OBJECT ORIENTED SOFTWARE DEVELOPMENT

Subash Manandhar

Chapter 1 – Introduction
1.2 Review of UML

- **Introduction**
  - UML stands for **U**nified **M**odeling **L**anguage.
  - UML is different from the other common programming languages like C++, Java, COBOL etc.
  - UML is a pictorial language used to make software blue prints.
  - UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
  - UML was created by Object Management Group and UML 1.0 specification draft was proposed to the OMG in January 1997.

Chapter 1 – Introduction
1.2 Review of UML

- **UML is a language for visualizing**
  - Modeling of any system requires certain degree of visualization of the system.
  - In a system development some things are best modeled textually, while some are best modeled graphically.
  - UML is such a language that is more than just a bunch of graphical symbols. Each symbol in contrast has a well defined semantics which is best for visualization
- **UML is a language for specifying**
  - Specifying means building models that are **precise, unambiguous and complete.**
  - UML addresses the specification of all the important analysis, design and implementation decisions that must be:
    - Scientific
    - Distributed web based services

Chapter 1 – Introduction
1.2 Review of UML

- **UML is a language for constructing**
  - The models of UML can directly be connected to a variety of programming languages i.e . It is possible to map a UML model to a programming language such as Java, C++, etc.
- **UML is a language for documenting**
  - software organization produces all sorts of artifacts in addition to executable code.
  - The UML addresses the documentation of a system's architecture and all of its details.
  - The UML also provides the language for expressing requirements and for test. Finally, the UML provides a language for modeling the activities of a project planning and release management.

Chapter 1 – Introduction
1.2 Review of UML

- **Introduction**
  - The OMG specification states:
    - *"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components ."*

Chapter 1 – Introduction
1.2 Review of UML

- **Goals of UML**
  - *A picture is worth a thousand words*
  - to define some general purpose modeling language which all modelers can use and also it needs to be made simple to understand and use.
  - **UML**
    - Helps to reduce cost and time-to-market.
    - Helps managing a complex project architecture.
    - Helps to convey ideas between developers\designers\etc.

Chapter 1 – Introduction
1.2 Review of UML

- **Unified** because it … – **Combines** main preceding OO methods (**Booch by Grady Booch, OMT by Jim Rumbaugh and OOSE by Ivar Jacobson**)
- **Modeling** because it is … – Primarily used for **visually modeling** systems. Many system views are supported by appropriate models
- **Language** because … – It **offers a syntax** through which to express modeled knowledge
- **UML is not:**
  - A **visual programming language** or environment
  - A **database specification** tool
  - A **development process** (i.e. an SDLC)
  - A **panacea** (The ultimate solution)
  - A **quality guarantee**

Chapter 1 – Introduction
1.2 Review of UML

- **Conceptual Model of UML**
  - To understand conceptual model of UML first we need to clarify *What is a conceptual model?* and *Why a conceptual model is at all required?*
  - A conceptual model can be defined as a model which is made of concepts and their relationships.
  - A conceptual model 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.
  - As UML describes the real time systems, it is very important to make a conceptual model and then proceed gradually.
  - Conceptual model of UML can be mastered by learning the following three major elements:
    - UML building blocks
    - Rules to connect the building blocks
    - Common mechanisms of UML

Chapter 1 – Introduction
1.2 Review of UML

- **Relation of UML with OO**
  - The relation between OO design and UML is very important to understand.
  - The OO design is transformed into UML diagrams according to the requirement.
  - Before understanding the UML in details the OO concepts should be learned properly.
  - Once the OO analysis and design is done the next step is very easy.
  - The input from the OO analysis and design is the input to the UML diagrams.
  - class diagram, object diagram, collaboration diagram, interaction diagrams all would basically be designed based on the objects.

Chapter 1 – Introduction
1.2 Review of UML

- **Building Blocks of UML**
  - Things
  - Relationships
  - Diagrams

Chapter 1 – Introduction
1.2 Review of UML

- **Building Blocks of UML**
  - **Things:**
    - are the most important building blocks of UML. Things can be:
      - Structural
      - Behavioral
      - Grouping
      - Annotational
    - **Structural:**
      - define the static part of the model.
      - They represent physical and conceptual elements.
      - ***Class***:
        - represents set of objects having similar responsibilities.

Chapter 1 – Introduction
1.2 Review of UML

- **Building Blocks of UML**
  - **Things:**
    - **Structural:**
      - ***Interface***:
        - defines a set of operations which specify the responsibility of a class.
      - ***Collaboration***:
        - defines interaction between elements.
      - ***Use case:***
        - represents a set of actions performed by a system for a specific goal.
      - ***Component***:
        - describes physical part of a system.
      - ***Node***:
        - physical element that exists at run time.

Chapter 1 – Introduction
1.2 Review of UML

- **Building Blocks of UML**
  - **Things:**
    - **Behavioral:**
      - consists of the dynamic parts of UML models.
      - ***Interaction:***
        - defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.
      - ***State Machine:***
        - defines the sequence of states an object goes through in response to events.
        - is useful when the state of an object in its life cycle is important.

Chapter 1 – Introduction
1.2 Review of UML

- **Building Blocks of UML**
  - **Things:**
    - **Group:**
      - can be defined as a mechanism to group elements of a UML model together.
      - ***Package:***
        - is the only one grouping thing available for gathering structural and behavioral things.
    - **Annotational:**
      - can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements.
      - *Note:*
        - is used to render comments, constraints etc of an UML element.

Chapter 1 – Introduction
1.2 Review of UML

- **Building Blocks of UML**
  - **Relationships:**
    - another most important building block of UML.
    - It shows how elements are associated with each other and this association describes the functionality of an application.
    - Different kinds of relationship are:
    - **Dependency**
      - is a relationship between two things in which change in one element also affects the other one.
    - **Association:**
      - is basically a set of links that connects elements of an UML model.
      - It also describes how many objects are taking part in that relationship.

Chapter 1 – Introduction
1.2 Review of UML

- **Building Blocks of UML**
  - **Relationships:**
    - **Generalization:**
      - can be defined as a relationship which connects a specialized element with a generalized element.
      - It basically describes inheritance relationship in the world of objects.
    - **Realization:**
      - can be defined as a relationship in which two elements are connected.
      - One element describes some responsibility which is not implemented and the other one implements them.
      - This relationship exists in case of interfaces.

Chapter 1 – Introduction
1.2 Review of UML

- **Building Blocks of UML**
  - **UML Diagrams :**
    - All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.
    - The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it a complete one.
    - Different UML diagrams are
      - Class diagram
      - Object diagram
      - Use case diagram
      - Sequence diagram
      - Collaboration diagram
      - Activity diagram
      - State chart diagram
      - Deployment diagram
      - Component diagram

Chapter 1 – Introduction
1.2 Review of UML

- UML plays an important role in defining different perspectives of a system.
- These perspectives are:
  - Design
  - Implementation
  - Process
  - Deployment
- And the centre is the **Use Case** view which connects all these four.
- A **Use case** represents the functionality of the system. So the other perspectives are connected with use case.
- **Design** of a system consists of classes, interfaces and collaboration.
  - UML provides class diagram, object diagram to support this.
- **Implementation** defines the components assembled together to make a complete physical system.
  - UML component diagram is used to support implementation perspective.
- **Process** defines the flow of the system.
  - So the same elements as used in *Design* are also used to support this perspective.
- **Deployment** represents the physical nodes of the system that forms the hardware.
  - UML deployment diagram is used to support this perspective.

Chapter 1 – Introduction
1.2 Review of UML

- **UML Modeling Types**
  - ***Structural Modeling***
    - captures the static features of a system
    - represents the framework for the system and this framework is the place where all other components exist.
    - never describes the dynamic behavior of the system.
    - Consists of
      - Classes diagrams
      - Objects diagrams
      - Deployment diagrams
      - Package diagrams
      - Composite structure diagram
      - Component diagram

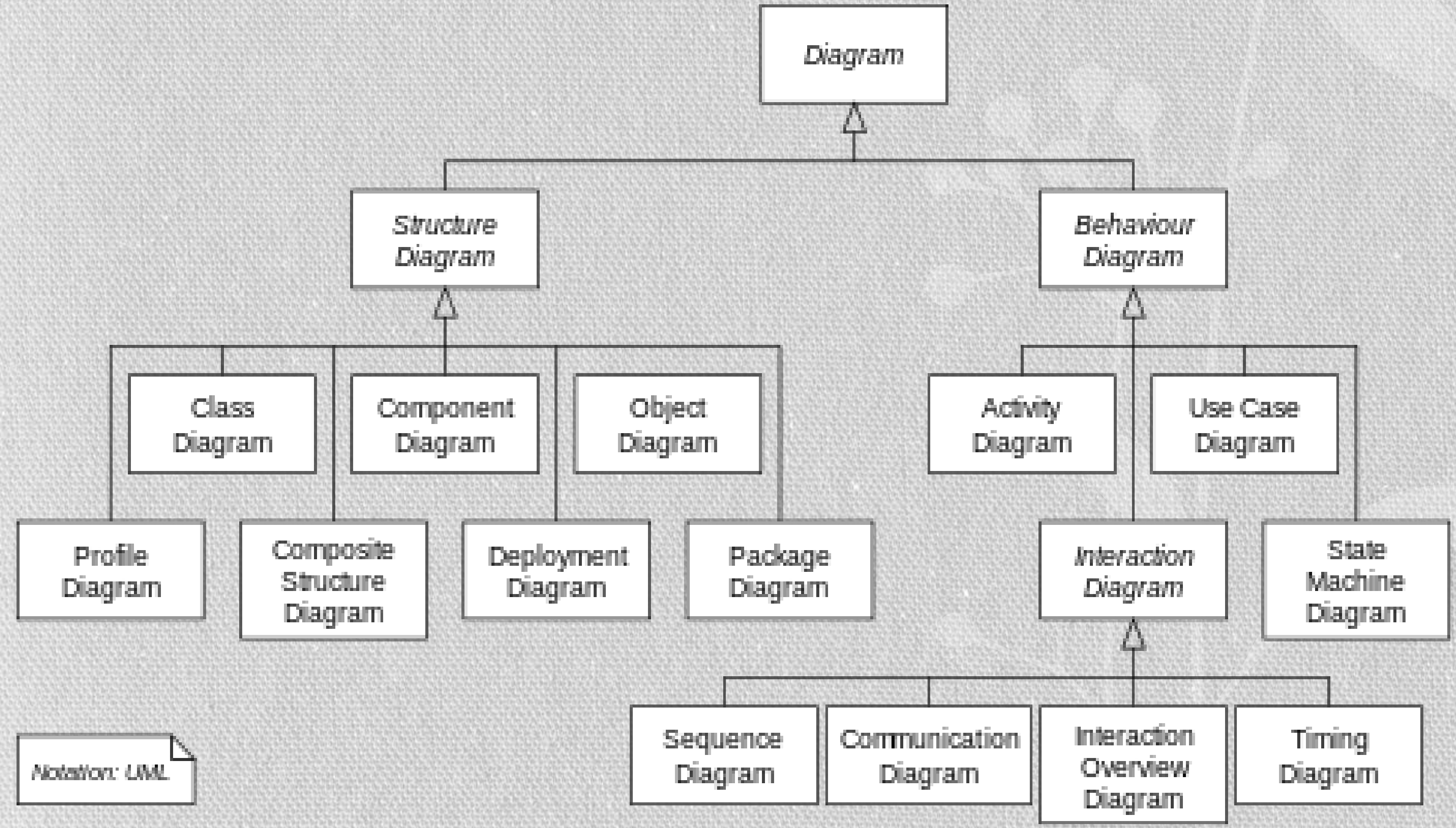
Chapter 1 – Introduction
1.2 Review of UML

- **UML Modeling Types**
  - ***Behavioral Modeling***
    - describes the interaction in the system.
    - It represents the interaction among the structural diagrams.
    - Behavioral modeling shows the dynamic nature of the system.
    - Consists of
      - Activity diagrams
      - Interaction diagrams
      - Use case diagrams

Chapter 1 – Introduction
1.2 Review of UML

- **UML Modeling Types**
  - ***Architectural Modeling***
    - represents the overall framework of the system.
    - It contains both structural and behavioral elements of the system.
    - Architectural model can be defined as the blue print of the entire system.
    - Package diagram comes under architectural modeling.

Chapter 1 – Introduction
1.2 Review of UML



Chapter 1 – Introduction
1.2 Review of UML

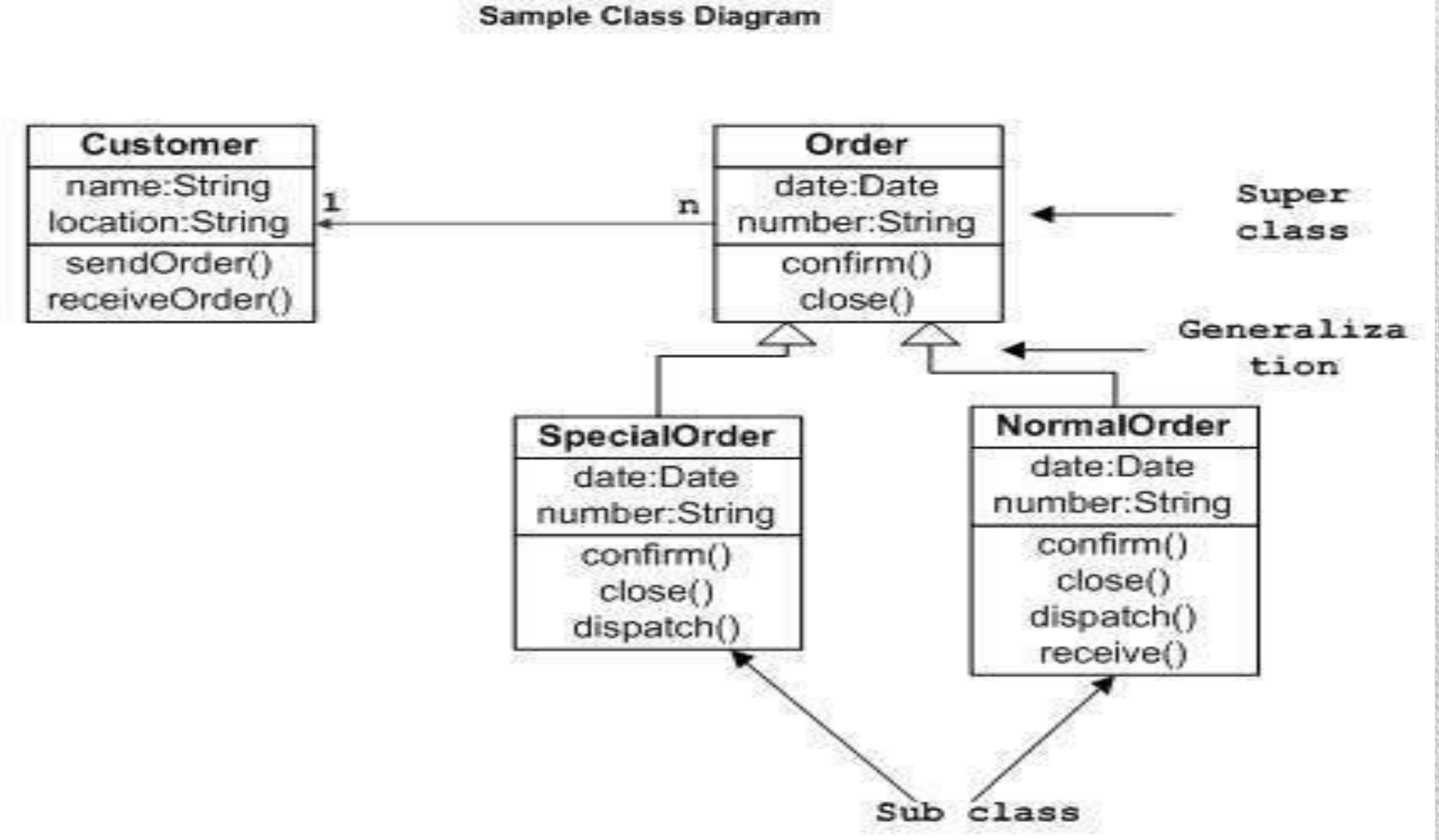
- **Class Diagram**
  - is a static diagram.
  - represents the static view of an application.
  - Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.
  - The class diagram describes the attributes and operations of a class and also the constraints imposed on the system.
  - The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.
  - The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a *structural diagram*.
  - Purpose:
    - Analysis and design of the static view of an application.
    - Describe responsibilities of a system.
    - Base for component and deployment diagrams.
    - Forward and reverse engineering.

Chapter 1 – Introduction
1.2 Review of UML

- **Class Diagram**
  - *How to draw?*
    - The name of the class diagram should be meaningful to describe the aspect of the system.
    - Each element and their relationships should be identified in advance.
    - Responsibility (attributes and methods) of each class should be clearly identified.
    - For each class minimum number of properties should be specified. Because unnecessary properties will make the diagram complicated.
    - Use notes when ever required to describe some aspect of the diagram. Because at the end of the drawing it should be understandable to the developer/coder.
    - Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

Chapter 1 – Introduction
1.2 Review of UML

- **Class Diagram**



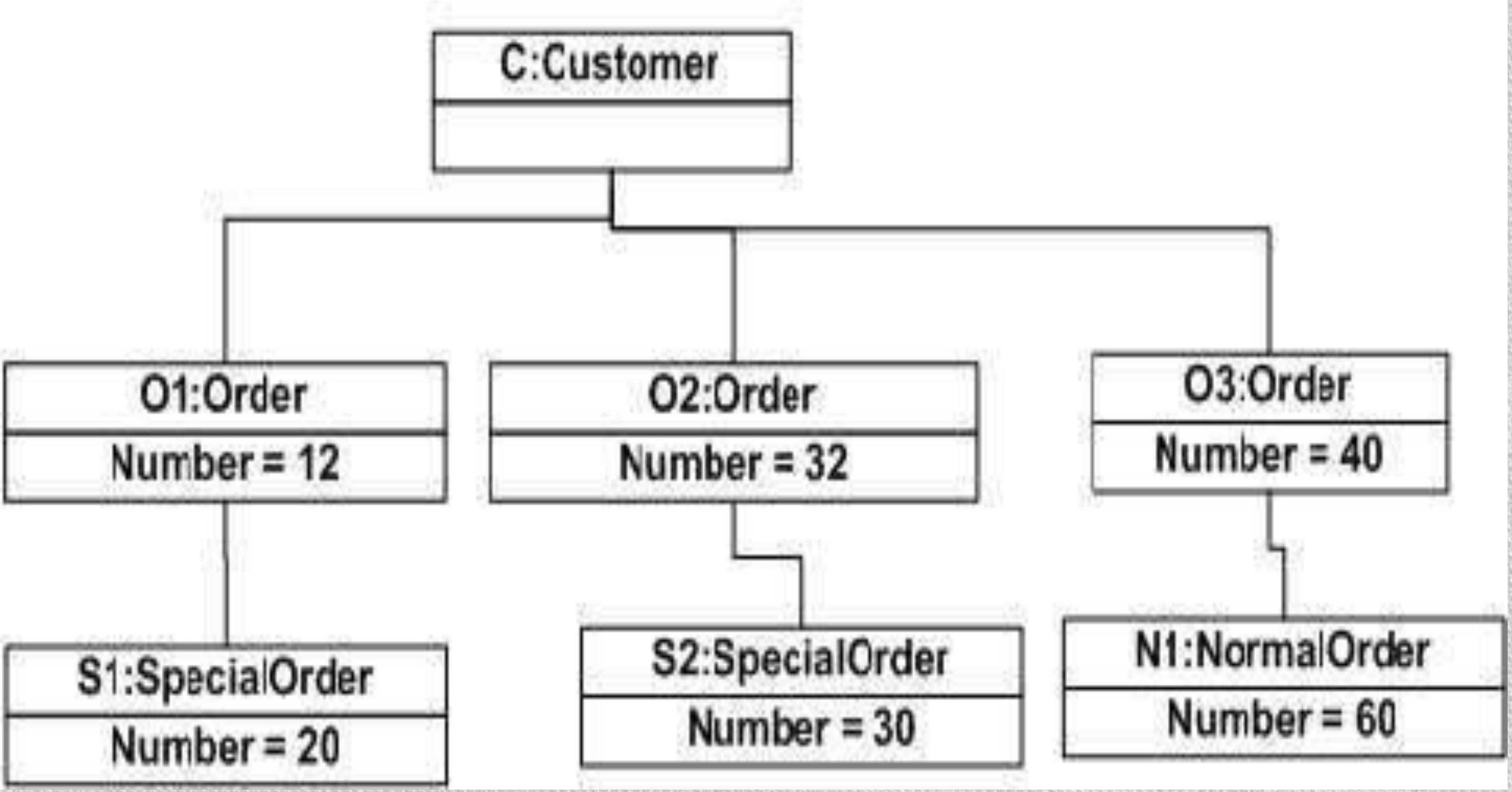
Chapter 1 – Introduction
1.2 Review of UML

- **Object Diagram**
  - represent an instance of a class diagram.
  - are used to render a set of objects and their relationships as an instance.
  - The difference between class and object diagram is that a class diagram represents an abstract model consisting of classes and their relationships. But an object diagram represents an instance at a particular moment which is concrete in nature.
  - Purpose:
    - Forward and reverse engineering.
    - Object relationships of a system
    - Static view of an interaction.
    - Understand object behavior and their relationship from practical perspective

Chapter 1 – Introduction
1.2 Review of UML

- **Object Diagram**

Object diagram of an order management system



Chapter 1 – Introduction
1.2 Review of UML

- **Use Case Diagram**
  - is dynamic in nature there should be some internal or external factors for making the interaction.
  - internal and external agents are known as actors.
  - So use case diagrams are consists of actors, use cases and their relationships.
  - The diagram is used to model the System/subsystem of an application.
  - A single use case diagram captures a particular functionality of a system.
  - Purpose:
    - Used to gather requirements of a system.
    - Used to get an outside view of a system.
    - Identify external and internal factors influencing the system.
    - Show the interacting among the requirements are actors.

Chapter 1 – Introduction
1.2 Review of UML

- **Use Case Diagram**
  - *How to draw???*
    - Functionalities to be represented as an use case
    - Actors
    - Relationships among the use cases and actors.
    - Then:
      - The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
      - Give a suitable name for actors.
      - Show relationships and dependencies clearly in the diagram.
      - Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
      - Use note when ever required to clarify some important points.

Chapter 1 – Introduction
1.2 Review of UML

- **Use Case Diagram**

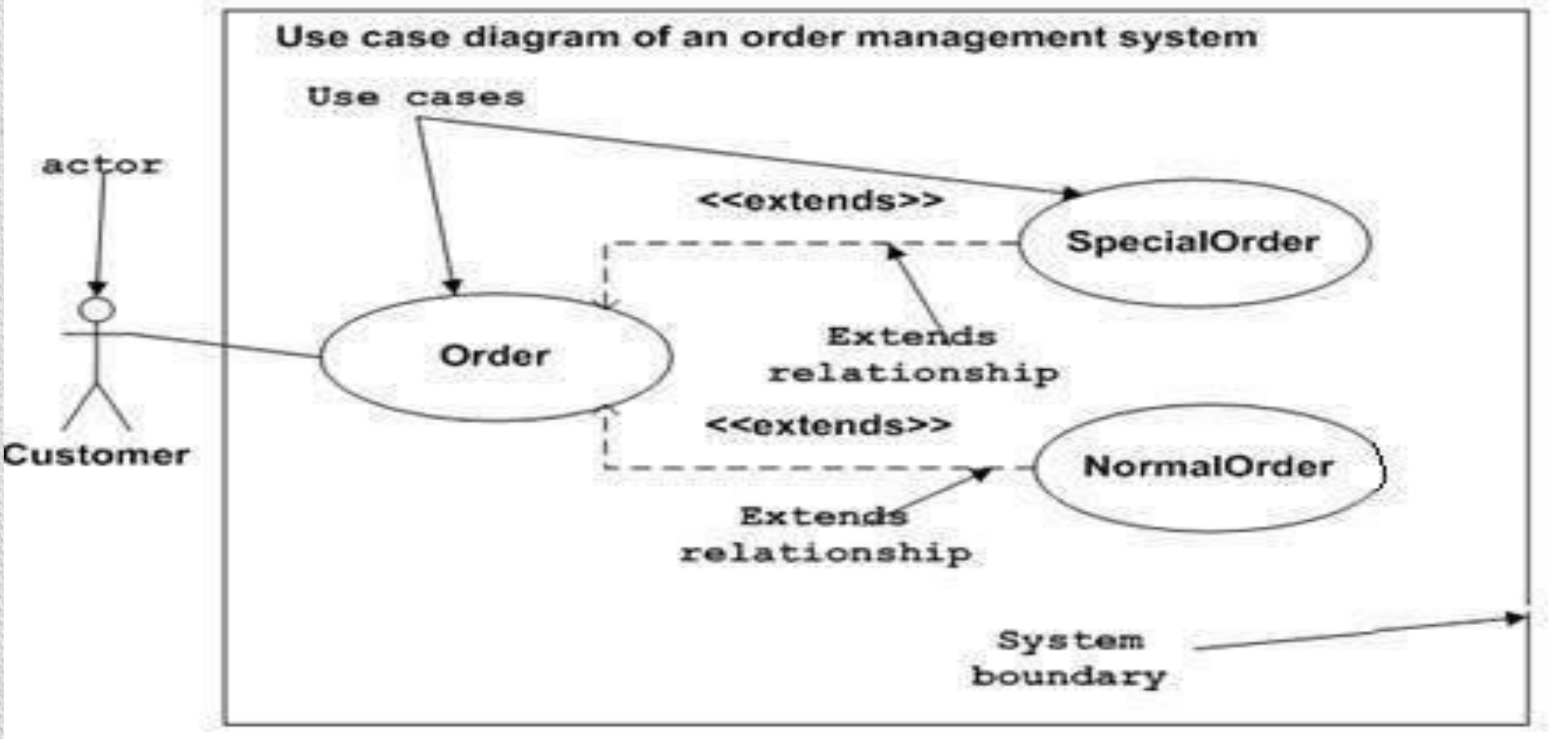


Figure: Sample Use Case diagram

Chapter 1 – Introduction
1.2 Review of UML

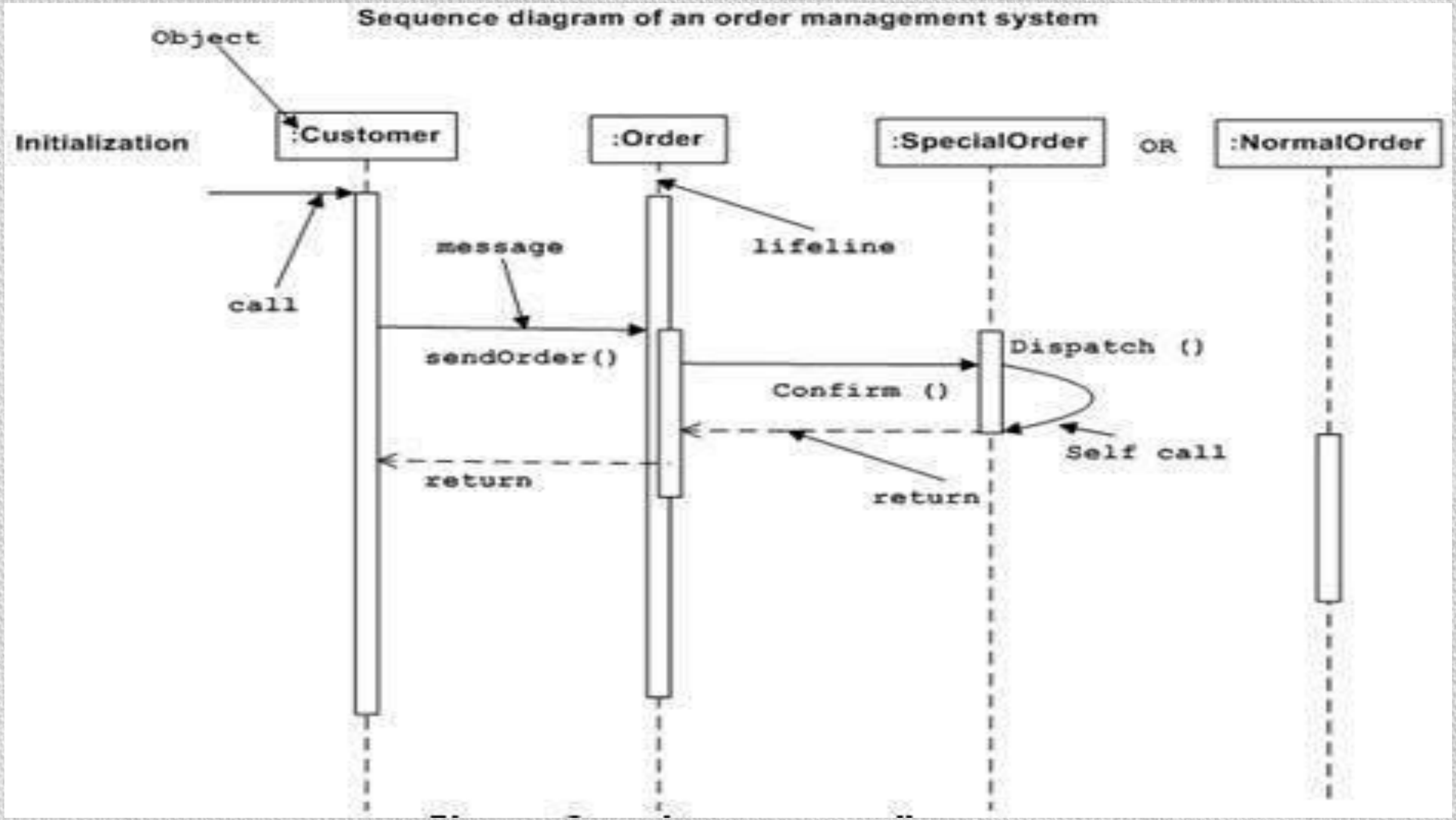
- **Interaction Diagram**
  - is used to describe some type of interactions among the different elements in the model.
  - Represents dynamic behavior.
  - interactive behavior is represented in UML by two diagrams known as *Sequence diagram* and *Collaboration diagram.*
  - The basic purposes of both 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.
  - Purpose:
    - To capture dynamic behavior of a system.
    - To describe the message flow in the system.
    - To describe structural organization of the objects.
    - To describe interaction among objects.

Chapter 1 – Introduction
1.2 Review of UML

- **Interaction Diagram**
  - *How to draw??*
    - Identify
      - Objects taking part in the interaction.
      - Message flows among the objects.
      - The sequence in which the messages are flowing.
      - Object organization.

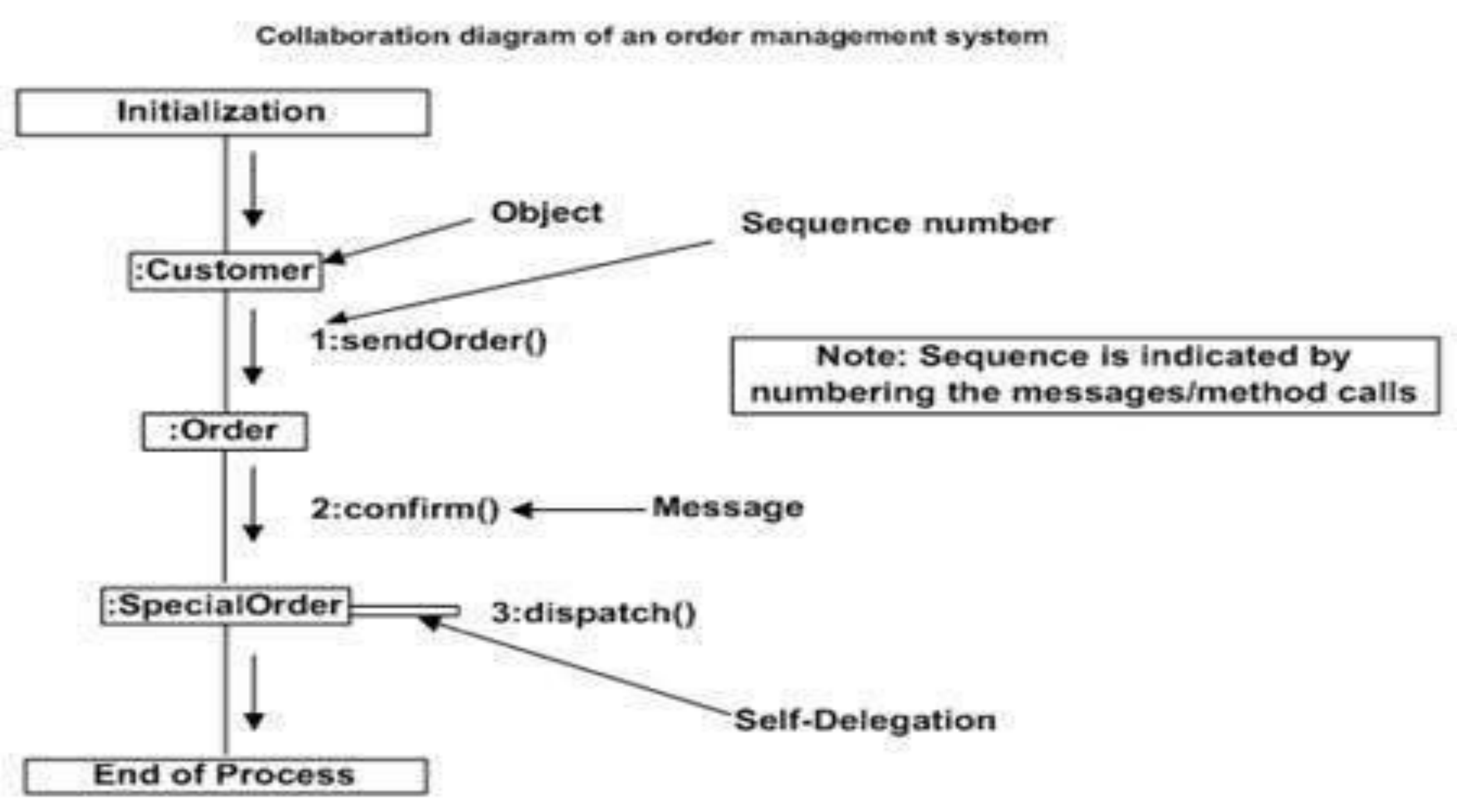
Chapter 1 – Introduction
1.2 Review of UML

- **Sequence Diagram**



Chapter 1 – Introduction
1.2 Review of UML

- **Collaboration Diagram**

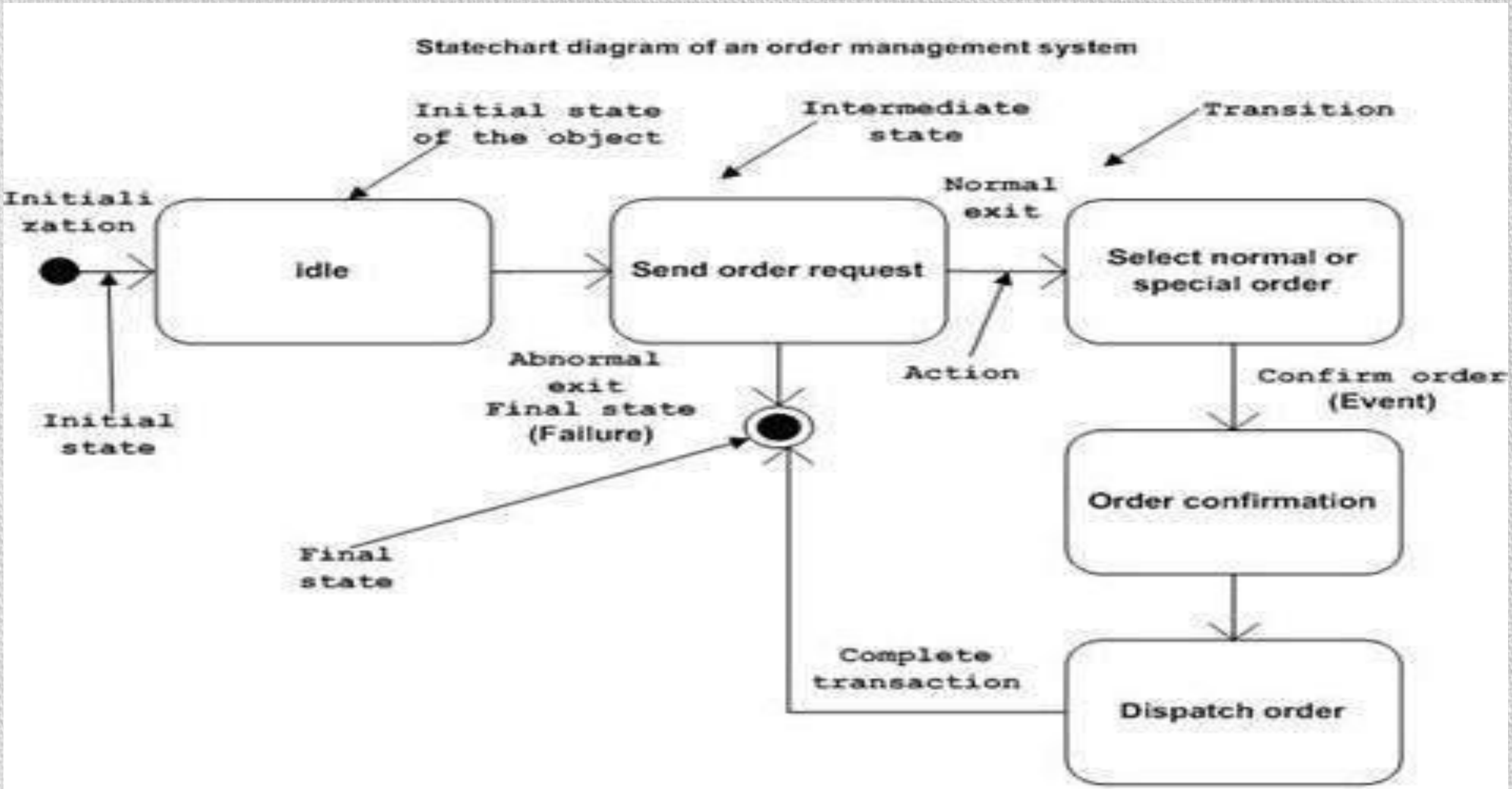


Chapter 1 – Introduction
1.2 Review of UML

- **State Chart Diagram**
  - describes different states of a component in a system.
  - The states are specific to a component/object of a system.
  - A State-chart diagram describes a state machine.
  - Purpose:
    - To model dynamic aspect of a system.
    - To model life time of a reactive system.
    - To describe different states of an object during its life time.
    - Define a state machine to model states of an object.
  - *How to draw??*
    - Identify important objects to be analyzed.
    - Identify the states.
    - Identify the events.

Chapter 1 – Introduction
1.2 Review of UML

- **State Chart Diagram**

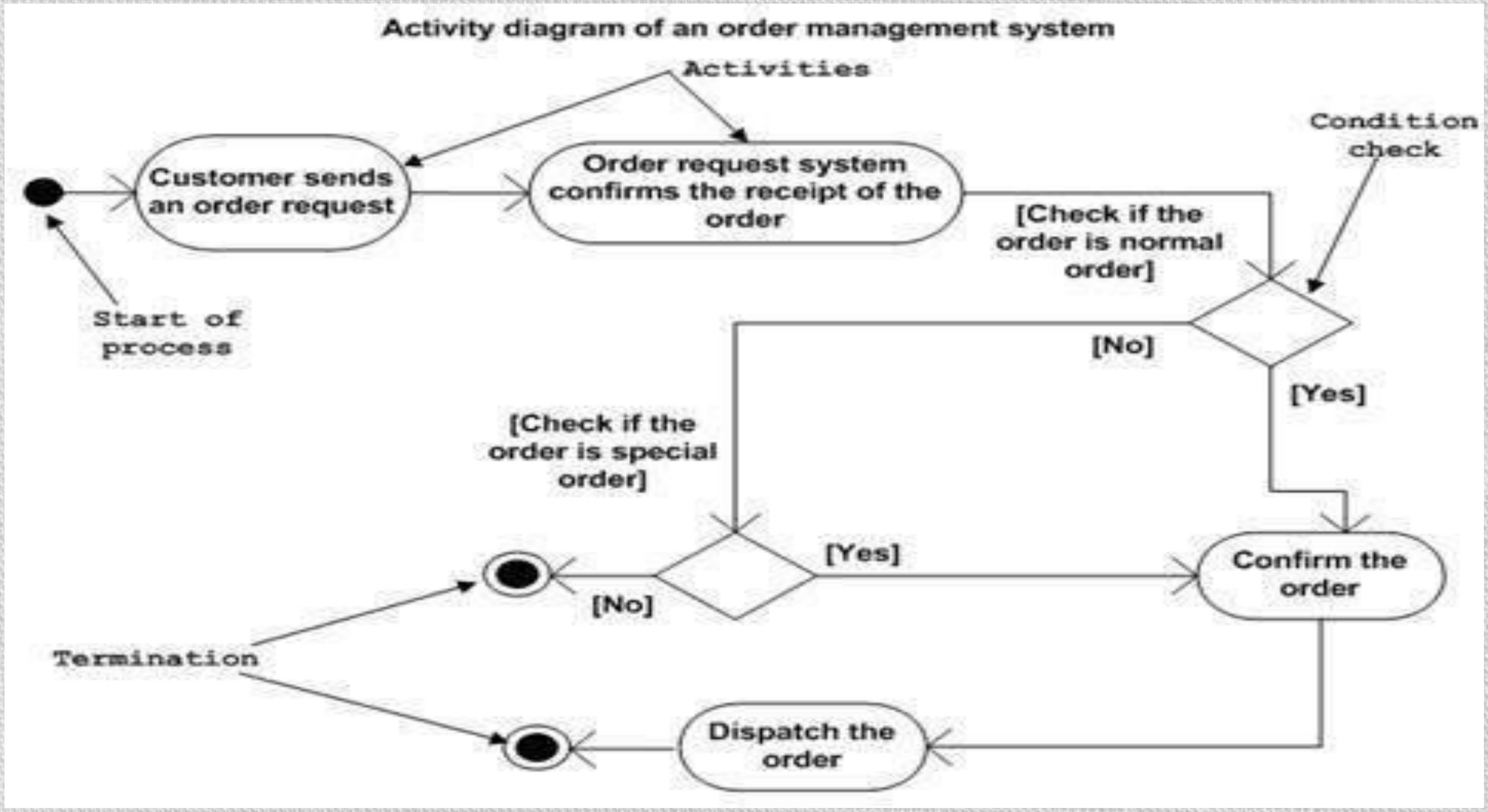


Chapter 1 – Introduction
1.2 Review of UML

- **Activity Diagram**
  - is basically a flow chart to represent the flow from one activity to another activity.
  - Purpose:
    - Draw the activity flow of a system.
    - Describe the sequence from one activity to another.
    - Describe the parallel, branched and concurrent flow of the system.
  - *How to draw??*
    - Identify
      - Activities
      - Association
      - Conditions
      - Constraints

Chapter 1 – Introduction
1.2 Review of UML

- **Activity Diagram**


Activity diagram of an order management system
Activities
Condition check
Customer sends an order request
Order request system confirms the receipt of the order
[Check if the order is normal order]
Start of process
[No]
[Yes]
[Check if the order is special order]
[Yes]
[No]
Confirm the order
Termination
Dispatch the order