JAIN DEEMED-TO-BE UNIVERSITY | SCHOOL OF COMPUTER SCIENCE AND IT

School of Computer Science & IT

Department of BCA

SOFTWARE ENGINEERING (22BCA3C01)

MODULE 2: Software Requirements

Session -13

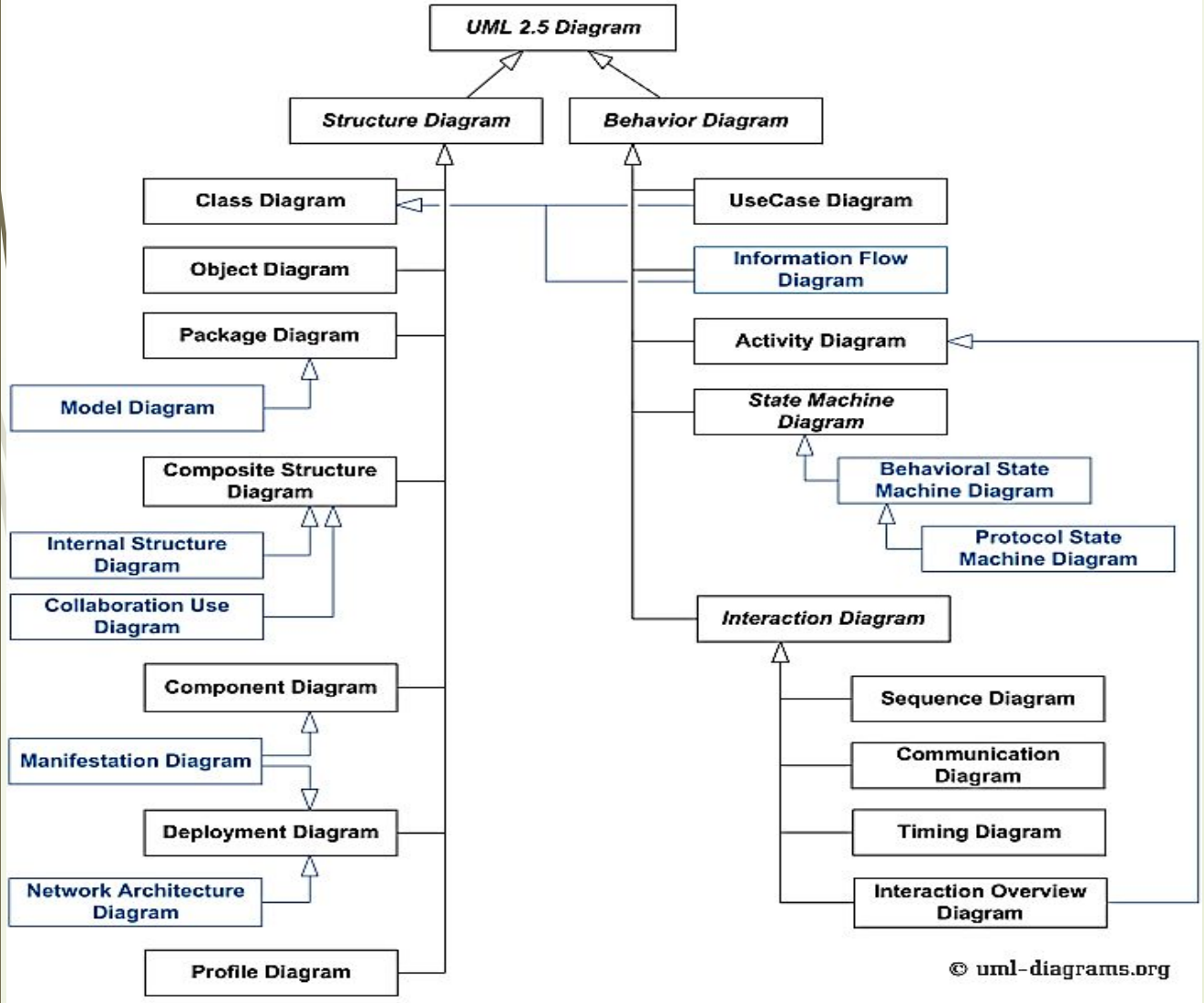
Requirements Modeling

- Building the Analysis Model
- Scenario-based Models

Requirements Modelling Approaches

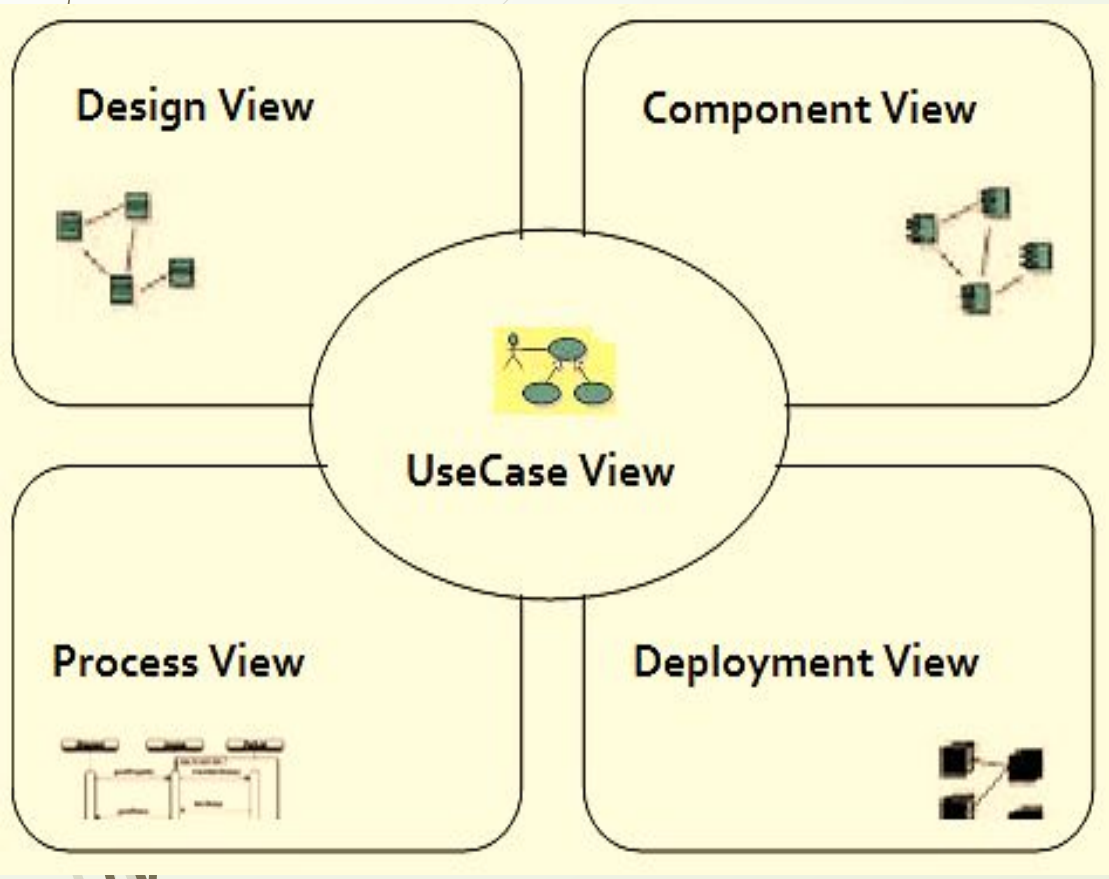
- Structured Analysis
  - One view of requirements modeling, called *structured analysis,* considers data and the processes that transform the data as separate entities.
  - Data Modeling: Data objects are modeled in a way that defines their attributes and relationships. (e.g. E-R Model)
  - Process Modeling: Processes that manipulate data objects are modeled in a manner that shows how they transform data as data objects flow through the system. (e.g. Data Flow Diagram)
- Object-Oriented Analysis
  - A second approach to analysis modeling, called *object-oriented analysis,* focuses on the definition of classes and the manner in which they collaborate with one another to effect software requirements.
  - UML (Unified Modeling Language) models are predominantly object oriented.

The Unified Modeling Language (UML)



- The UML is a general-purpose standard visual modeling language in software engineering used to specify, visualize, construct, and document software system.
- The UML has been standardized by Object Management Group (OMG) in 1997 and periodically revise it. The current version is UML 2.5.1 released in 2017.
- UML specification defines two major kinds of UML diagram: **structure diagrams** and **behavior diagrams**.
- **Structure diagrams** show the **static structure** of the system and its parts on different abstraction and implementation **levels** and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.
- **Behavior diagrams** show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time**.

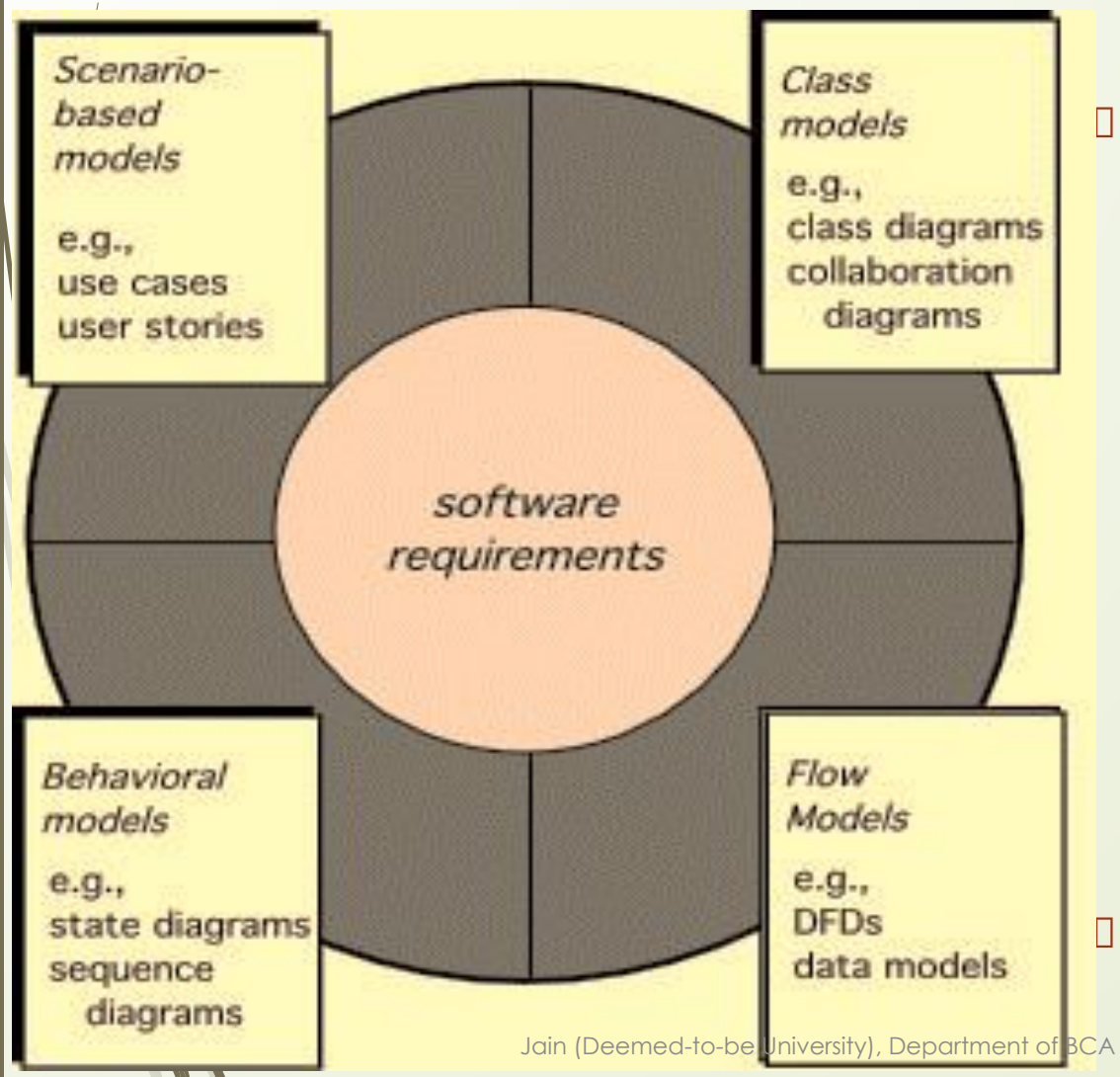
4+1 View of the System



- **Use case View:** Use case diagrams are used to view a system as a set of discrete activities or functions.
- **Process View:** The dynamic behavior of a system can be seen using the process view. The different diagrams such as the state chart diagram, activity diagram, sequence diagram, and collaboration diagram are used in this view.
- **Design View:** The design view of a system is the structural view of the system. This gives an idea of what a given system is made up of. Class diagrams and object diagrams form the design view of the system.
- **Component View:** The component view shows the grouped modules of a given system modeled using the component diagram.
- **Deployment View:** The deployment diagram of UML is used to identify the deployment modules for a given system.

UML Tools: Visual Paradigm, Enterprise Architect, Star UML, Lucid Chart, Magic Draw, Visio, Creately

Building Requirements Analysis Models

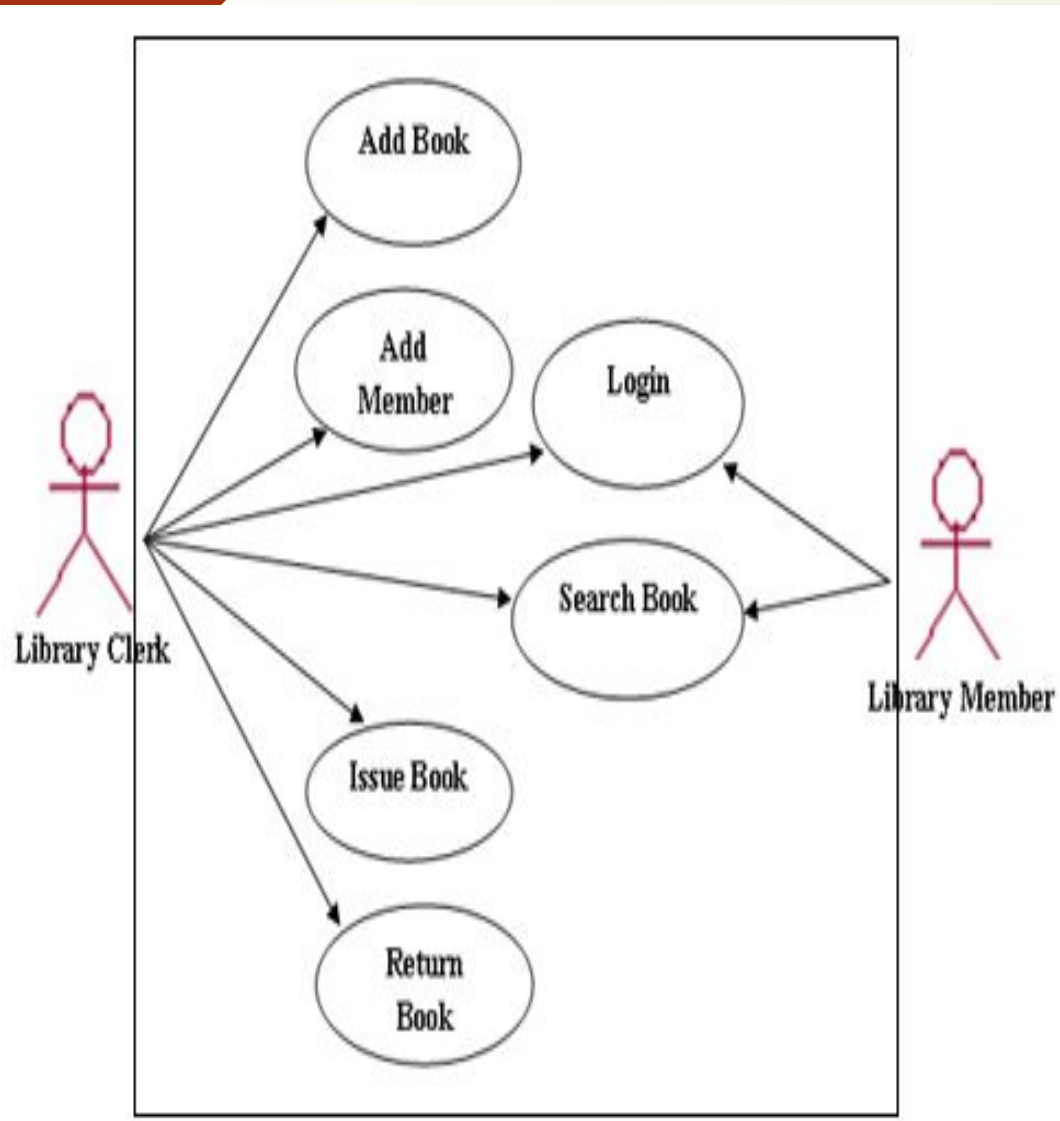


- Requirements analysis results in the specification of software's operational characteristics, indicates software's interface with other system elements, and establishes constraints that software must meet.
- Requirements analysis allows a software engineer to elaborate on basic requirements established during requirements engineering process.
- Elements of the analysis model
  - Scenario-based Models
    - Provide processing narratives for software functions
    - Use-case Model, Activity Diagram
  - Class-based Models
    - Represent object-oriented classes and the manner in which classes collaborate to achieve system requirements.
  - Behavioral Models
    - Depict how the software behaves as a consequence of external events.
    - State diagram, Sequence diagram
  - Flow-oriented Models
    - Data flow diagram
  - Data Models that depict the information domain for the system: E-R diagram
- These models provide a software designer with information that can be translated to architectural design, interface designs, and component-level designs.

SCENARIO-BASED MODELING

- Although the success of a computer-based system or product is measured in many ways, user satisfaction resides at the top of the list.
- If you understand how end users (and other actors) want to interact with a system, your software team will be better able to properly characterize requirements and build meaningful analysis and design models.
- Hence, requirements modeling with UML begins with the creation of scenarios in the form of use cases, activity diagrams, and swimlane diagrams.

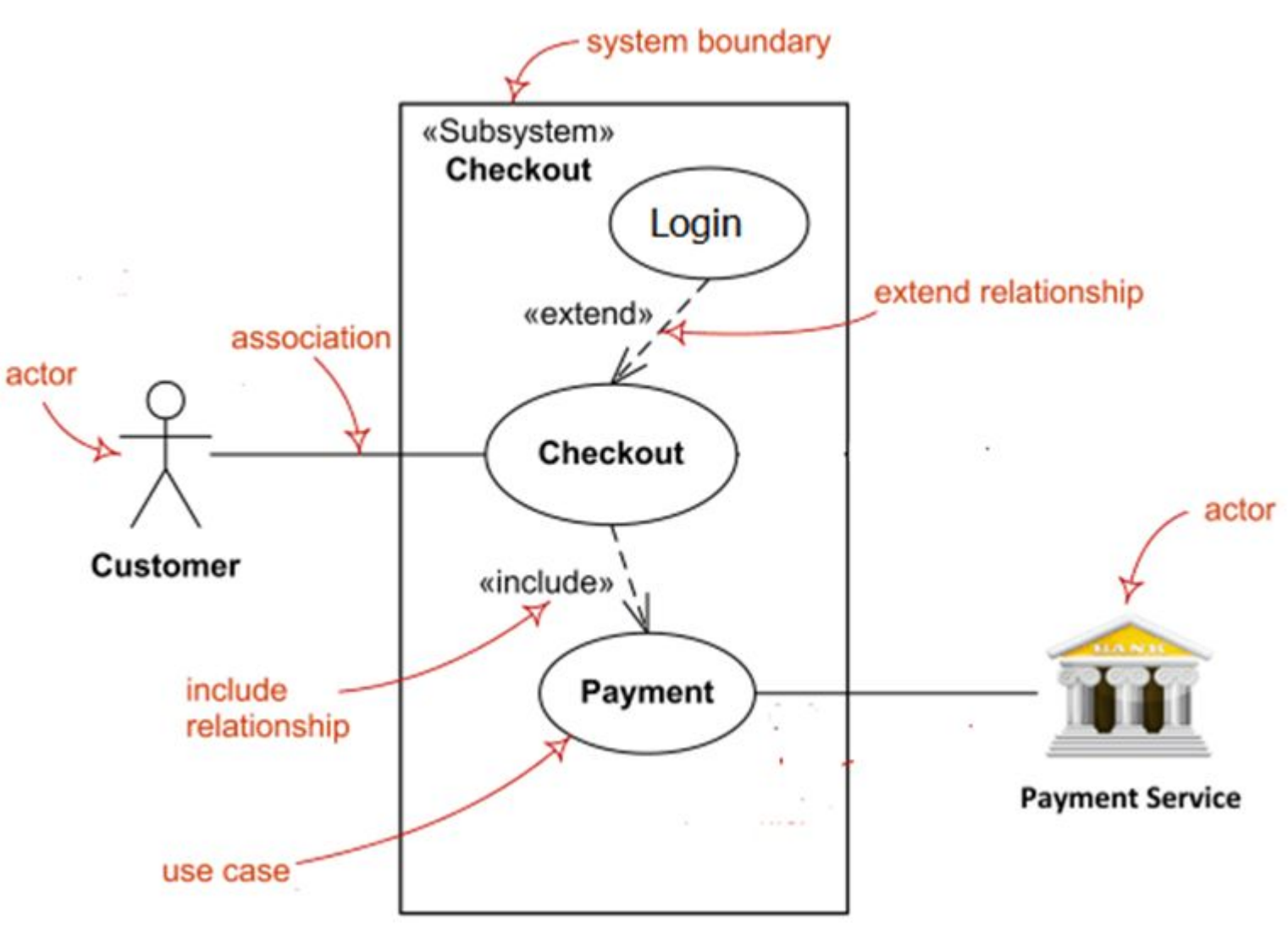
Use Case Diagram



Library Case Study for details

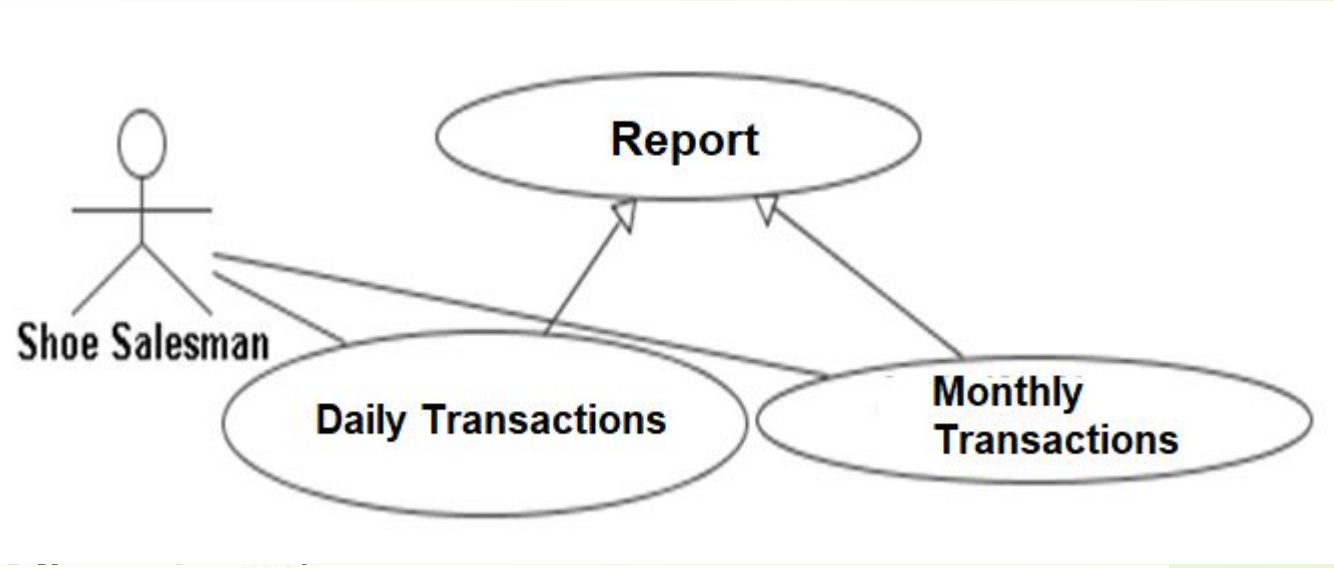
- A use-case model describes a system's functional requirements in terms of use cases. It is a model of the system's intended functionality (use cases) and external users in its environment (actors).
- UseCase Model has two parts:
  - UseCase Diagram (Graphical Part, optional)
  - UseCase Specification (Textual Part, mandatory)
- Each UseCase in a UseCase diagram represents one and only one *functionality*.
- An UseCase describes an interaction between a user (called an Actor) and the *functionality* provided by the system.
- It models the interaction as a dialogue between an actor and the system
- Actor is an external entity that interacts with the system.
- A boundary is the dividing line between the system and its environment.
- Use cases are within the boundary.
- Actors are outside of the boundary.

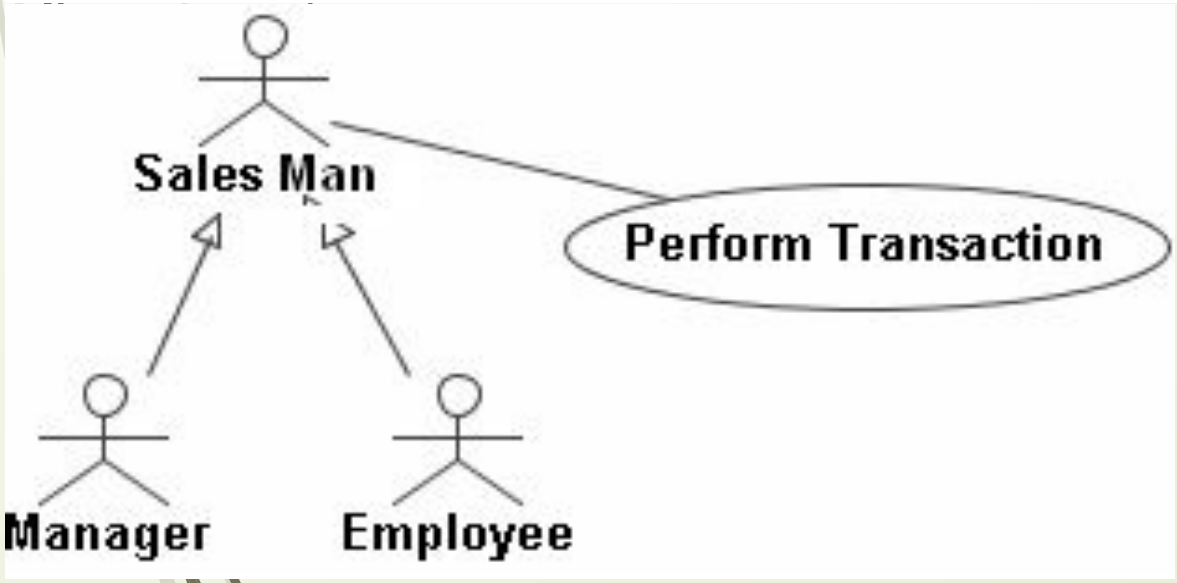
Reuse in Use Case



- Reuse in uses case model is achieved through **include and exclude use cases.**
- **Include**: Include is a directed relationship between two use cases, which implies that the behavior of the included use case is inserted into the behavior of the including use case. It is a Mandatory Relation.
- **Extend** (or Exclude): Extend is a relationship between two use cases, which specifies how and when the extended use case insert the behavior defined in the extending use case. (Normally Exceptions)
- The extend relationship **adds new incremental** behavior to a use case. Specialized use case extends the general use case. It is an Optional Relation.

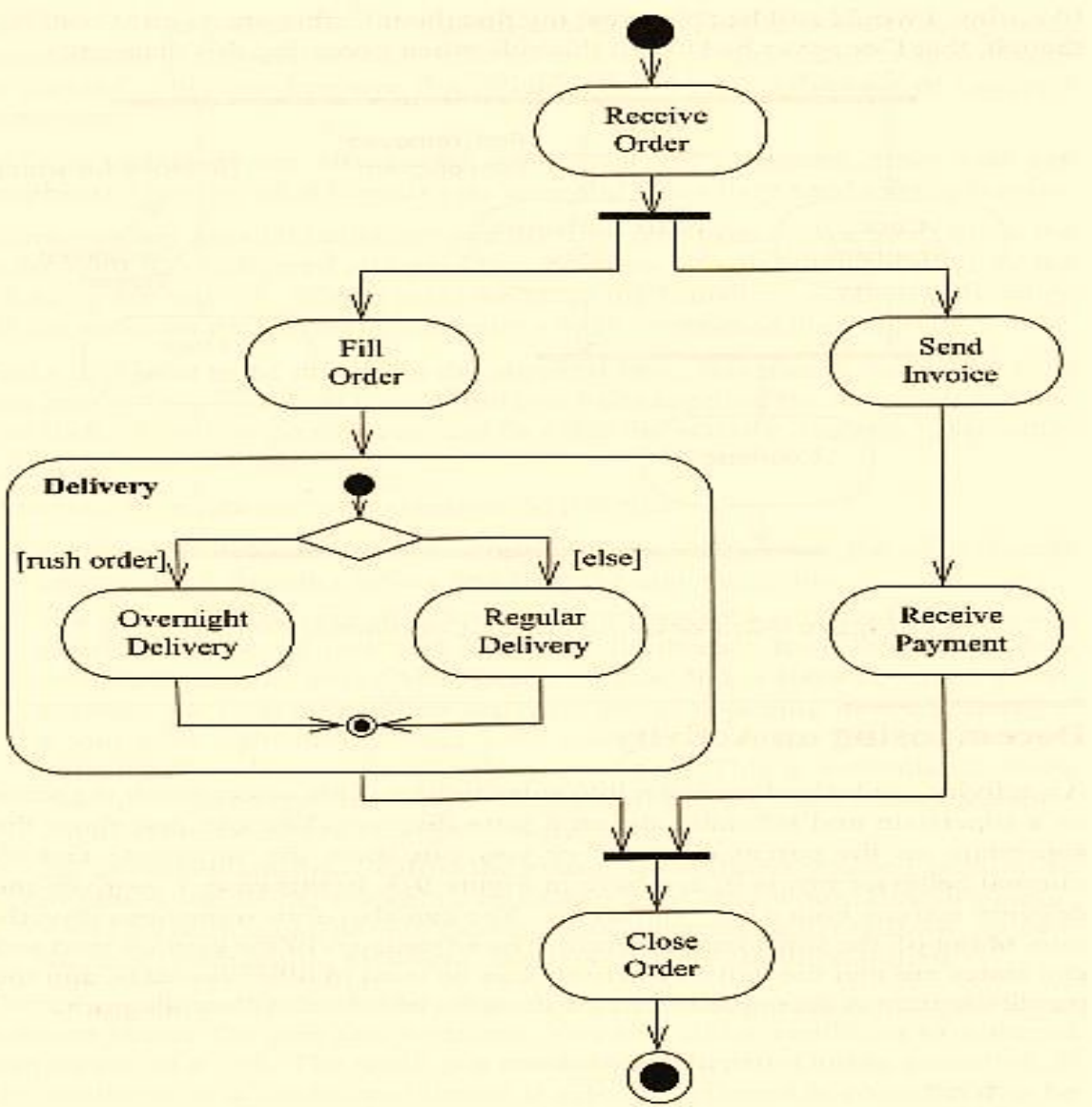
Generalization and Specialization





- Direct access to the generalized use case is avoided and replaced with direct access to its specializing use
- When a use case is associated with more than one actor, the overlapping roles between the two actors should be generalized into a separate actor.
- In library case study,:
  - Library Member: Student, Faculty, Staff

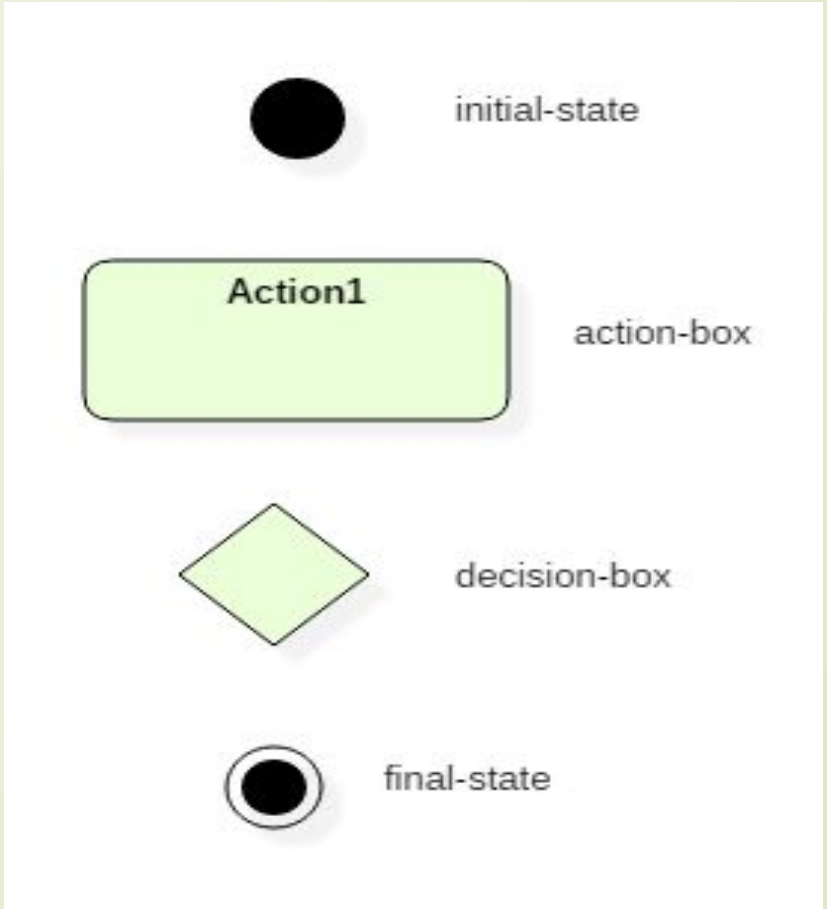
Activity Diagram



- It supplements the use case by providing a graphical representation of the flow of interaction within a specific scenario.
- It is like flow chart, but more formal and expressive (due to model elements).
- Activity Diagrams
  - model activities in one use case
  - model complex workflows in operations on objects
  - show how a collection of use cases coordinate to create a workflow for an organisation/ system

Elements of Activity Diagram

- Activities
- Transitions
- Control Nodes
  - Initial and Final Nodes
  - Forks and Joins
  - Decision and Merge Points
- Swimlanes



Elements and symbols

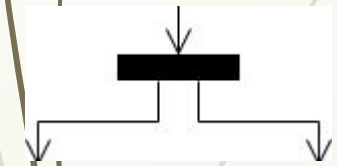
- **Activity**
  - An Activity is a unit of work that needs to be carried out
  - Any Activity takes time
  - Activities constitute the process being modelled
  - Activities are the vertices of the diagram
  - Rounded rectangle is used for Activity

- **Transition**
  - A Transition is the movement from one activity to another
  - Transitions take place when one activity is complete and the next can commence
  - Transitions are shown using unlabeled arrows from one activity to the next

Initial and Final Nodes

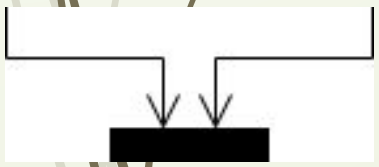
- An initial node is the starting point for an activity
- An activity final node terminates the entire activity
- It is possible to have multiple initial nodes and final nodes

Forks

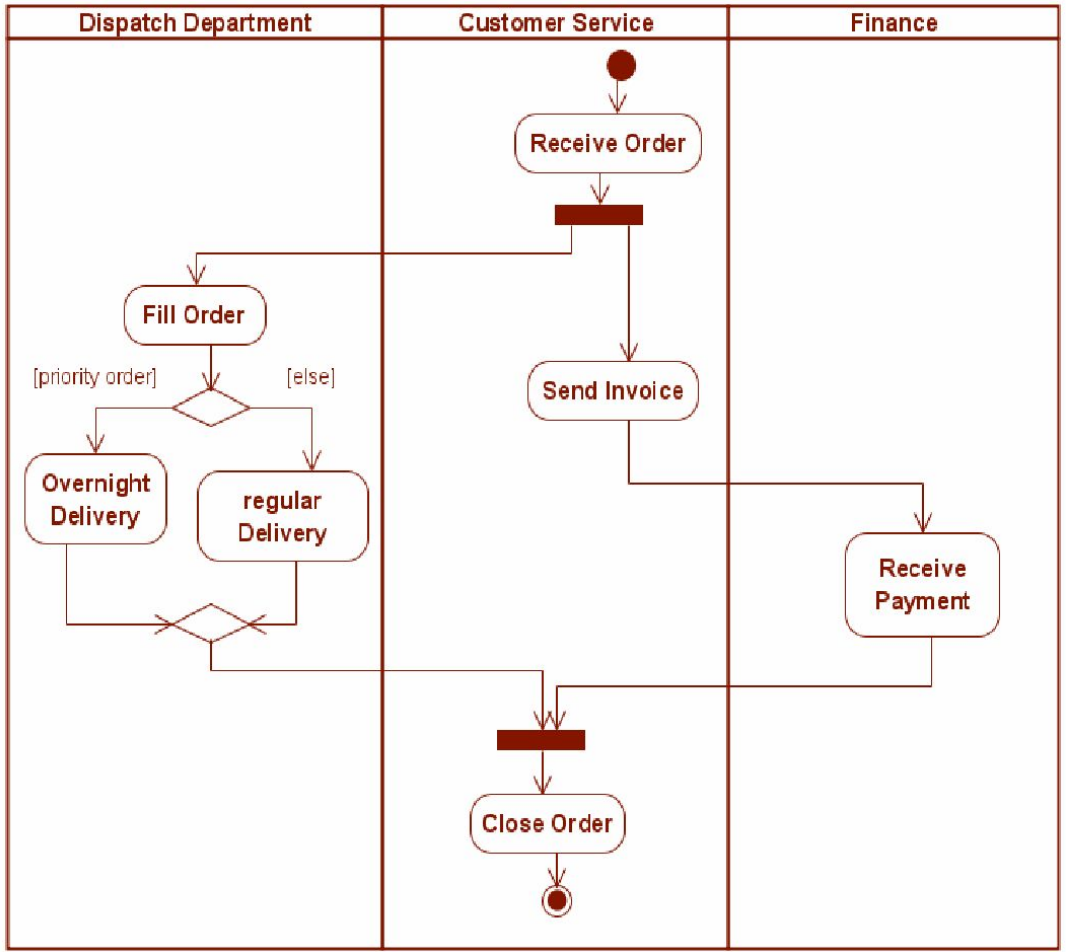


- A transition can be split into multiple paths and multiple paths combined into a single transitions by using a synchronization bar
- A synchronization may have many in-arcs from activities and a number of out-arcs to activities
- A fork is where the paths split
- A fork node splits the current flow through an activity into multiple concurrent flows.

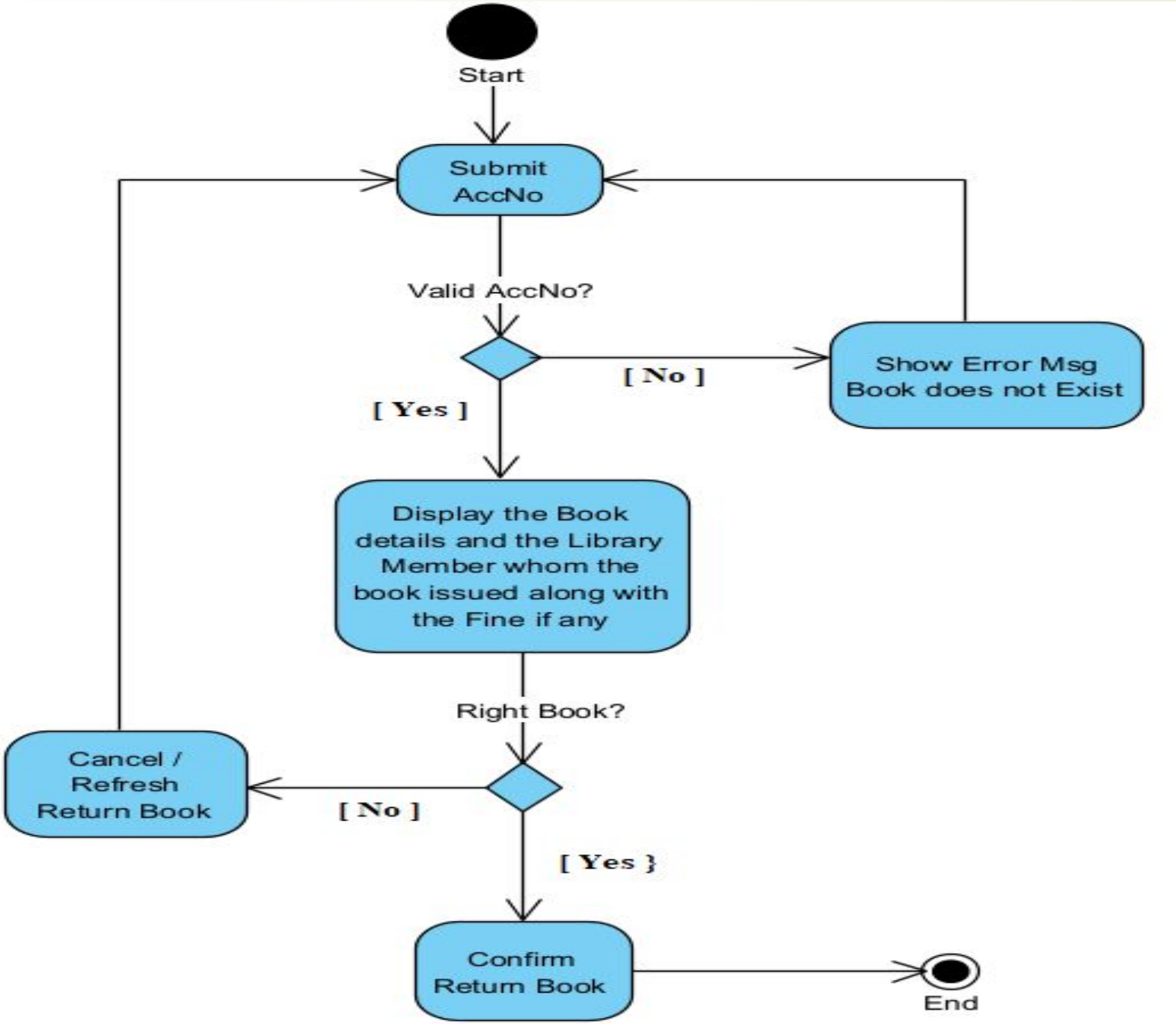
Joins



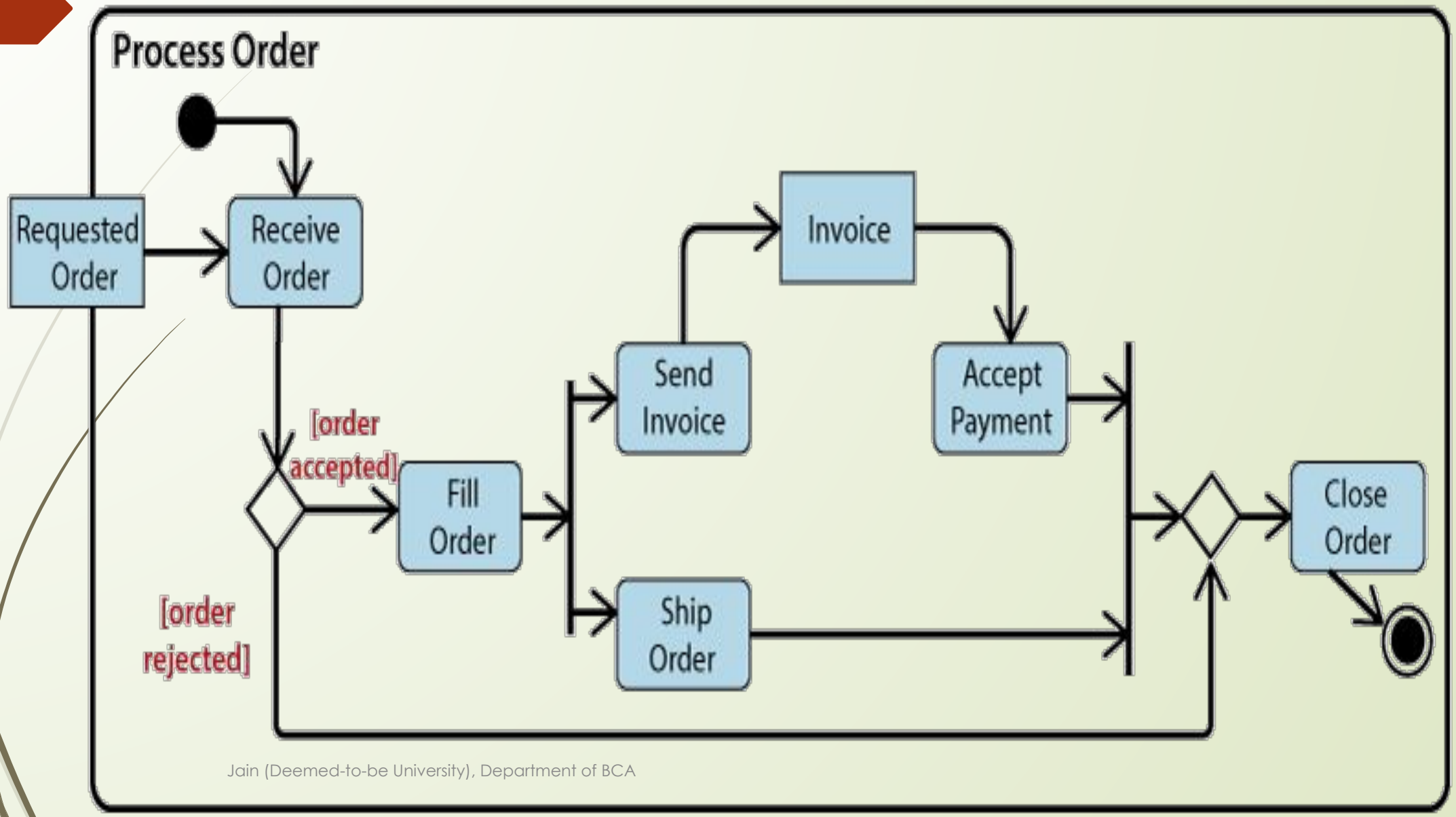
- A join is where the paths meet
- The bar represents synchronization of the completion of those activities with arcs into the transition



- Decision and Merge Points
  - A decision point shows where the exit transition from a state or activity may branch in alternative directions depending on a condition
  - A decision involves selecting one control-flow transition out of many control-flow transitions based on a condition
  - Each branched edge contains a guard condition
  - Guard expressions (inside [ ]) label the transitions coming out of a branch
- A merge point brings together alternate flows into a single output flow – note that it does not synchronize multiple concurrent flows
- Swimlanes
  - Swimlanes (or activity partitions) indicate where activities take place.
  - Swimlanes allow to partition an activity diagram so that parts of it appear in the swimlane relevant to that partition

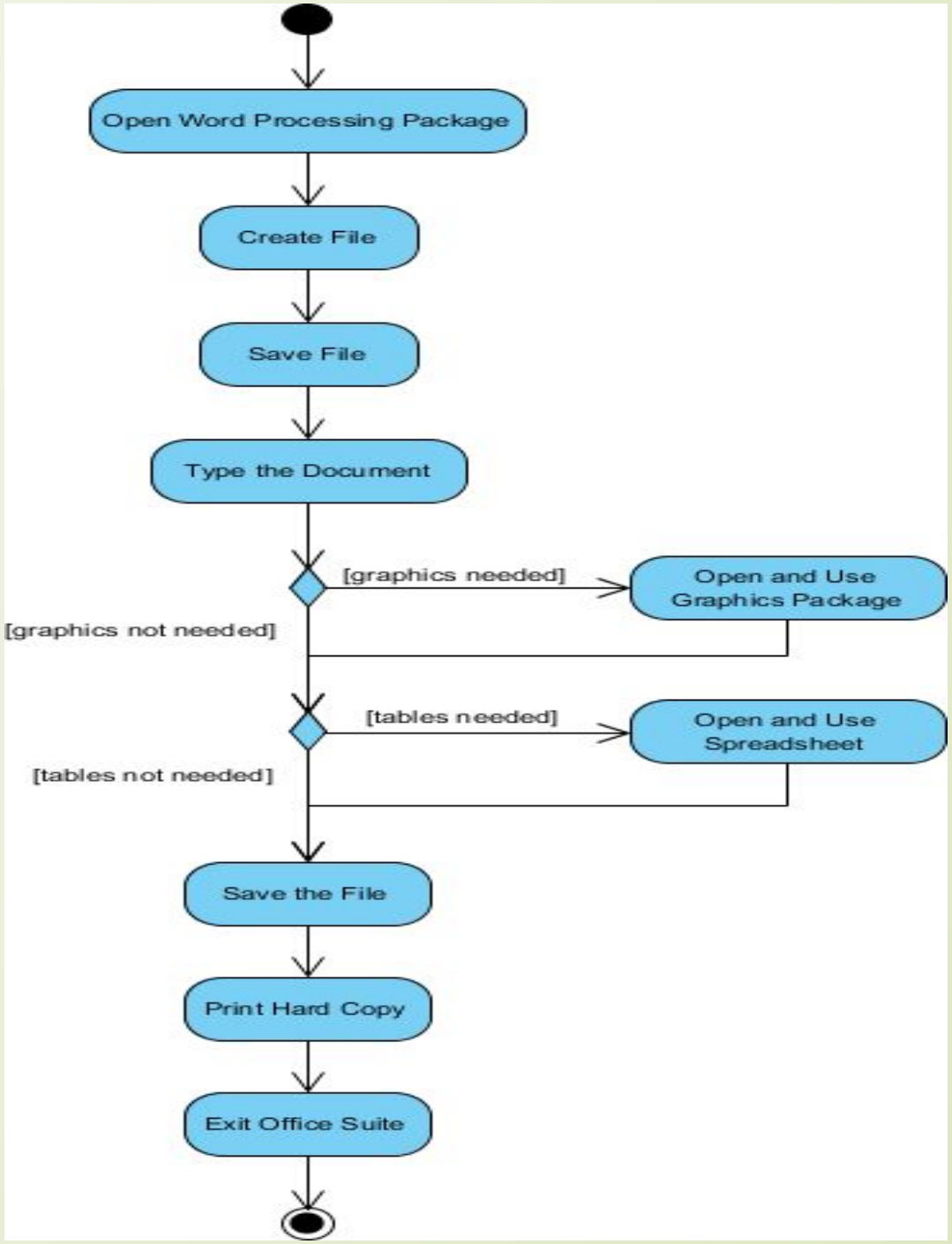


Activity Diagram of Return Book Use Case in Simple Library Management System Case Study


Process Order
Requested Order
Receive Order
Invoice
Send Invoice
Accept Payment
[order accepted]
Fill Order
Close Order
[order rejected]
Ship Order

Processing a word document activity diagram

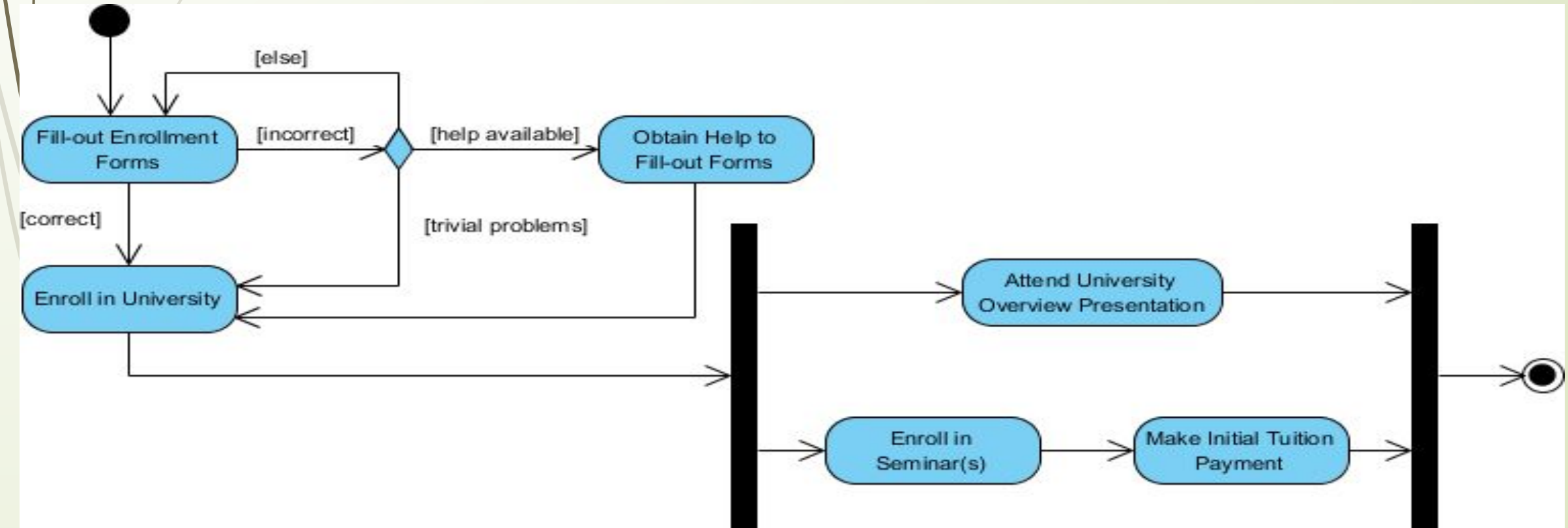
- Open the word processing package.
- Create a file.
- Save the file under a unique name within its directory.
- Type the document.
- If graphics are necessary, open the graphics package, create the graphics, and paste the graphics into the document.
- If a spreadsheet is necessary, open the spreadsheet package, create the spreadsheet, and paste the spreadsheet into the document.
- Save the file.
- Print a hard copy of the document.
- Exit the word processing package



Activity Diagram Example - Student Enrollment

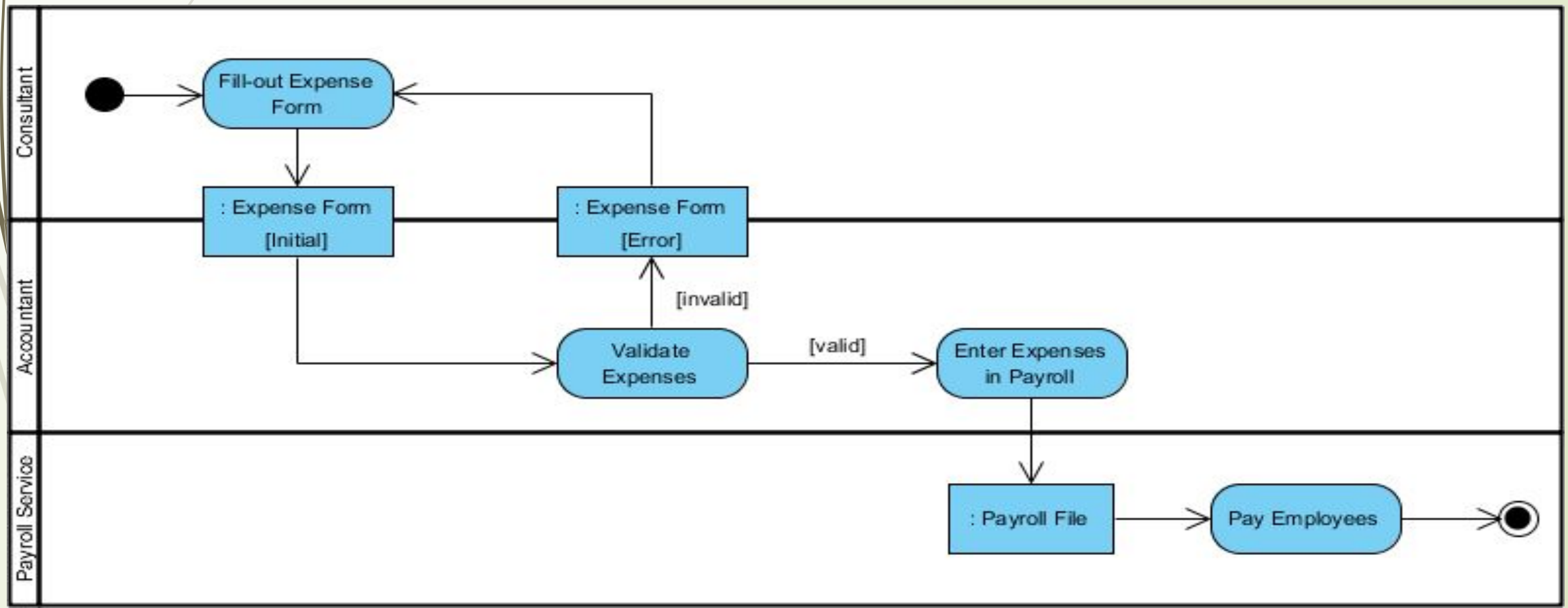
This UML activity diagram example describes a process for student enrollment in a university as follows:

- An applicant wants to enroll in the university.
- The applicant hands a filled out copy of Enrollment Form.
- The registrar inspects the forms.
- The registrar determines that the forms have been filled out properly.
- The registrar informs student to attend in university overview presentation.
- The registrar helps the student to enroll in seminars
- The registrar asks the student to pay for the initial tuition.

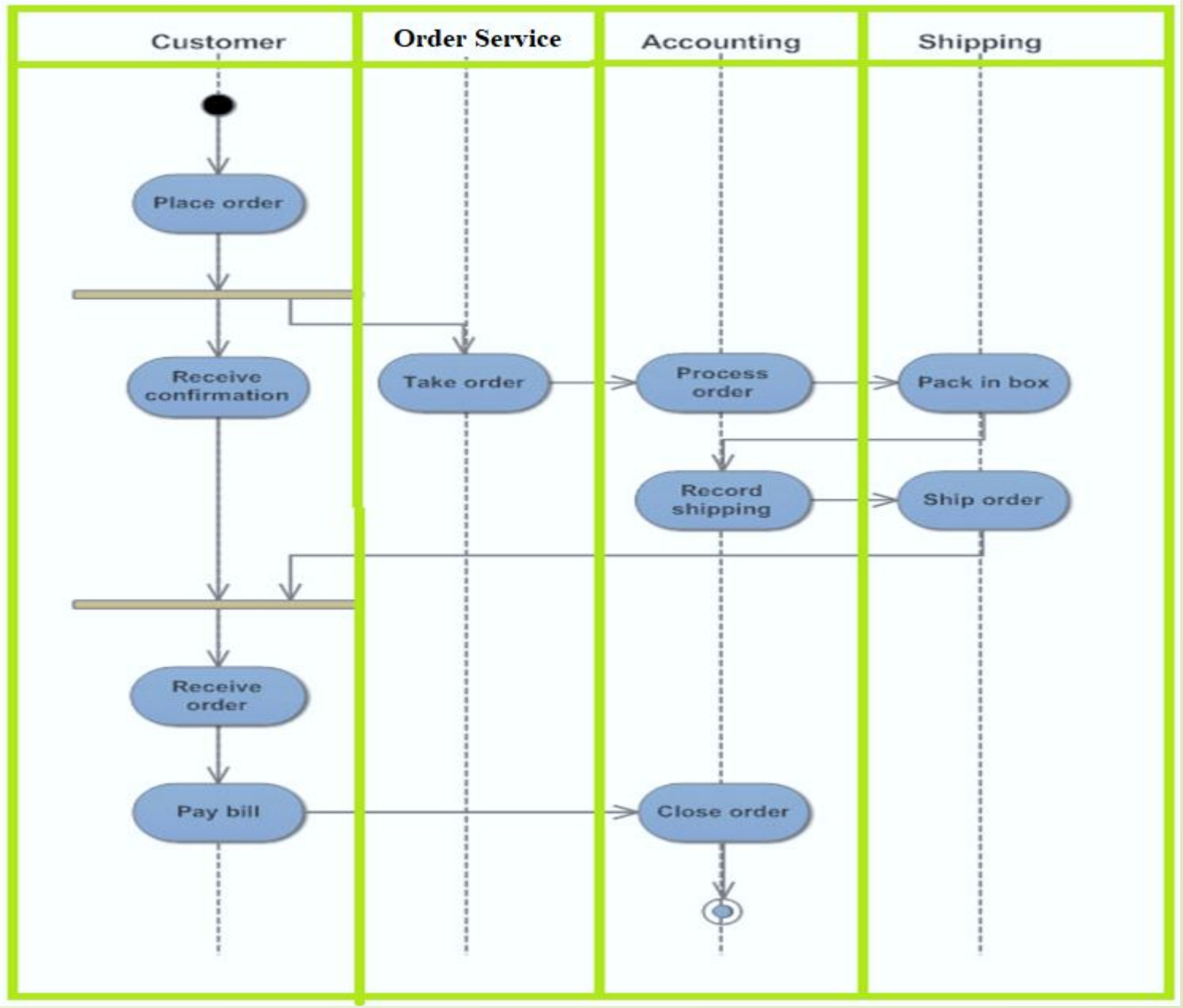


Activity Diagram - Swimlane

A swimlane is a way to group activities performed by the same actor on an activity diagram or activity diagram or to group activities in a single thread. Here is an example of a Swimlane activity diagram for modeling Staff Expenses Submission:

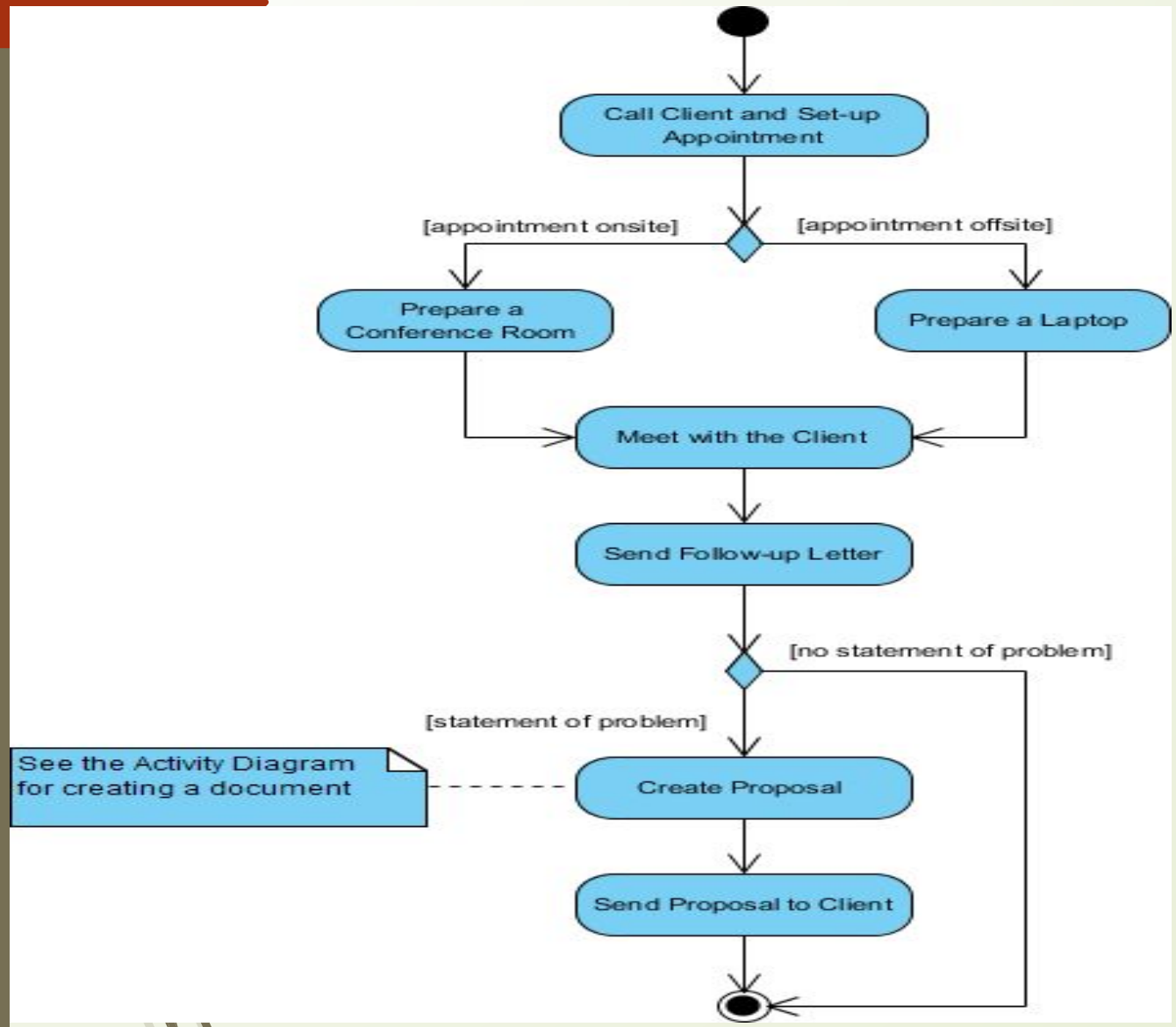


Customer Oder Processing
(with pay on delivery)

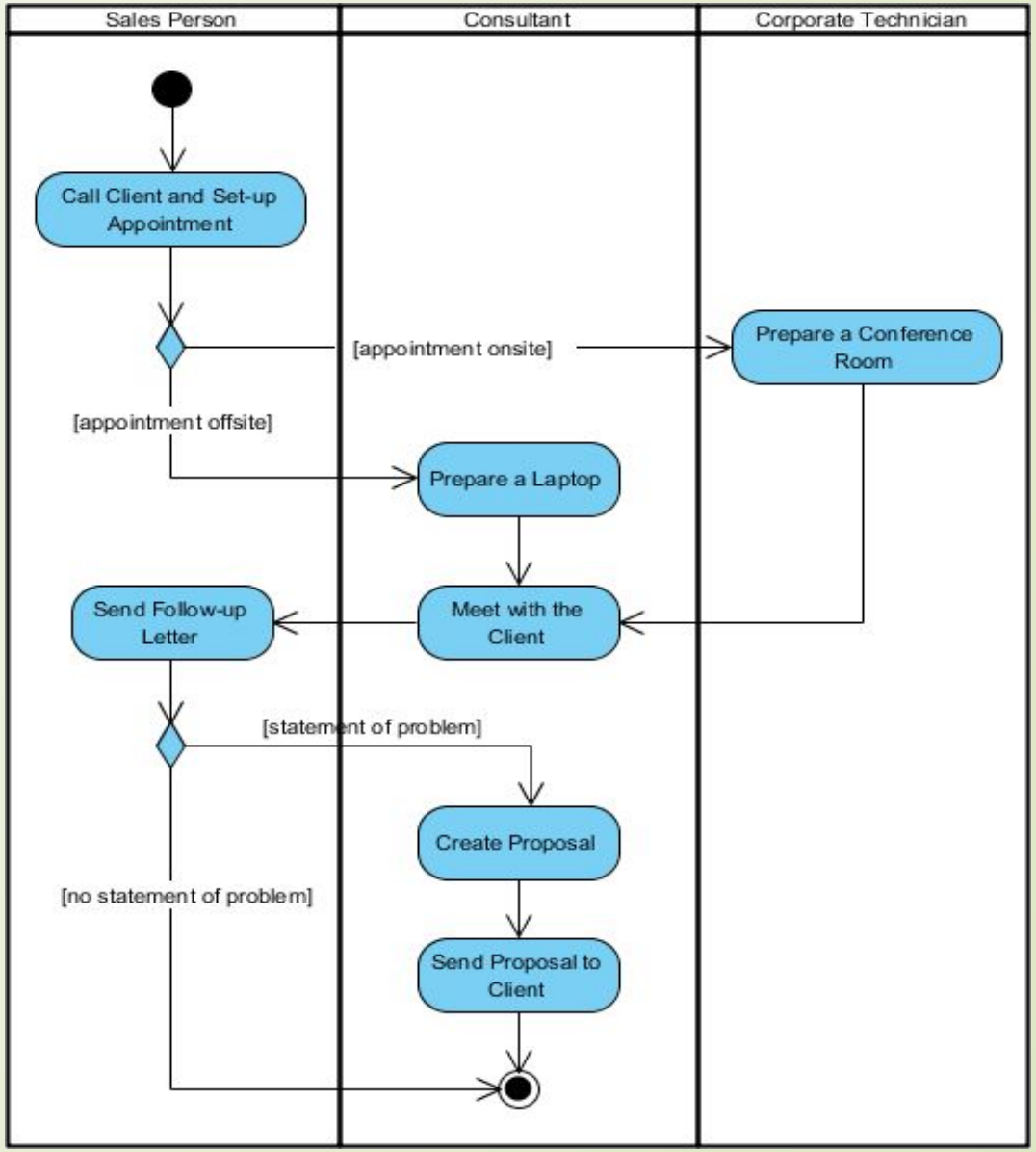


Swimlane and Non-Swimlane Activity Diagram

The activity diagram example below describes the business process for meeting a new client using an activity Diagram without Swimlane and with Swimlane.



An activity Diagram without Swimlane.



An activity Diagram with Swimlane.

Student registration process activity diagram

This activity diagram shows the series of actions performed by the student, the student registration system and the registration division to complete the student registration process.