ACTIVITY MODELS

- Activity diagrams like **flow chart** in that it shows flow of control & focuses on operations rather than objects
- it shows
  - **sequential flow of control**
  - **concurrent flow of control**

ACTIVITY DIAGRAM

- An activity diagram is a flowchart, showing flow of control from activity to activity.
- With an activity diagram, you can also model the flow of an object as it moves from state to state at different points in the flow of control
- It looks at the operations that are passed among objects

Contents of Activity Diagram

- Activity diagrams commonly contain
  - Activity states and action states
  - Transitions
  - Objects

INTIATION & TERMINATION

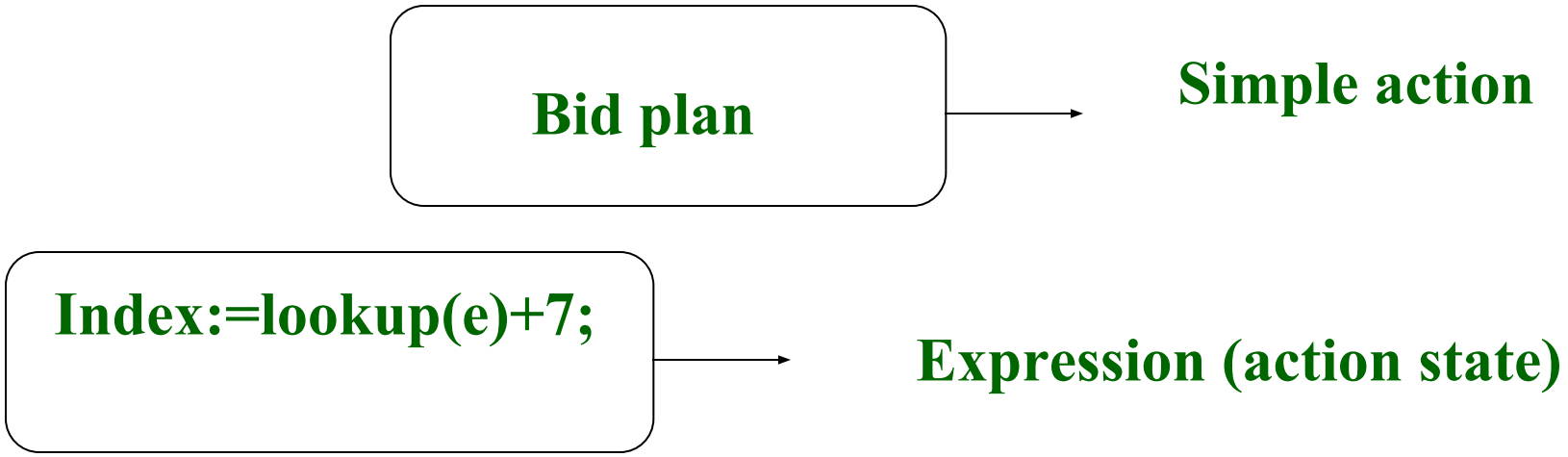
- ● **A solid circle** with outgoing arrow shows **starting point of activity diagram** so control starts at solid circle.

- ◉ **A bull's eye** shows **termination point** which has incoming arrow.
  - at bull's eye activity is completed & execution is completed

Action states and Activity States

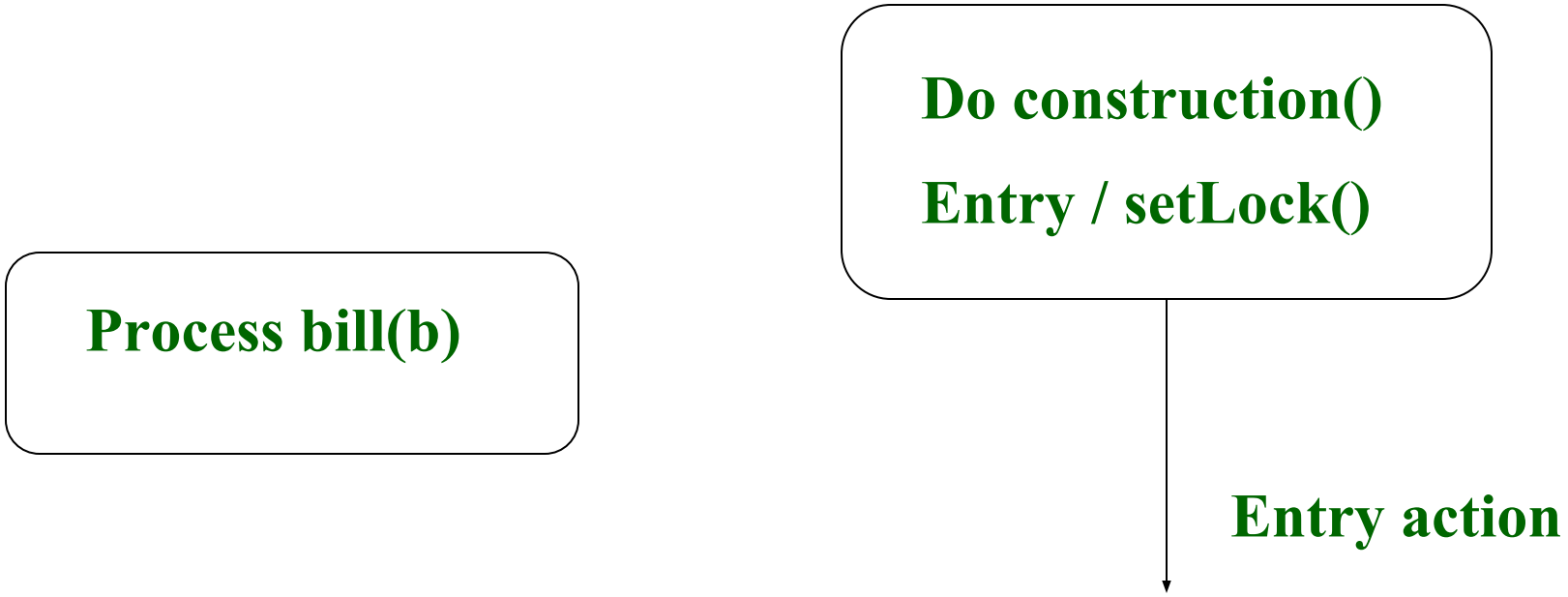
- You might evaluate some expression that sets the value of an attribute or that returns some value.
- (or) You might call an operation of an object, send a signal to an object or even create or destroy an object.
- These executable atomic computation are called action states because they are state of the system, each representing the execution of an action.

- Action states can’t be decomposed.
- Action state are atomic meaning that events may occur but the work of the action state is not interrupted.



Activity states

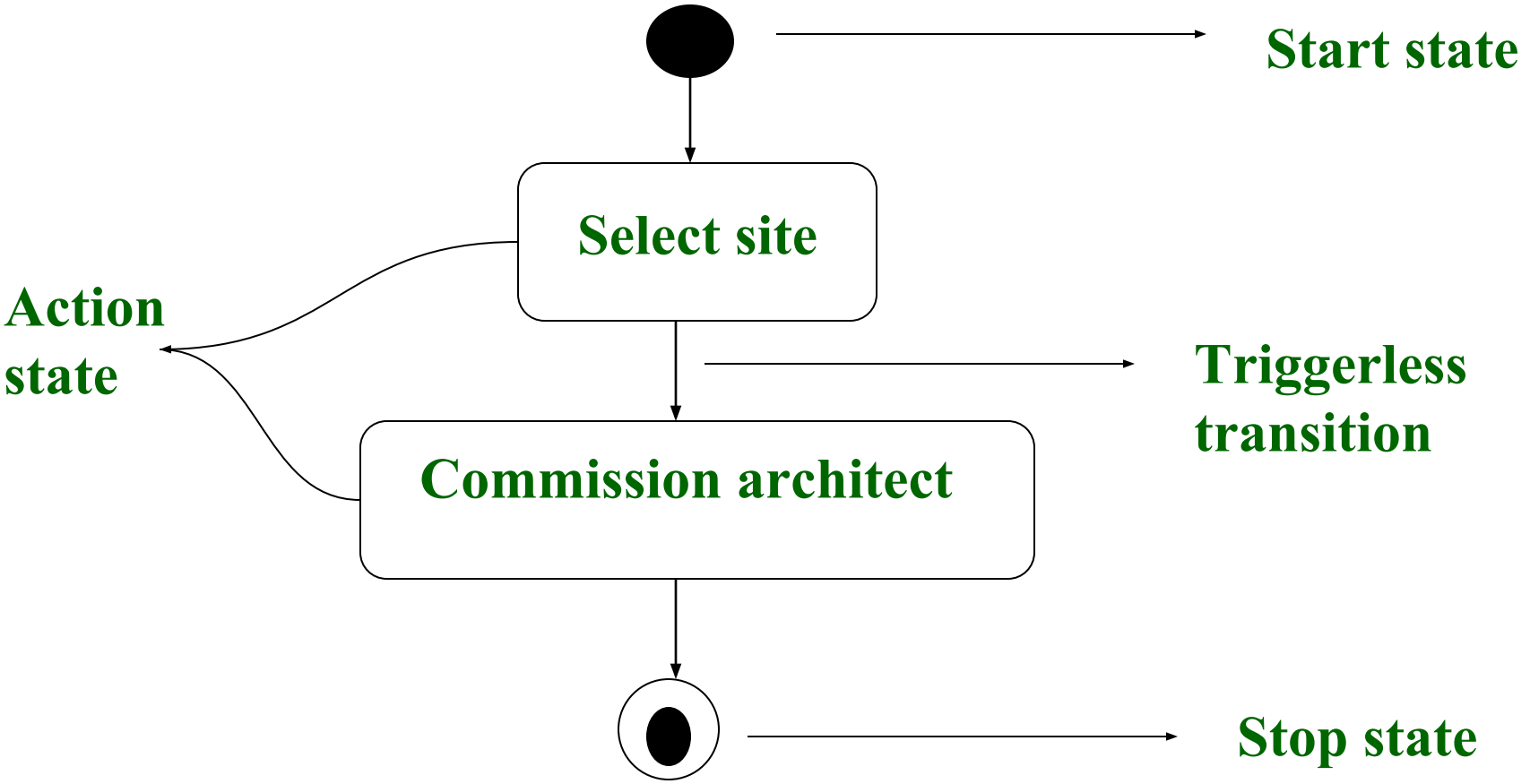
- Actions which are involved on entering and leaving the state are called entry and exit actions.



Transitions

- When the action or activity of a state completes, flow of control passes immediately to the next action or activity state.
- You specify this flow by using transitions, to show the path from one action or activity state to the next action or activity state.
- Transition can be represented as directed line.

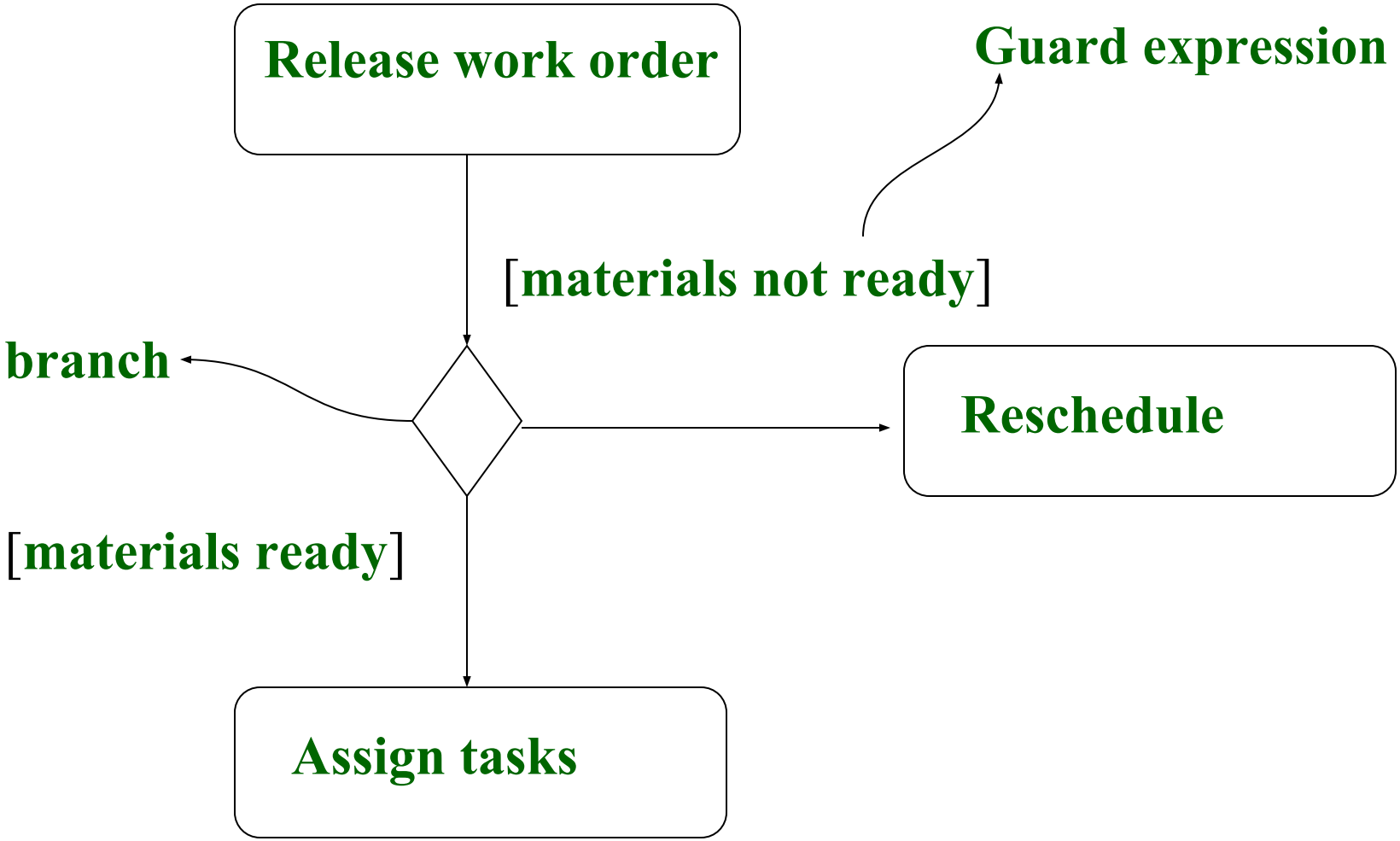
Example


Start state
Select site
Action state
Triggerless transition
Commission architect
Stop state

Branching

- If you want to specify some boolean expression in activity diagram you can make use of Branching, it will represent as diamond.
- A branch may have one incoming transition and two or more outgoing ones.
- The guards should not overlap.

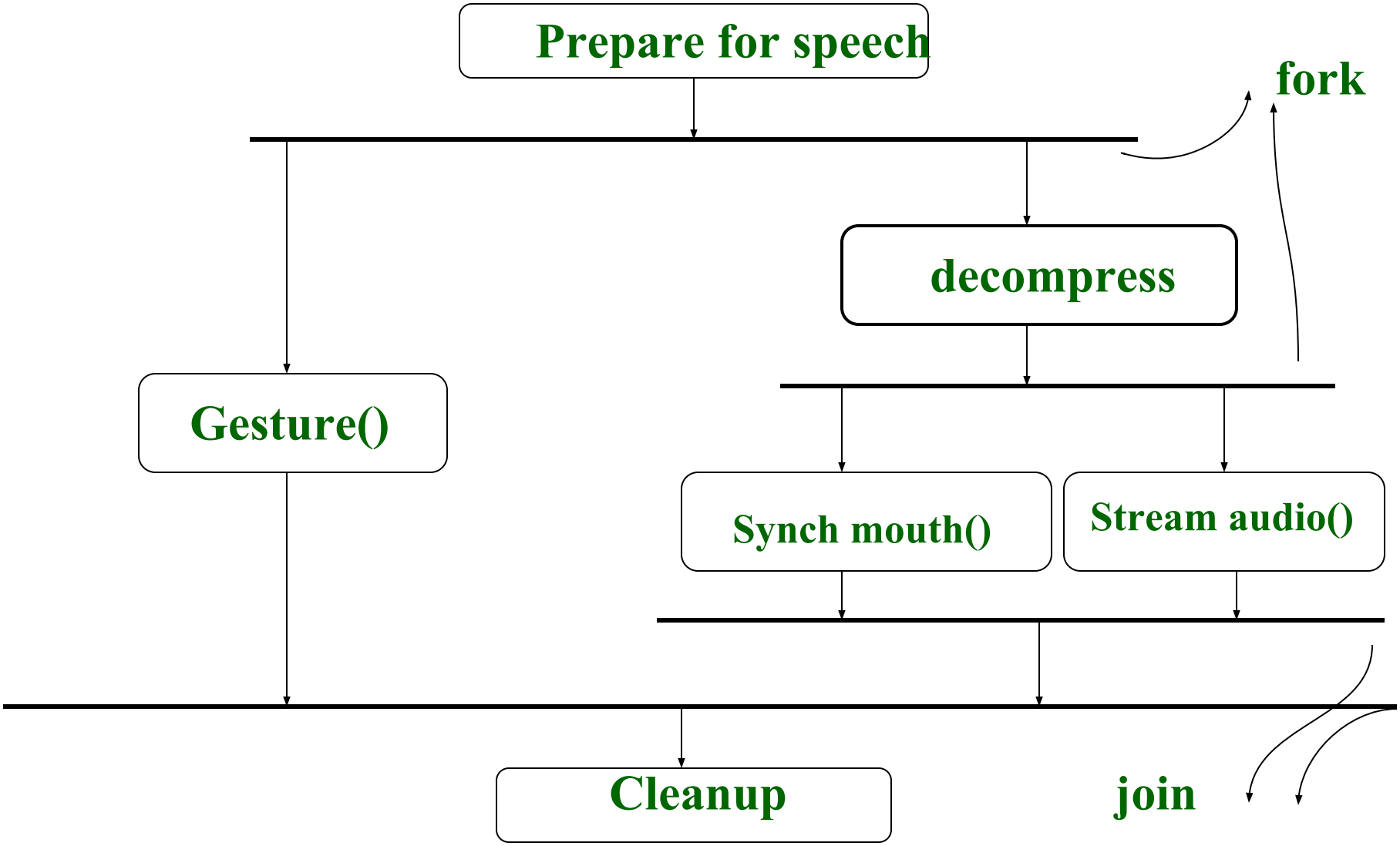
Example


Release work order
Guard expression
[materials not ready]
branch
Reschedule
[materials ready]
Assign tasks

Forking and joining

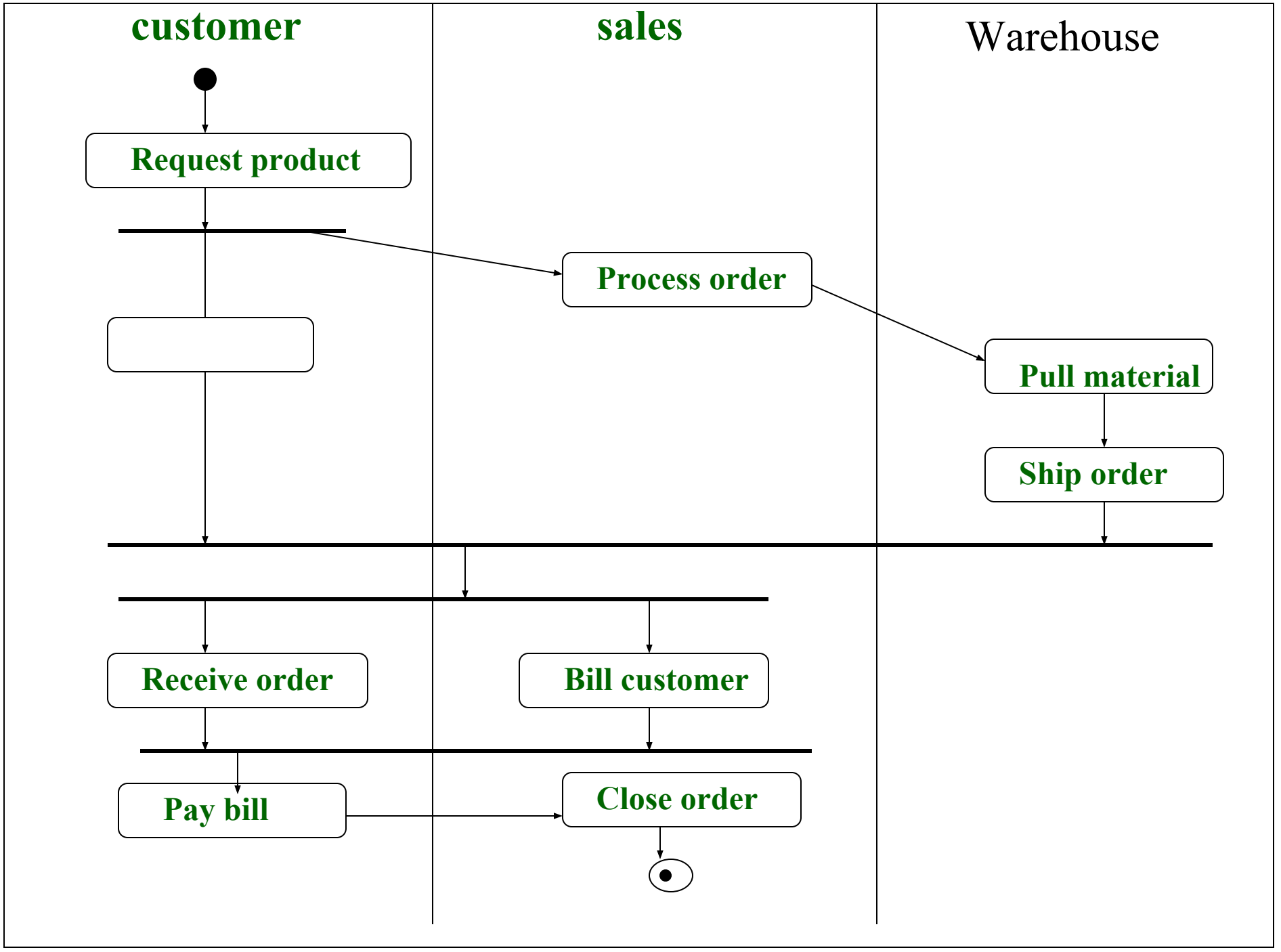
- In UML , you use a synchronization bar to specify the forking and joining of these parallel flows of control.
- A synchronization bar is rendered as a thick horizontal or vertical line.
- FORK : It may have one incoming transition and two or more outgoing transitions; each of which represents an independent flow or control.
- JOIN : It may have two or more incoming transitions and one outgoing transitions.

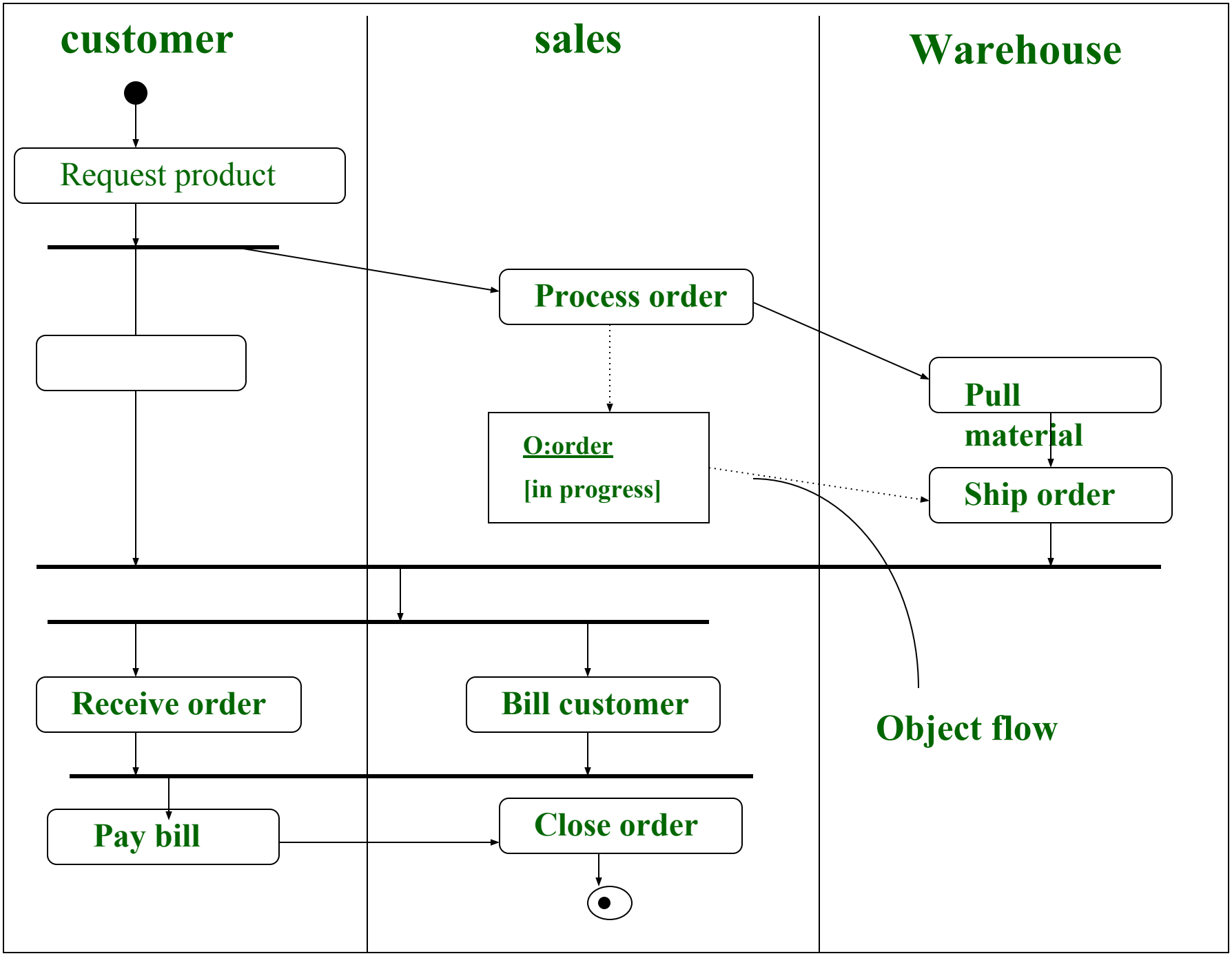
Example


Prepare for speech
fork
decompress
Gesture()
Synch mouth()
Stream audio()
Cleanup
join

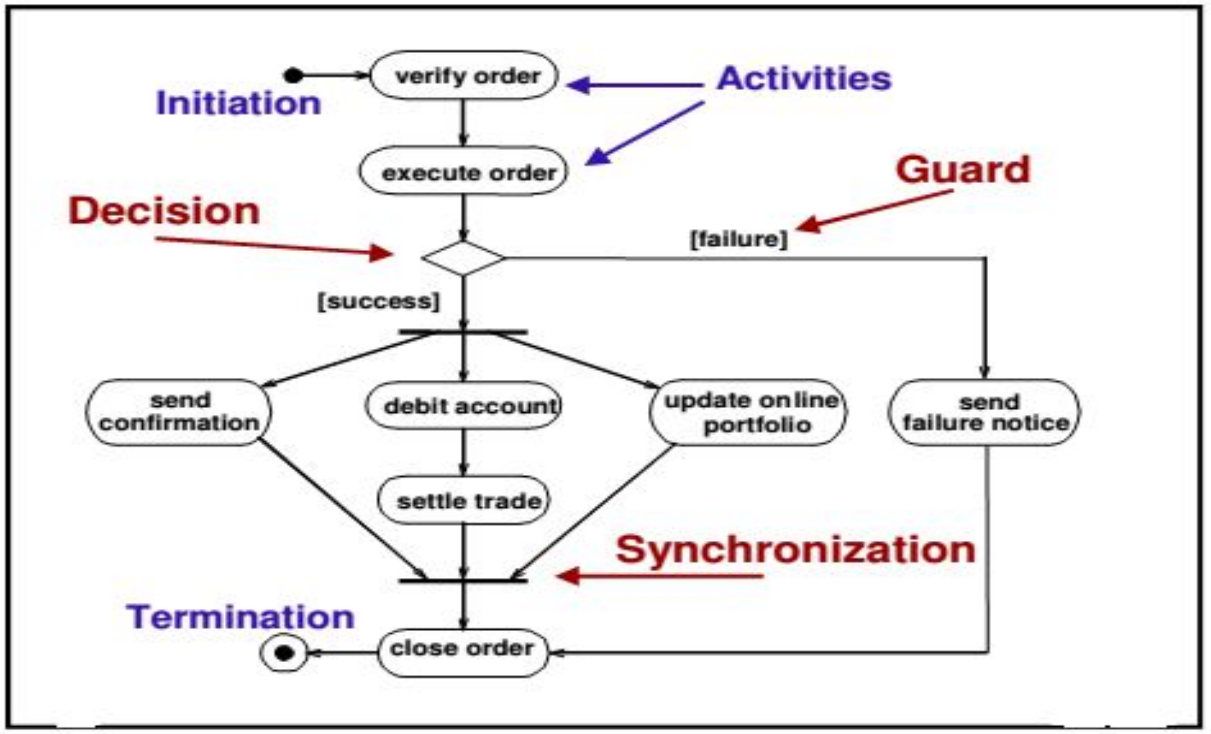
Swimlanes and object flow

- To partition the activity states on an activity diagram into groups, each group representing the business organization responsible for those activities and each group is called a swimlane.(each group is divided from its neighbor by a vertical solid line)
- Each swimlane may eventually be implemented by one or more classes.
- The activity diagram can be partitioned into swinlanes , every activity belongs to exactly one swimlane but transitions may cross lanes.

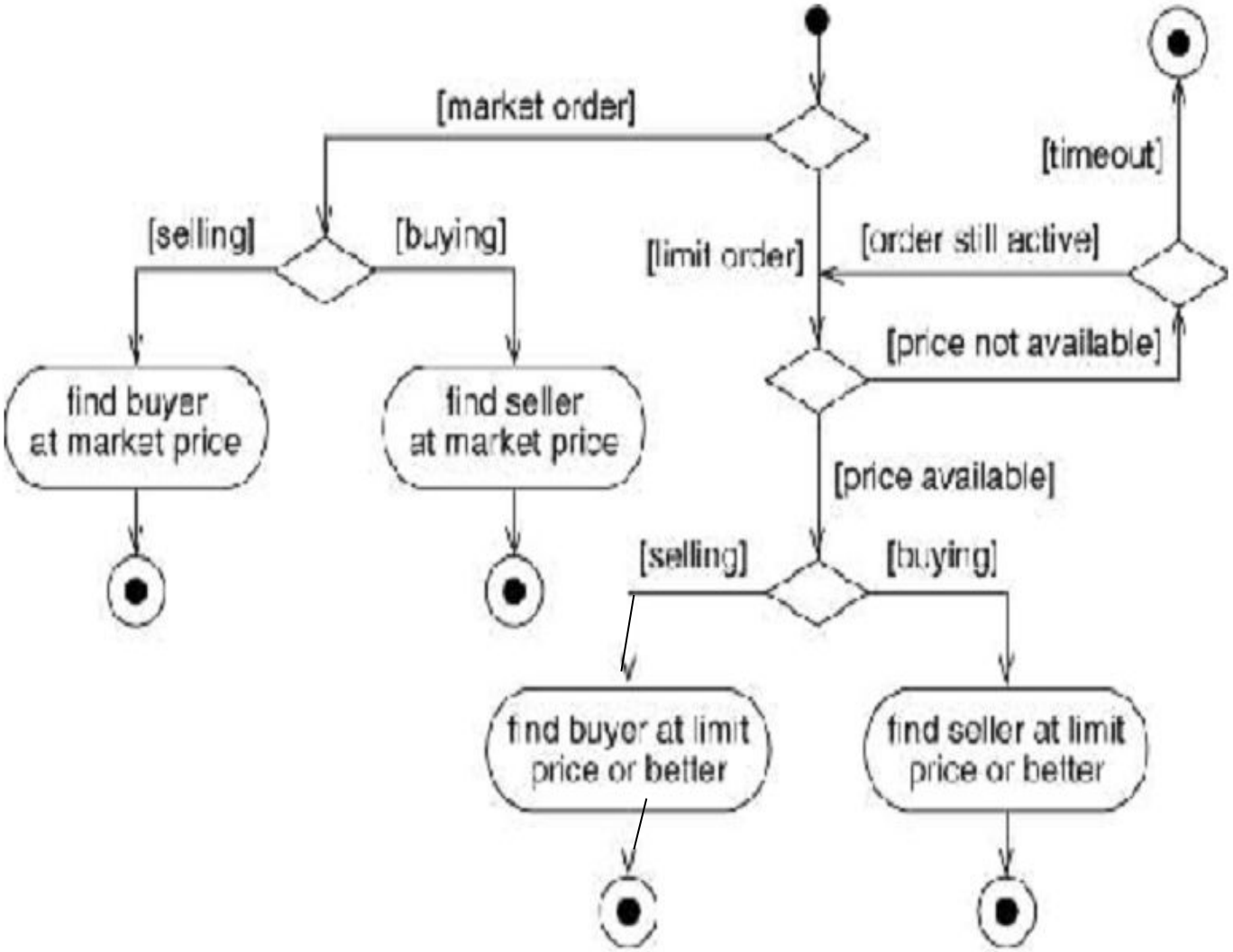

customer
sales
Warehouse
Request product
Process order
Pull material
Ship order
Receive order
Bill customer
Pay bill
Close order


customer
sales
Warehouse
Request product
Process order
Pull material
O:order
[in progress]
Ship order
Receive order
Bill customer
Object flow
Pay bill
Close order

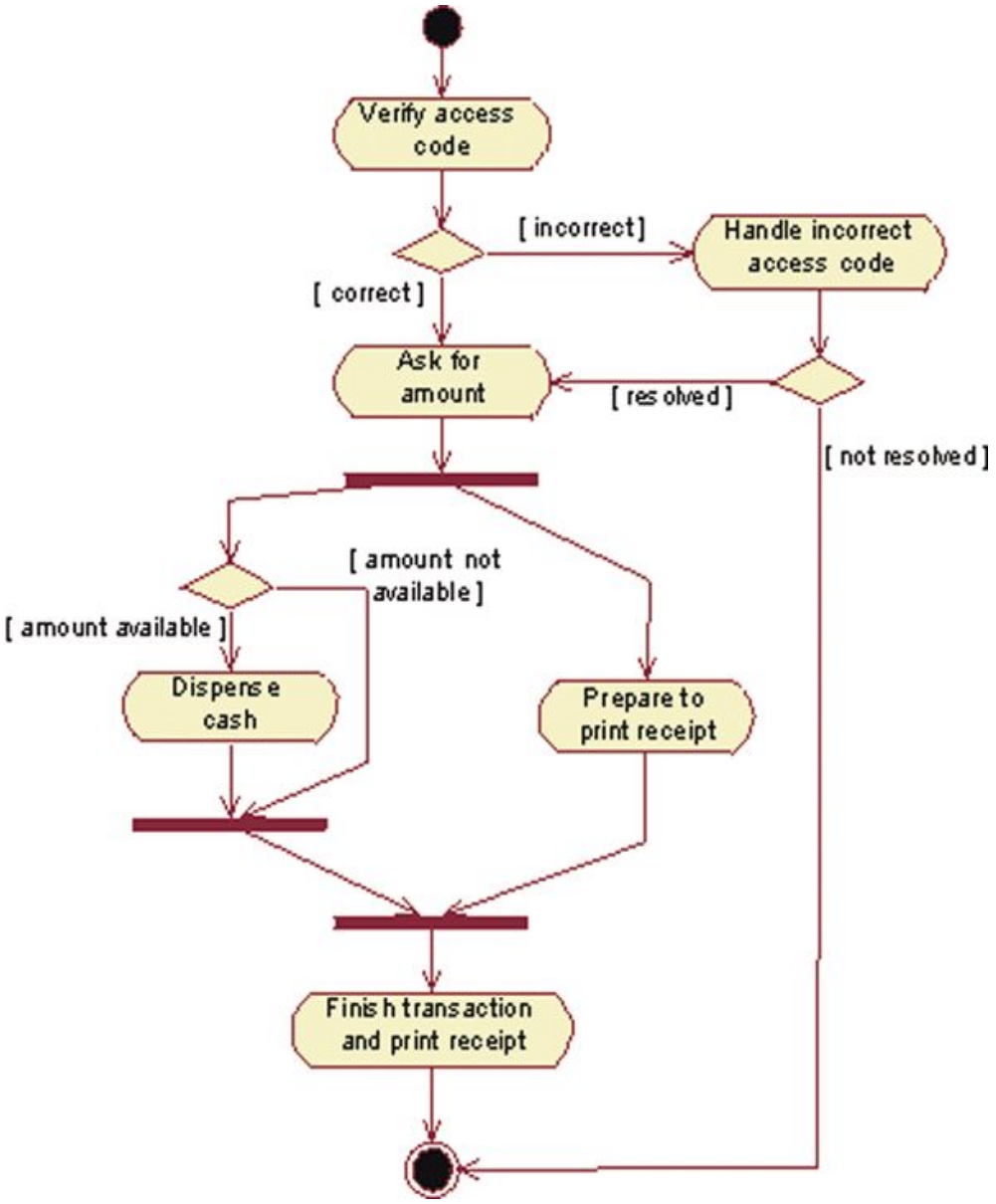
Activity diagram for stock trade processing



Activity diagram for execute order

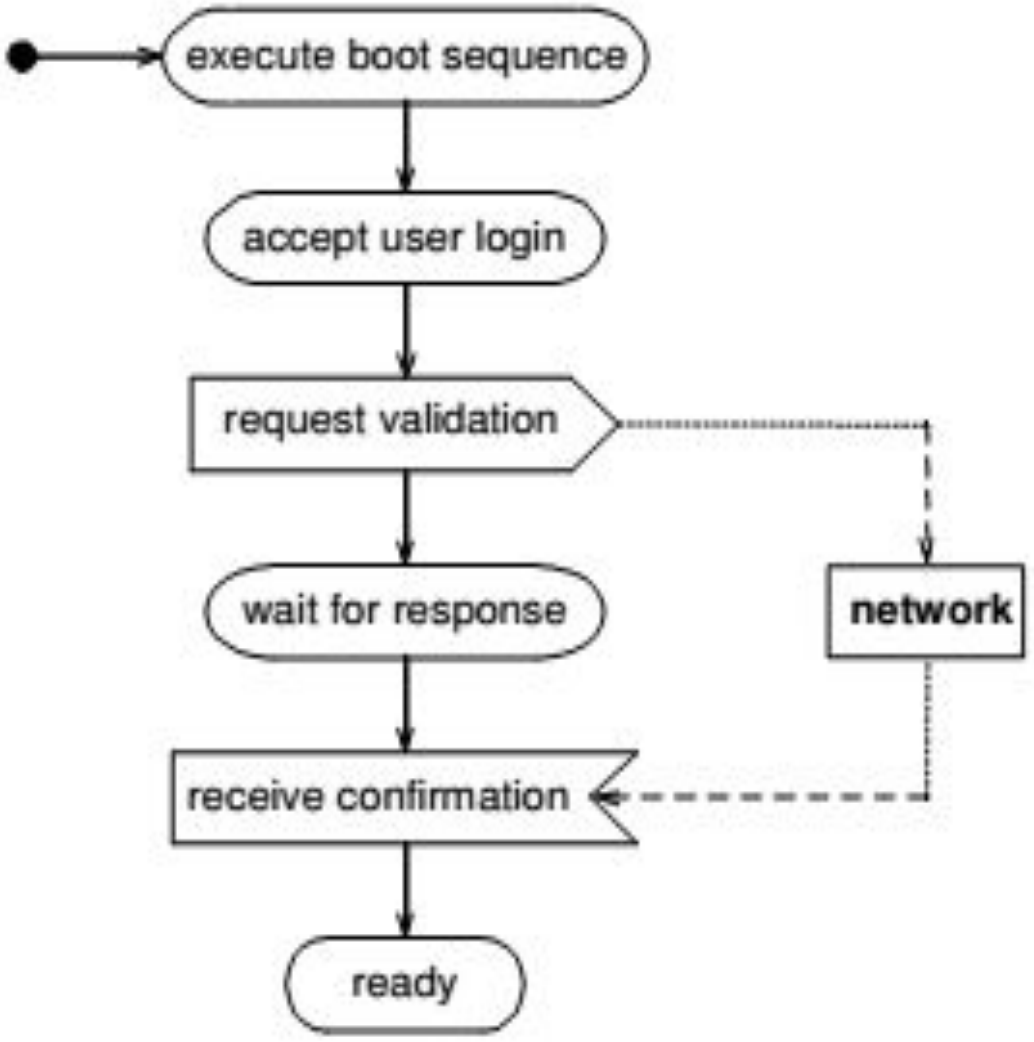


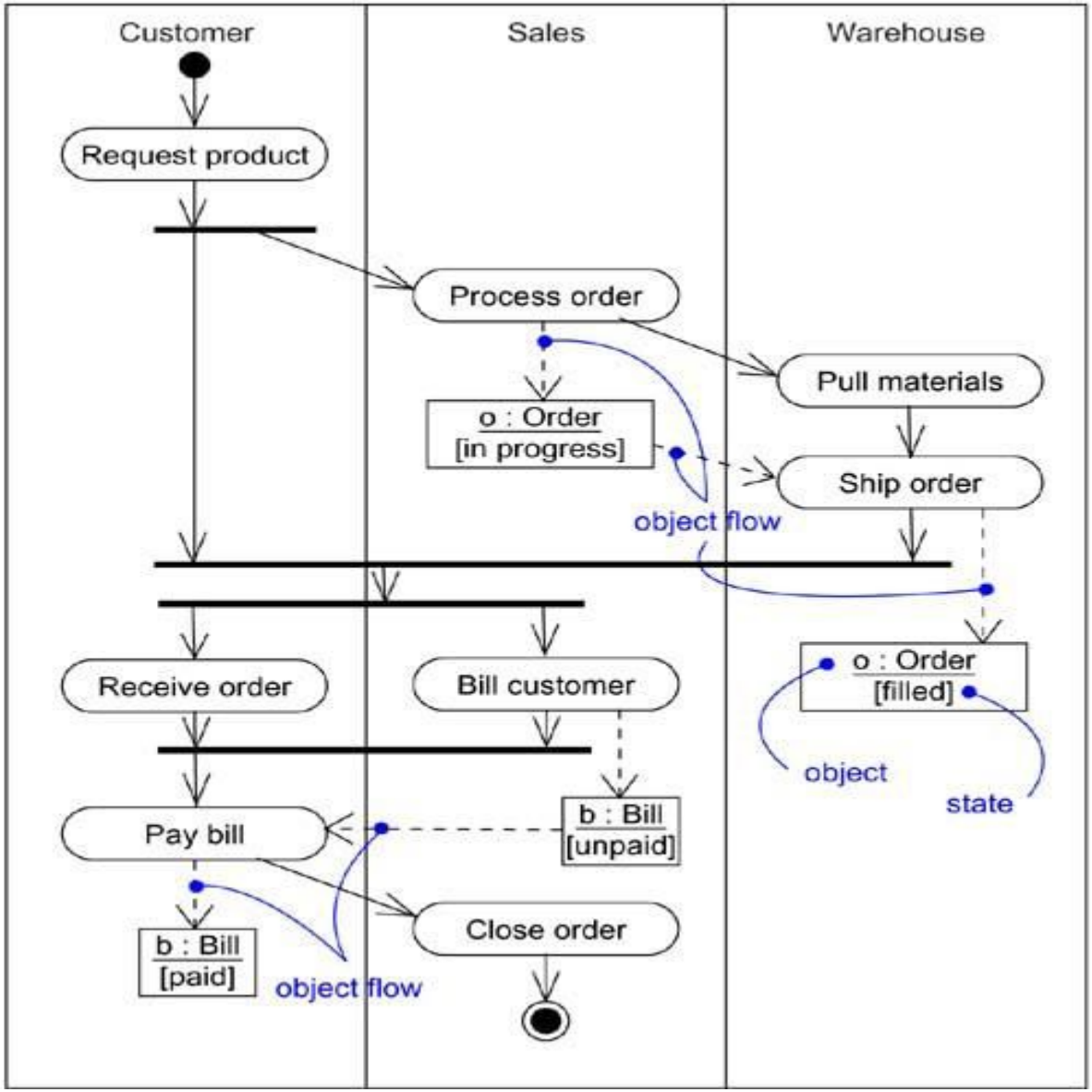
A Simplified Activity Diagram for the Use Case "Withdraw Money" in the Use-Case Model of an Automated Teller Machine (ATM)

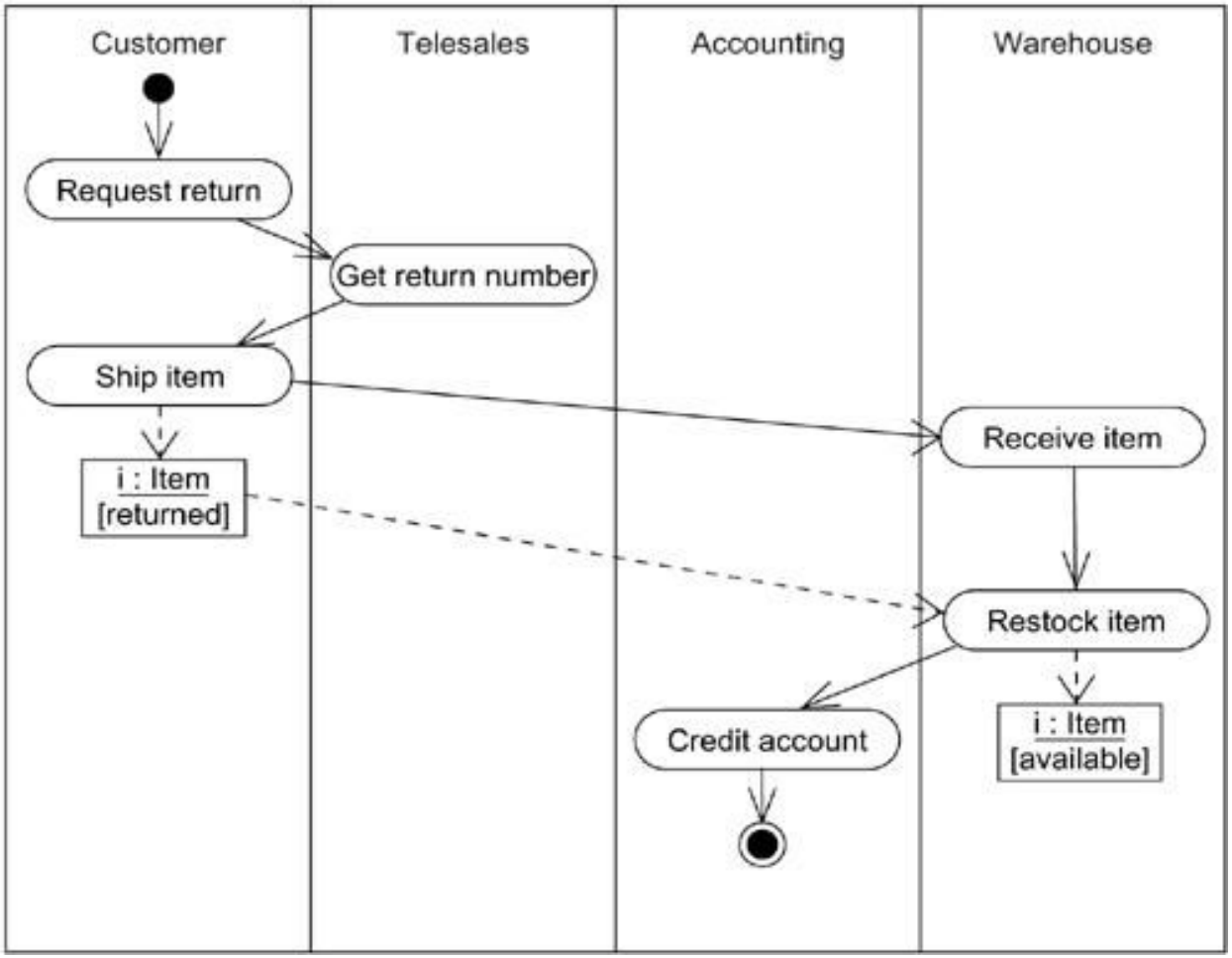


Sending and Receiving Signals

UML shows sending signals as a convex pentagon and receiving signals as concave pentagon.




Customer
Sales
Warehouse
Request product
Process order
Pull materials
o : Order
[in progress]
Ship order
object flow
Receive order
Bill customer
o : Order
[filled]
object
state
Pay bill
b : Bill
[unpaid]
Close order
b : Bill
[paid]
object flow


Customer
Telesales
Accounting
Warehouse
Request return
Get return number
Ship item
i : Item
[returned]
Receive item
Restock item
Credit account
i : Item
[available]