Chapter Eight

Determining What to Build: OO Analysis

Compiled Yilkal B.

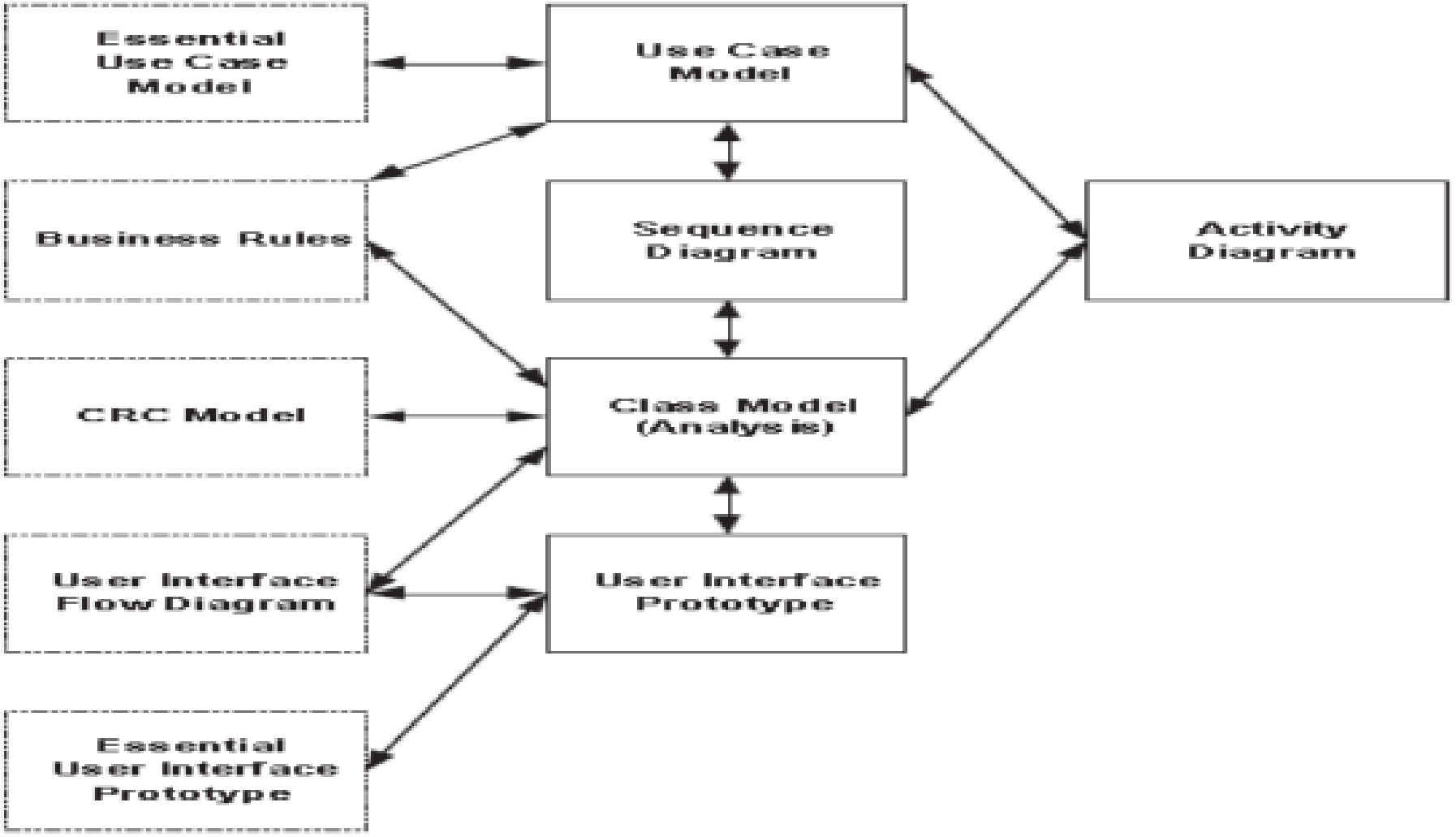
Chapter Outline

- *System Use Case Modeling*
- *Sequence Diagrams: From Use Cases to Classes*
- *Conceptual Modeling: Class diagrams*
- *Activity diagramming*
- *User interface prototyping Evolving your supplementary specification*
- *Applying Analysis patterns Effectively*
- *Organizing your models with packages*

Introduction

- ***The purpose of analysis*** *is to understand what will be built.*
- *This is similar to* ***requirements gathering****.*
- *The main difference is that:*
  - *The focus of requirements gathering is on understanding your users and their potential usage of the system,*
  - *Whereas,* ***the focus of analysis shifts to understanding the system itself.***
- *The following picture depicts the main artefacts of your analysis efforts and the relationships between them*

Overview of Analysis artifacts and their Relationships



Cntd…

The picture has three important implications.

1. *Analysis is an iterative process.*

2. ***Requirements gathering and analysis*** *are* ***highly interrelated and iterative****.*

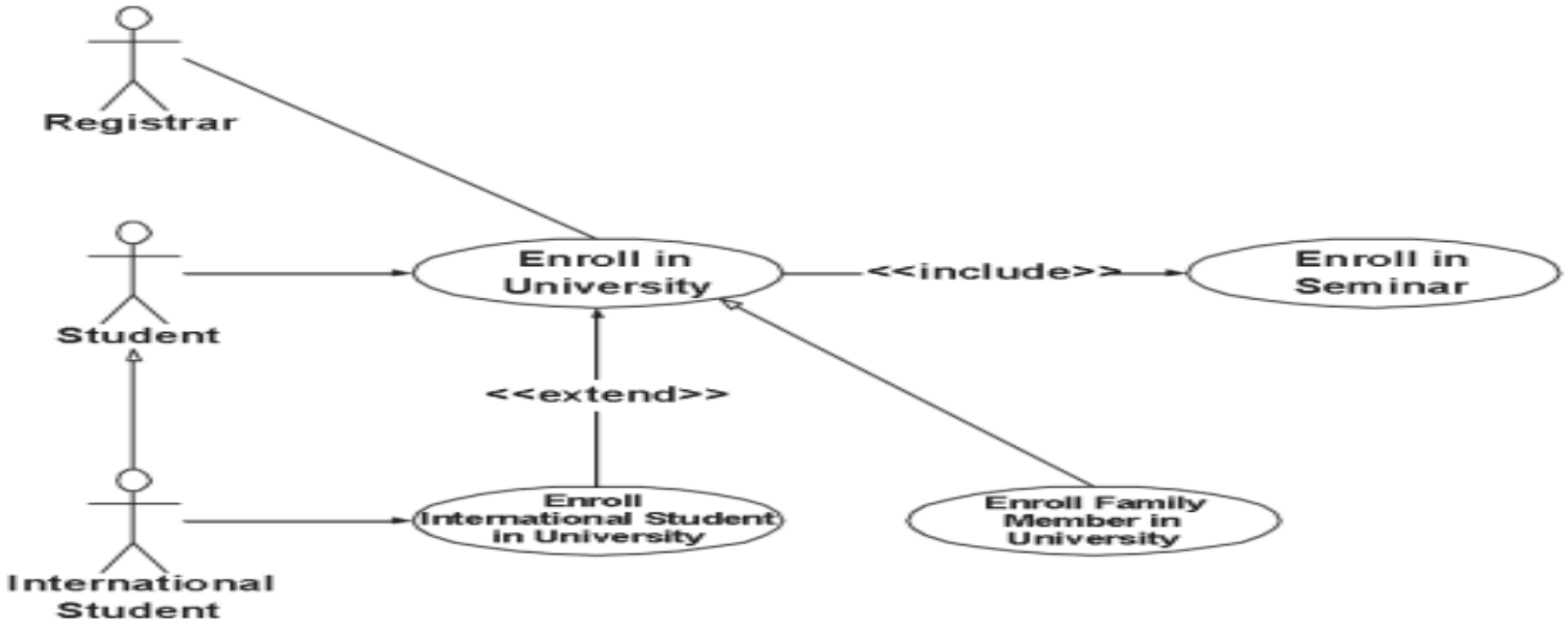
3. *"essential" models, such as* ***essential use case*** *model and essential* ***user interface prototype****, evolve into corresponding analysis artefacts respectively, in to* ***system use case*** *model and* ***user interface prototype****.*

- *During analysis, your main goal is to evolve your essential use cases into system use cases.*
- ***The main difference between an essential use case and a system use case is, in the system use case, you include high-level implementation decisions***

Cntd…

- *E.g., **a system use case** refers to specific user interface components such as screens, HTML pages, or reports-something **you wouldn't do** in an essential use case.*
- ***During analysis,** you make decisions regarding what will be built or design.*
- *Because your UI will work differently depending on the implementation technology*
- ***Likewise an essential use case model,** **a system use case model** is composed of a use case diagram and the accompanying documentation describing the **use cases**, **actors**, and **associations.***
- *The following figures provides an example of **a use case diagram,** depicts a collection of **use cases**, **actors,** their associations, a system boundary box (optional), and packages (optional).*

Cntd…



- *The **rectangle around** the use cases is called the **system boundary box** and, as the name suggests, it delimits the scope of your system.*
- *The **use cases inside the rectangle represent the** functionality you intend to implement.*
- ***Packages** are UML constructs that enable you to organize model elements (such as use cases) into groups.*

Reuse in Use Case Models: <<extend>>, <<include>>, and Inheritance

- ***Potential reuse*** *can be modelled through four generalization relationships supported by the UML use case models:*
  - *extend relationships between use cases,*
  - *include relationships between use cases,*
  - *inheritance between use cases,*
  - *inheritance between actors.*

Good Things to Know About Use Case Modelling

- *An important thing to understand about use case models is that* ***the associations between actors and use cases*** *indicate the need for interfaces.*
- ***When the actor is a person,*** *then to support the association, you need to develop user interface components, such as* ***screens and reports.***

The Unified Modeling Language (UML)

- The UML is a graphical language for OOAD **that gives a standard way to write a software system's blueprint.**
- It helps to visualize, specify, construct, and document the artifacts of an OO system.
- **It is used to depict the structures and the relationships in a complex system.**

Systems and Models in UML

- **System:** A set of elements organized to achieve certain objectives form a system. Systems are often divided into subsystems and described by a set of models.
- **Model :** Model is a simplified, complete, and consistent **abstraction of a system**, created for better understanding of the system.
- **View** : A view is a projection of a system's model from a **specific perspective**

Conceptual Model of UML

- The Conceptual Model of UML encompasses **three major elements:**
  1. Basic building blocks
  2. Rules
  3. Common mechanisms

I. Basic Building Blocks: The **three building blocks** of UML are:

1. Things
2. Relationships
3. Diagrams

Cntd…

1.Things: There **are four kinds** of things in UML, namely:

i. **Structural Things:** These are **the nouns of the UML** models representing the static elements that may be either physical or conceptual. The structural things are **class**, object…

ii. **Behavioral Things :** These are **the verbs of the UML** models representing the dynamic behavior **over time and space**.

The two types of behavioral things are interaction and machine.

iii. **Grouping Things:** They comprise the organizational parts of the UML models. There is only one kind of grouping thing, i.e., **package.**

iv. **Notational Things:** These are the explanations in the UML models representing the **comments** applied to describe elements.

Cntd…

2. Relationships : Relationships are the connection between things.

- The **four types** of relationships that can be represented in UML are:

I. **Dependency:** This is a semantic relationship between two things such that a change in one thing brings a change in the other.

II. **Association:** This is a structural relationship that represents a group of links having common structure and common behavior.

III. **Generalization:** This represents a generalization/specialization relationship in which subclasses inherit structure and behavior from super-classes.

IV. **Realization:** This is a semantic relationship between two or more classifiers such that one classifier lays down a contract that the other classifiers ensure to stand.

Cntd…

3. **Diagrams :** *A diagram is a graphical representation of a system.*

- *It comprises of a group of elements generally in the form of a graph.*
- *UML includes nine diagrams in all, namely:*

1. *Class Diagram*
2. *Object Diagram*
3. *Use Case Diagram*
4. *Sequence Diagram*

6. *Collaboration Diagram*
6. *State Chart Diagram*
7. *Activity Diagram*
8. *Component Diagram*
9. *Deployment Diagram*

Cntd…

II. Rules: *UML has a number of rules so that the models are semantically self-consistent and related to other models in the system harmoniously.*

UML has semantic rules for the following:

- *Names*
- *Scope*
- *Visibility*

III. Common Mechanisms: *UML has* ***four*** *common mechanisms:*

- *Specifications*
- *Adornments*
- *Common Divisions*
- *Extensibility Mechanism*

Cntd…

I. Specifications: *In UML, behind each graphical notation, there is a textual statement denoting the* ***syntax and semantics.***

II. Adornments: *Each element in UML has a* ***unique graphical notation****. Besides, there are notations to represent the important aspects of an element like name, scope, visibility, etc.*

III. Common Divisions: *Object-oriented systems can be divided in many ways.*

The ***two common ways*** *of division are:*

1. ***Division of classes and objects*** *: A class is an abstraction of a group of similar objects. An object is the concrete instance that has actual existence in the system.*
2. ***Division of Interface and Implementation*** *: An interface defines the rules for interaction.*

- ✓ *Implementation is the concrete realization of the rules defined in the interface.*

IV. Extensibility Mechanisms

- UML is an **open-ended language**.
- It is possible to extend the capabilities of UML in a controlled manner to suit the requirements of a system.
- The extensibility mechanisms are:
  - ✓ **Stereotypes:** It extends **the vocabulary** of the UML, through which new building blocks can be created out of existing ones.
  - ✓ **Tagged Values:** It extends the properties of UML building blocks.
  - ✓ **Constraints :** It extends the semantics of UML building blocks.

UML Basic Notations

UML defines specific notations for each of the building blocks.

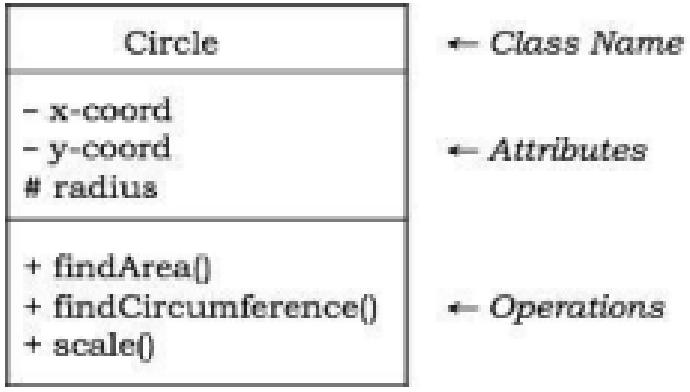
- **Class:** *A class is represented by a* ***rectangle having three sections****:*
  - *the top section containing the name of the class*
  - *the middle section containing class attributes*
  - *the bottom section representing operations of the class*

The visibility of the attributes and operations can be represented in the following ways:

- ✓ ***Public :*** *A public member is visible from anywhere in the system. In class diagram, it is prefixed by the symbol '+'.*
- ✓ ***Private:*** *A private member is visible only from within the class. It cannot be accessed from outside the class. A private member is prefixed by the symbol '–'.*
- ✓ ***Protected*** *: A protected member is visible from* ***within the class*** *and* ***from the subclasses*** *inherited from this class, but not from outside. It is prefixed by the symbol '#'.*

Cntd…

- ***An abstract class*** *has the class name written in italics.*
- *Example : Let us consider the Circle class introduced earlier. The attributes of Circle are x-coord, y-coord, and radius. The operations are findArea(), findCircumference(), and scale(). Let us assume that x-coord and y-coord are private data members, radius is a protected data member, and the member functions are public. The following figure gives the diagrammatic representation of the class*

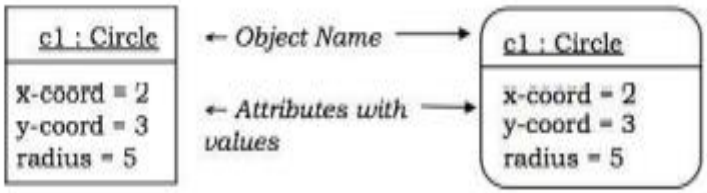


Cntd…

Object: *An object is represented as a rectangle with two sections:*

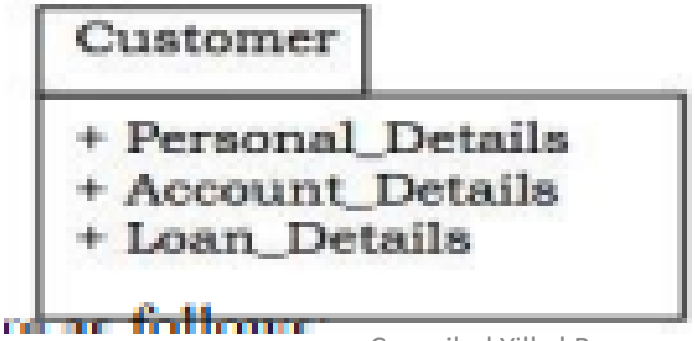
- *The top section contains the name of the object with the name of the class or package of which it is an instance of. The name takes the following forms:*
  - ***object-name:*** *class-name*
  - ***object-name:*** *class-name :: package-name*
  - ***class-name:*** *in case of anonymous objects*
- *The* ***bottom section*** *represents the values of the* ***attributes.*** *It takes the form attribute-name = value.* ***Sometimes objects*** *are represented using* ***rounded rectangles.***

Example : Let us consider an object of the class Circle named c1. We assume that the center of c1 is at (2, 3) and the radius of c1 is 5. The following figure depicts the object.



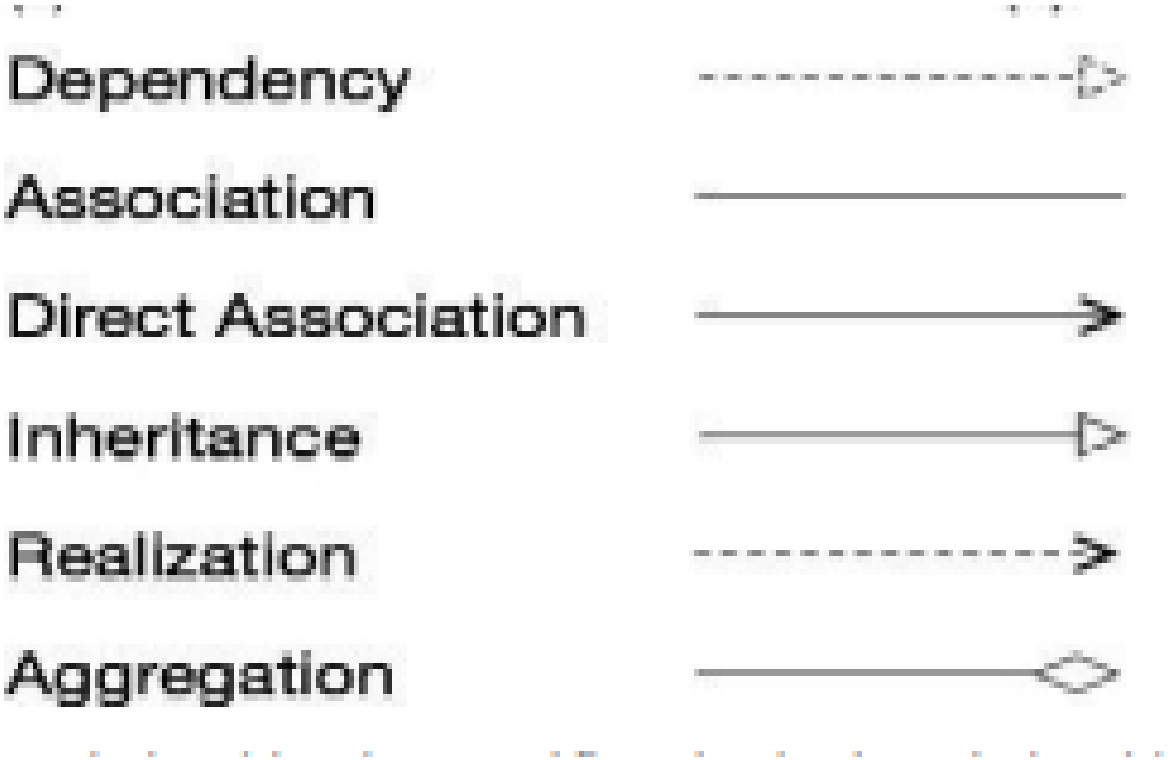
Package

- A package is an organized group of elements.
- A package may contain structural things like **classes, components,** and other packages in it.
- **Notation:** Graphically, a package is represented by a **tabbed folder**.
- A package is generally drawn with only its name.
- However it may have additional details about the contents of the package.



Cntd…

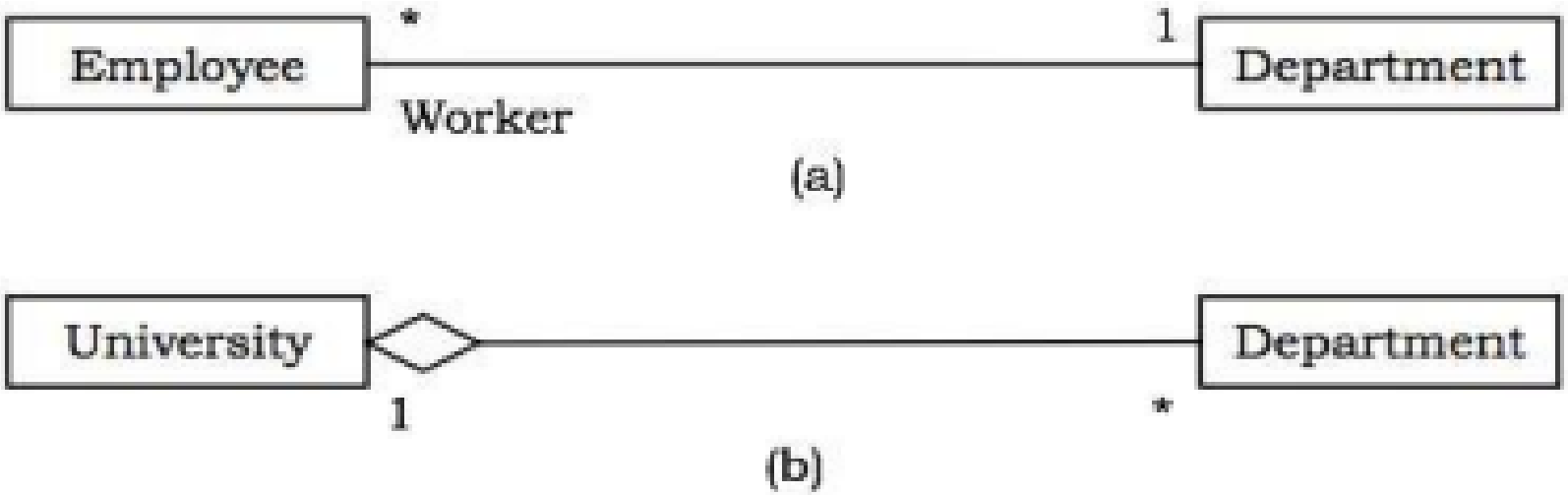
Relationship: *the notion for the different types of relationships are as follows*



- *Usually, elements in a relationship play specific roles in the relationship. A role name signifies the behavior of an element participating in a certain context.*

Cntd…

- *Example : The following figures show examples of different relationships between classes. The first figure shows an association between two classes, Department and Employee, wherein a* ***department may have a number of employees working in it****. Worker is the role name. The '1' alongside Department and '*' alongside Employee depict that the cardinality ratio is one–to– many.* ***The second figure represents the aggregation relationship, a University is the "whole– of" many Departments***



UML structured diagrams

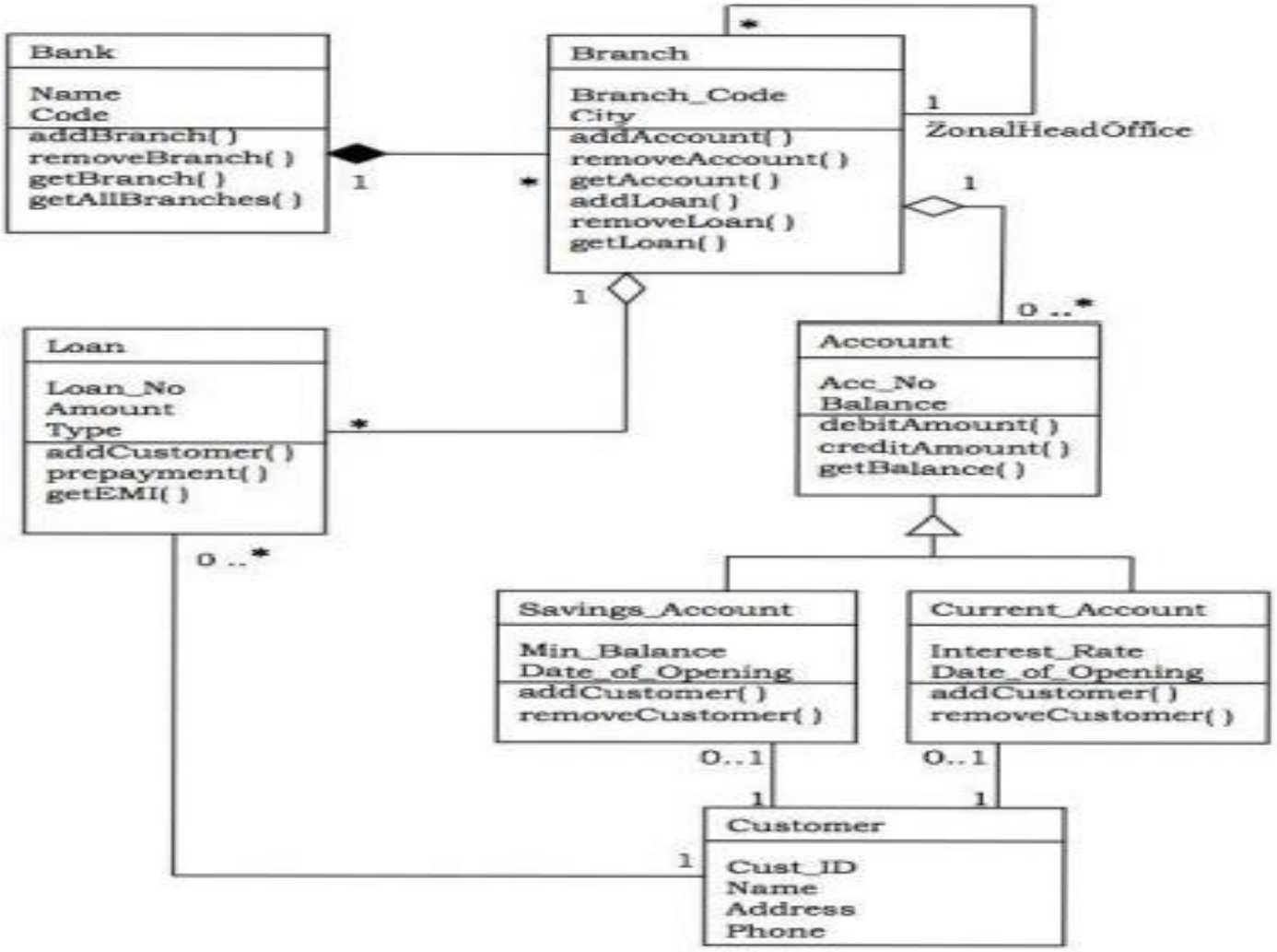
- ***UML structural diagrams*** *are categorized as follows: class diagram,* ***object diagram,*** ***component diagram****, and deployment diagram.*

I. Class Diagram

- *A class diagram models the* ***static view of a system.***
- *It includes of the classes, interfaces, and collaborations of a system; and the relationships between them.*
- *E.g., bank system.*

A bank has many branches. In each zone, one branch is designated as the zonal head office that supervises the other branches in that zone. Each branch can have multiple accounts and loans. An account may be either a savings account or a current account. A customer may open both a savings account and a current account. Etc.

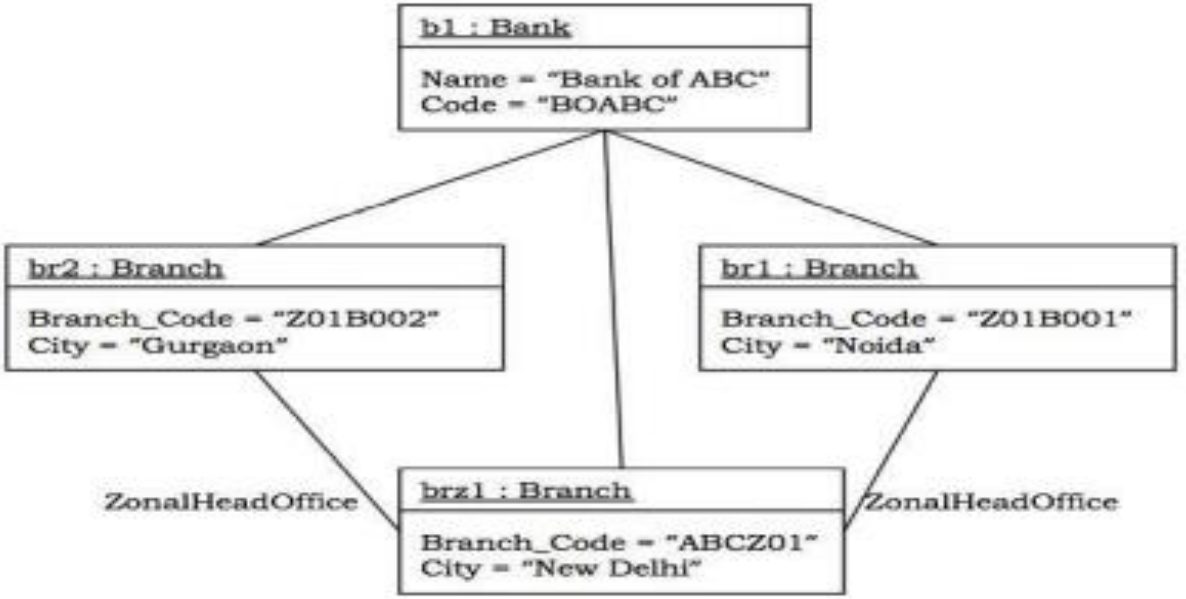
Cntd…


Bank
Name
Code
addBranch( )
removeBranch( )
getBranch( )
getAllBranches( )
Branch
Branch_Code
City
addAccount( )
removeAccount( )
getAccount( )
addLoan( )
removeLoan( )
getLoan( )
ZonalHeadOffice
Loan
Loan_No
Amount
Type
addCustomer( )
prepayment( )
getEMI( )
Account
Acc_No
Balance
debitAmount( )
creditAmount( )
getBalance( )
Savings_Account
Min_Balance
Date_of_Opening
addCustomer( )
removeCustomer( )
Current_Account
Interest_Rate
Date_of_Opening
addCustomer( )
removeCustomer( )
Customer
Cust_ID
Name
Address
Phone
1
*
0..*
0..1

Cntd…

2. Object Diagram: *An object diagram models a group of objects and their links at a point of time.*

- *It shows the* ***instances of the things in a class diagram.***
- *Object diagram is the* ***static part of an interaction diagram.***
- *Example : The following figure shows an object diagram of a portion of the class diagram of the Banking System*



3. Component Diagram: ***it show the organization and dependencies among a group of components.***

- *Component diagrams comprise of:*
  - *Components*
  - *Interfaces*
  - *Relationships*
  - *Packages and Subsystems (optional) Component diagrams are used for:*
  - *Constructing systems through forward and reverse engineering.*
  - *Modeling configuration management of source code files while developing a system using an object-oriented programming language.*
  - *Representing schemas in modeling databases.*
  - *Modeling behaviors of dynamic systems.*

4. Deployment Diagram: *it puts emphasis on the* ***configuration of runtime processing*** *nodes and their components that live on them.*

- *They are commonly comprised of nodes and dependencies, or associations between the nodes.*
- ***Deployment diagrams are used to:***
  - *Model devices in embedded systems that typically comprise of software-intensive collection of hardware.*
  - *Represent the topologies of client/server systems.*
  - *Model fully distributed systems.*

UML Behavioral Diagrams

- UML behavioral diagrams visualize, specify, construct, and document the **dynamic aspects of a system**.
- The behavioral diagrams are categorized as follows: **use case diagrams**, interaction diagrams, **state–chart diagrams**, and **activity diagrams**.

Use Case Model

- **Use Case:** A use case describes the sequence of actions a system performs yielding visible results.
- It shows the interaction of things outside the system with the system itself. Use cases may be applied to the whole system as well as a part of the system.
- **Actor:** An actor represents the roles that the users of the use cases play. An actor may be a person (e.g. student, customer), a device (e.g. workstation), or another.

Cntd…

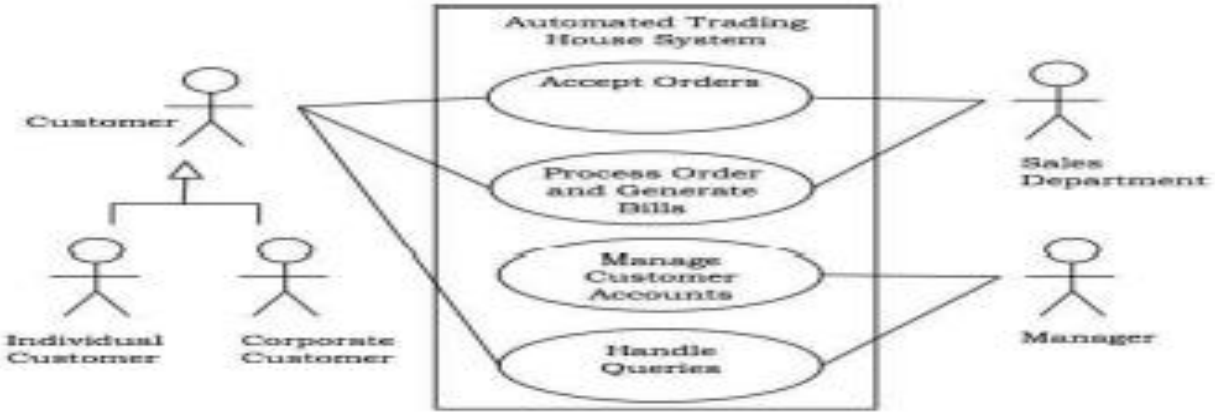
- *The following figure shows the notations of an actor named Student and a use case called Generate Performance Report*

5. Use Case Diagrams

- Use case diagrams present an outside view of the manner the elements in a system behave and how they can be used in the context.
- Use case diagrams comprise of:
  - Use cases
  - Actors
  - Relationships like dependency, generalization, and association Use case diagrams are used:
- Model the context of a system by enclosing all the activities of a system within a rectangle and focusing on the actors outside the system by interacting with it.
- To model the requirements of a system from the outside point of view

Cntd…

- *E.g., Let us consider an Automated Trading House System. We assume the following features of the system:*
  - *The trading house has transactions with two types of customers, individual customers and corporate customers.*
  - *Once the customer places an order, it is processed by the sales department and the customer is given the bill.*
  - *The system allows the manager to manage customer accounts and answer any queries posted by the customer.*



Cntd…

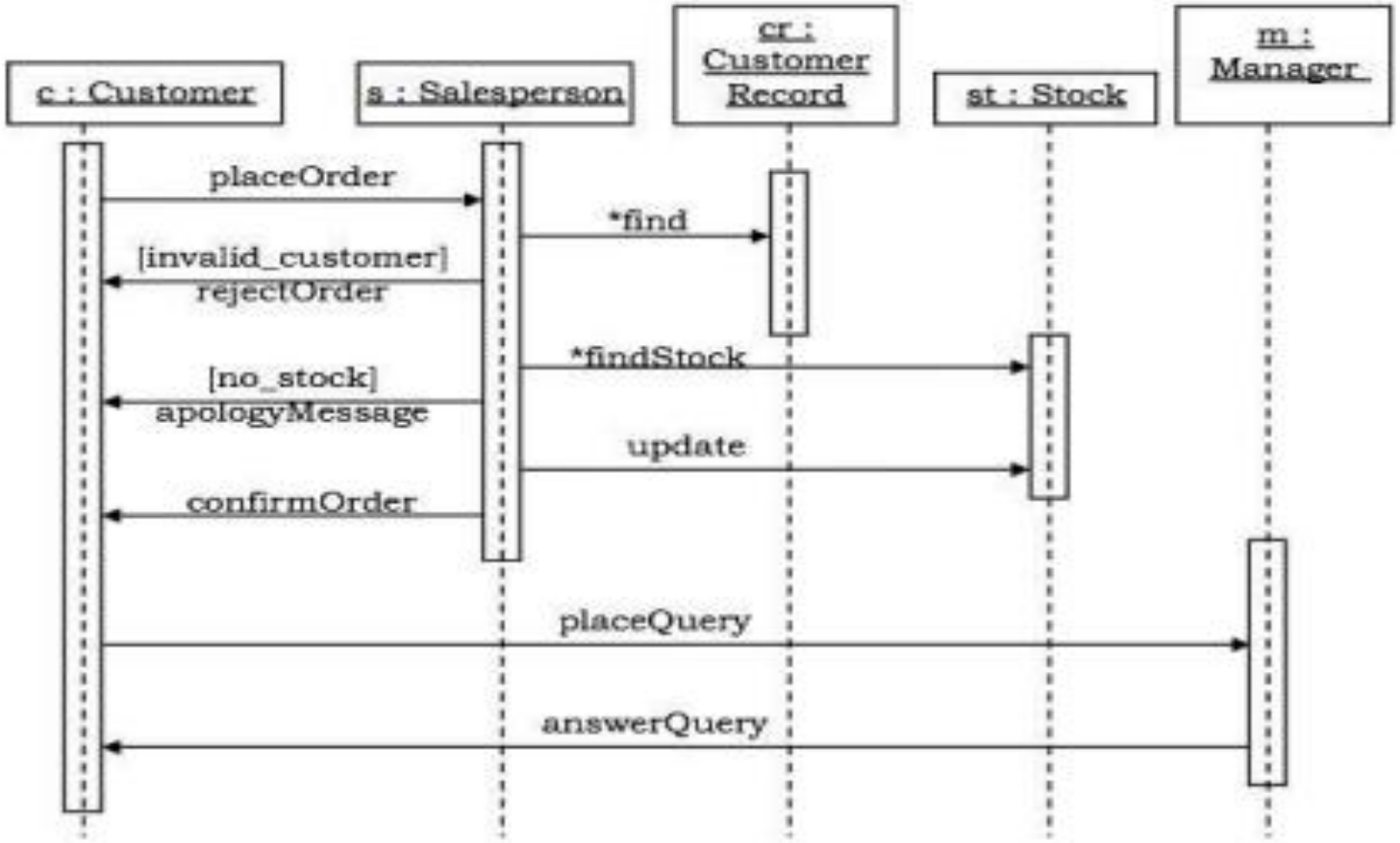
- ***Interaction Diagrams:*** *it depict interactions of objects and their relationships.*
- *They also include the messages passed between them. There are two types of interaction diagrams:*
  - *Sequence Diagrams and Collaboration Diagrams*
- ***Interaction diagrams are used for modeling:***
  - *the control flow by time ordering using sequence diagrams.*
  - *the control flow of organization using collaboration diagrams.*

6. Sequence Diagrams: *are interaction diagrams that illustrate the ordering of messages according to time.*

- ***Notations:*** *These diagrams are in the form of two-dimensional charts. The objects that initiate the interaction are placed on the* ***x–axis****.*
- *The messages that these objects send and receive are placed along the* ***y-axis,*** *in the order of increasing time from top to bottom*

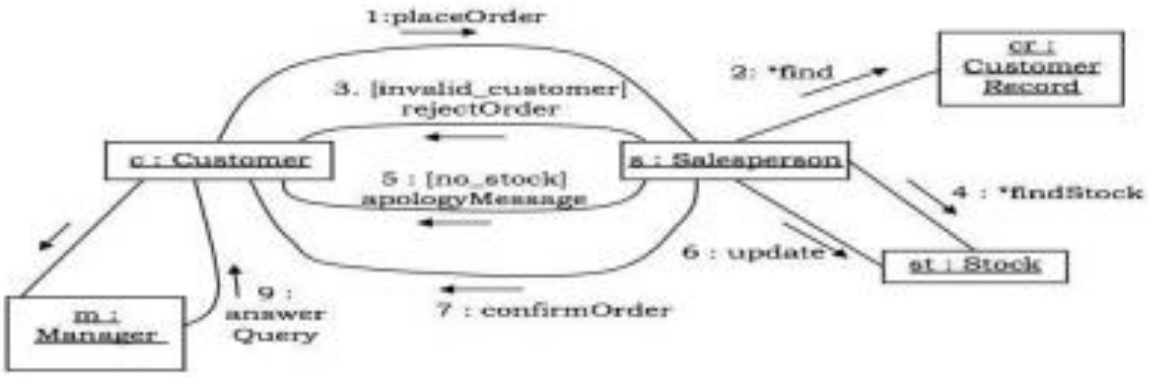
Cntd…

E.g., A sequence diagram for the Automated Trading House System is shown in the following figure.



7. Collaboration Diagrams *are interaction diagrams that illustrate the structure of the* ***objects*** *that send and receive messages.*

- ***Notations :*** *In these diagrams, the objects that participate in the interaction are shown using vertices.*
- *The links that connect the objects are used to send and receive messages. The message is shown as a labeled arrow.*
- *Example: Collaboration diagram for the Automated Trading House System is illustrated*

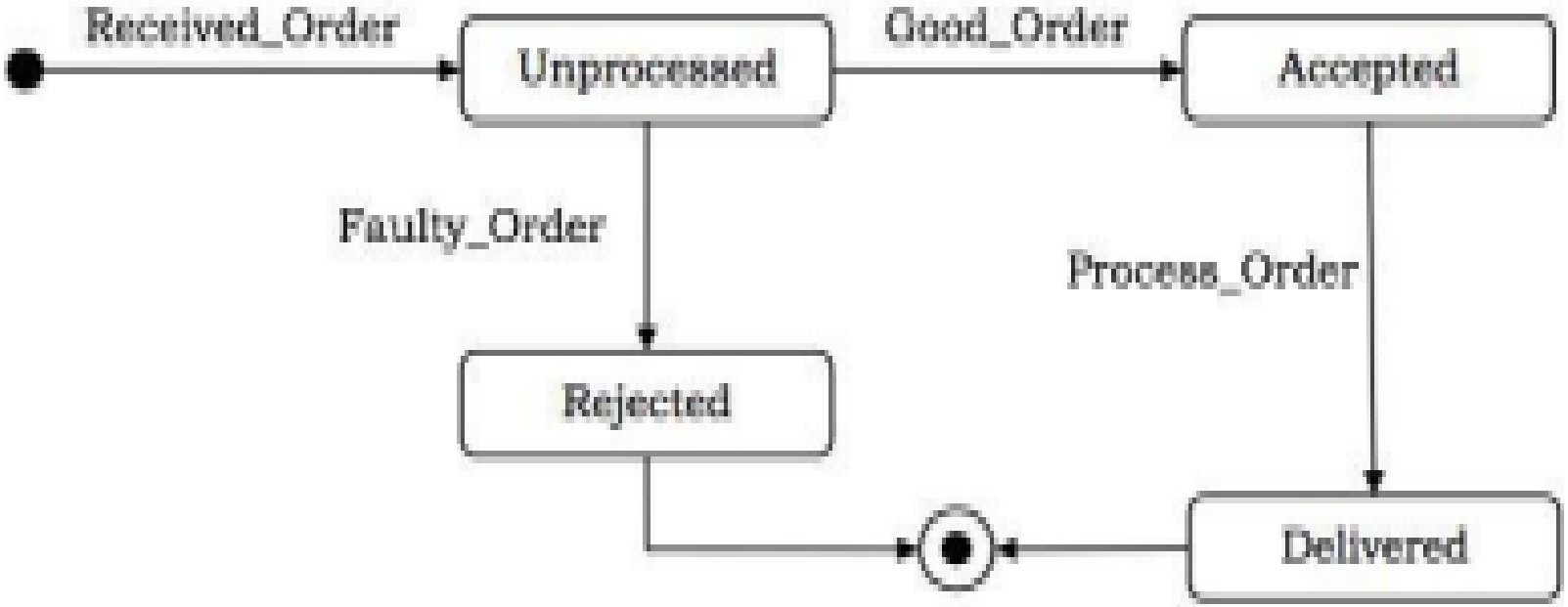


8. State–Chart Diagrams

- *A state–chart diagram shows a state machine that depicts the control flow of an object from one state to another.*
- ***A state machine depicts the sequences of states which an object undergoes due to events and their responses to events.***
- *State–Chart Diagrams include of:*
  - ***States:*** *Simple or Composite*
  - *Transitions between states*
  - *Events causing transitions*
  - *Actions due to the events*
- ***State-chart diagrams*** *are used for* ***modeling objects which are reactive in nature***

Cntd…

- *Example, In the Automated Trading House System, let us model Order as an object and trace its sequence. The following figure shows the corresponding state–chart diagram.*

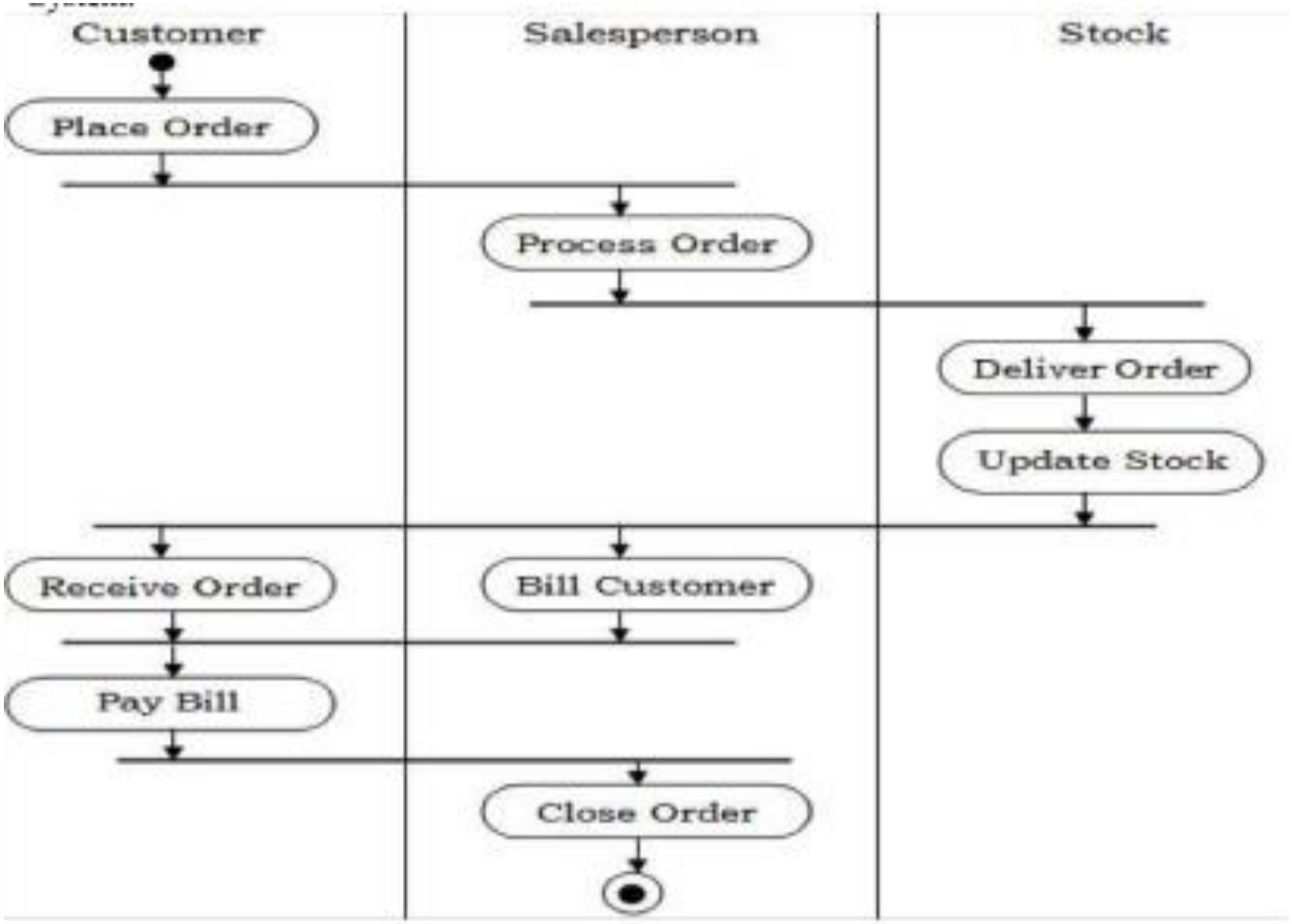


9. Activity Diagrams: *An activity diagram depicts the flow of activities which are ongoing non-atomic operations in a state machine.*

- *Activities result in actions which are atomic operations.*
- *Activity diagrams include of:*
  - *Activity states and action states*
  - *Transitions*
  - *Objects*
- *Activity diagrams are used for modeling:*
  - *Workflows as viewed by actors, interacting with the system.*
  - *details of operations or computations using flowcharts.*

Cntd…

Example, The following figure shows an activity diagram of a portion of the Automated Trading House System.



End of chapter Eight

Any Question?