Object Oriented Analysis and Design using Java

Prof. Vinay Joshi

Department of Computer Science and Engineering

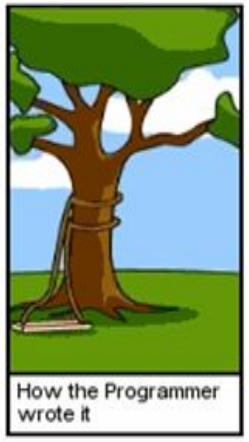
Requirement Engineering

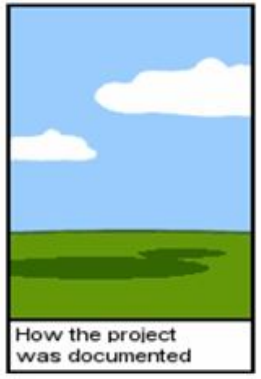
RE is usually the first step in any software intensive development lifecycle irrespective of model

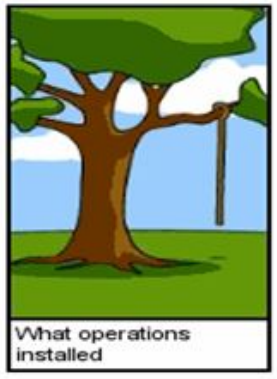
- Usually difficult, error prone and costly
- Critical for successful development of all down stream activities
- Errors introduced during requirements phase if not handled properly will propagate into the subsequent phases
- Unnecessary, Late or invalid requirements can make the system cumbersome or slip
- Requirement errors are expensive to fix at a later stage

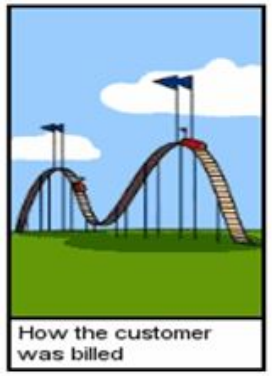
Object Oriented Analysis and Design using Java

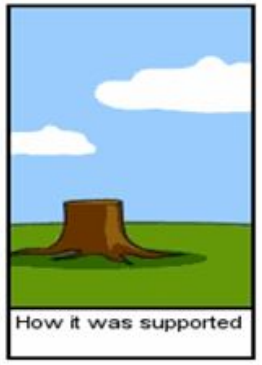
Requirement Engineering

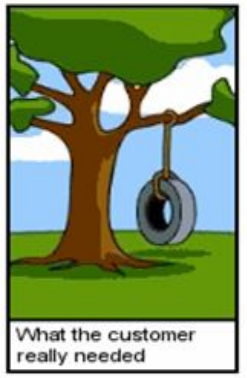












Requirement Engineering

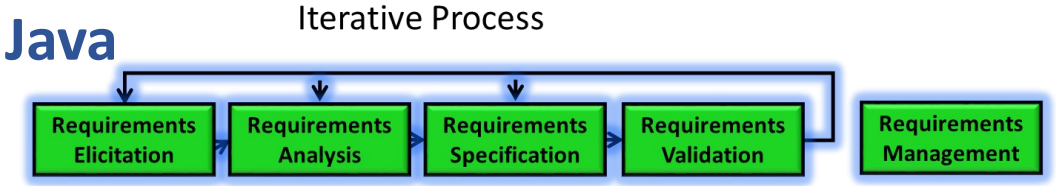
- Requirement is the Property which must be exhibited by software developed/adapted to solve a particular problem
- Requirement should specify the externally visible behavior of **what** and **not how**
- Requirements can be looked at as
  - Individual requirements
  - Set of requirements

Requirement Engineering

Is a process of working proactively with all stakeholders gathering their needs, articulating their problem, identify and negotiate potential conflicts thereby **establishing a clear scope and boundary for a project.**

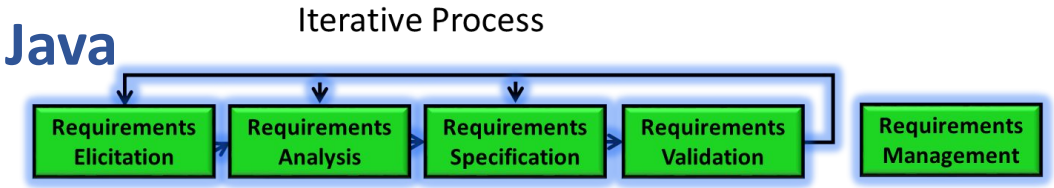
It can also be described as a process of ensuring that the stakeholders have been identified and they have been given an **opportunity to explain their problem and needs** and describe what they would like the new system to do.

Requirement Analysis Process



1. Understand the requirements in depth, both from a product and process perspective
2. Classify and Organize the requirements into coherent clusters
   - Functional, Non-Functional & Domain requirements
   - System and User Requirements
3. Model the requirements
4. Analyze the requirements (if necessary) using fish bone diagram

Requirement Analysis Process



5. Recognize and resolve conflicts (e.g., functionality v. cost v. timeliness)
6. Negotiate Requirements
7. Prioritize the requirements (MoSCoW -Must have, Should have, Could have, Wont have)
8. Identify risks if any
9. Decide on Build or Buy (Commercial Of The Shelf Solution) and refine requirements

Requirement Modelling

A Model is a representation of a system in some form.
A is a Model of B if A can be used to answer questions about B
Part of Requirement Analysis and Specification phases.

Couple of important goals of Modelling

- Providing an Understanding (existing) System
  - Analyzing and Validating the requirements in terms of visible requirements within the problem
- Communicating the requirements in terms of who, what and interpreting it in the same way

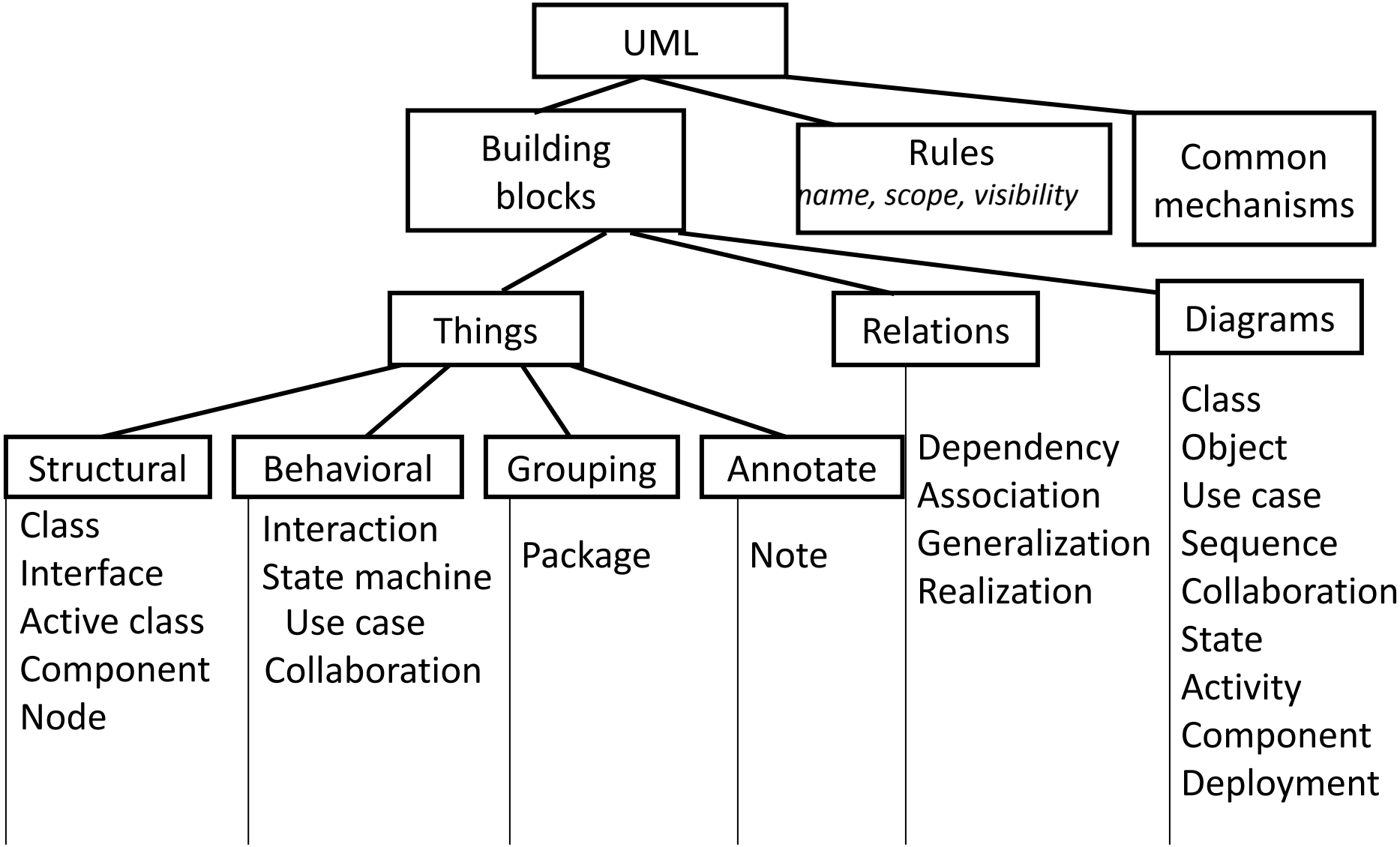
- **Discussed different kinds of Models**
  - Structural Models and Behavioral models

What is UML

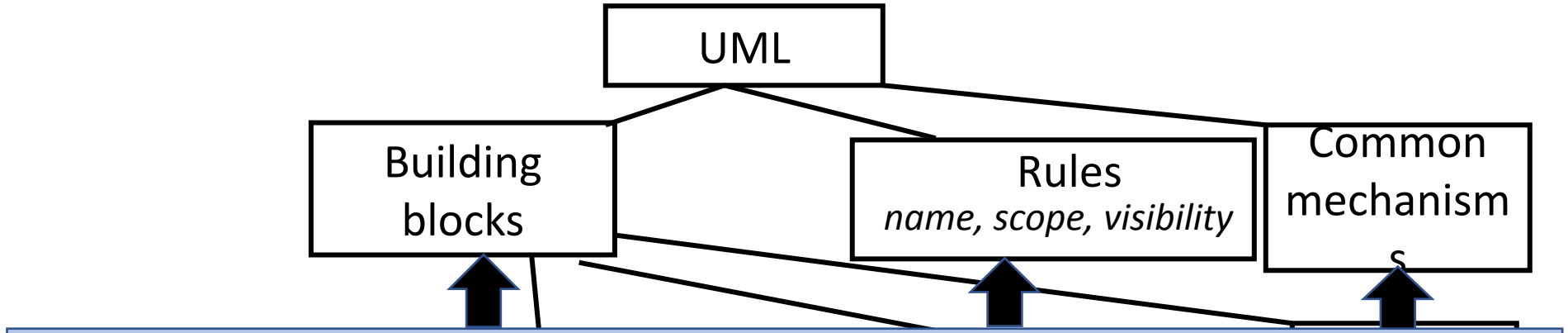
- **Unified Modeling Language**
  - Language to express different types of models
  - Language defines
  - Syntax – Symbols and rules for using them
  - Semantics – What these symbols represent
- **UML can be used to**
  - Visualize (Graphical Notation)
  - Specify (Complete and Unambiguous)
  - Construct (Code Generation)
  - Document (Design, Architecture, etc.)

  the artifacts of software-intensive systems

Conceptual Models of UML



Conceptual Models of UML

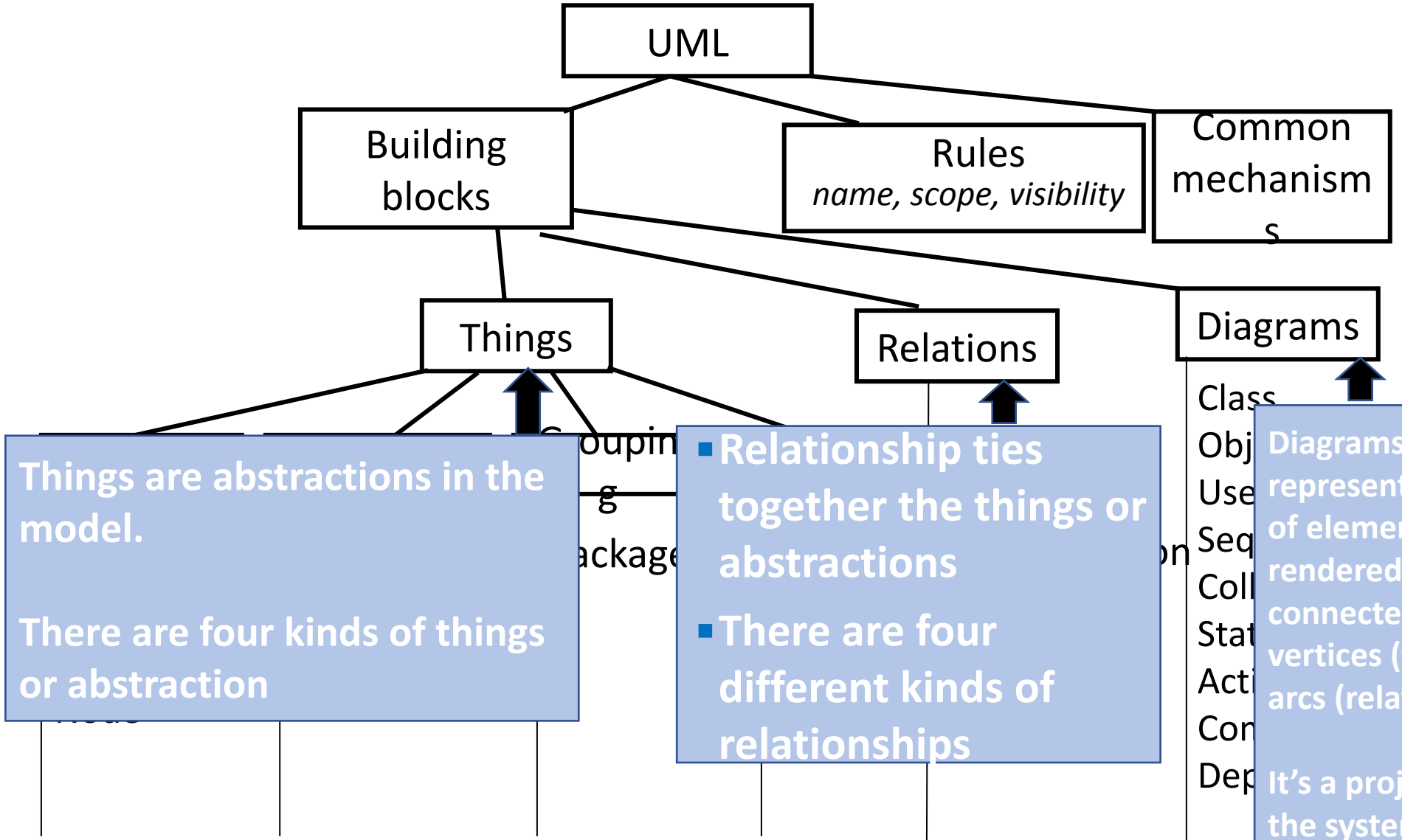


UML is made up of three conceptual elements. They are

- **Building blocks which make up the UML**
- **Rules that dictate how these building blocks can be put together**
- **Common mechanisms that apply through out UML**

Interface Active class Component Node	State machine Use case Collaboration	Package	Note	Generalization Realization	Collaboration State Activity Component Deployment

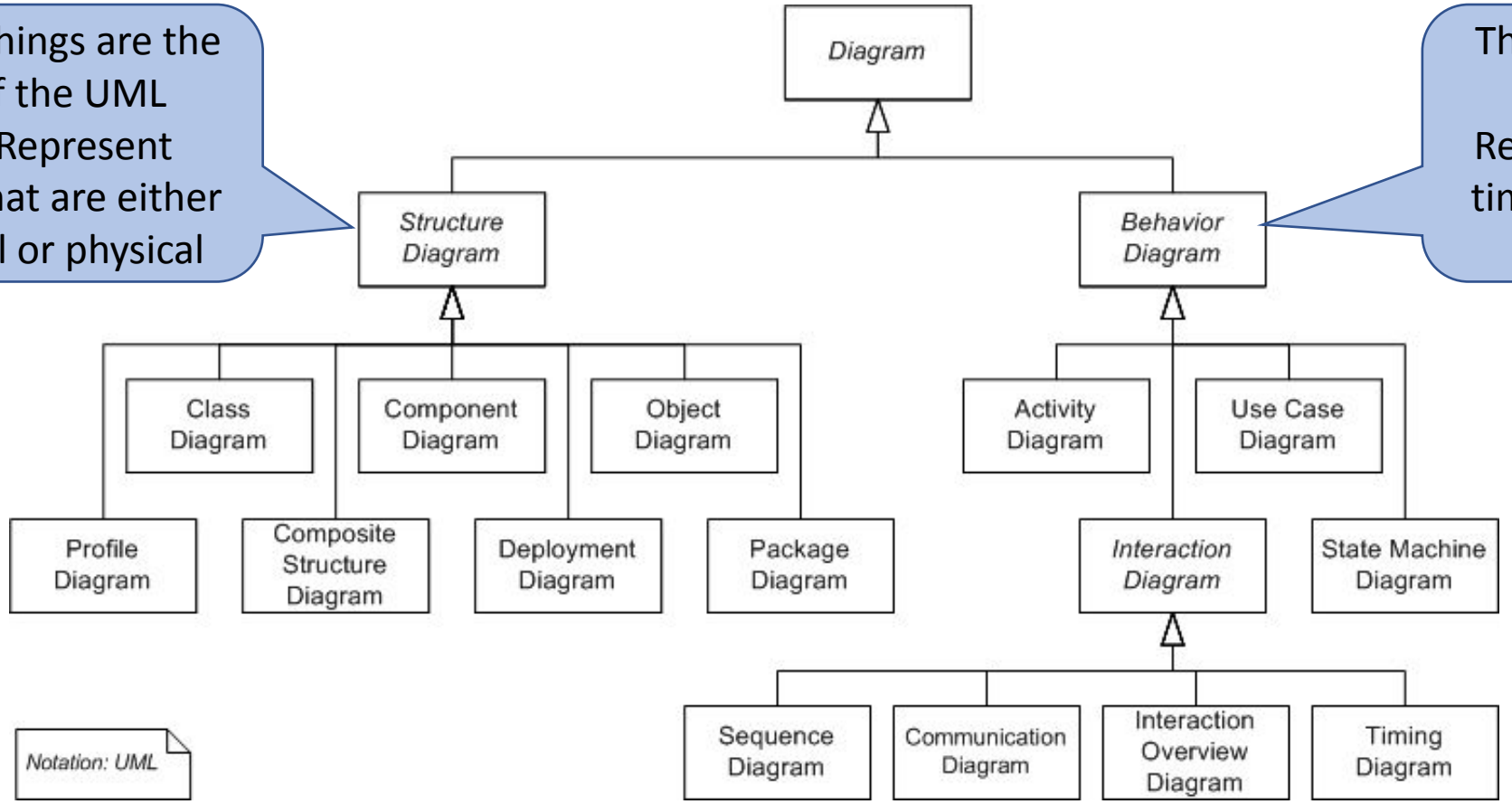
Conceptual Models of UML



Object Oriented Analysis and Design using Java

UML Diagrams

Structural things are the nouns of the UML models. Represent elements that are either conceptual or physical



These are dynamic parts of the UML models. Represent behavior over time and space. Typically verbs of the model

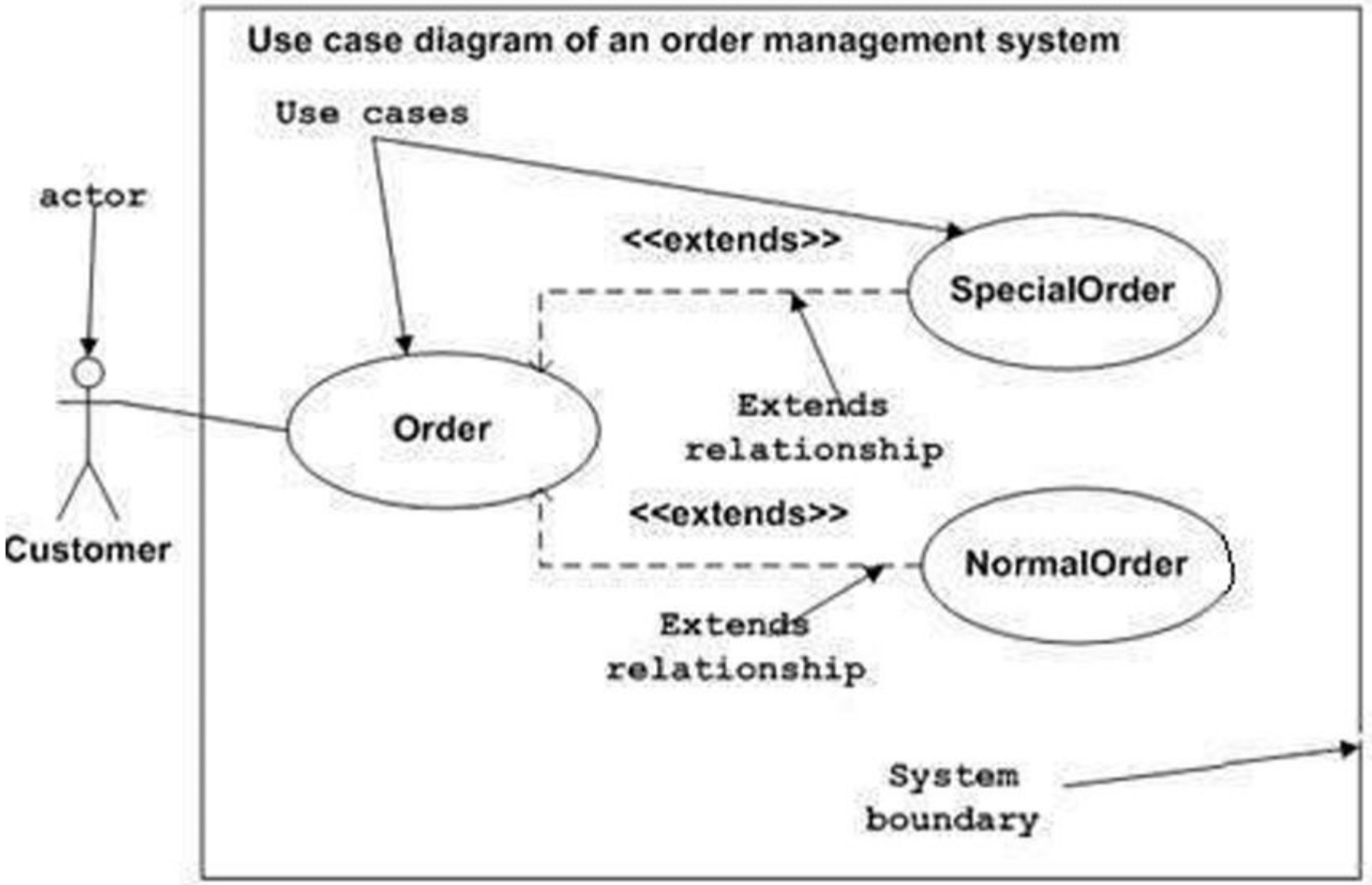
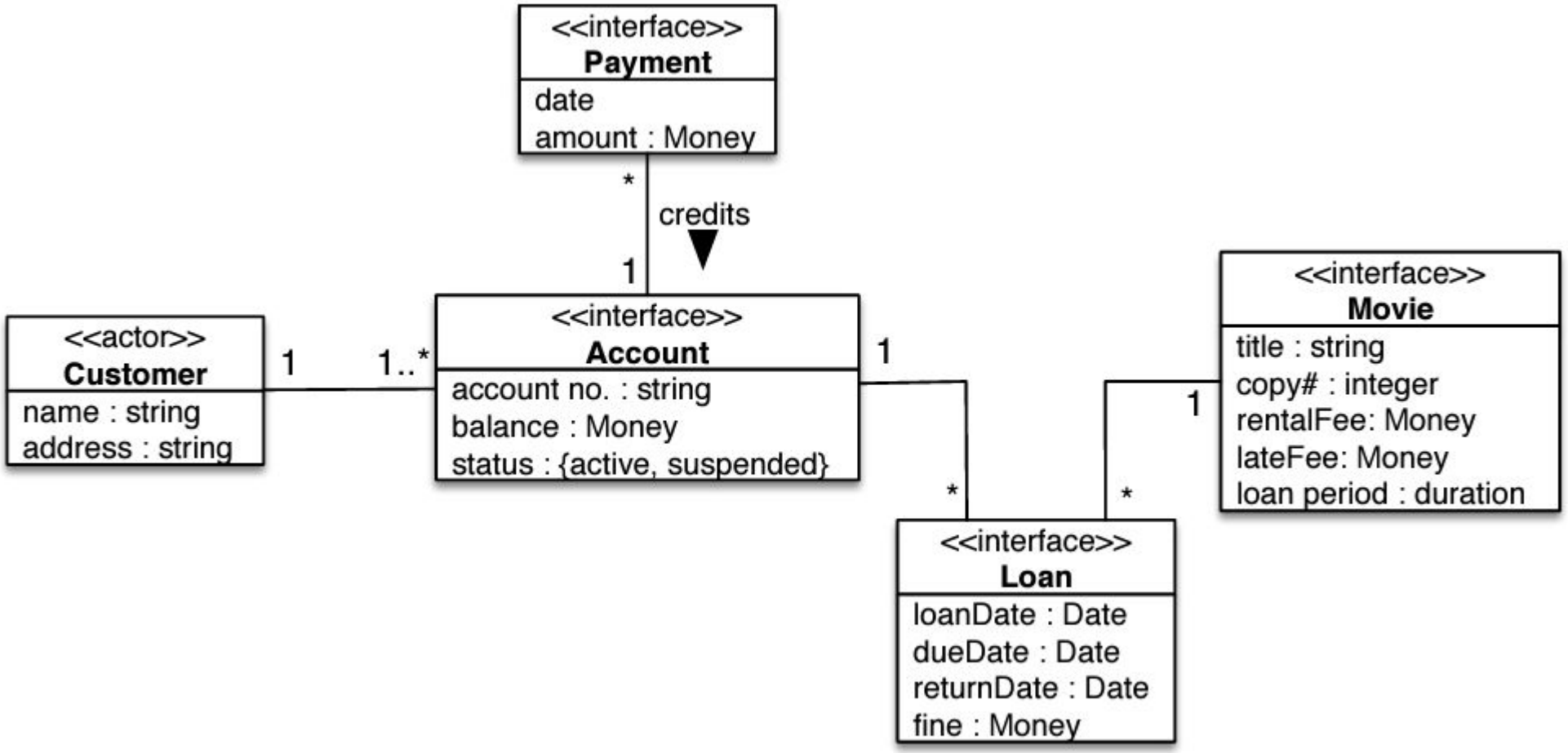


Figure: Sample Use Case diagram

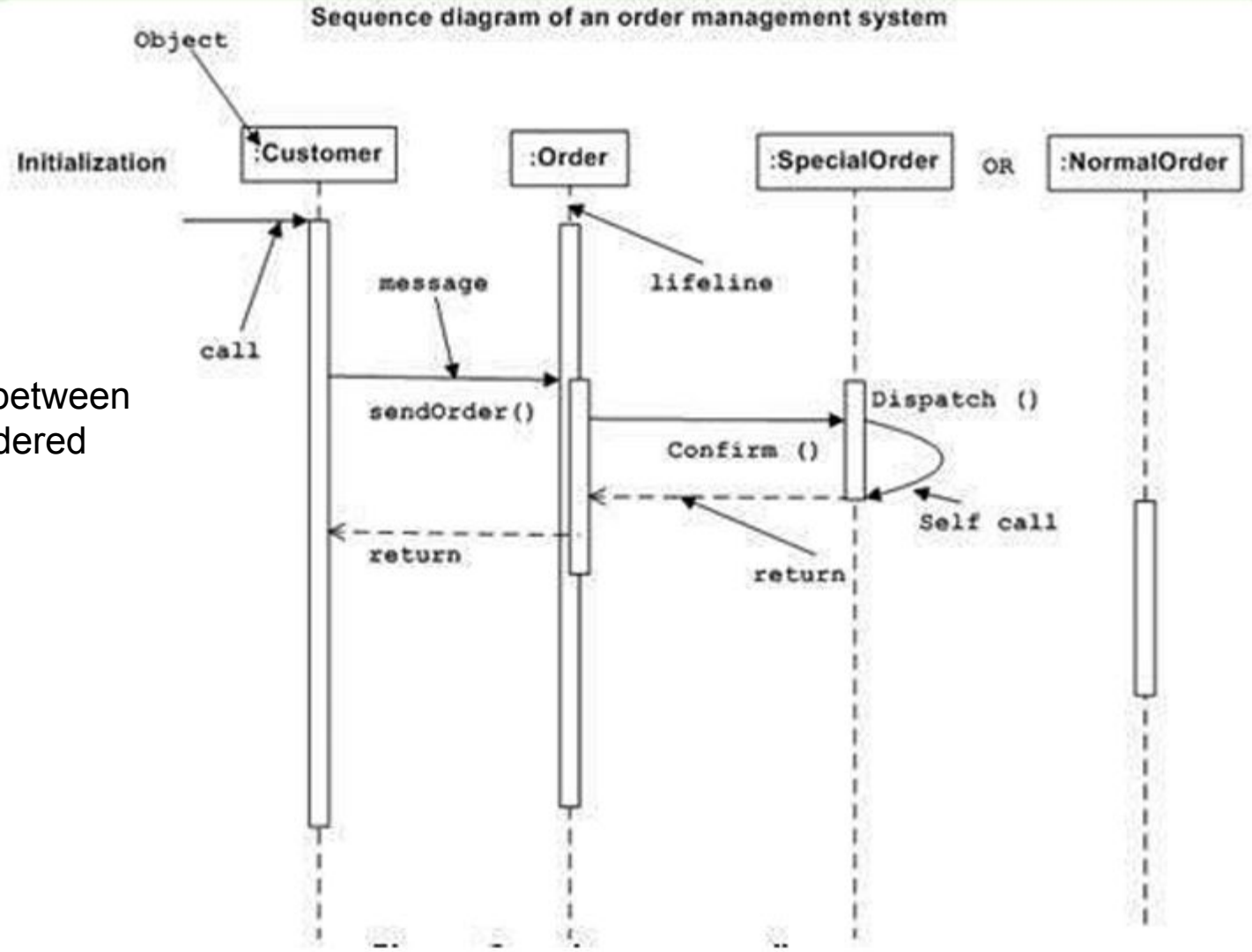
Class Diagram



Interaction diagram

- This interactive behaviour is represented in UML by two diagrams known as
  - Sequence diagram
  - Collaboration diagram
- The basic purpose of both the diagrams are similar.
- Sequence diagram emphasizes on time sequence of messages and
  - collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.
- Following things are to be identified clearly before drawing the interaction diagram
  - Objects taking part in the interaction.
  - Message flows among the objects.
  - The sequence in which the messages are flowing.
  - Object organization.

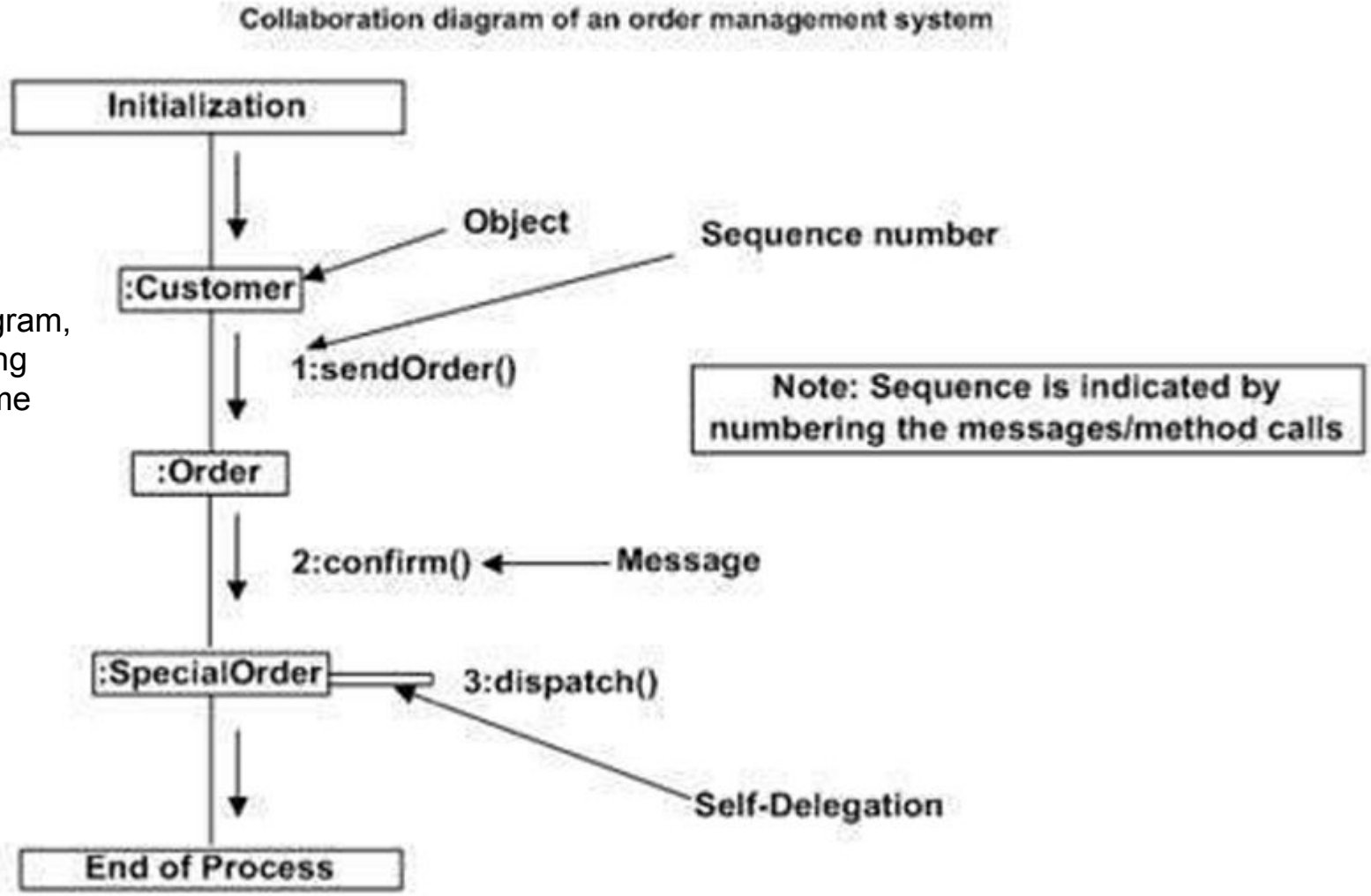
Sequence diagram



Shows interactions between objects as a time-ordered view

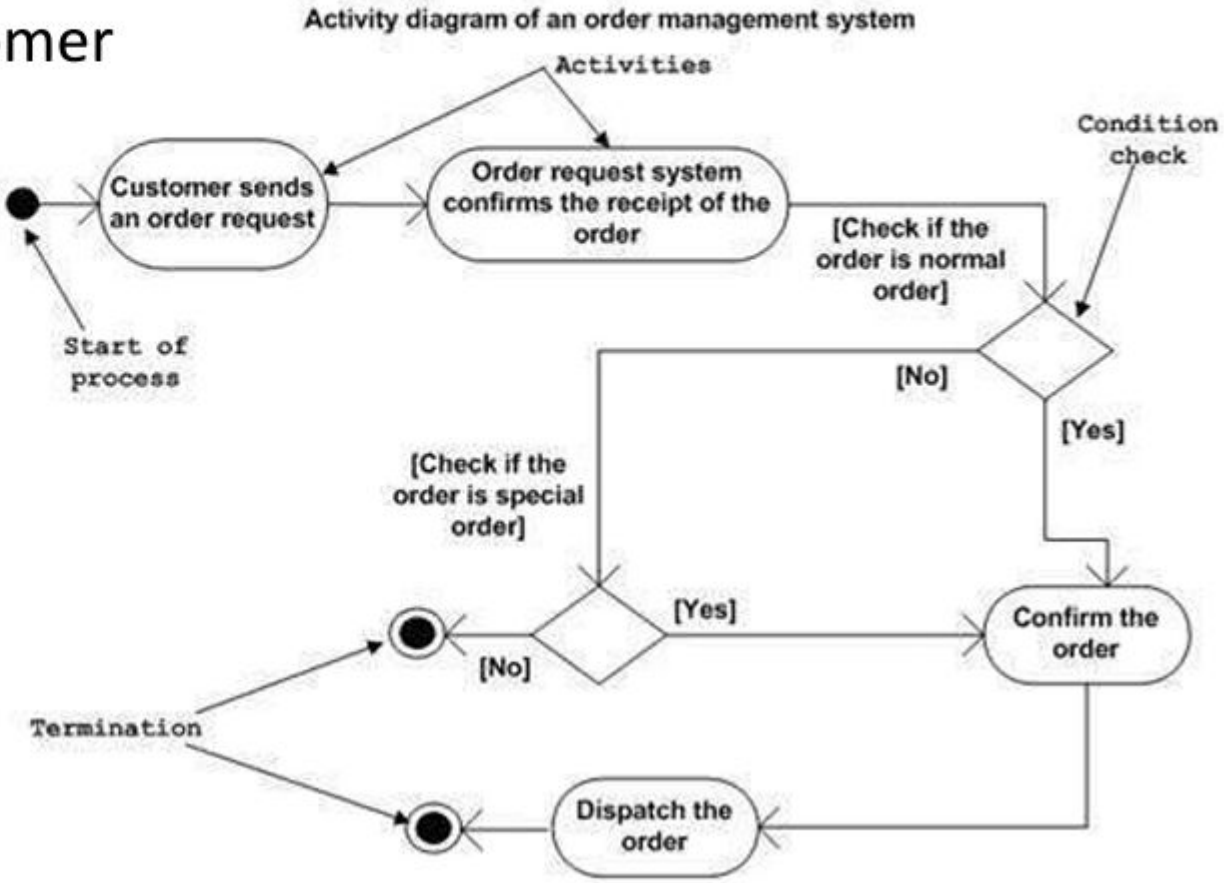
Collaboration diagram

Similar to sequence diagram, but in a spatial view, using numbering to indicate time order



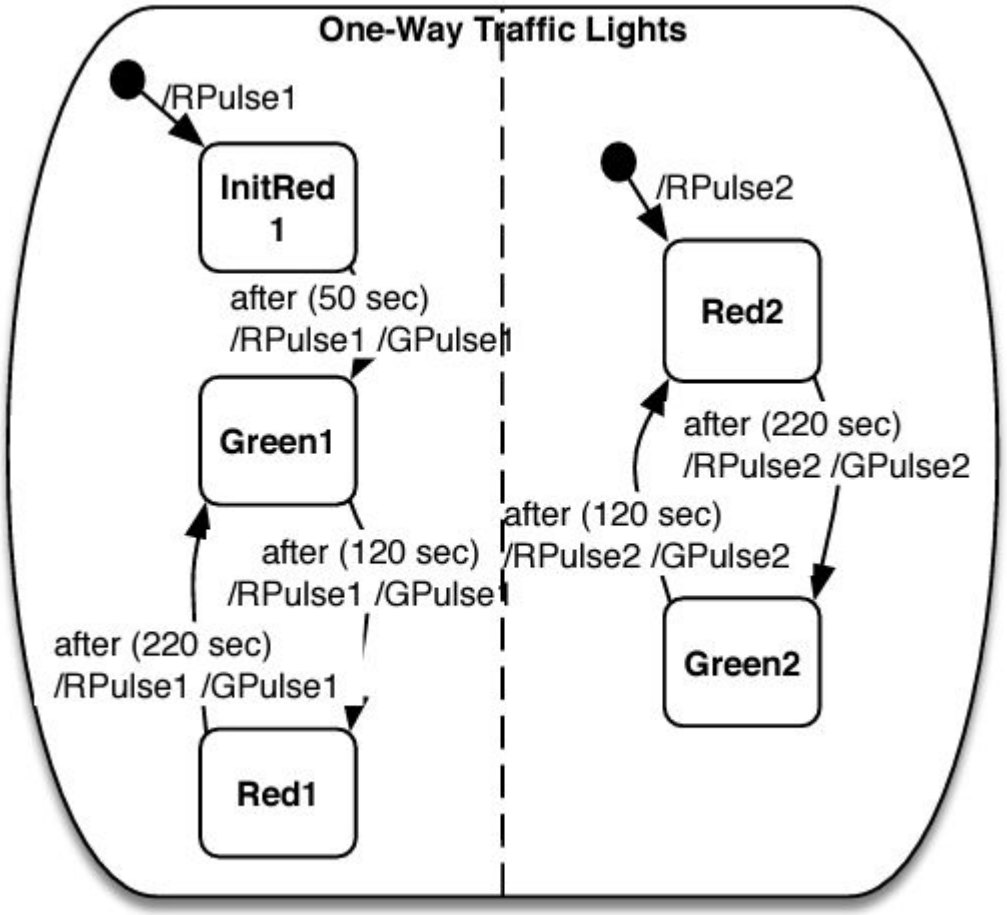
Activity diagram

- Following diagram is drawn with the four main activities –
  - Send order by the customer
  - Receipt of the order
  - Confirm the order
  - Dispatch the order



State Diagram

Shows states, transitions, events, and activities depicting the dynamic view of internal object states



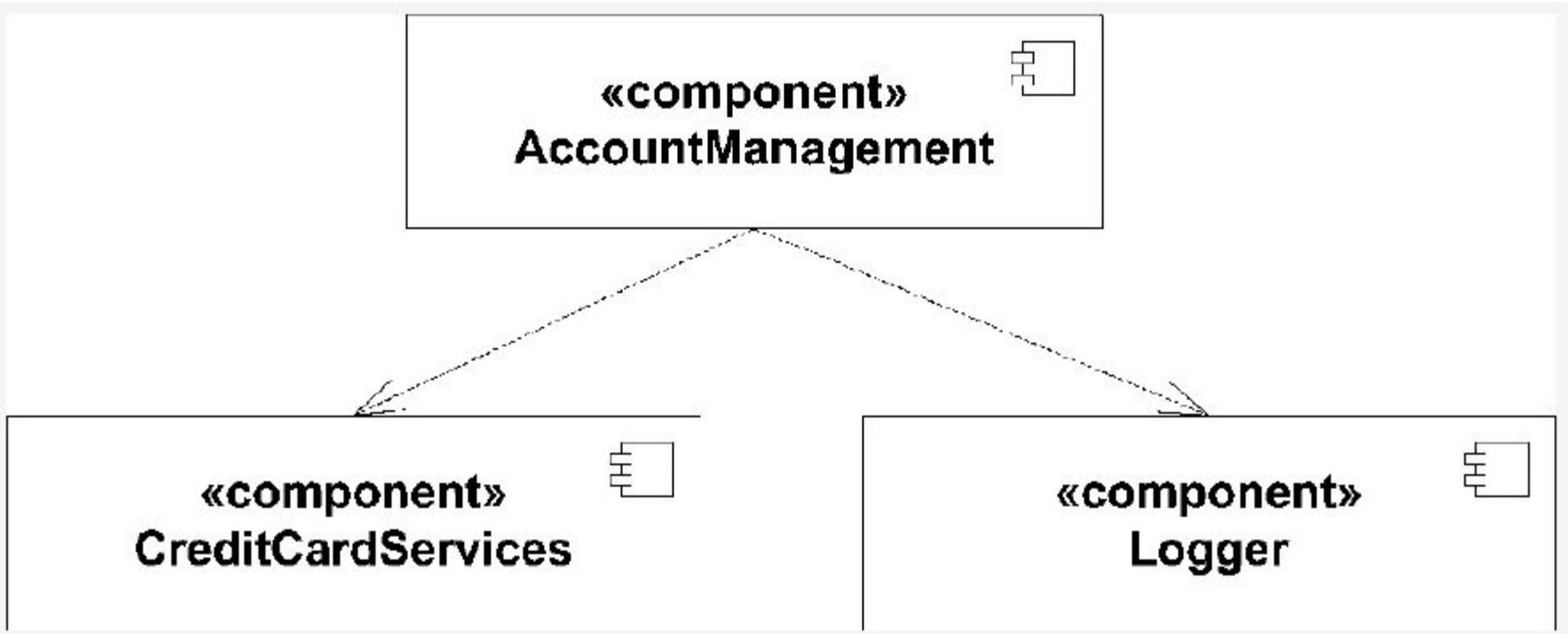
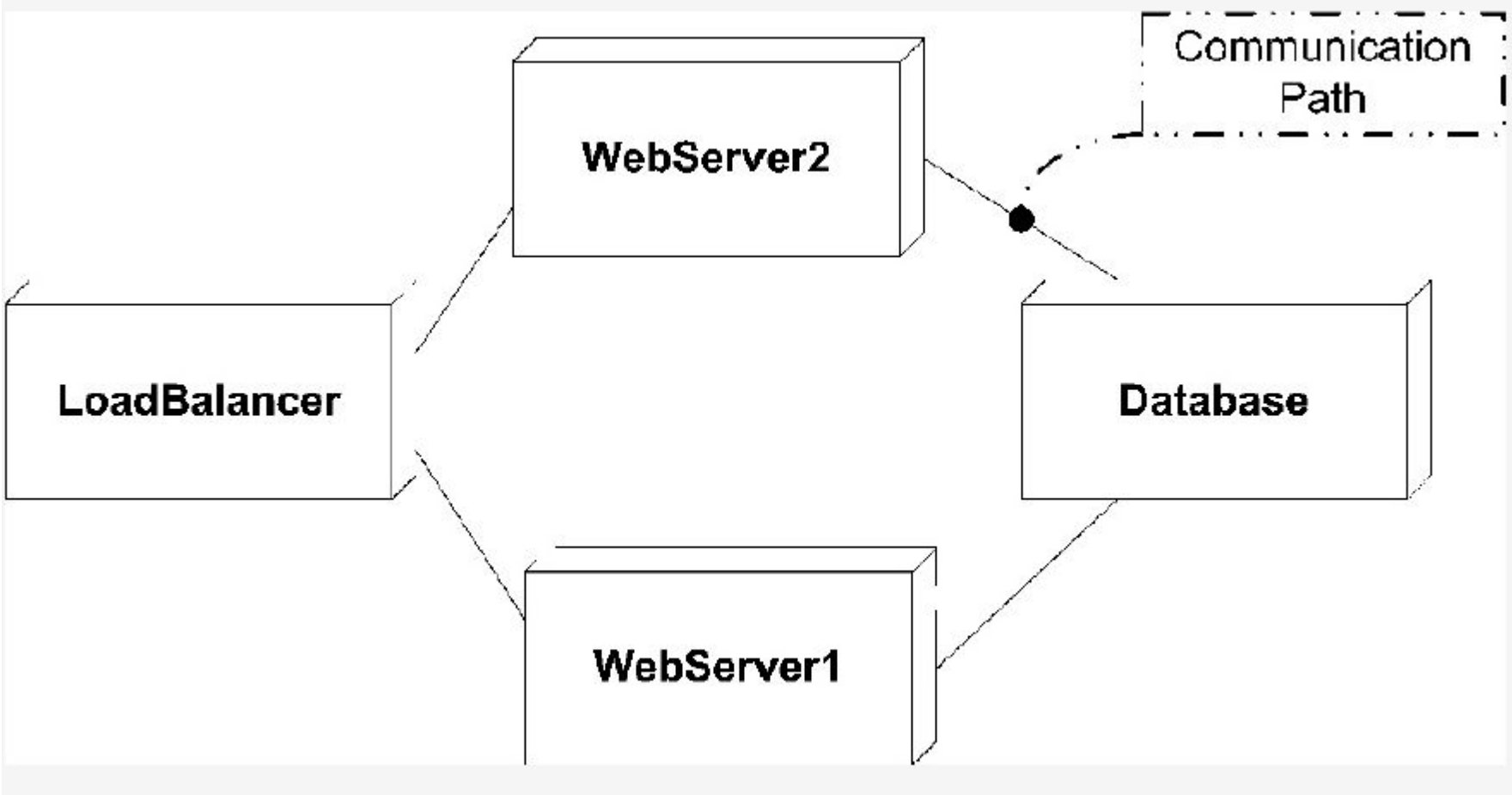


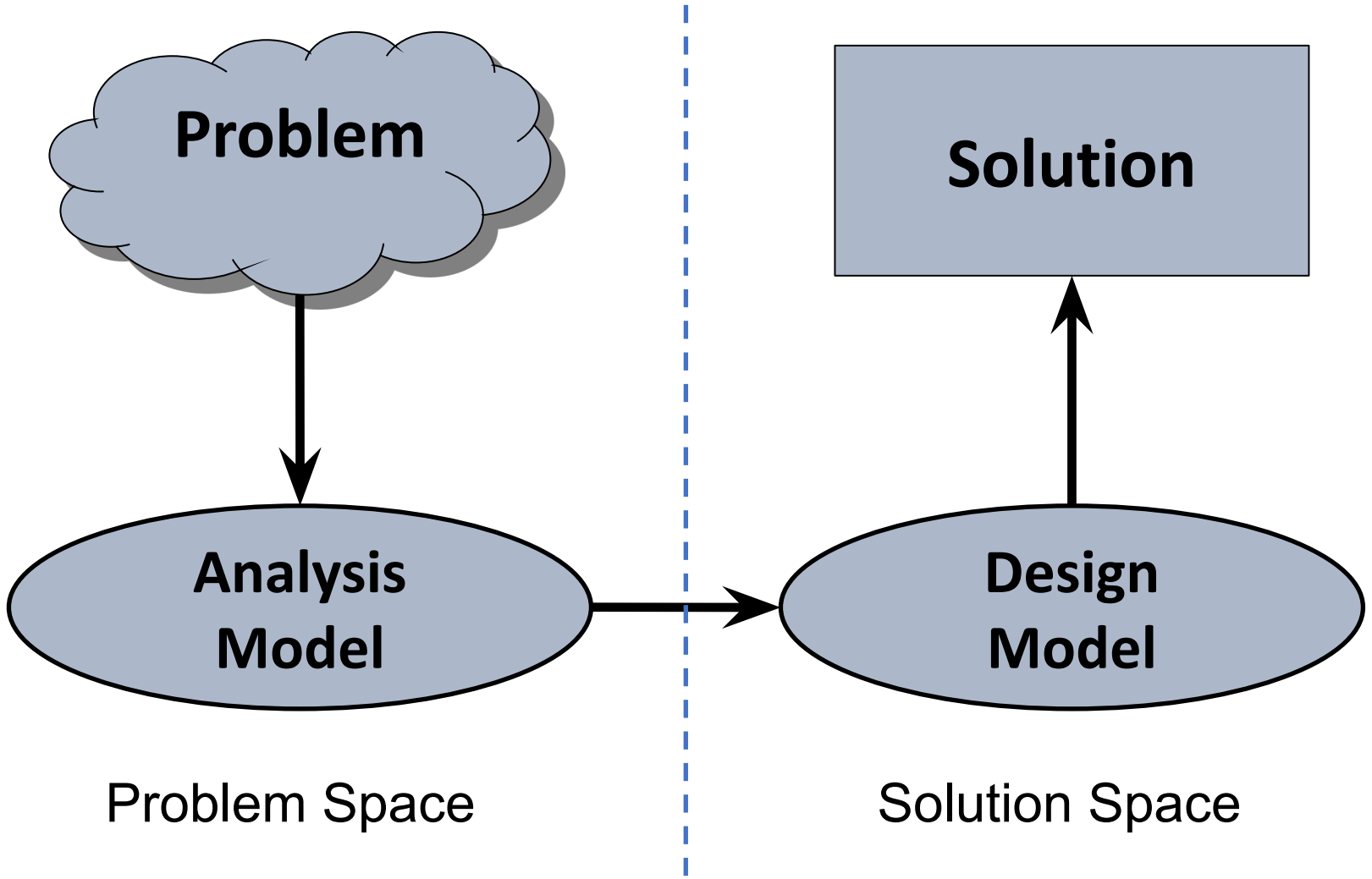
Figure 5-2. Component dependency

Deployment Diagram

Shows the configuration of run-time processing nodes and the components that are deployed on them

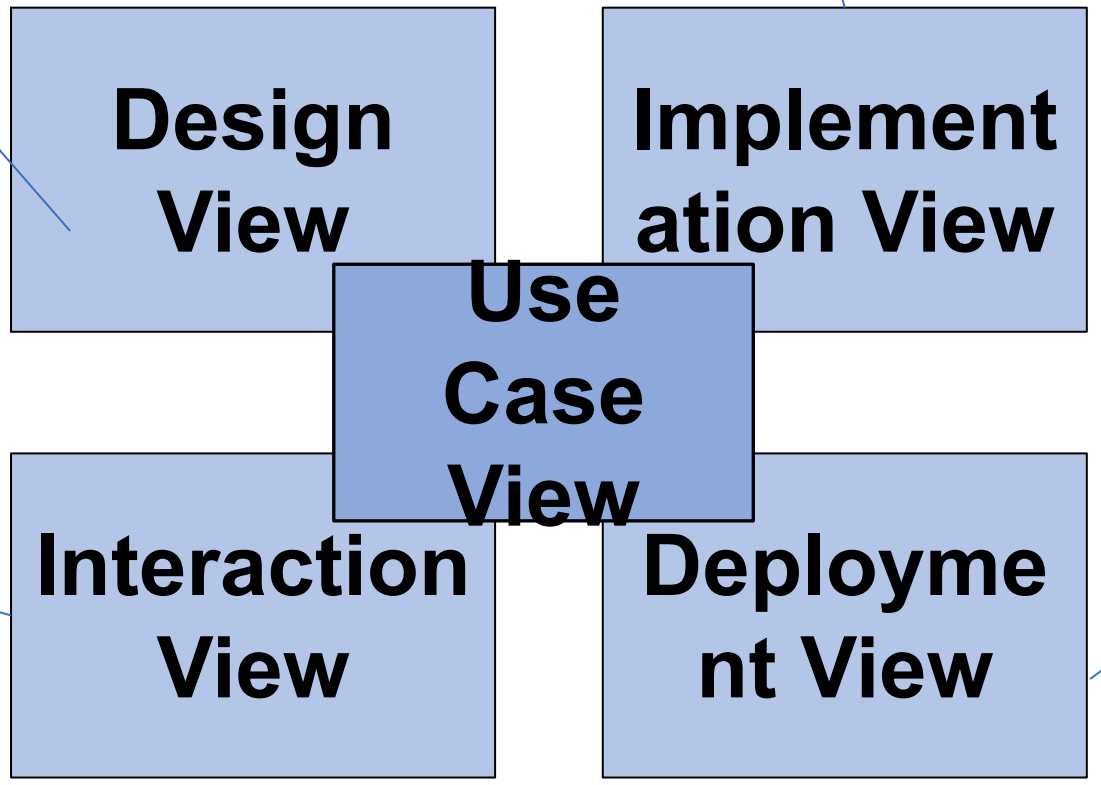


Models in Solution Development



Object Oriented Analysis and Design using Java

Views of a Solution System



captures the classes, interfaces, and patterns that describe the representation of the problem domain and how the software will be built to address it.
class diagrams, object diagrams, activity diagrams, composite structure diagrams, and sequence diagrams to convey the design of a system.

emphasizes the components, files, and resources used by a system. what components depend on what, what source files implement what classes, etc. Implementation views almost always use one or more component diagrams and may include interaction diagrams, statechart diagrams, and composite structure diagrams.

Helps the modeler choose diagrams that help convey the correct information depending on his goals.
models are often divided into *4+1* *views* of a system.
The 4+1 notation represents four distinct views of a system and one overview of how everything fits together.

intended to capture concurrency, performance, and scalability information. Process views often use some form of interaction diagrams and activity diagrams to show how a system actually behaves at runtime.

how a system is configured, installed, and executed. It often consists of component diagrams, deployment diagrams, and interaction diagrams. The deployment view captures how the physical layout of the hardware communicates to execute the system, and can be used to show failover, redundancy, and network topology.

Views of a Solution System

- Use Case View
  - Describes behavior of system as seen by end users, analysts, and testers
- Design View
  - Describes Classes, Interfaces, and Collaborations that form the vocabulary of the problem and its solution
- Interaction View
  - Describes flow of control among its various parts
  - Address performance, scalability, and throughput

UML Diagrams –Summary

- State Diagram
  - Shows states, transitions, events, and activities depicting the dynamic view of internal object states
- Sequence Diagram
  - Shows interactions between objects as a time-ordered view
- Collaboration Diagram
  - Similar to sequence diagram, but in a spatial view, using numbering to indicate time order
- Deployment Diagram
  - Shows the configuration of run-time processing nodes and the components that are deployed on them

THANK YOU

Vinay Joshi
Department of Computer Science and Engineering

vinayj@pes.edu