ✓ Is an interaction between users & the system
     - ✓ Capture the goal of the user & the responsibility of the system to its users.
     - ✓ In the requirements analysis, the use cases are described as one of the following
       - Nonformal text with no clear flow of events
       - Text, easy to read but with a clear flow of event to follow
       - Formal style using pseudo code
     - ✓ Use case description must contain,
       - How & when the use case begins and ends
       - Interaction between the use-case & its actors, including when the interaction occurs & what is exchanged
       - How & when the use-case will need data stored in the system or will store data in the system

- Exceptions to the flow of events
- How & when the concepts of problem domain are handled

- ✓ Every single use-case should have one main flow of events.
- ✓ Use-case would be viewed as concrete or abstract.(is not complete and has no actors that initiate it but is used by another user case

2:Object Oriented Software Engineering:objectory (OOSE)

- ✓ Also called objectory is a method of OO development with the specific aim to fit the development of large, real-time systems.
- ✓ Development process, called use-case driver development, stresses that use-cases are involved in several phases of the development, including analysis, design, and validation & testing.
- ✓ OOSE- is a disciplinary process for the industrialized development of s/w, based on a use-case driven design.
- ✓ Jacobson objectory has been developed & applied to numerous application areas & embedded in the CASE tools system.
- ✓ Objectory is built around several different models.
  - Use case model: defines outside (actor) & inside (use case) of the systems behaviour.
  - Domain object model: object of real world are mapped into the domain object model.
  - Analysis object model: it presents how the source code should be carried out & written.
  - Implementation model: rep implementation of the system.
  - Test model: constitutes the test plans, specification & reports.

- ✓ Maintenance of each model is specified in its associated process.

3. Object oriented business Engineering. (OOBE):

- ✓ OOBE is object modeling at the enterprise level.

  - Analysis phase
    - ▪ Object Model
    - ▪ Requirement
    - ▪ Analysis
  - Design & Implementation phase
    - ▪ DBMS
    - ▪ Distribution of Process
  - Testing phase
    - ▪ Unit Testing
    - ▪ Integration testing
    - ▪ System Testing

Reference Books:

1. Rambaugh: Object Oriented modeling and design PHI
2. Booch, Object Oriented analysis and design with applications, Addison Wesley

Online References:

1. https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_object_oriented_analysis.htm
2. http://www.srmuniv.ac.in/sites/default/files/files/UNIT%20II.pdf