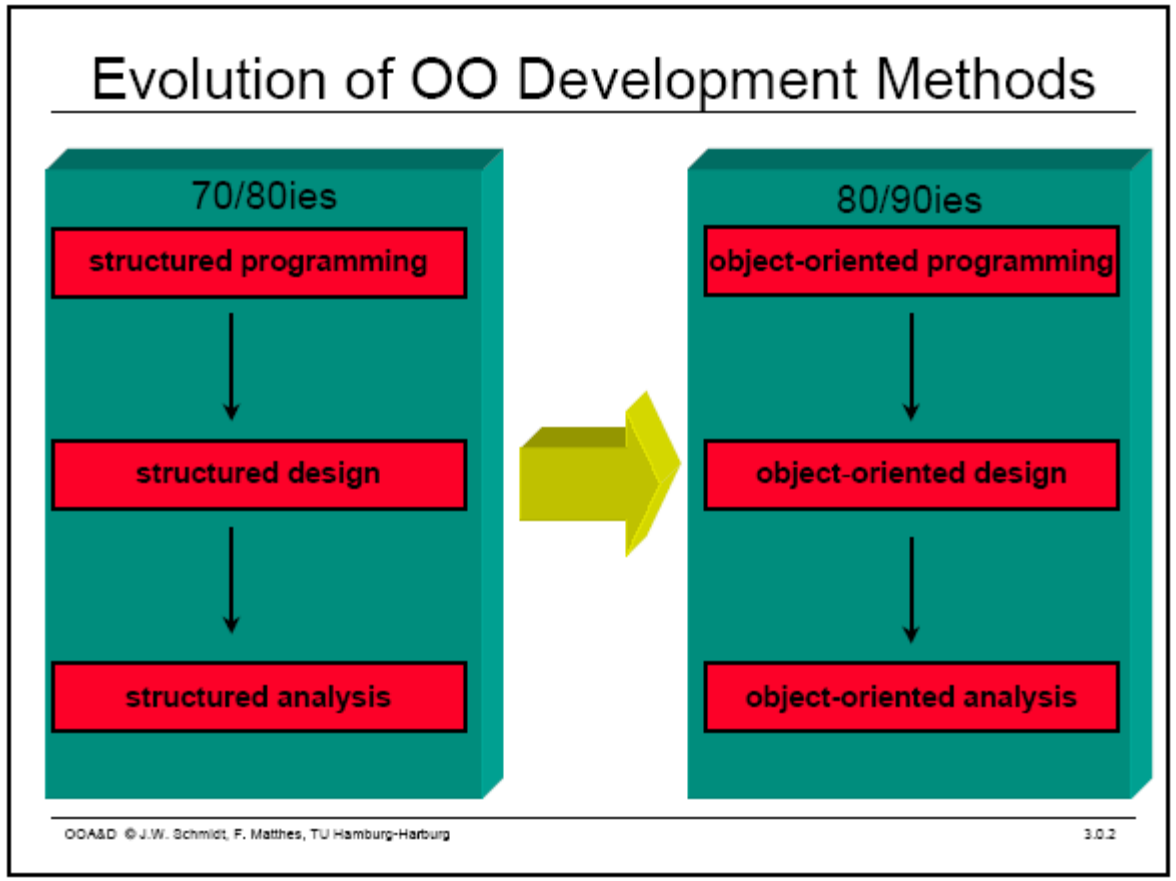
Introduction to UML

Prof. Maitreyee Ganguly

Object Oriented Modeling



What is UML?

- UML stands for “Unified Modeling Language”
- It is a industry-standard graphical language .
- UML is a pictorial language used to make software blue prints
- It is used for specifying, visualizing, constructing, and documenting the artifacts of software systems
- UML is different from the other common programming languages
- It uses mostly graphical notations.
- Simplifies the complex process of software design

Why UML for Modeling

- Use graphical notation to communicate more clearly than natural language (imprecise) and code(too detailed).
- Help acquire an overall view of a system.
- Tools can be used to generate code in various languages using UML diagrams
- UML is *not* dependent on any one language or technology.
- *A picture is worth than thousand words*
- UML can be defined as a simple modeling mechanism to model all possible practical systems in today's complex environment.

Conceptual Model of UML

- It is a model which is made of concepts and their relationships.
- It is the first step before drawing a UML diagram.
- It helps to understand the entities in the real world and how they interact with each other
- It can be mastered by learning the following three major elements: UML building blocks

  Rules to connect the building blocks

  Common mechanisms of UML

Object Oriented Concepts

- Object contains both data and methods that control the data.
- The data represent the state of the object
- Data can also describe the relationships between this object and other objects
- objects are the real world entities

Object Oriented Concepts

- Every object belongs to (is an instance of) a class
- An object may have fields, or variables
- The class describes those fields
- An object may have methods
- The class describes those methods
- A class is like a template, or cookie cutter
- You use the class’s constructor to make objects

Example: A "Rabbit" object

- You could (in a game, for example) create an object representing a rabbit
- **It would have data:**
  - How hungry it is
  - How frightened it is
  - Where it is
- **And methods:**
  - eat, hide, run, dig

Building blocks of UML

- The building blocks of UML can be defined as:
- Things
- Relationships
- Diagrams

Things

- **Things** are the most important building blocks of UML. Things can be:
- Structural
- Behavioral
- Grouping
- Annotational

Structural things

- The **Structural things** define the static part of the model.
- They represent physical and conceptual elements.

Class:

- Class represents set of objects having similar responsibilities.

Interface:

- Interface defines a set of operations which specify the responsibility of a class

Collaboration:

- Collaboration defines interaction between elements.

Structural things

Use case:

- Use case represents a set of actions performed by a system for a specific goal.

Component:

- Component describes physical part of a system.

Node:

- A node can be defined as a physical element that exists at run time.

Behavioral things

- It consists of the dynamic parts of UML models.

Interaction:

- It is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.

State machine:

- It is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events.

Grouping things

- **Grouping things** can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available:

Package:

- Package is the only one grouping thing available for gathering structural and behavioral things.

Annotational things

- **Annotational things** can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements.

Note

- It is the only one Annotational thing available.
- A note is used to render comments, constraints etc of an UML element.

Phases of System Development

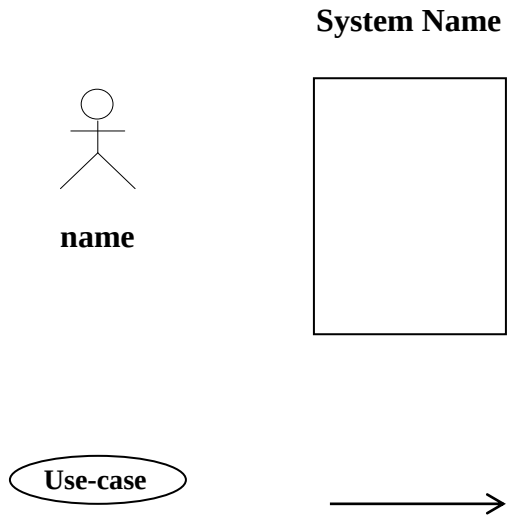
- Requirement Analysis
  - The functionality users require from the system
  - **Use-case model**
- OO Analysis
  - Discovering classes and relationships
  - Class diagram
- OO Design
  - Result of Analysis expanded into technical solution
  - Sequence diagram, state diagram, etc.
  - Results in detailed specs for the coding phase
- Implementation (Programming/coding)
  - Models are converted into code
- Testing
  - Unit tests, integration tests, system tests and acceptance tests.

Use-Case Modeling

- In use-case modeling, the system is looked upon as a black box whose boundaries are defined by its functionality to external stimuli.
- The actual description of the use-case is usually given in plain text. A popular notation promoted by UML is the stick figure notation.
- We will look into the details of text representation later. Both visual and text representation are needed for a complete view.
- A use-case model represents the use-case view of the system. A use-case view of a system may consist of many Use-case diagrams.
- An use-case diagram shows (the system), the actors, the use-cases and the relationship among them.

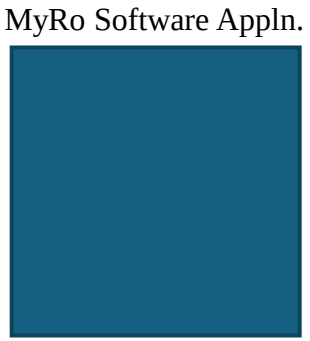
Components of Use-case Model

- The components of a Use-case model are:
  - System Modeled
  - Actors
  - Use-cases
  - Stimulus



System

- As a part of the use-case modeling, the boundaries of the system are developed.
- System in the use-case diagram is a box with the name appearing on the top.
- Define the scope of the system that you are going to design with your MyRo. (software scope).



Actors

- An actor is something or someone that interacts with the system.
- Actor communicates with the system by sending and receiving messages.
- An actor provides the stimulus to activate an Use-case.
- Message sent by an actor may result in more messages to actors and to Use-cases.
- Actors can be ranked: primary and secondary; passive and active.
- Actor is a role not an individual instance.

Prof. Maitreyee Ganguly

Finding Actors

- The actors of a system can be identified by answering a number of questions:
  - Who will use the functionality of the system?
  - Who will maintain the system?
  - What devices does the system need to handle?
  - What other system does this system need to interact?
  - Who or what has interest in the results of this system?

Use-cases

- A Use-case in UML is defined as a set of sequences of actions a system performs that yield an observable result of value to a particular actor.
- Actions can involve communicating with number of actors as well as performing calculations and work inside the system.
- A Use-case
  - is always initiated by an actor.
  - provides a value to an actor.
  - must always be connected to at least one actor.
  - must be a complete description.

Finding Use-cases

- For each actor ask these questions:
  - Which functions does the actor require from the system?
  - What does the actor need to do?
  - Could the actor’s work be simplified or made efficient by new functions in the system?
  - What events are needed in the system?
  - What are the problems with the existing systems?
  - What are the inputs and outputs of the system?

Prof. Maitreyee Ganguly

Describing Use-cases

- Use-case Name:
- Use-case Number: system#.diagram#.Use-case#
- Authors:
- Event(Stimulus):
- Actors:
- Overview: brief statement
- Related Use-cases:
- Typical Process description: Algorithm
- Exceptions and how to handle exceptions:

Example

- Number: A.132.4
- Name: Buy book online
- Author: B.Ramamurthy
- Event: Customer request one or more books
- System: Amazon.com
- Overview: Captures the process of purchasing one or more books and the transactions associated with it.
- Related Use-case: A.132.5, A.132.8
- Typical Process Description with exceptions handled.

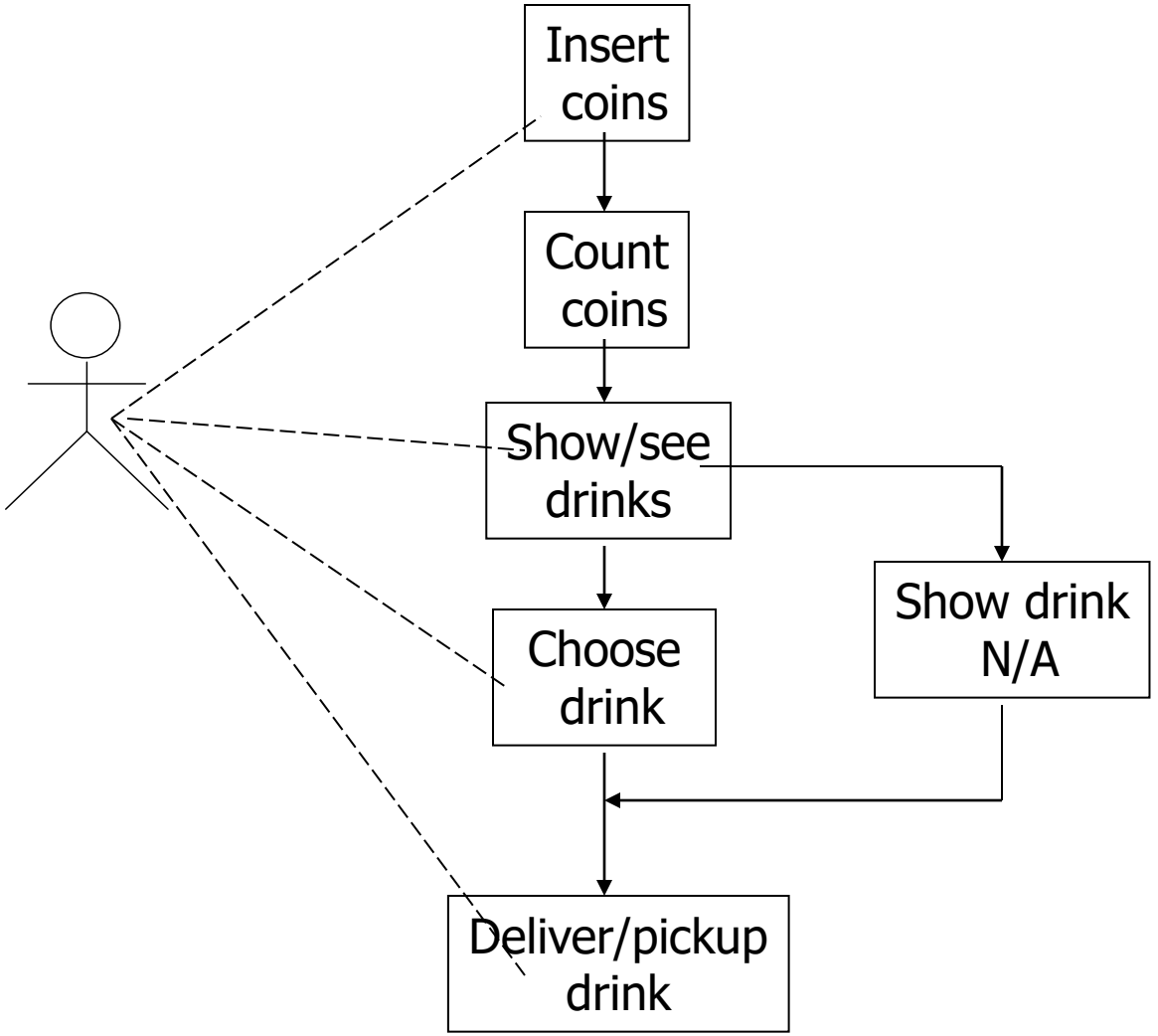
NOTE : All these can be in a tabular form, say, in an Excel worksheet for example.

Realizing Use-cases

- Validation is done up front. As soon as the model is ready it has to be presented and discussed with the customers.
- Use-cases are implementation independent descriptions of the functionality of the system.
- Use-case can be realized in the next stages of software development using, say, a class diagram.

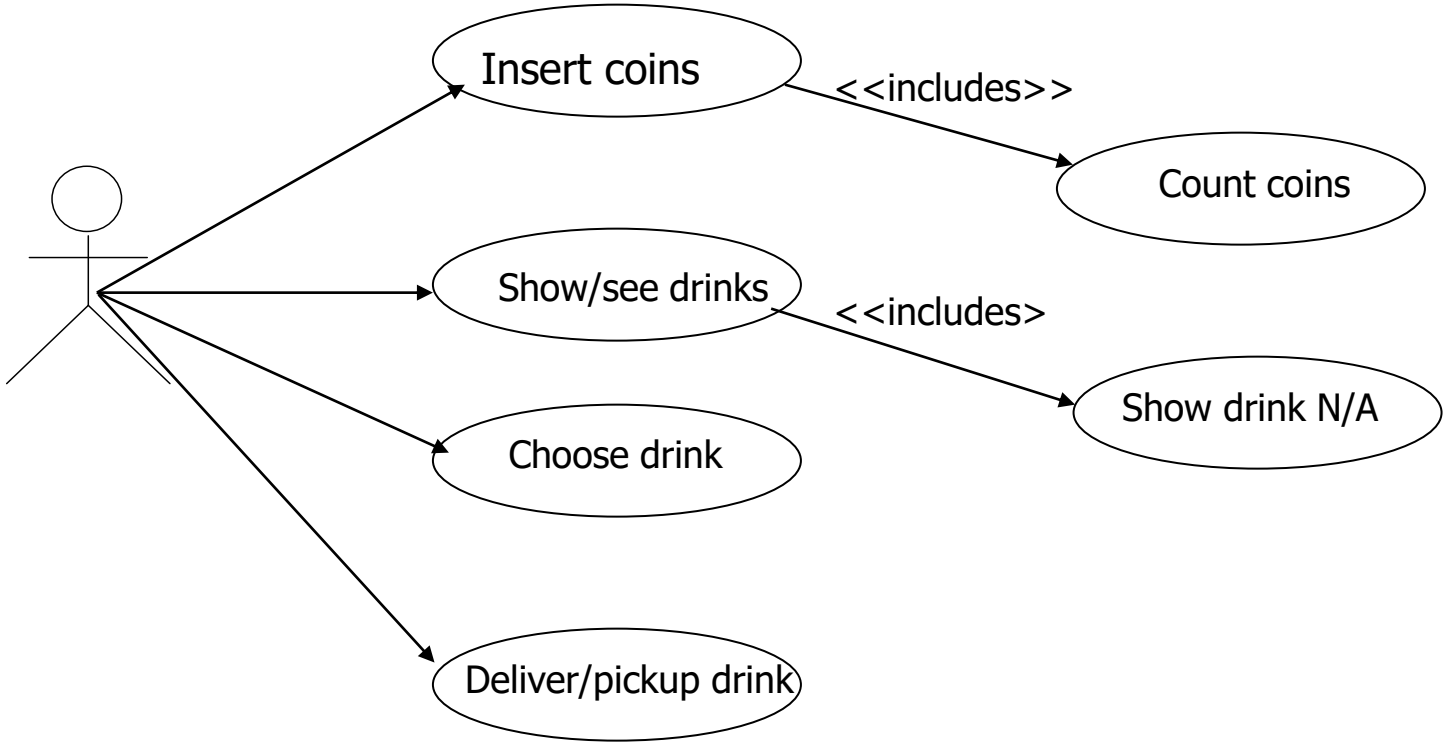
Prof. Maitreyee Ganguly

Interaction between user and Use-case



Prof. Maitreyee Ganguly

Use Case Diagram



Case Studies

- Ticket counter for basketball game
- TIVO: Video recorder/controller
- Weather Station.
- ATM Machine: Description given as data dictionary.
- 4-cycle lawnmower engine
- Burger queen fast food restaurant’s hand-held order device

Prof. Maitreyee Ganguly

Use Case Example: ATM System