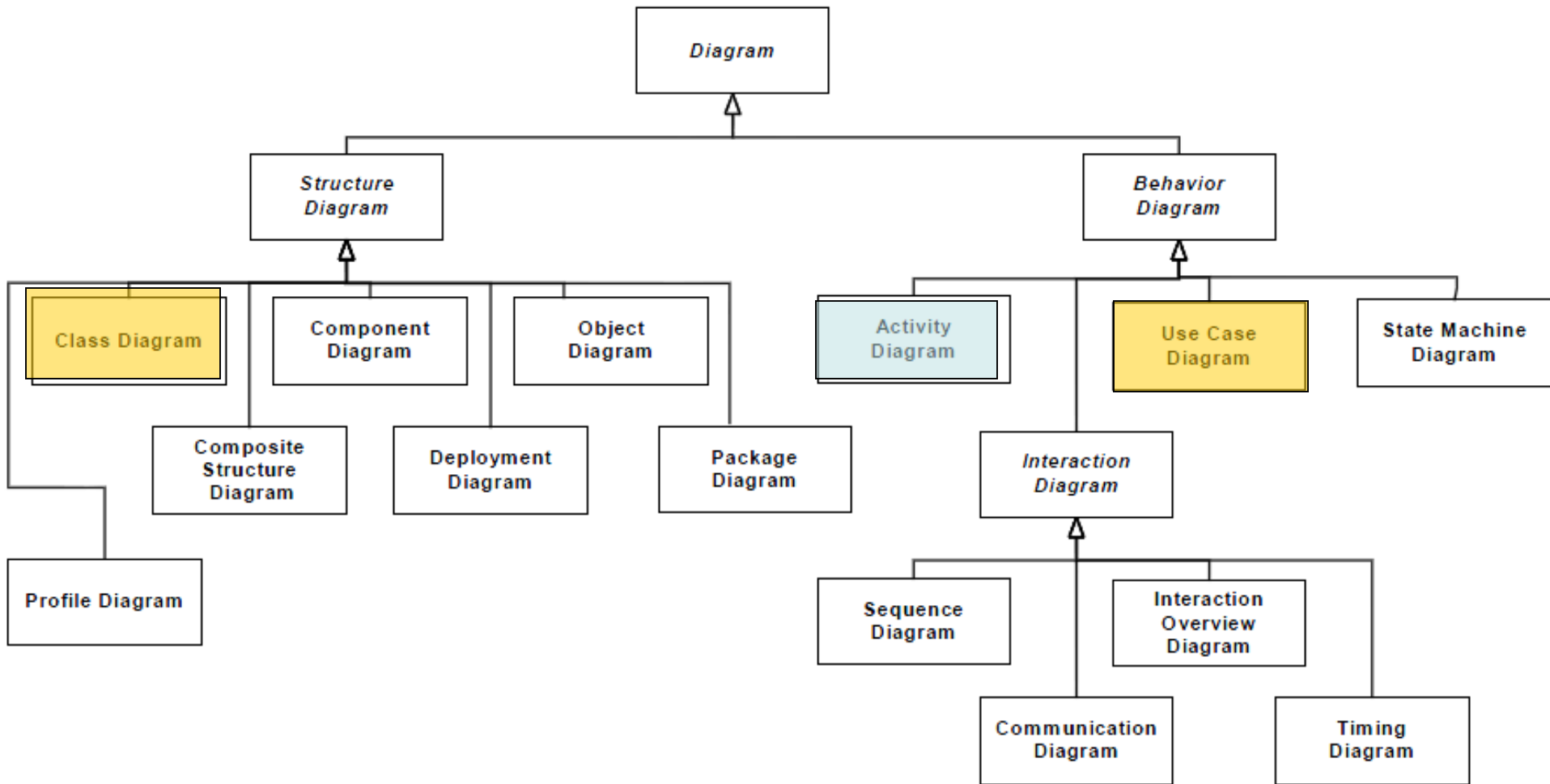


# Object-Oriented Technology and UML Activity Diagram

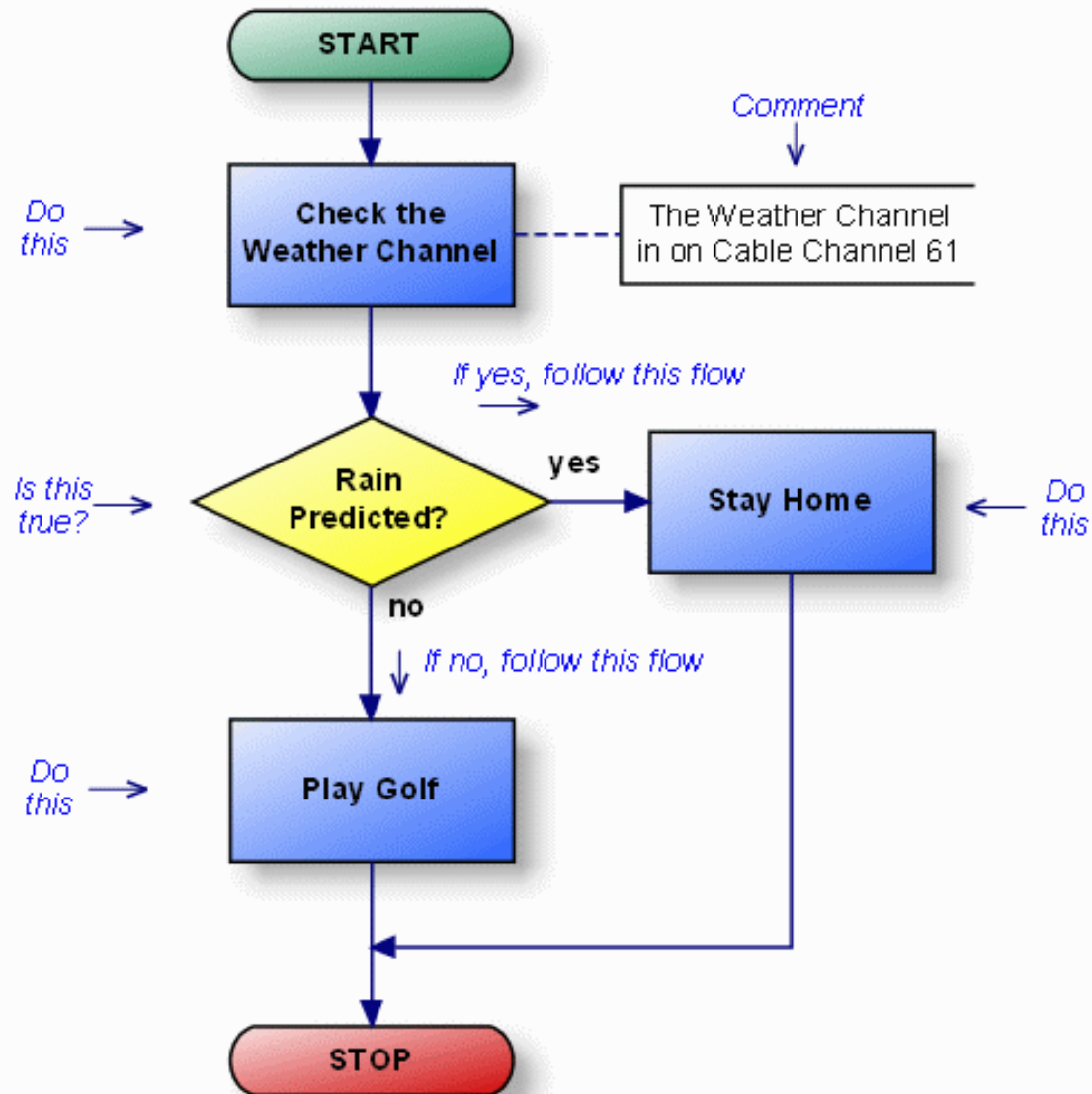
# Diagrams of UML2.X



# Topics

- Basic concepts
- Constituent elements
- Representation
- Reading method
- Modeling

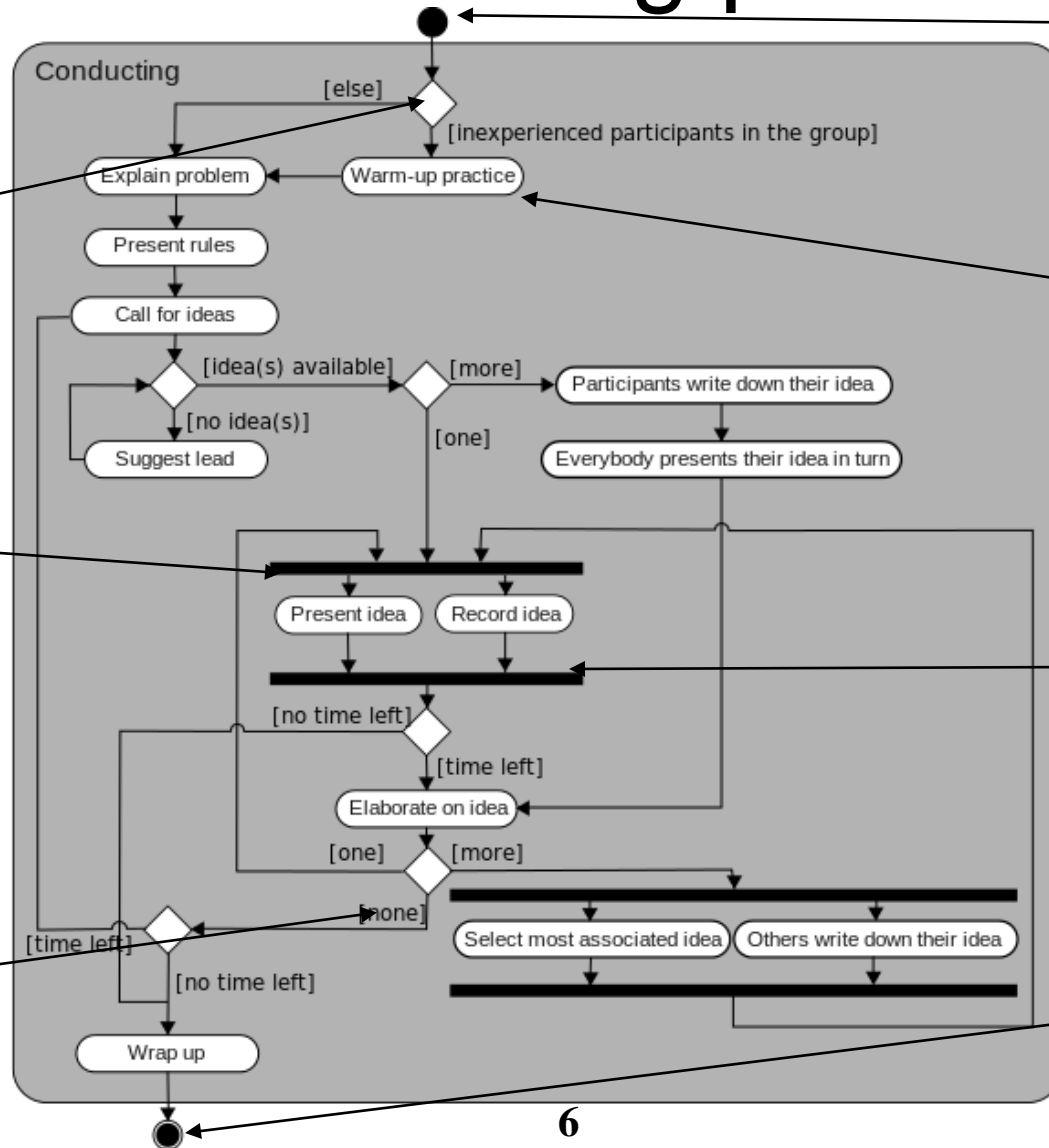
# Flowchart



# Activity Diagram

- An activity diagram shows the flow from activity to activity
- You use activity diagrams to model the dynamic aspects of a system. For the most part, this involves modeling the sequential (and possibly concurrent) steps in a computational process
- You can use activity diagrams to model a workflow or an operation

# Activity diagram for a guided brainstorming process



Initiation

Sequential  
branch

Activity node

Fork

Join

Guard  
expression


Completion

# Constituent Elements of Activity Diagram



- Initial node
- Final node
- Activity node
- Control flows
- Decision node and guard expression
- Merge node
- Fork node and join node

# Initial node and Final node

## ● Initial node

- An initial node acts as a starting point for executing an activity
- Initial nodes are notated as a solid circle 


## ● Final node

- A final node is a control node at which a flow in an activity stops
- Activity final nodes are notated as a solid circle within a hollow circle 
- Flow final nodes are noted as a circle with an “X” cross inside it 



# Activity Node

- An activity node is an organizational unit within an activity
- In general, activity nodes are nested groupings of **actions** or other nested activity nodes



Do construction()

The diagram shows two activity nodes, each represented as a rounded rectangle with a thin black border. The first node on the left contains the text 'Do construction()'. The second node on the right contains the text 'Get next order'.

Get next order

# Control Flows

- When an action or activity node completes execution, flow of control passes immediately to the next action or activity node
- You specify this flow by using flow arrows to show the path of control from one action or activity node to the next action or activity node
- In the UML, you represent a flow as a simple arrow from the predecessor action to its successor, without an event label



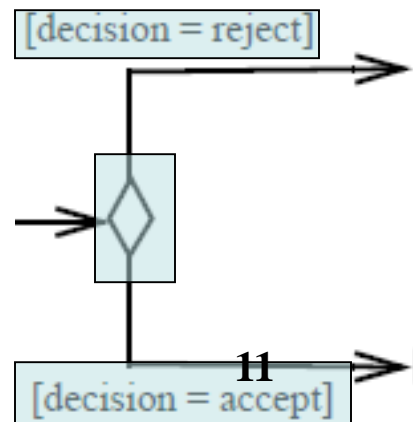
# Decision Node and Guard Expression

## ● Decision node

- A Decision Node is a Control Node that chooses between outgoing flows

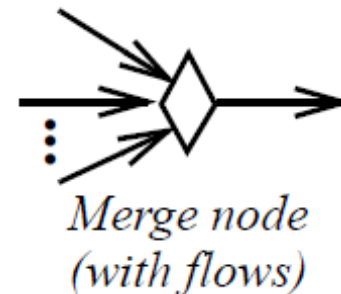
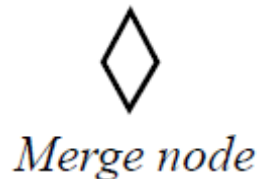
## ● Guard condition

- A condition that must be satisfied to enable an associated transition to fire



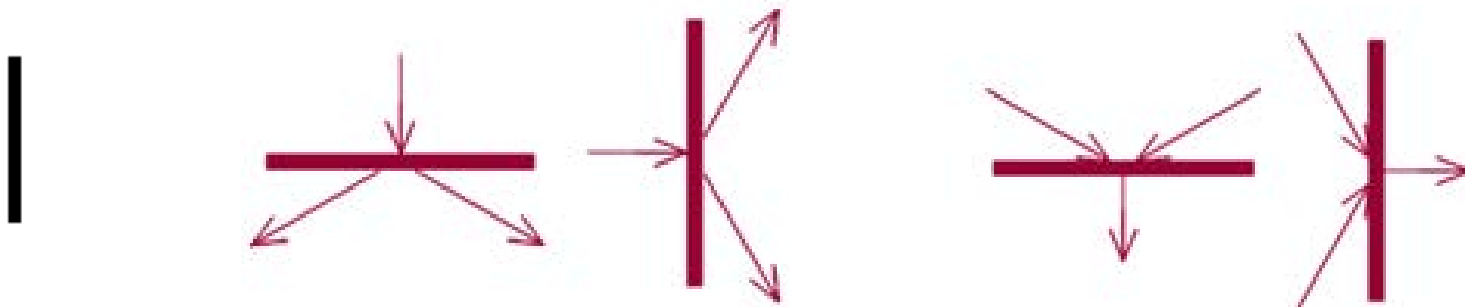
# Merge

- A MergeNode is a control node that brings together multiple flows without synchronization
  - When two paths of control merge back together, you can also use a diamond symbol with two input arrows and one output arrow. No guards are necessary on merge



# Fork Node and Join Node

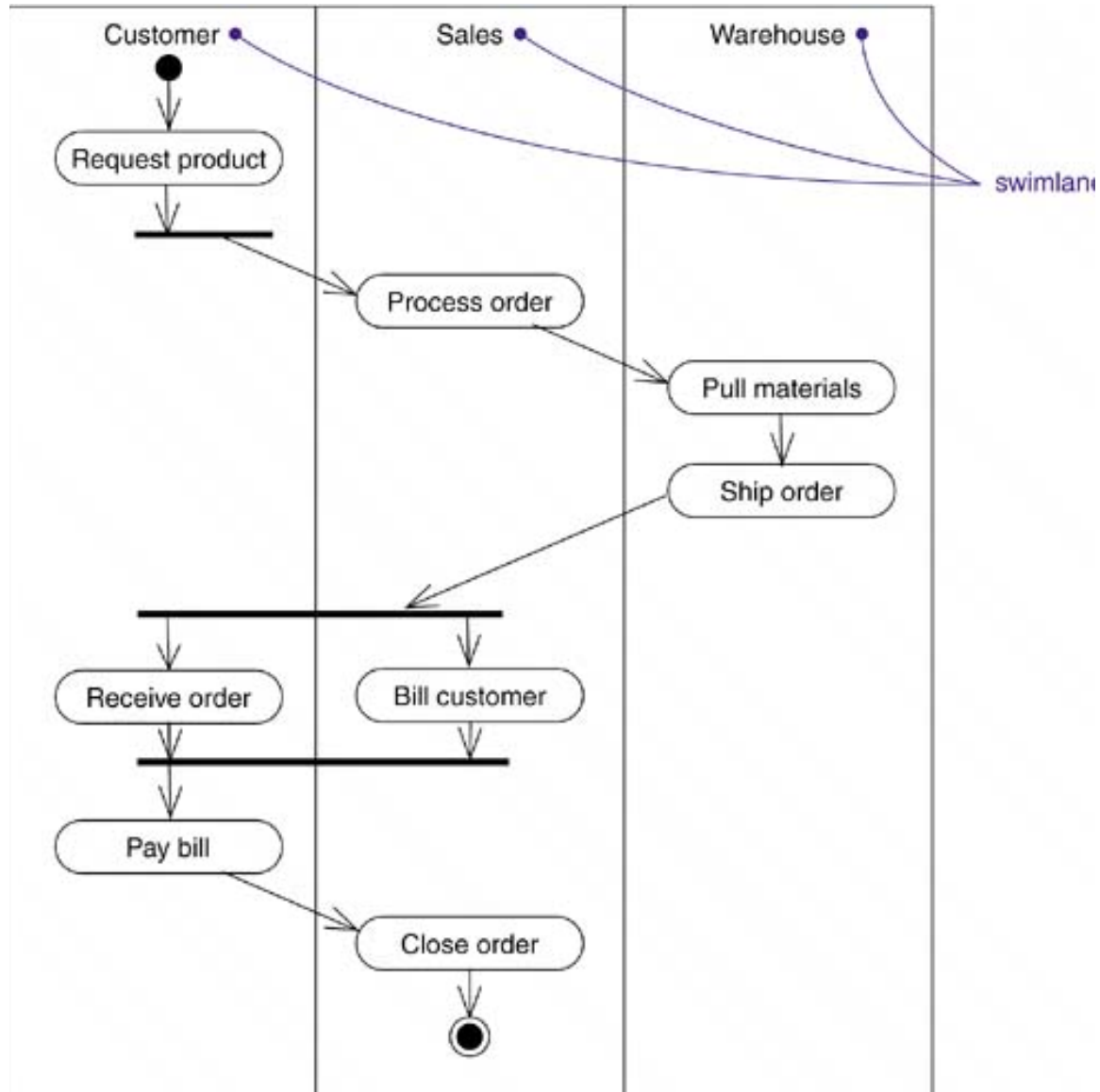
- In the UML, you use a synchronization bar to specify the forking and joining of these parallel flows of control
- A synchronization bar is rendered as a thick horizontal or vertical line



# Fork Node and Join Node

- A fork represents the splitting of a single flow of control into two or more concurrent flows of control
  - A fork may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control
- A join represents the synchronization of two or more concurrent flows of control
  - A join may have two or more incoming transitions and one outgoing transition

# Swimlane

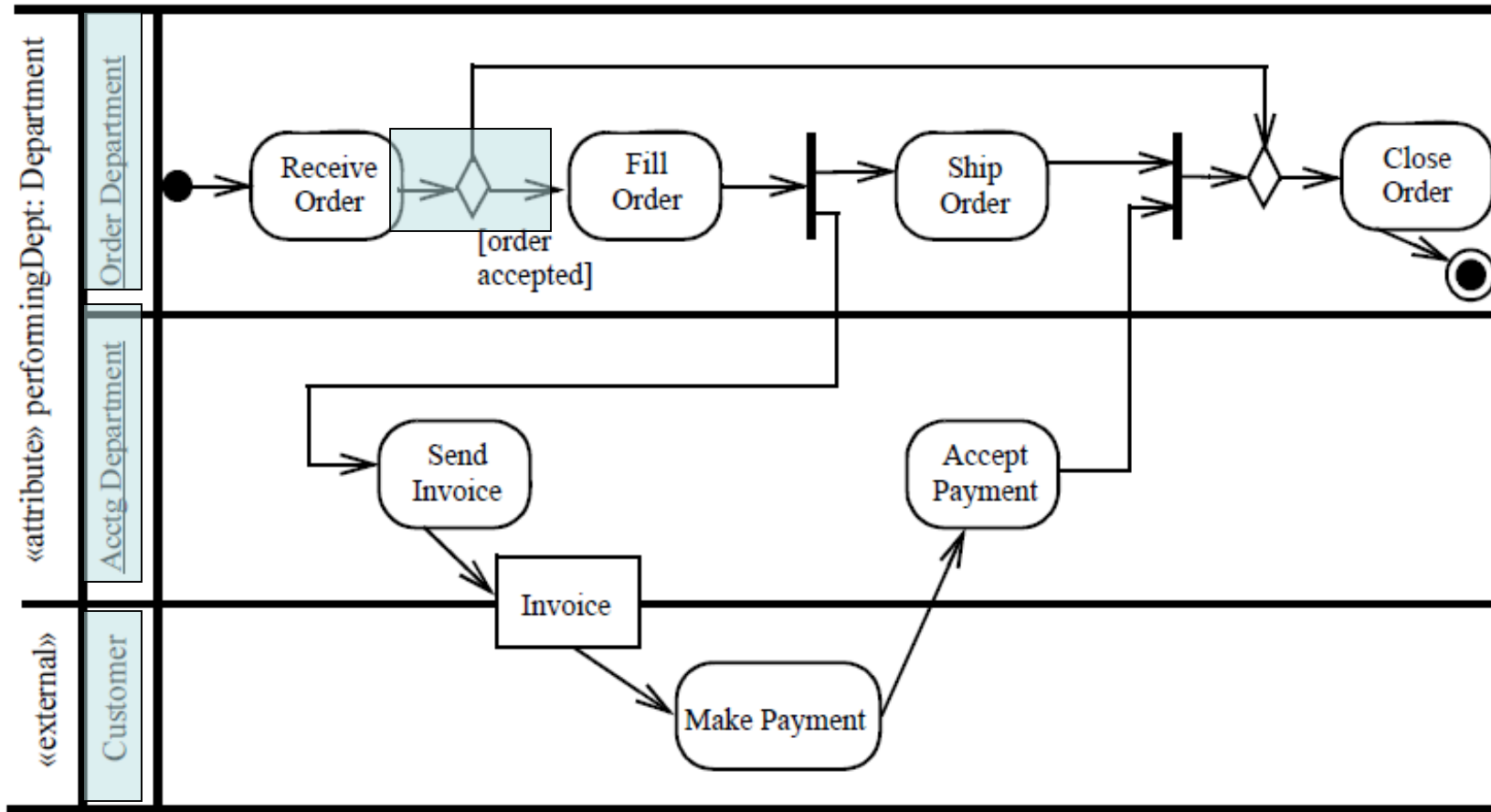


# Swimlane

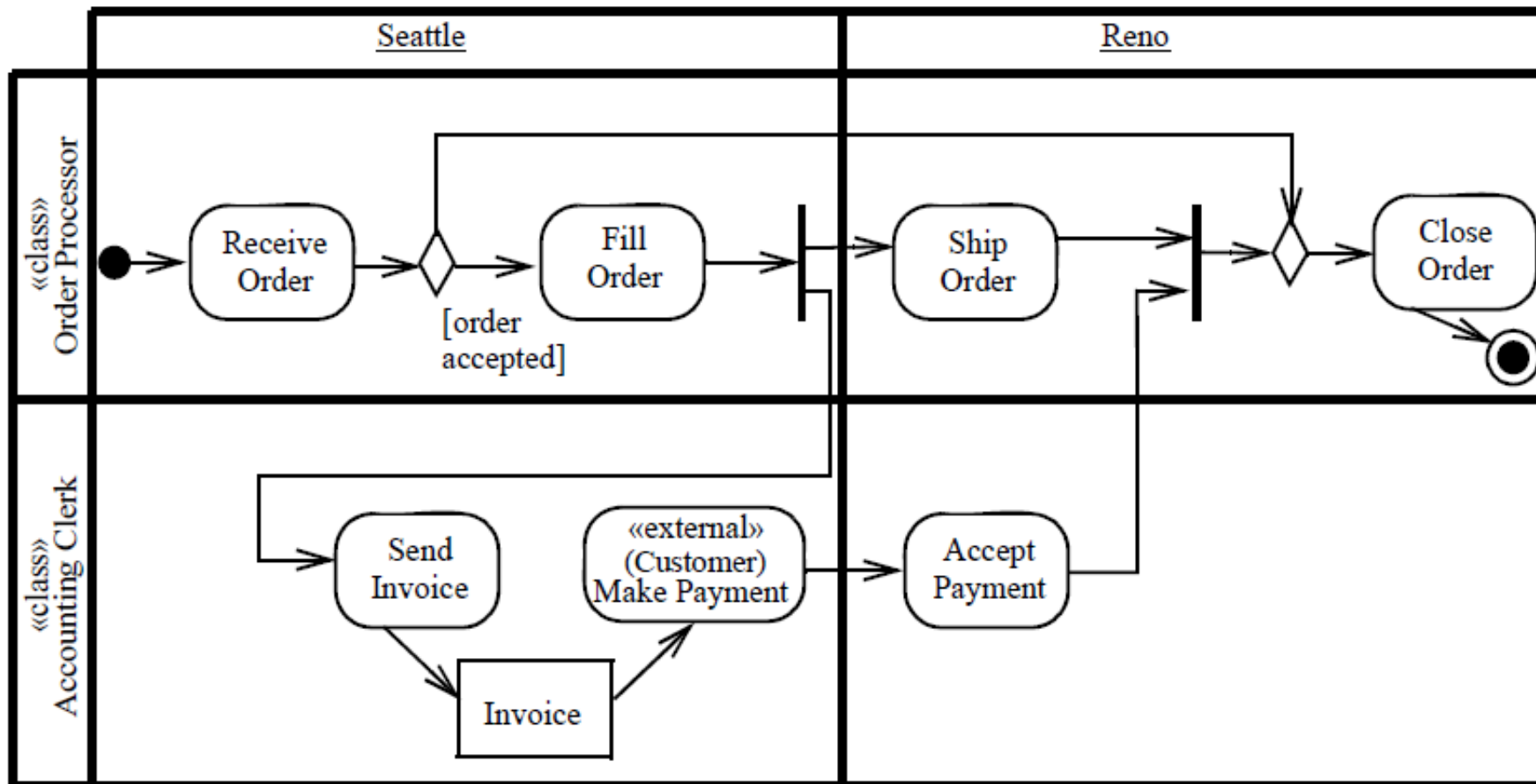
- You'll find it useful, especially when you are modeling workflows of business processes, to partition the activity states on an activity diagram into groups, each group representing the business organization responsible for those activities
- Each swimlane has a name unique within its diagram



# Swimlane

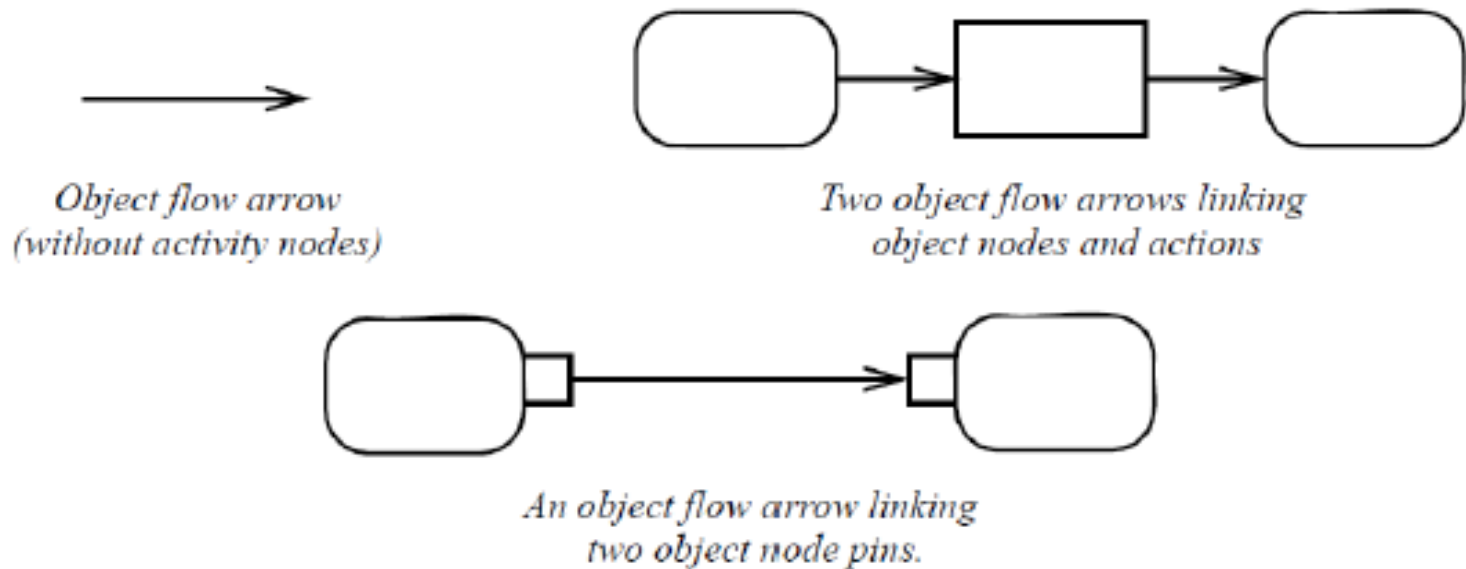


# Swimlane

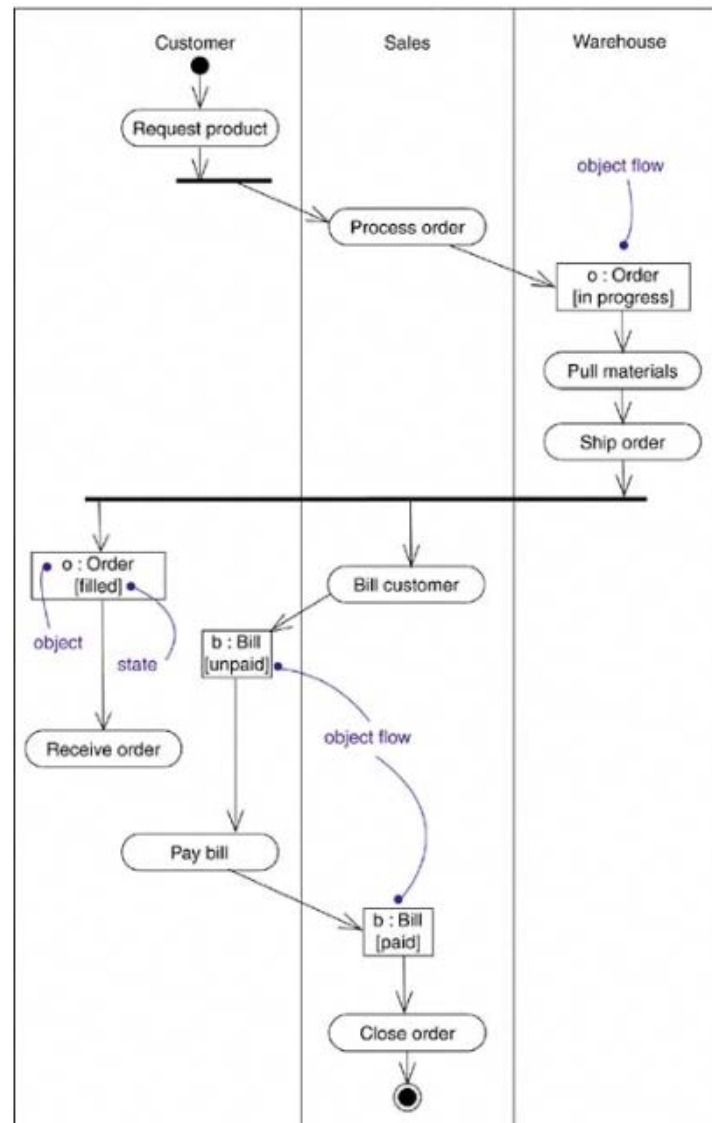


# Object Flow

- Objects may be involved in the flow of control associated with an activity diagram



# Object Flow



# Activity Diagram

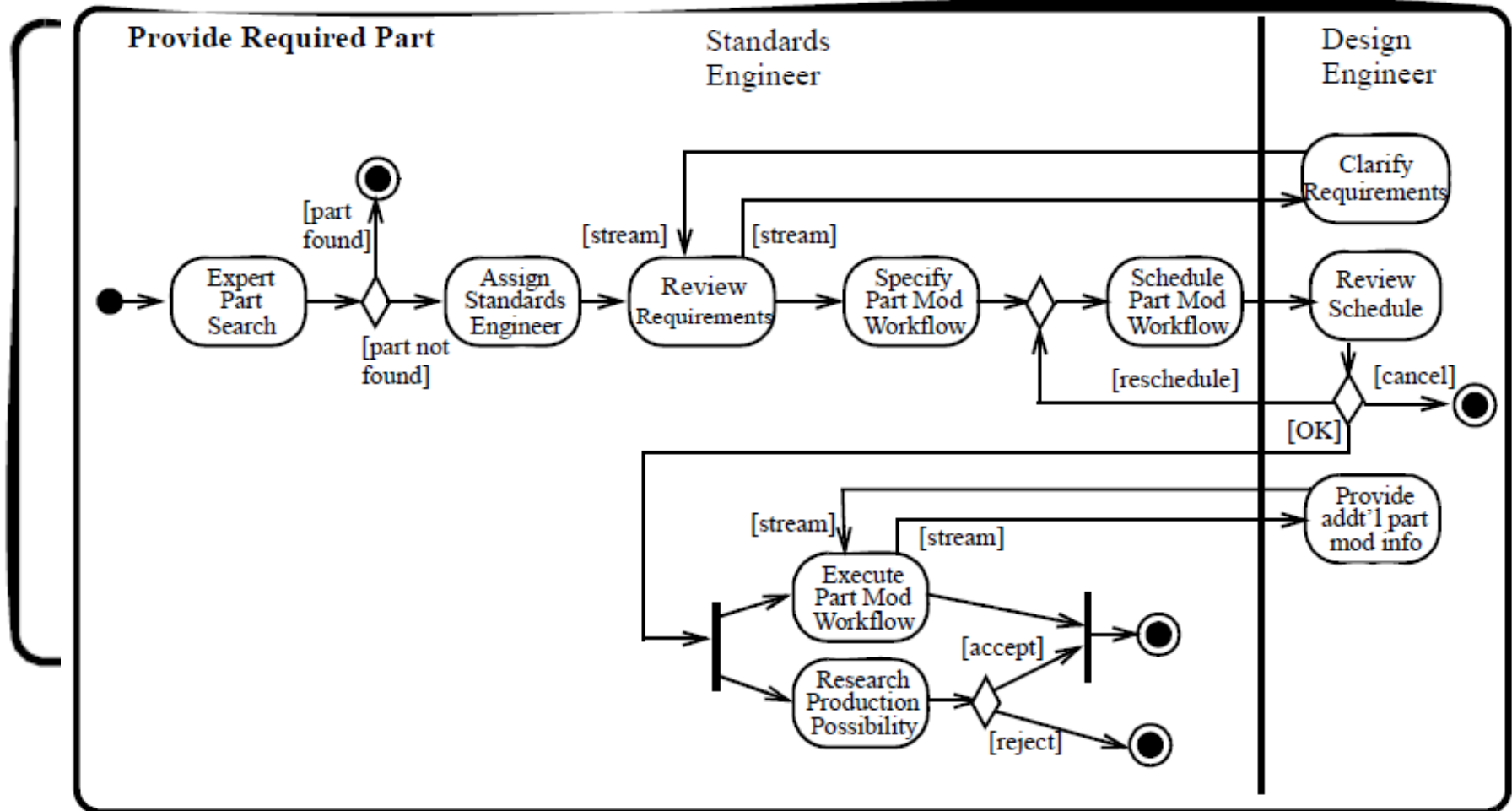
- Sub-activity
- Join Specification
- Signal
- Pin
- Expansion Region

# Sub-activity

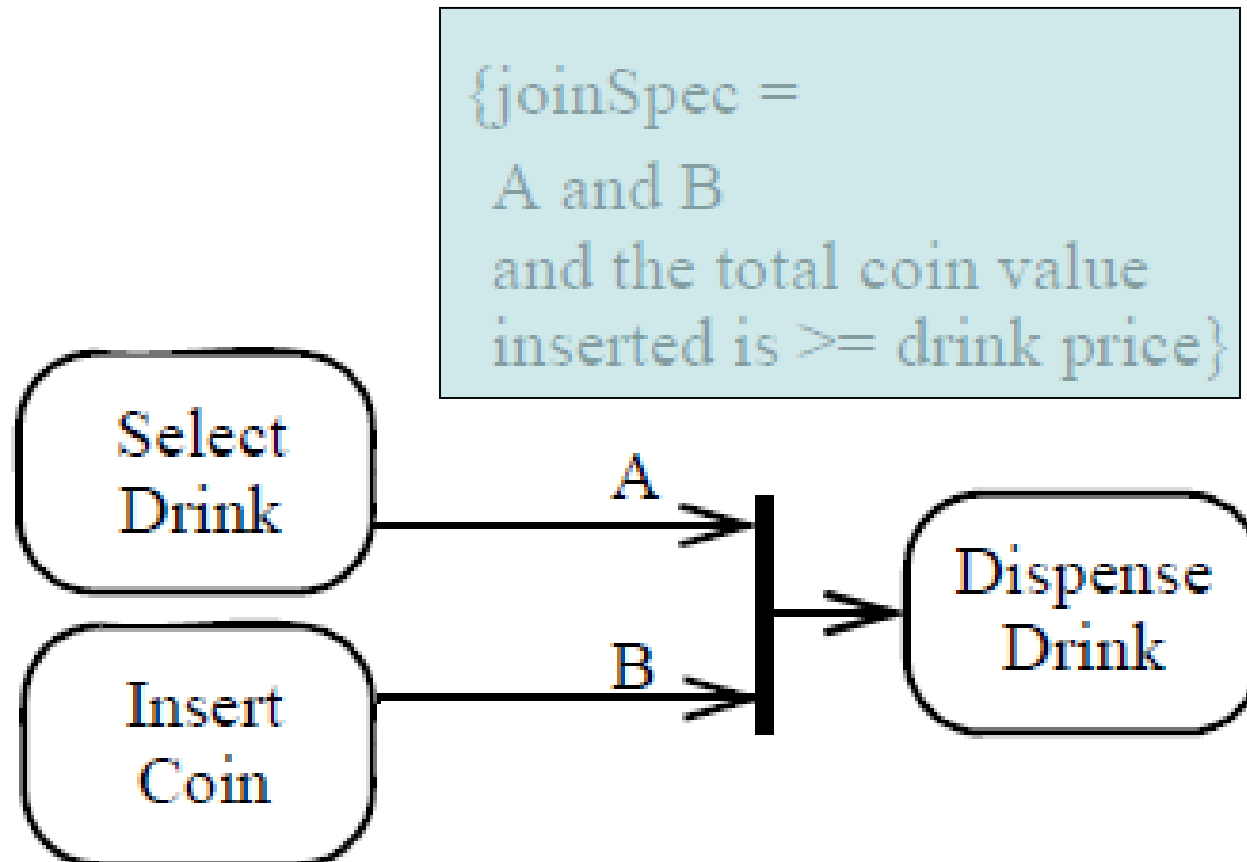
- The rake in the bottom corner of an activity indicates that the activity is described by a more finely detailed activity diagram



# Sub-activity Example



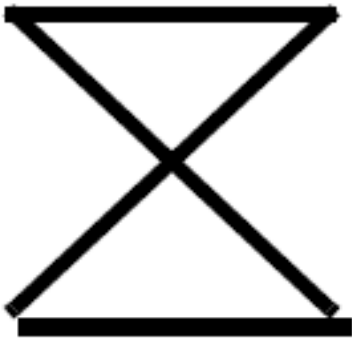
# Join Specification



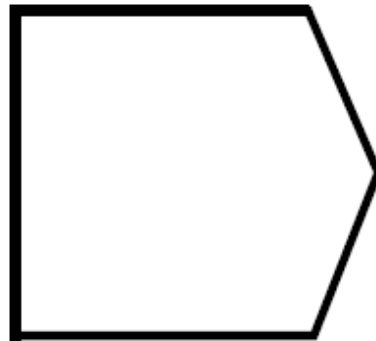


# Signal Action

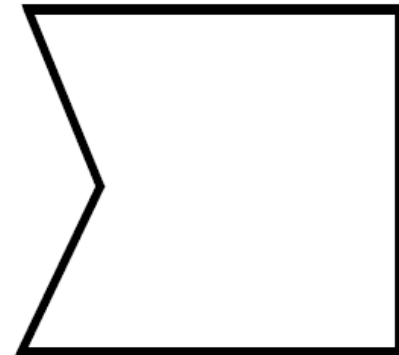
- A signal is a kind of event that represents the specification of an asynchronous message communicated between instances



Accept Time

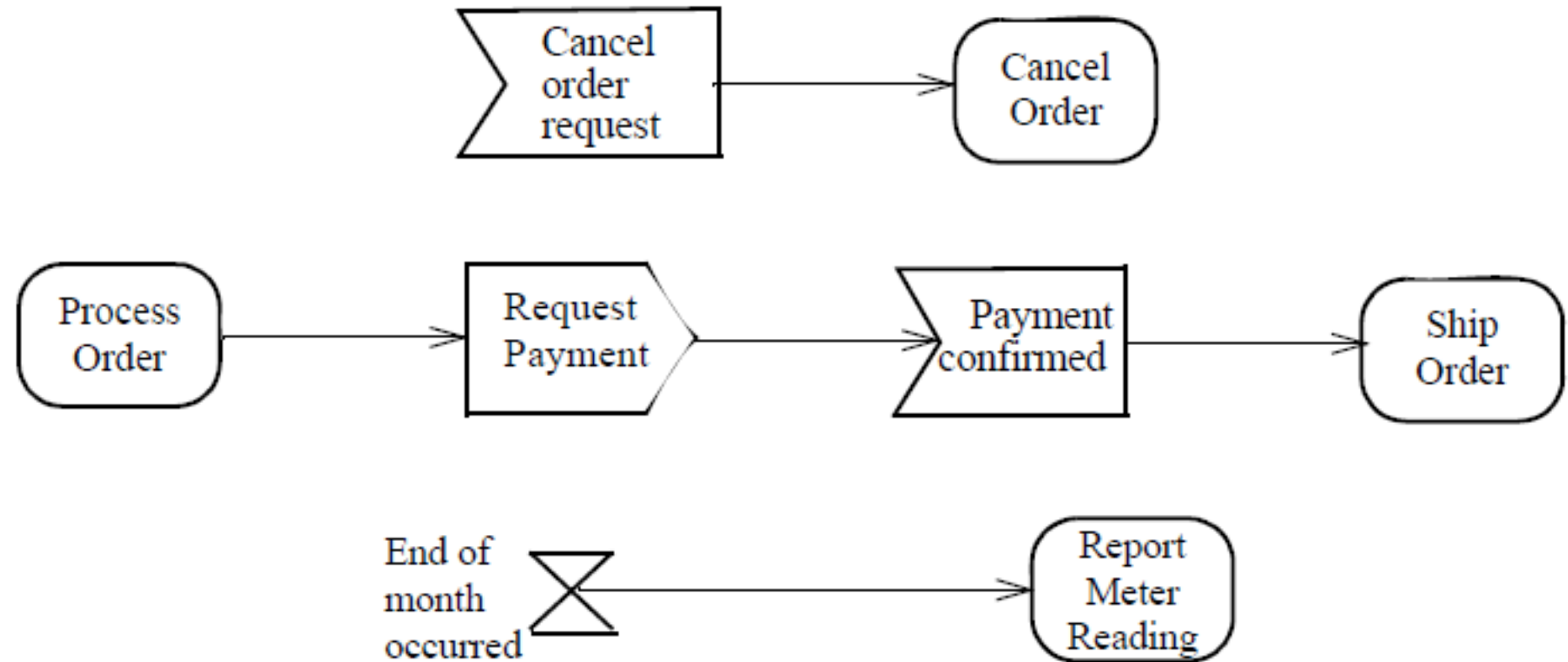


Sending



AcceptEventAction

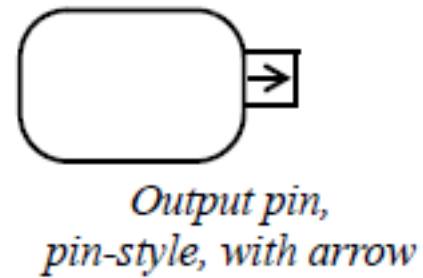
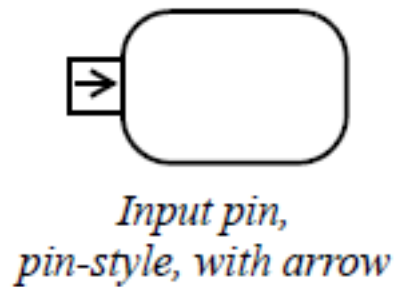
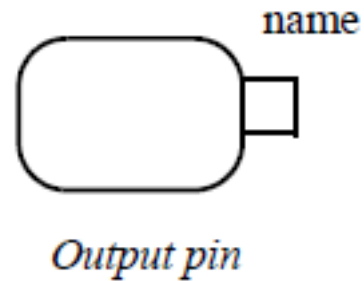
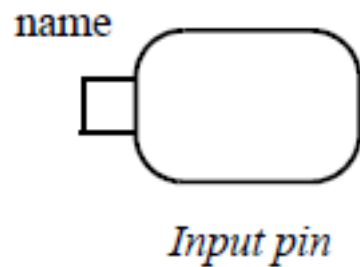
# Signal



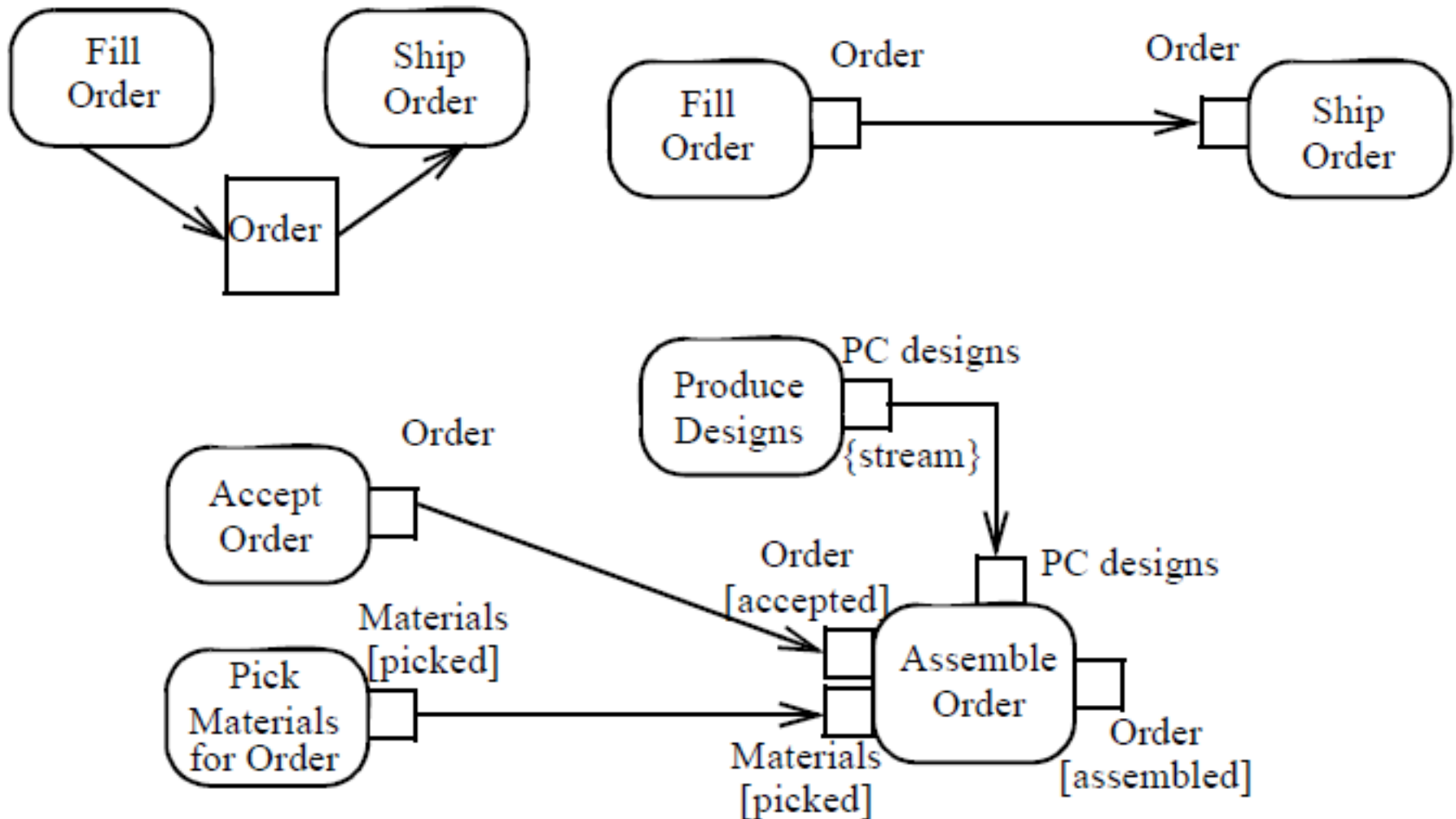
# Pin

- A Pin represents an input to an Action or an output from an Action
  - An InputPin represents an input, while an OutputPin represents an output
  - Each of the sets of inputs and outputs owned by an Action are ordered
  - The InputPins and OutputPins of an Action are determined by the kind of Action it is

# Pin



# Pin Examples

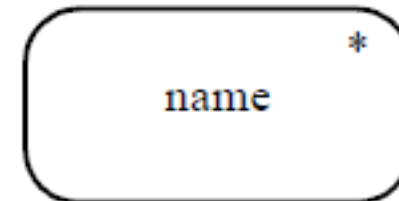
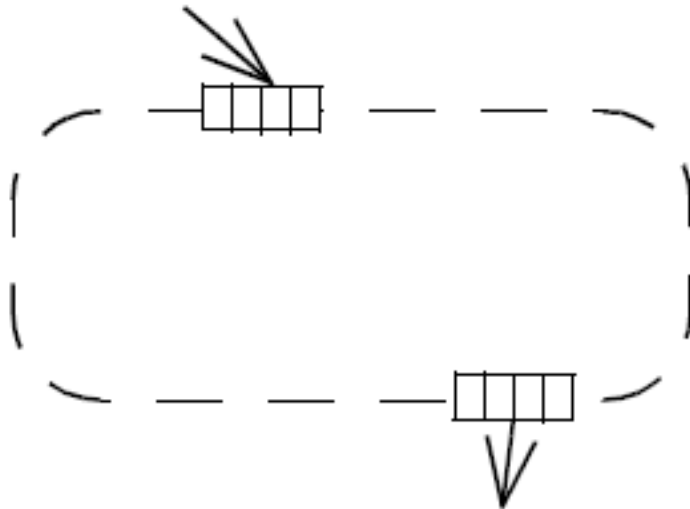


# Expansion Region

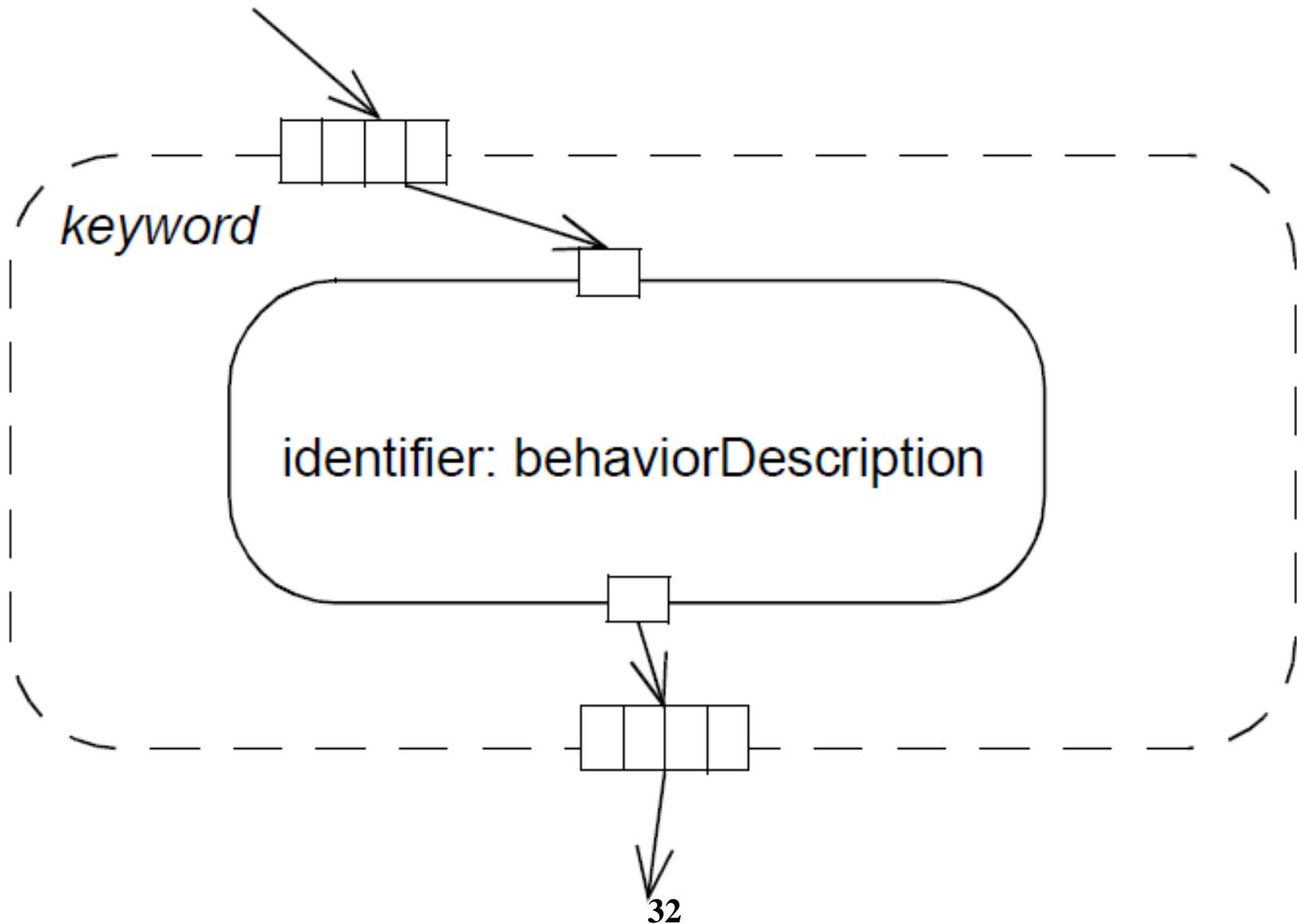
- An Expansion Region is a Structured Activity Node that executes its contained elements multiple times corresponding to elements of an input collection

# Expansion Region

- The Expansion Node symbols are placed on the boundary of the dashed box
  - Usually, Activity Edge arrows inside and outside the Expansion Region will distinguish input and output expansion nodes

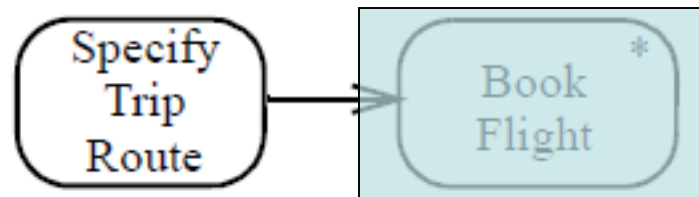
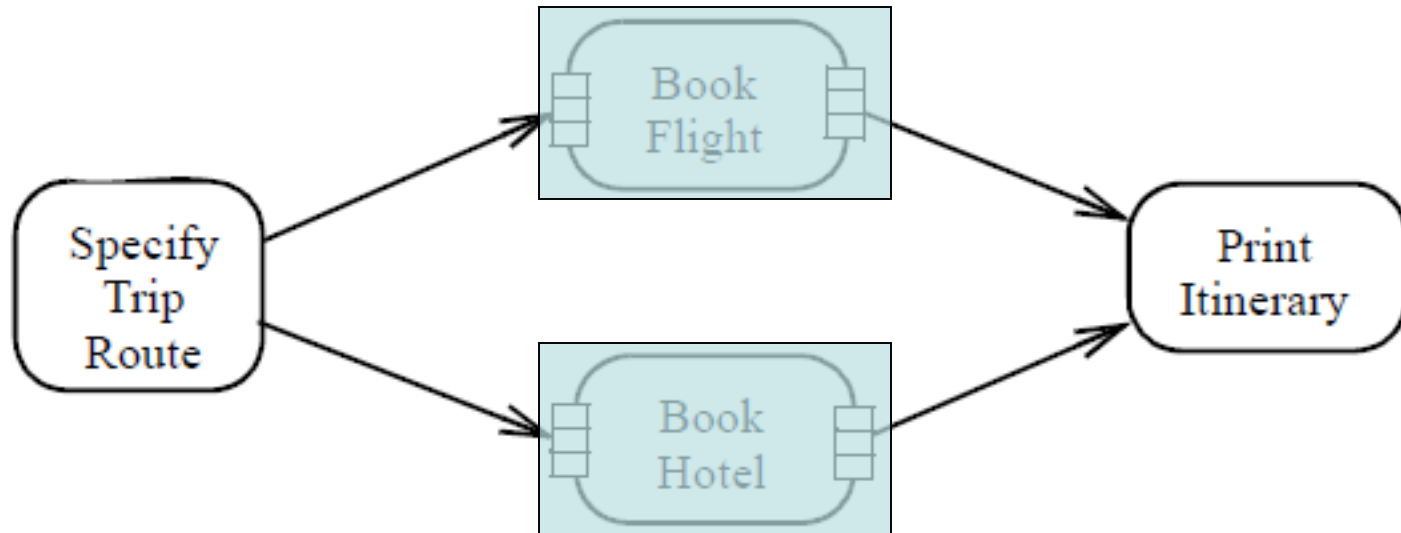


# Expansion Region Example





# Expansion Region Example



# Attention Points in Activity Diagram

- Use the basic modeling elements to draw a activity diagram, such as branch, fork node and join node
- Use the swimlane according to whether the activity diagram should reflect the different actors
- If necessary, add the object flow
- If necessary, add the advanced modeling elements, such as Sub-activity, Join Specification, Signal, Pin and Expansion Region

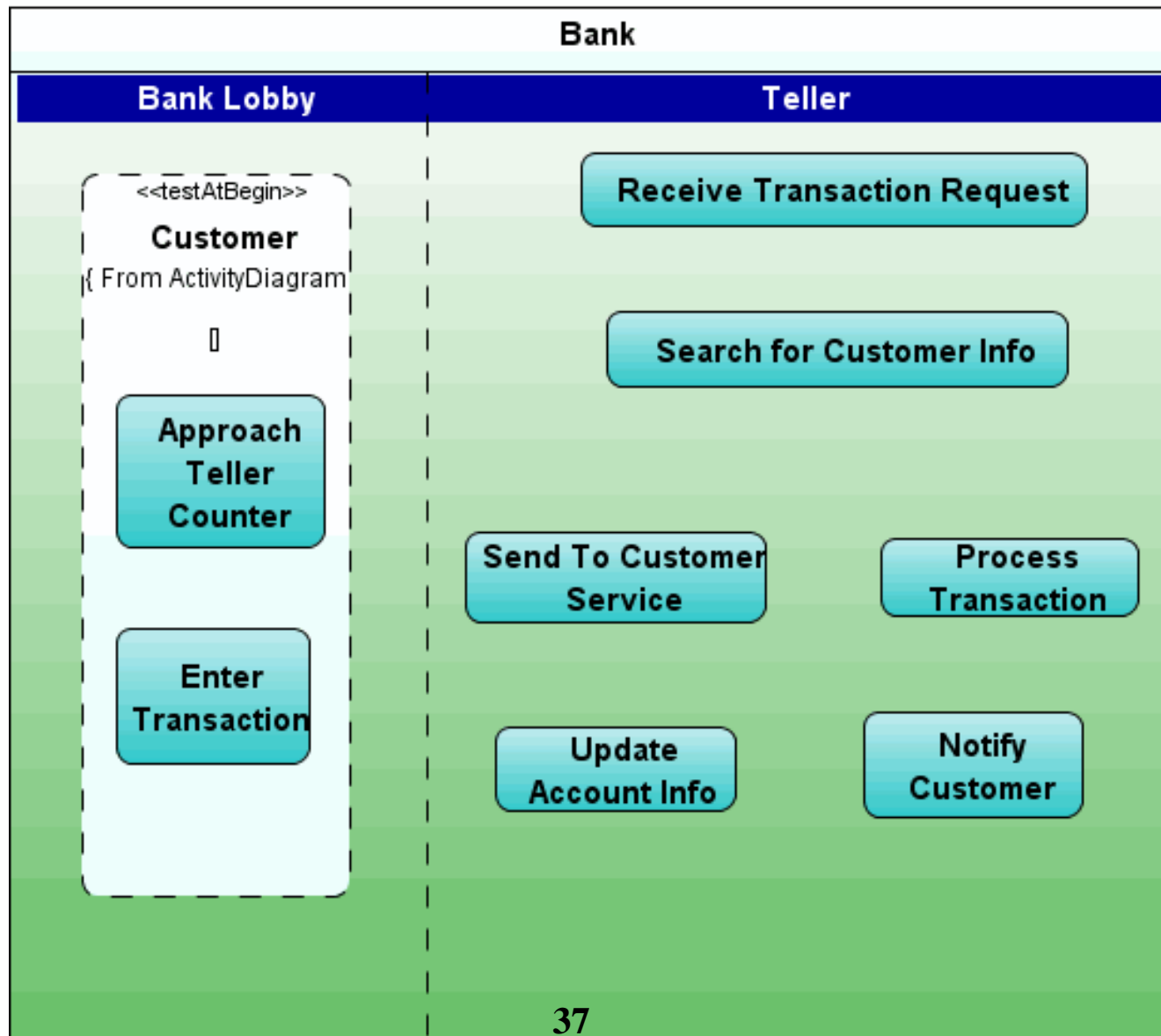
# Steps of Activity Diagram Modeling

- Define the scope of the activity diagram
- Add the initial node and final node
- Add the activity node
- Add the decision node
- Identify parallel activities
- Add the control flows

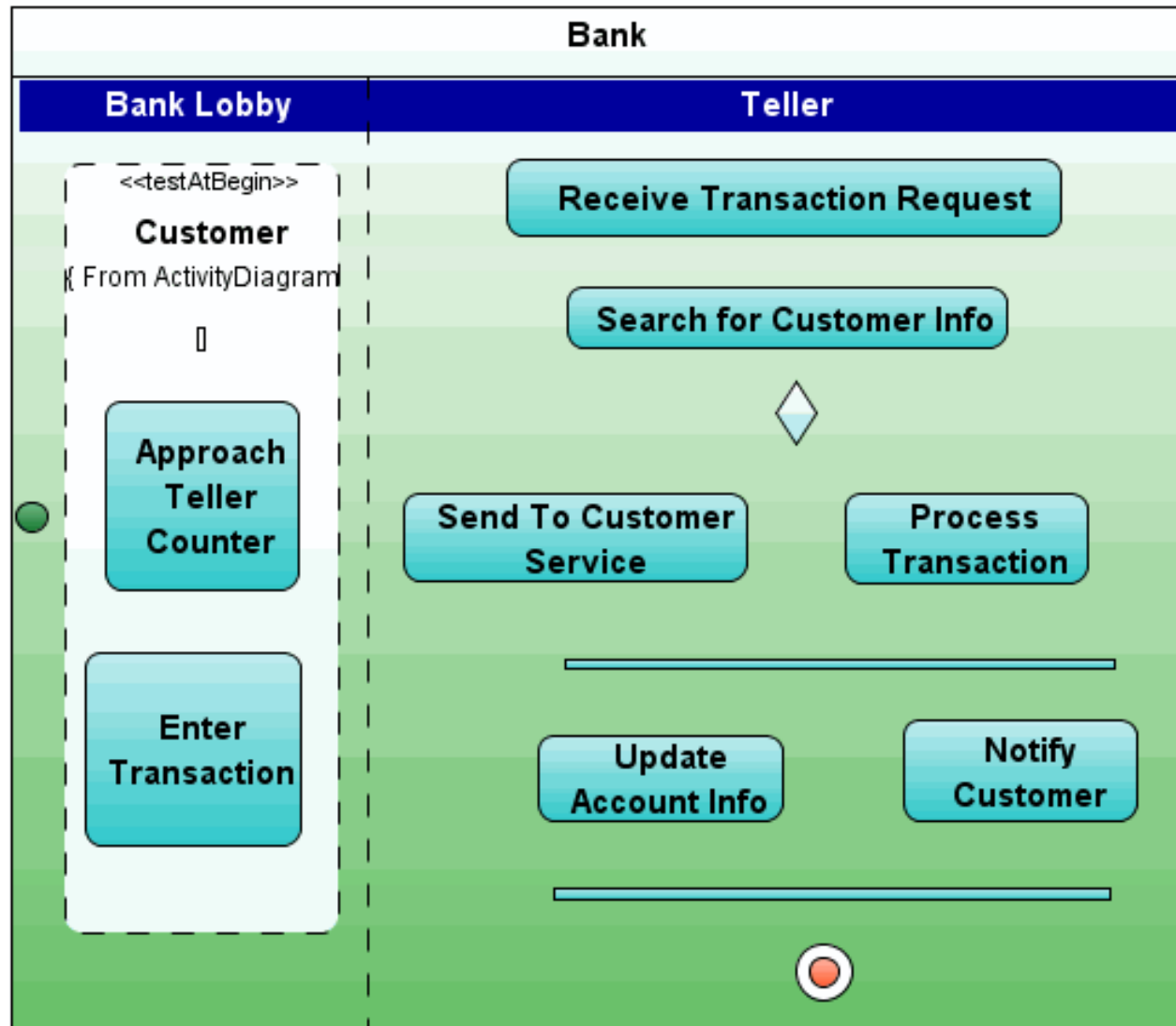
# Activity Diagram Modeling Example



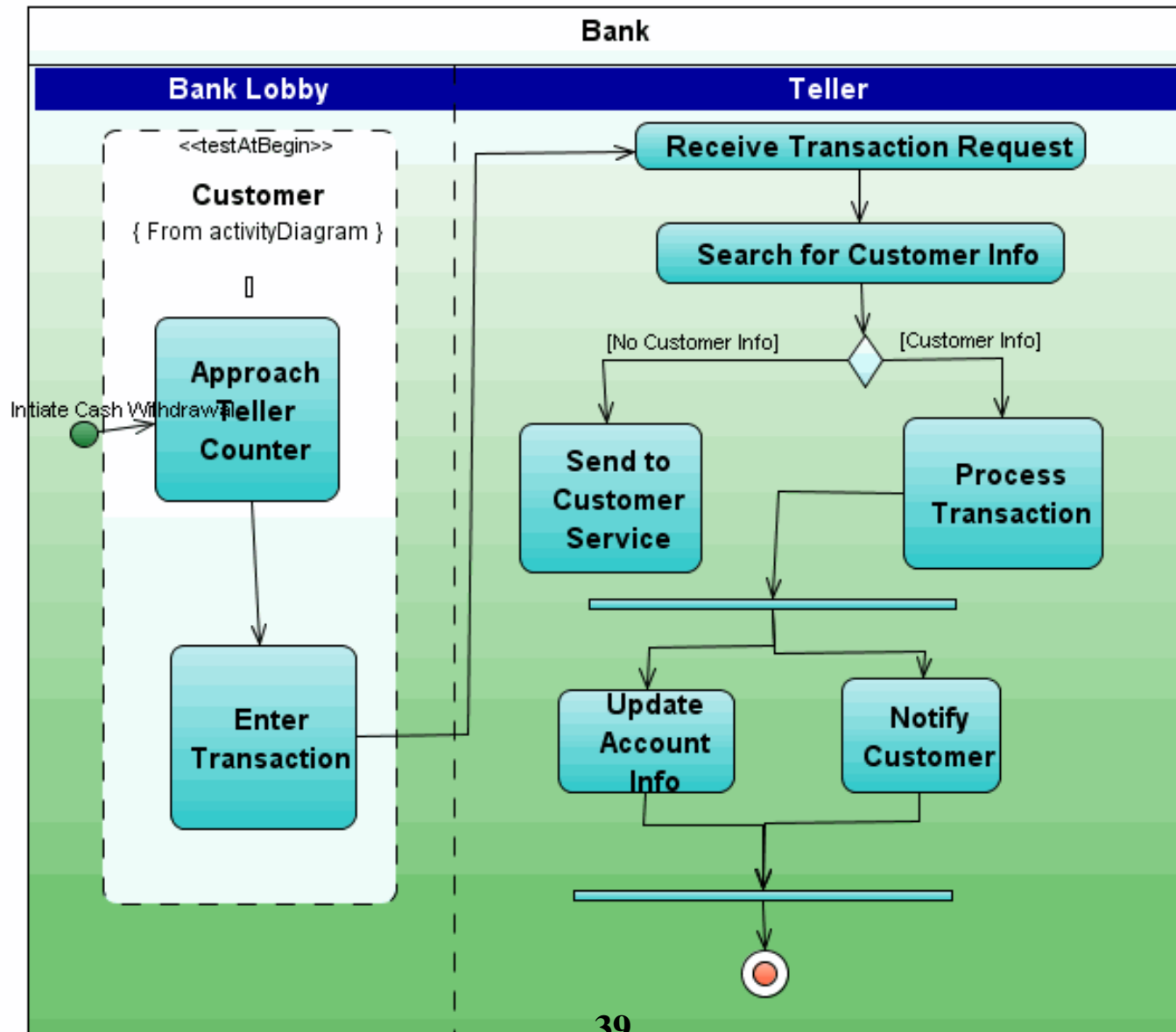
# Activity Diagram Modeling Example



# Activity Diagram Modeling Example



# Activity Diagram Modeling Example



# Style of UML Activity Diagram

- Ensure That Each Activity Edge Leaving a Decision Point Has a Guard
  - This ensures that you have thought through all possibilities for that decision point
- Ensure That Guards on Decision Points Form a Complete Set
  - It must always be possible to leave a decision point
  - Therefore, the guards on its exit activity edges must be complete. For example, guards such as  $x < 0$  and  $x > 0$  are not complete because it isn't clear what happens when  $x$  is 0
- Do Not Overlap Guards
  - The guards on the activity edges leaving a decision point, or an activity, must be consistent with one another
  - For example, guards such as  $x < 0$ ,  $x = 0$ , and  $x > 0$  are consistent



# Style of UML Activity Diagram

- Ensure That Forks Have Corresponding Joins
  - In general, for every start (fork), there is an end (join)
- Ensure That a Fork Has Only One Entry
- Ensure That a Join Has Only One Exit

# Style of UML Activity Diagram

- Have Fewer than Five Swim Lanes
- Model the Key Activities in the Primary Swim Lane