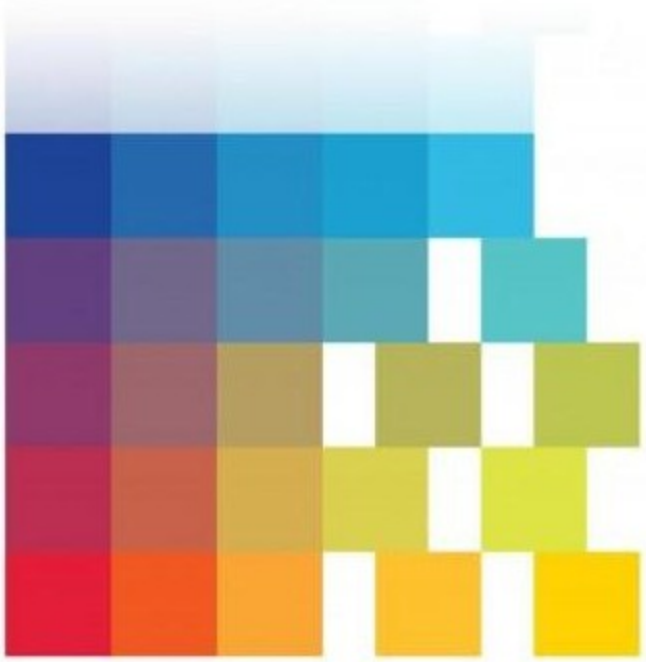


Activity Diagrams

Lecture 4

Session Outcomes

- Introduction
- Elements of Activity diagram
  - Actions
  - Nodes
  - Edges
  - controls
- Partitioning an activity diagram
- Applying activity diagrams in real world applications

What is an activity diagram?

- Models the dynamic aspects of a system.
- plays a role similar to flowcharts
- principal difference is that they support

parallel behaviour

When to Use Activity Diagrams ?

- Modeling business **processes**
- Analyzing **system functionality to** identify the use cases
- Analyzing **individual use cases in** detail
- Clarifying **concurrency issues.**

Elements of an Activity Diagram

- Activity
  - Nodes
    - Action
      - SimpleAction
      - CallAction
      - Send signal action
      - Accept EventAction
      - WaitTime Action
    - Pin
    - Object
    - Data Source
  - Activity Edges
    - Control flow edges
    - Object flow edges
    - Interrupting edges

- Activity
  - Controls
    - Start
    - End
    - Flow End
    - Nodes
      - Decision/Branch
      - Merge
      - Fork
      - Join

Activity Diagrams Notation

- Initial & Final Nodes

  Beginning & end of diagram

  Final node is within circle

- Actions

  Rounded rectangles

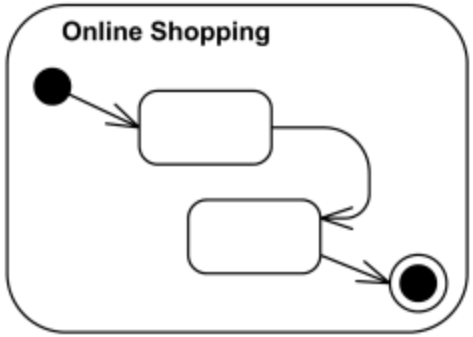
  An activity is a sequence of actions

- Transitions-Directed arrows (flow/edge)

  Show sequence of activities

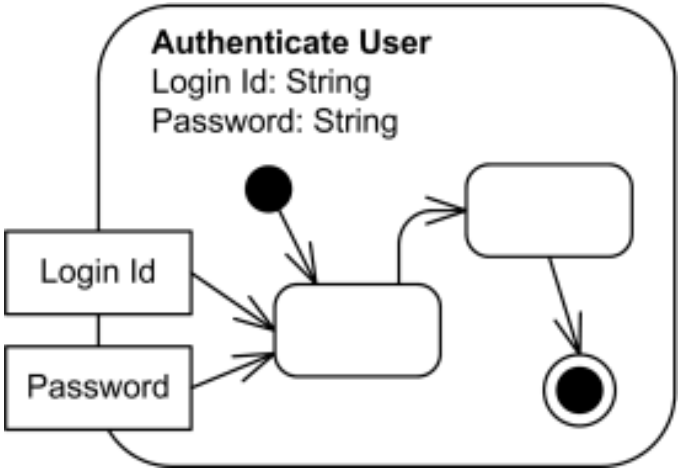
Activity

- **Activity** is a parameterized behavior represented as coordinated flow of actions.
- Shows the flow of execution.
- Activity could be rendered as round-cornered rectangle with activity name in the upper left corner and nodes and edges of the activity inside.



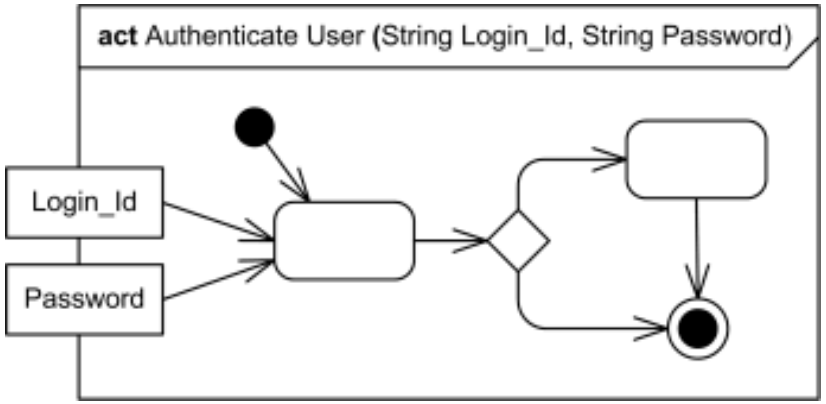
Activity… UML 2.4

- Activity parameters are displayed on the border and listed below the activity name as: ***parameter-name*: *parameter-type***.

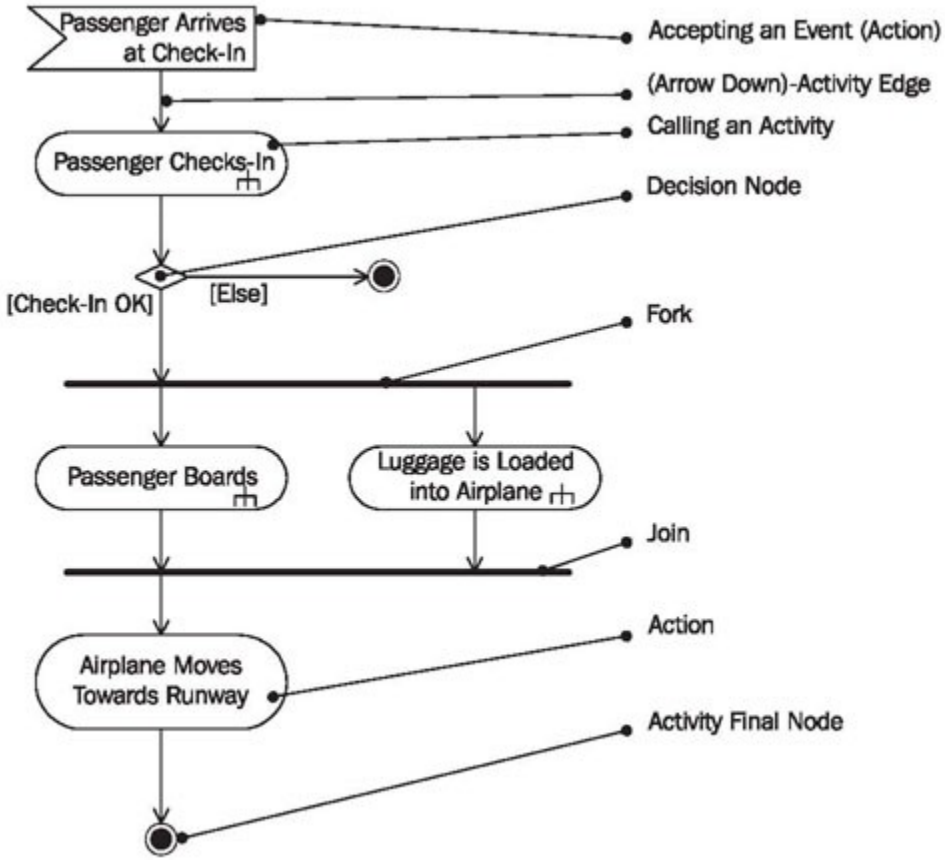


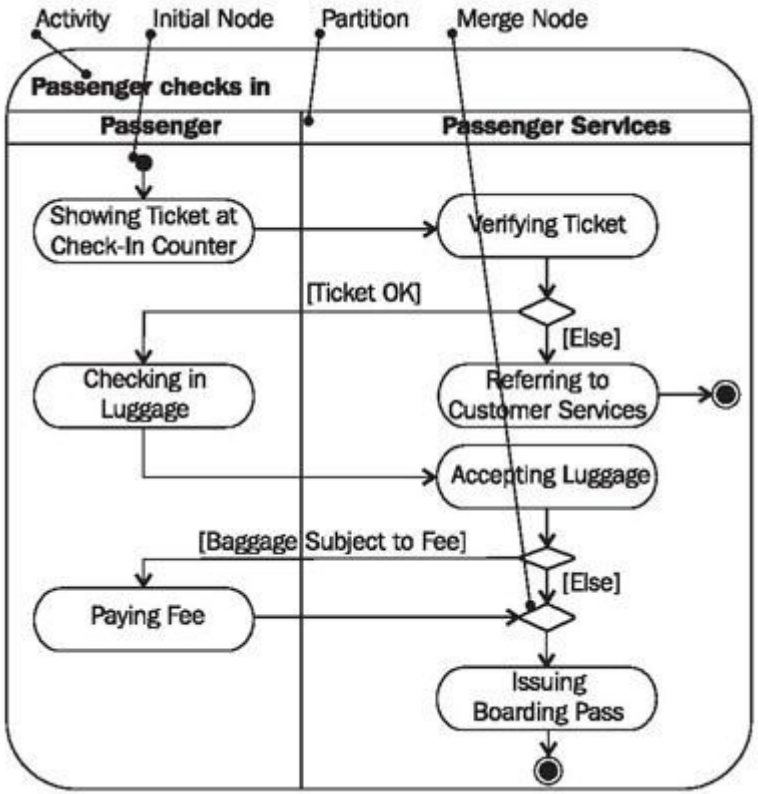
Activity… UML 2.4

- The round-cornered activity border may be replaced with the diagram frame. The kind of the frame in this case is **activity** or **act** in short form. Activity parameters if any are displayed on the frame.



Activity an example





ref:http://sourcemaking.com/uml/modeling-business-systems/external-view/activity-diagrams

Activity

- An Activity contains **actions.**
- **Actions** can be of various kinds:
  - Occurrences of primitive functions, such as arithmetic functions.
  - Invocations of behavior.
  - Communication actions, such as sending of signals.
  - Manipulations of objects, such as reading or writing attributes or associations.

Action types (1) – Action node

- SimpleAction

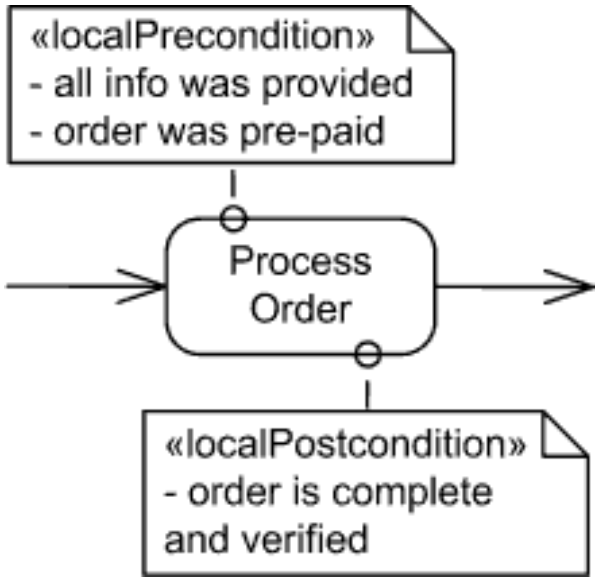
Fill Order

- **Action** could be expressed in some application-dependent **action language.**

```
For (Order o:order)
    o.fillOrder();
end_for
```

Pre and Post conditions of Actions

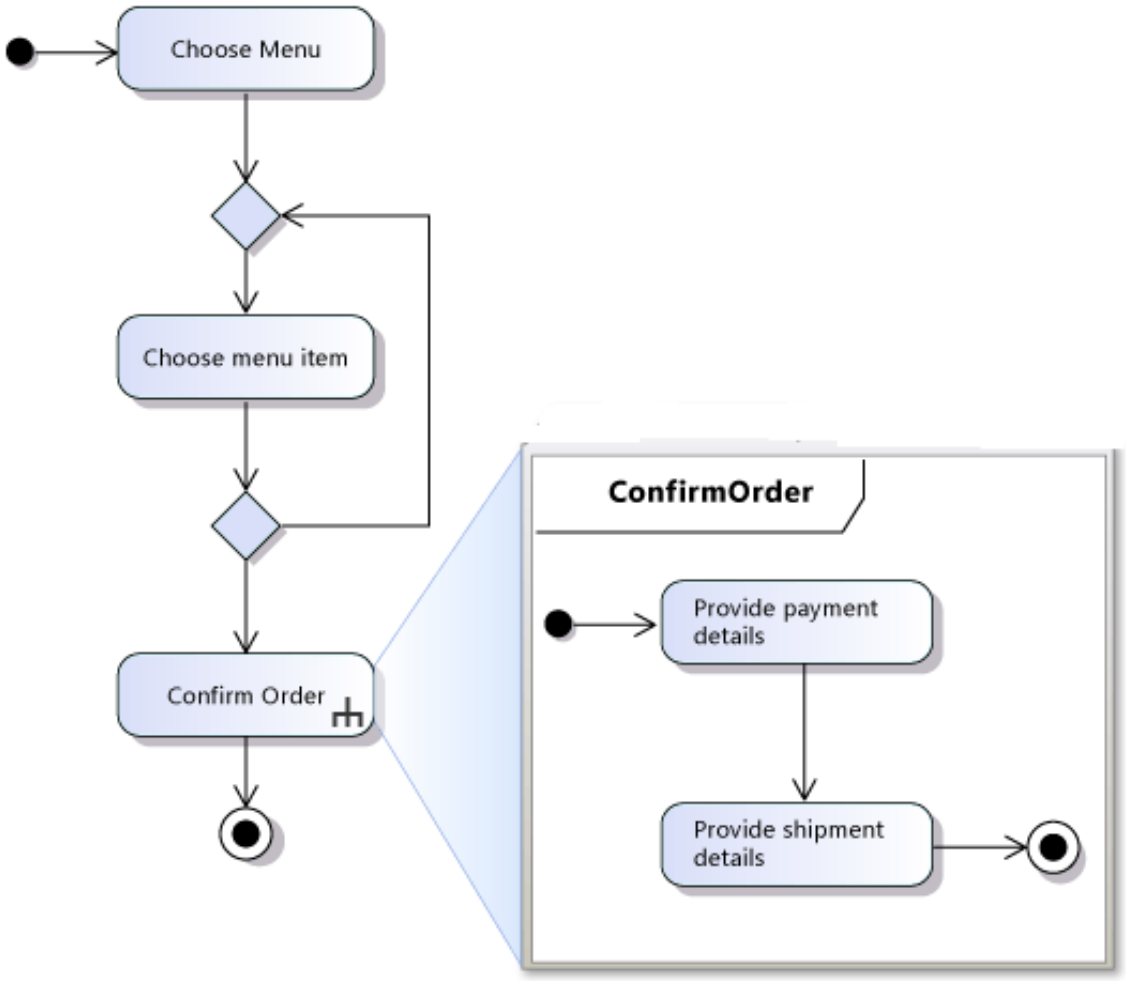
- **Shown as notes**



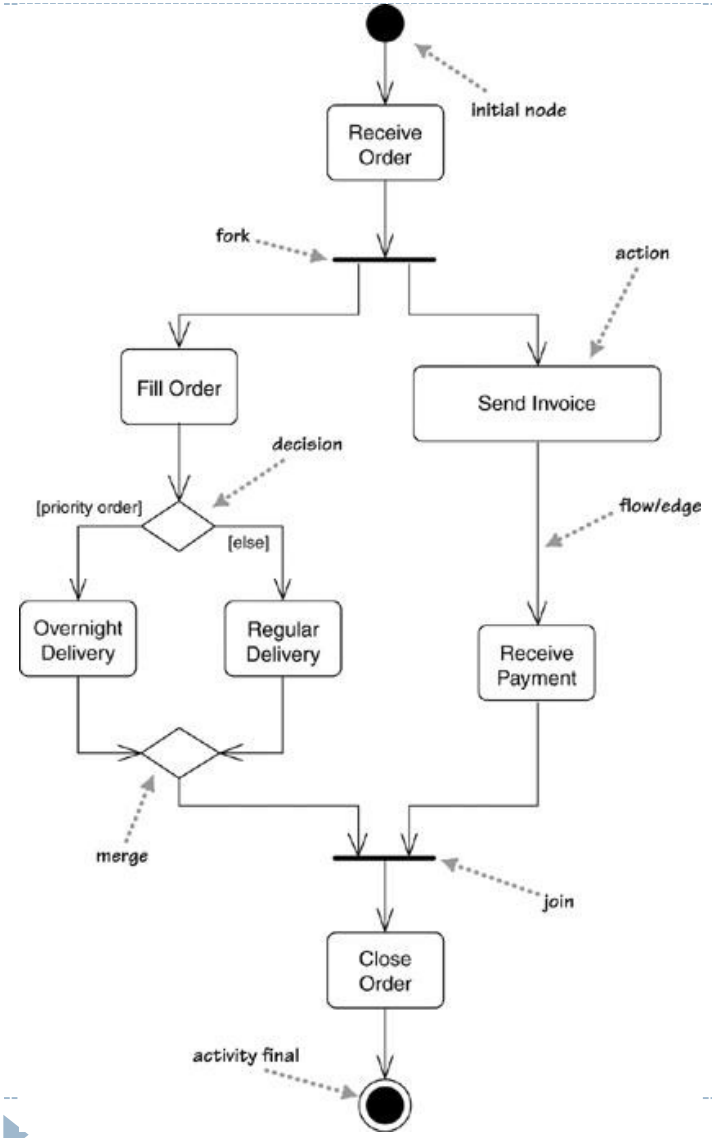
- **Local pre-conditions** and **local post-conditions** are shown as **notes** attached to the invocation with the keywords «localPrecondition» and «localPostcondition», respectively.

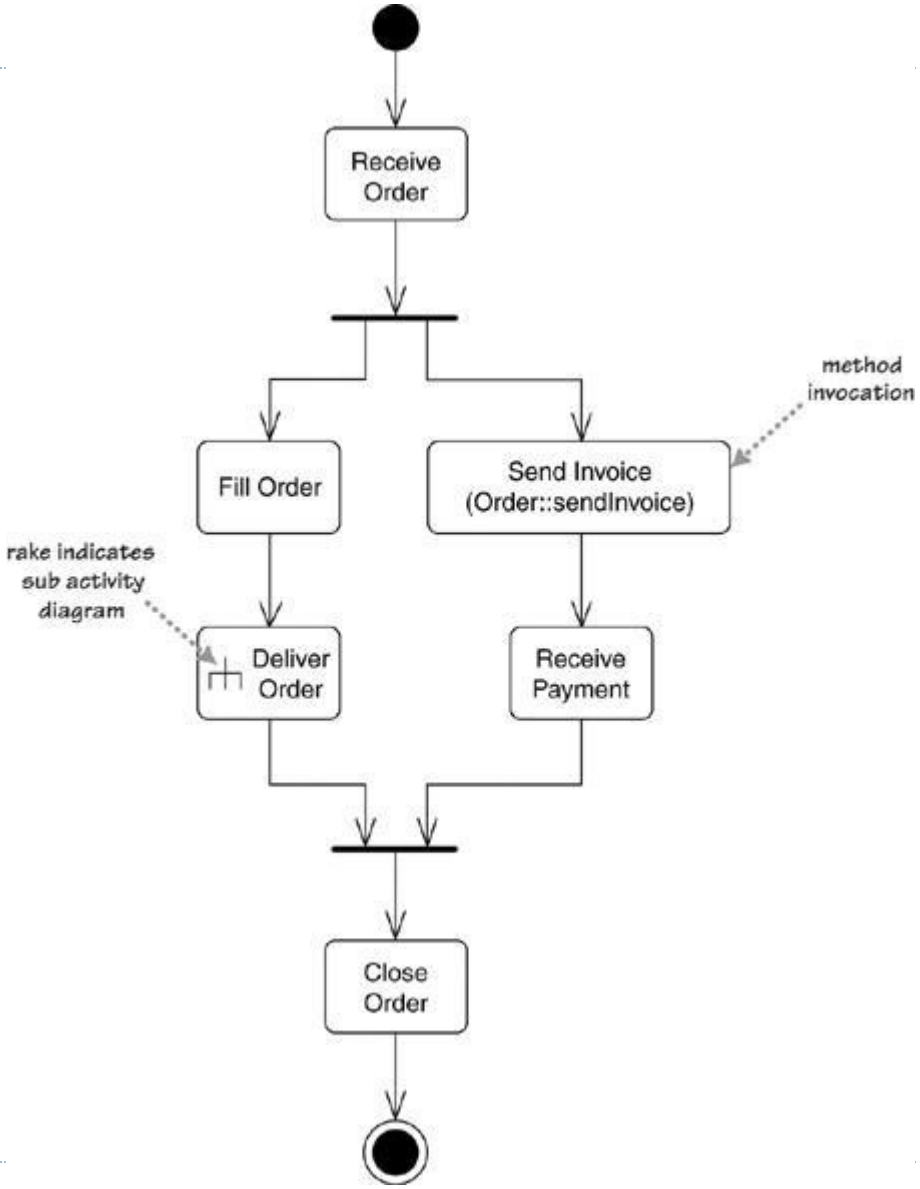
Action types (2) – Call activity action /Sub Activity

Call activity action is an action that is defined in more detail on another activity diagram. It is indicated by a rake-style symbol within the action symbol

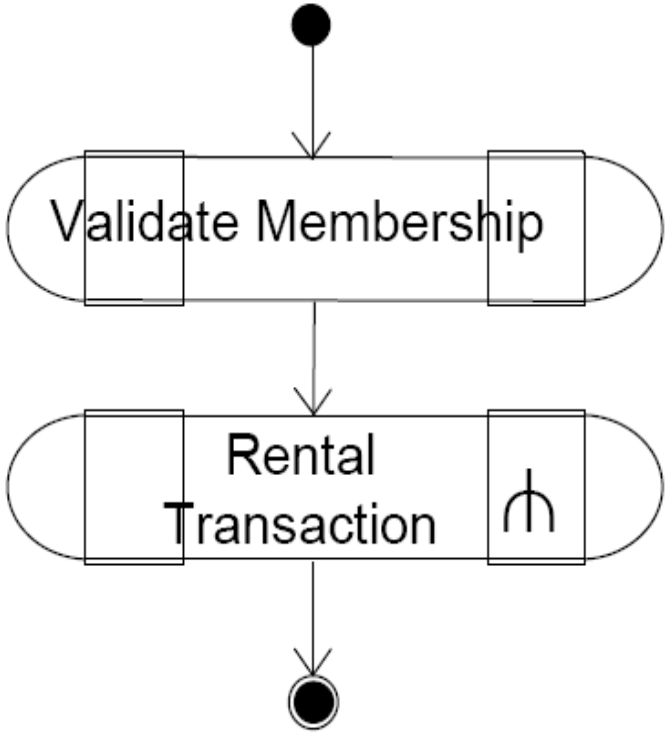


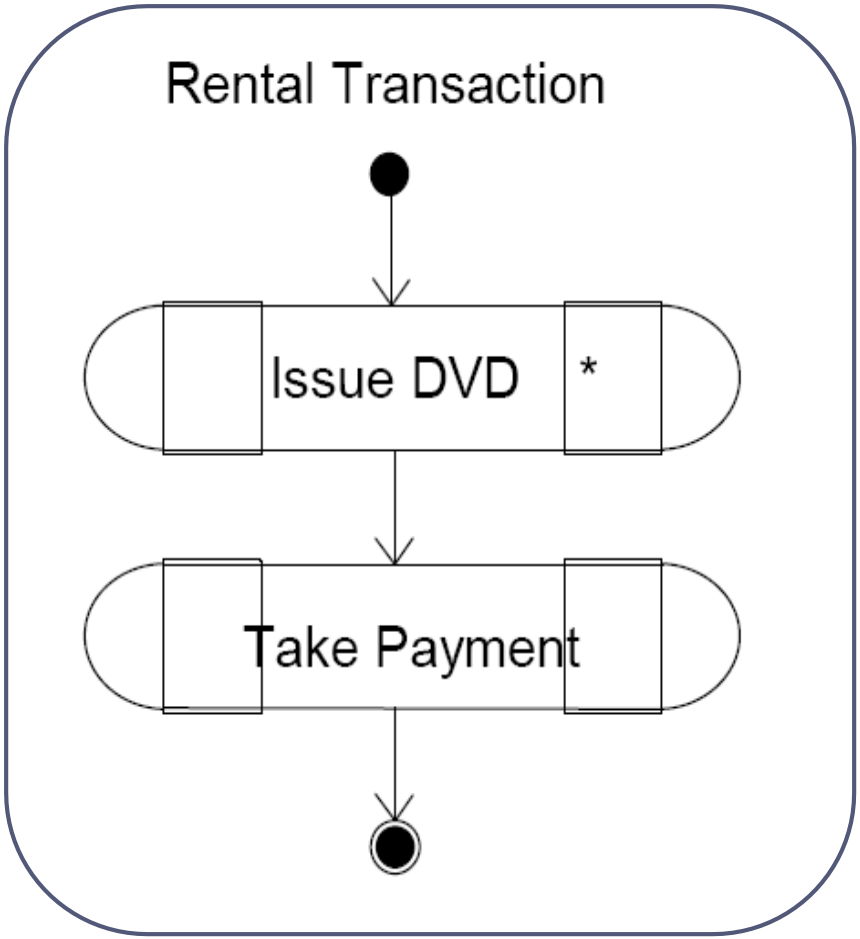
Call activity action : Example 1





Call activity action :
Example 2



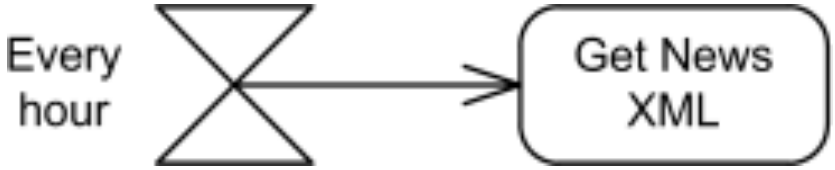


Signal Actions

- A signal indicates that the activity receives an event from an outside process. This indicates that the activity constantly listens for those signals
  - **Send signal action -** indicates that a signal or message is sent to other activities or processes.

  - **Accept Event Action -** waits for some external event or incoming message.

  - **Wait Time Action -** represents the acceptance of a time event



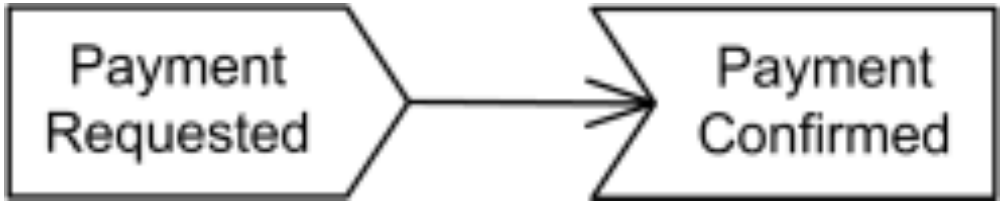
Action types (3) - Send Signal Action

- **Send signal action** is an invocation action that creates a signal from its inputs, and transmits it to the specified **target** object, where it may cause the firing of a state machine transition or the execution of an activity.
- When all the prerequisites of the action execution are satisfied, a signal is generated from the arguments and is transmitted to the identified target object.

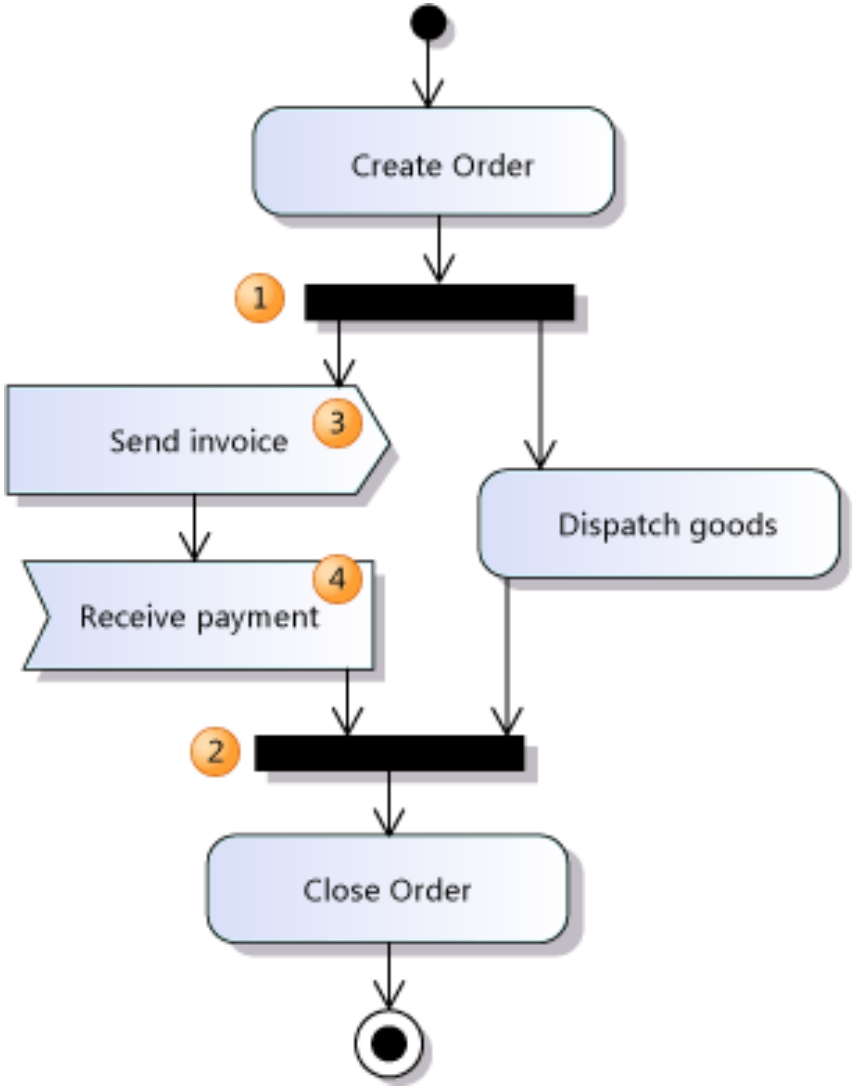
Action types (4) – Accept Event Action

- If an accept event action has **no incoming edges**, then the action starts when the containing activity or structured node does, whichever most immediately contains the action.
- In addition, an accept event action with no incoming edges remains enabled after it accepts an event. It does not terminate after accepting an event and outputting a value, but continues to wait for other events.

- **Accept event action** could have **incoming edges.** In this case the action starts after the previous action completes



Signal Actions – Example 1



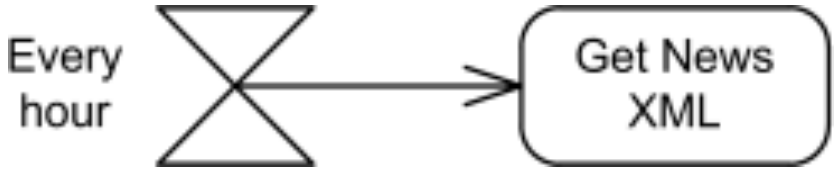
Signal Actions – Activity 1

- Amara decided on going to Australia next week.
  1. He reserves an itinerary through his travel agent
  2. Amara sends all the details of his itinerary together with the confirmation to the travel agent
  3. When the travel agent gets the confirmation, he books the itinerary for Amara.

Draw the partial activity diagram for this scenario.

Action types(5) – Wait Time Action

- If the event is a **time event occurrence**, the result value contains the time at which the occurrence happened. Such an action is informally called a **wait time action**.



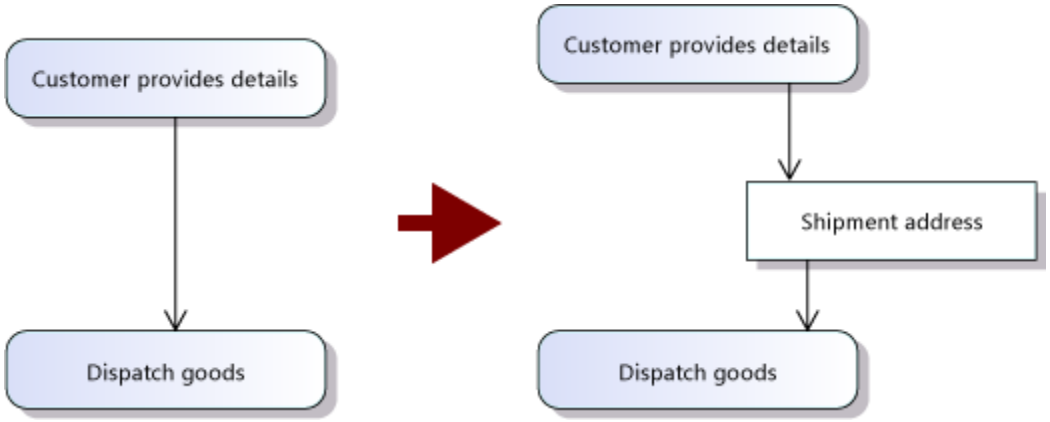
Signal Actions – Activity 2

- Amara thought about the things he has to do for the trip and came up with the following plan
  1. Pack the bags two hours before the flight
  2. Leave for airport.

Draw the partial activity diagram for this scenario.

Other Nodes – Object Node

- An Object Node represents something that stores one or more values that are passing from one action to another.

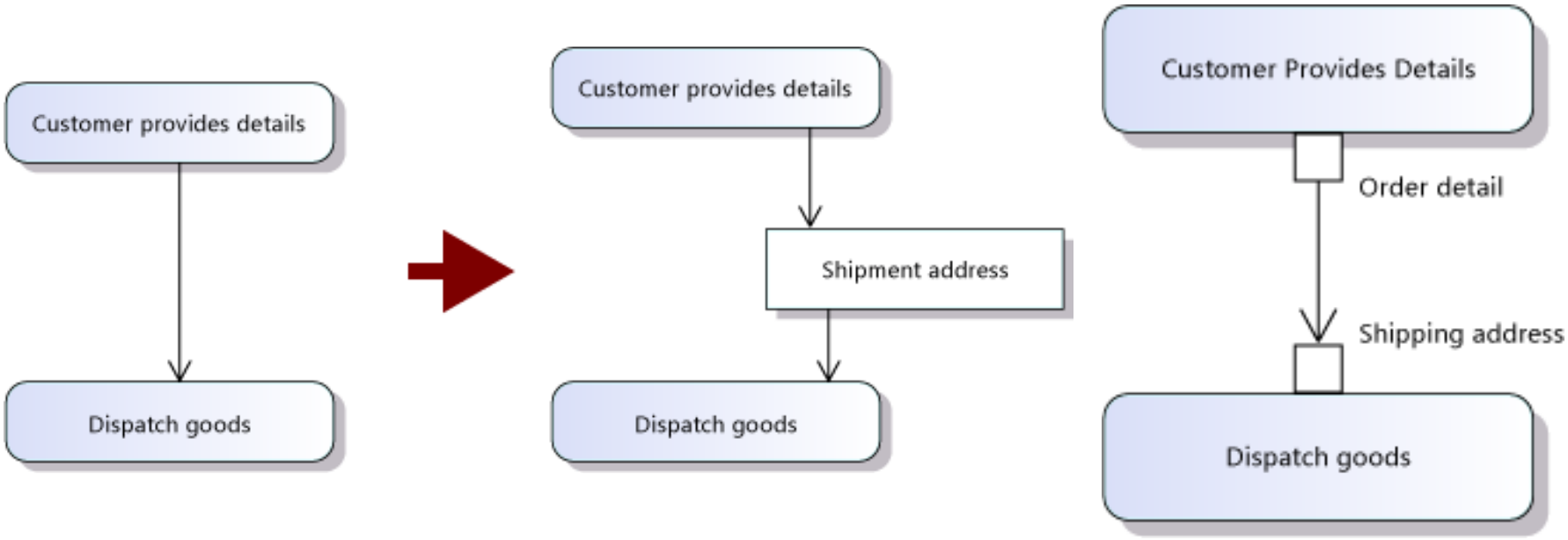


ref:http://msdn.microsoft.com/en-us/library/vstudio/dd409465.aspx

Other Nodes – Pin

- A **pin** is an **object node** for inputs and outputs to **actions**.
- Pin is usually shown as a small rectangle attached to the action rectangle. The name of the pin can be displayed near the pin.

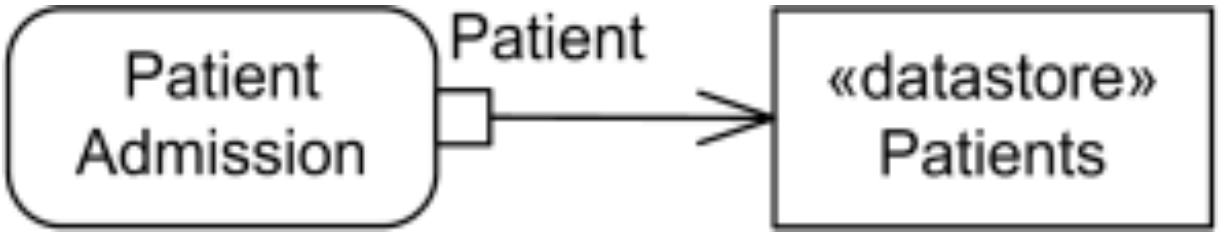
Pin - example



ref:http://msdn.microsoft.com/en-us/library/vstudio/dd409465.aspx

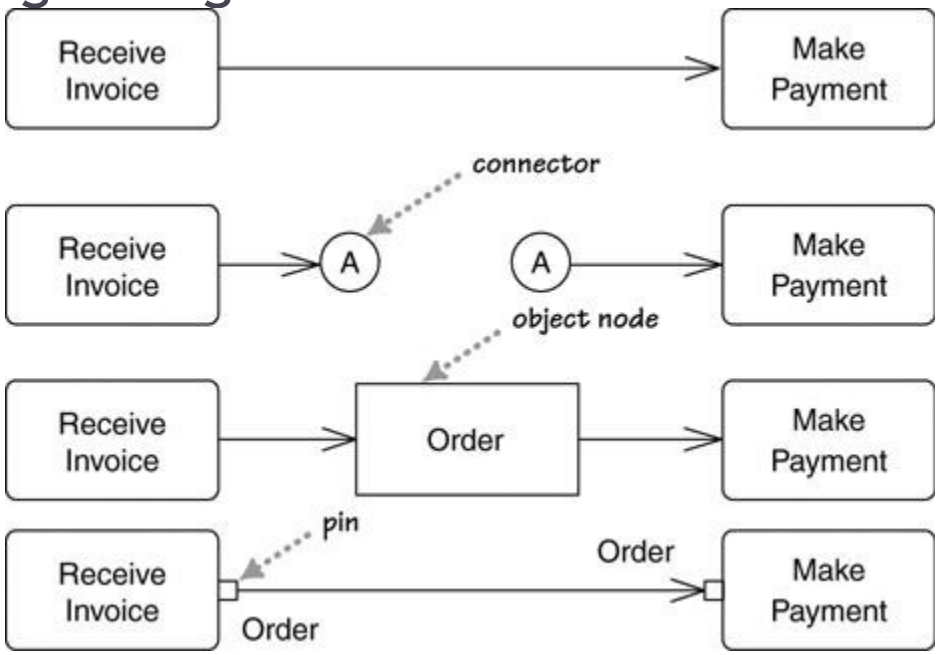
Other Nodes – Data Store

- A **data store** is a central buffer node for non-transient information.
- The data store is notated as an object node with the keyword «datastore».



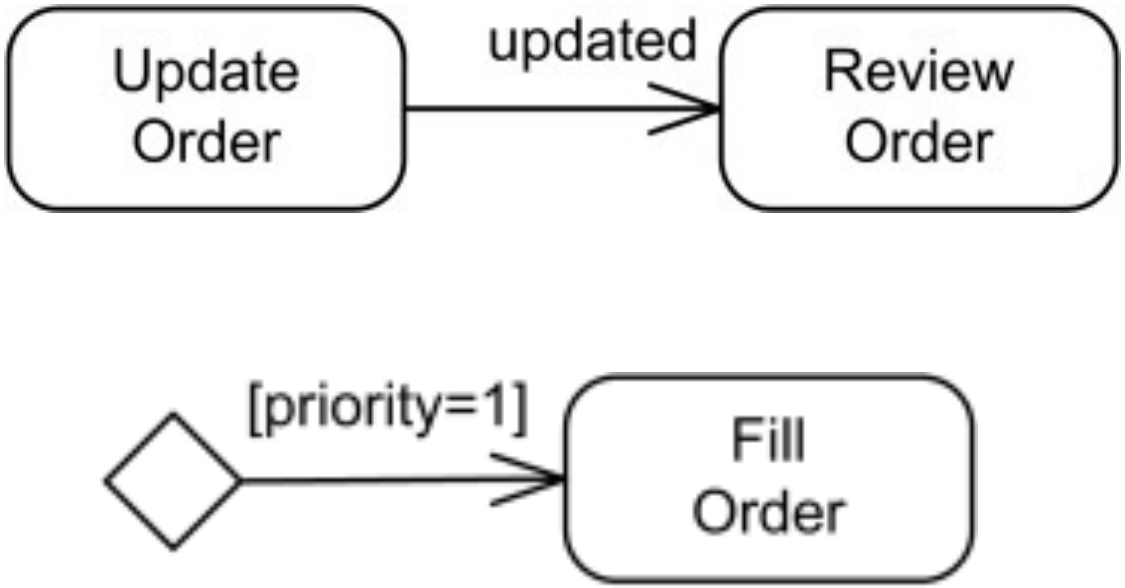
Activity Edges

- Edge is a directed connection between actions.
  - Four ways of showing an edge



- There are 3 types,
  - Control flow edges
  - Object flow edges
  - Interrupting edges

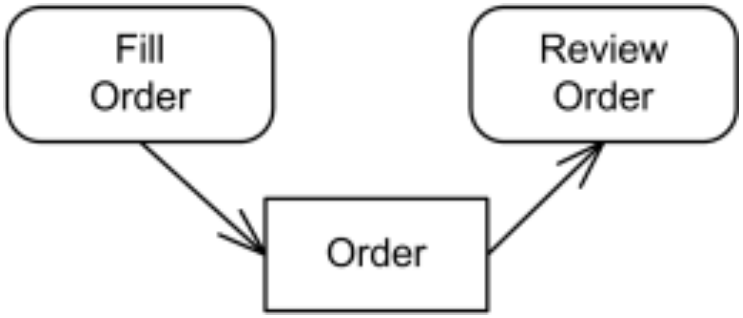
Control Flow Edge



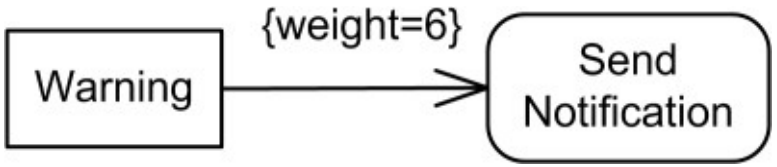
- Used to show the execution control.
- Edges can be named.
- Edge contains a guard conditions (In a selection)

Object Flow Edge

- Used to show data flow of object and data tokens between action nodes.
- Object flow edges have no specific notation.

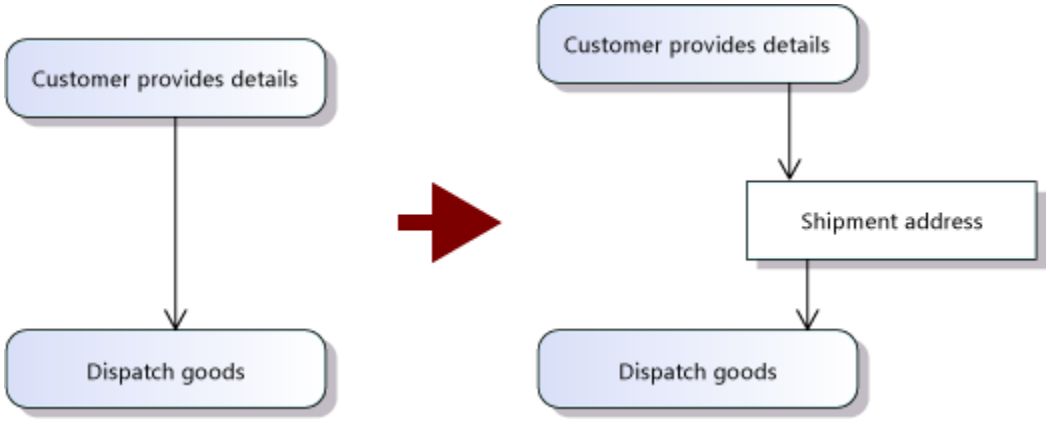


Order is a token sent between Fill order and Review order



Weight is a value specification (variable) – When Number of warnings are = 6 , send notifications

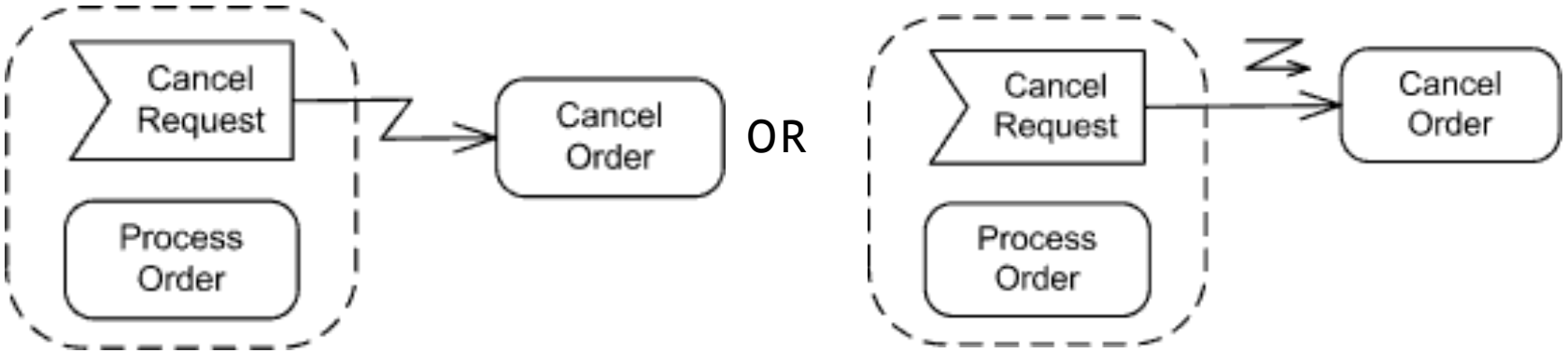
Object Flow Edge - example



ref:http://msdn.microsoft.com/en-us/library/vstudio/dd409465.aspx

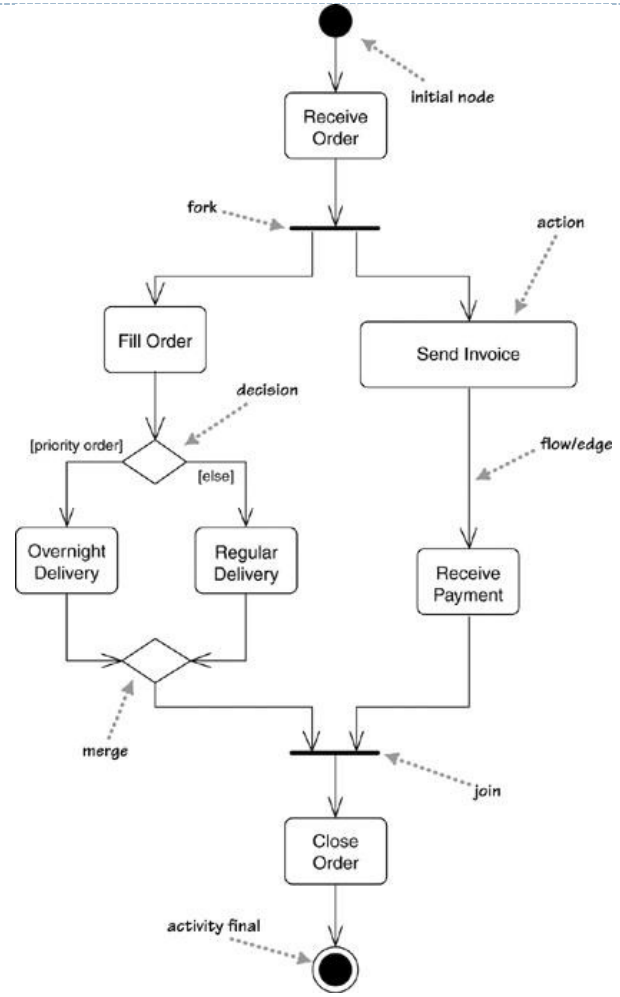
Interruption Edge

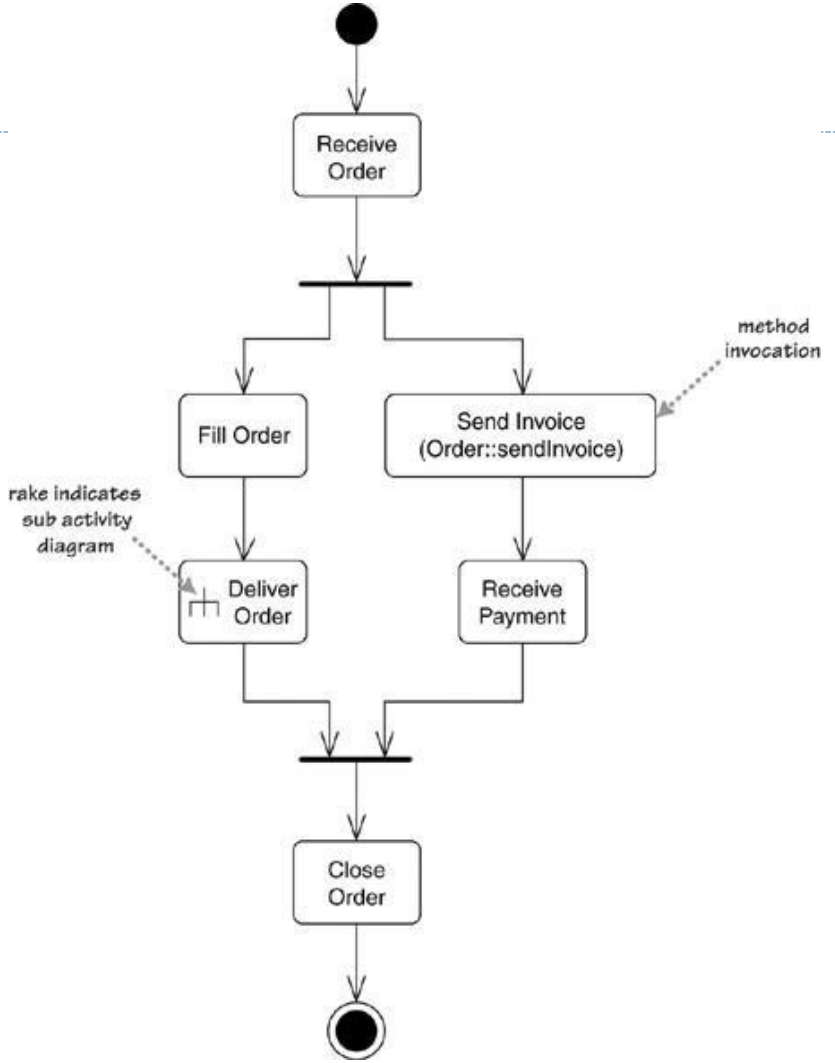
- Express interruption for regions that has interruptions. It is rendered as a lightning-bolt



Cancel Request signal causes interruption resulting in Cancel Order.

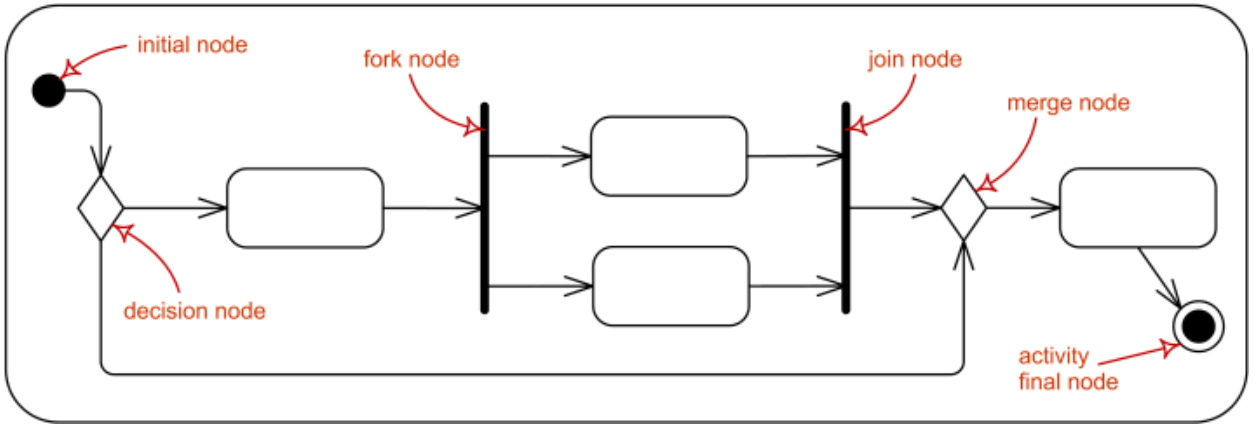
Activity 3

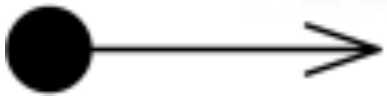




Draw the Delivery Order sub activity diagram assuming you are taking the Order object as the input parameter.

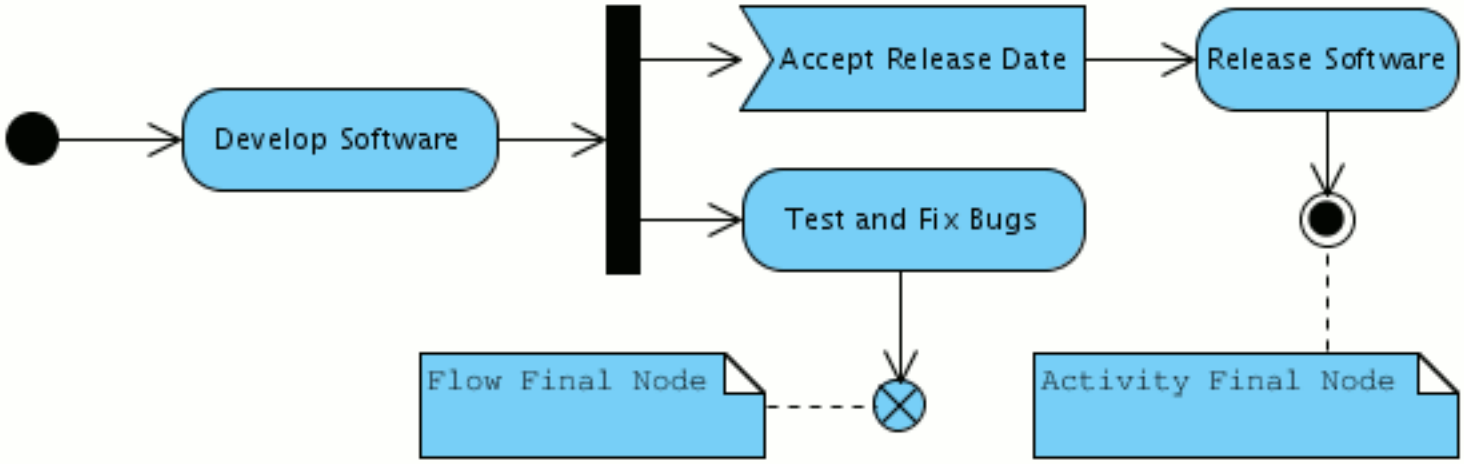
Activity Diagram Controls – start, end and flow end

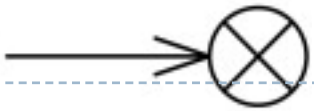




Start /Initial node

End /Termination node

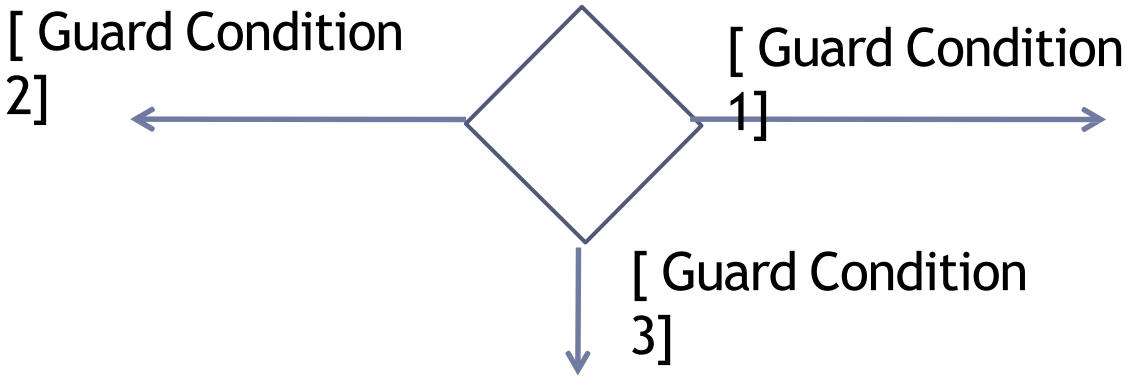




Flow end / Termination of flow

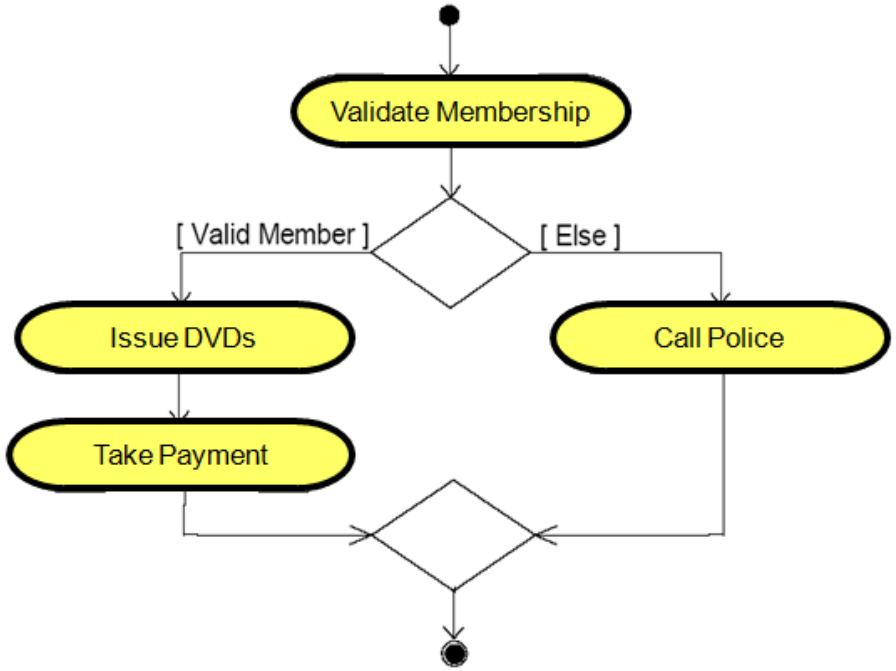
Activity Diagram Controls – Nodes
Decision Node

- A branch represents a conditional flow of control.
- Each branch must have a **guard condition.**
- The **flow of control** flows down the **single** path where the branch condition is true.
- There is **no limit** on the number of **branches** each branch point may have.



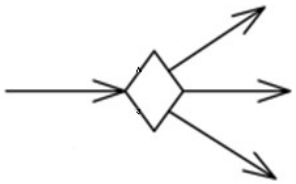
Decision Node

- The **'else' branch**:
  - always has an [else] guard expression;
  - is taken if all the other guard conditions are false;
  - is analogous to the DEFAULT option in SWITCH statement.

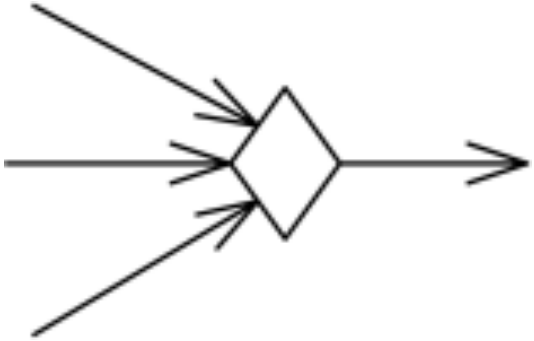


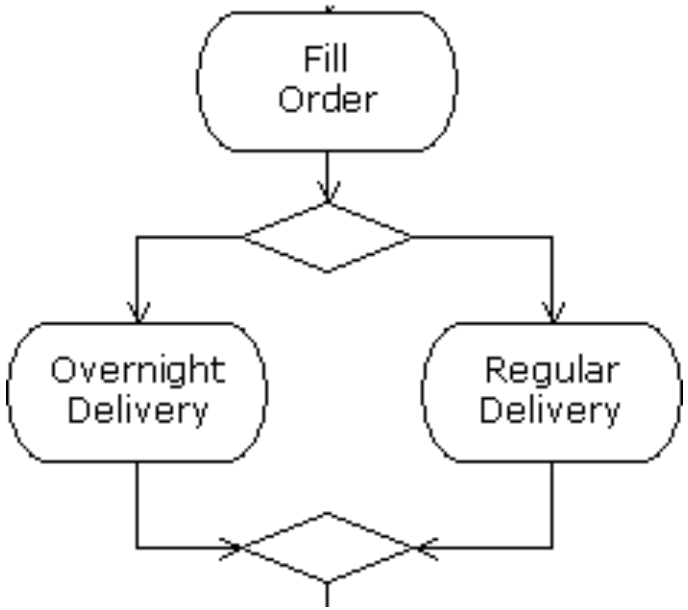
Activity Diagram Controls - Nodes
Decision and Merge Nodes

Decision node



Merge node





Every Decision node should have a merge node

Activity Diagram Controls - Nodes Fork and Join

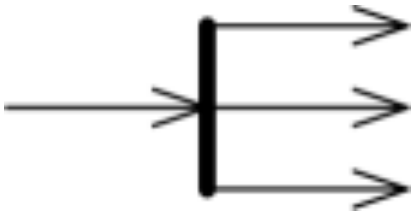
Fork node

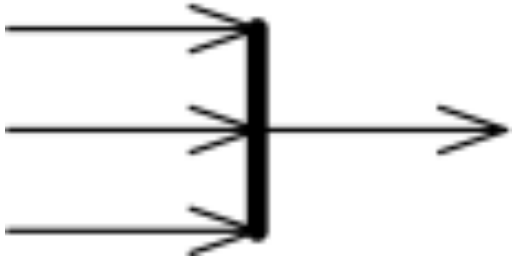
Has one incoming edge and multiple outgoing edges and is used to split incoming flow into multiple concurrent flows.

Join node

used to synchronize incoming concurrent flow

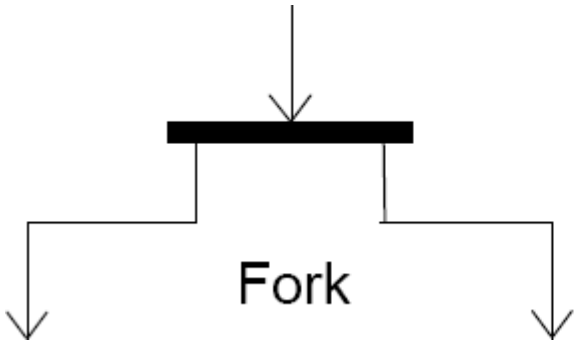
Both Fork and Join nodes are introduced to support **parallelism** in activities

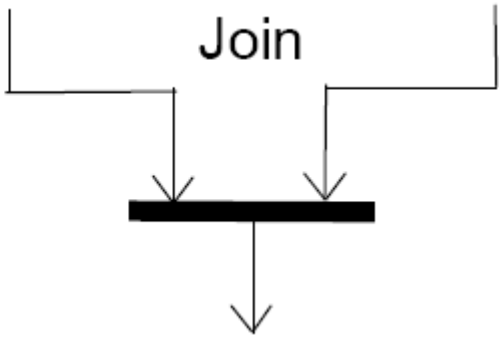




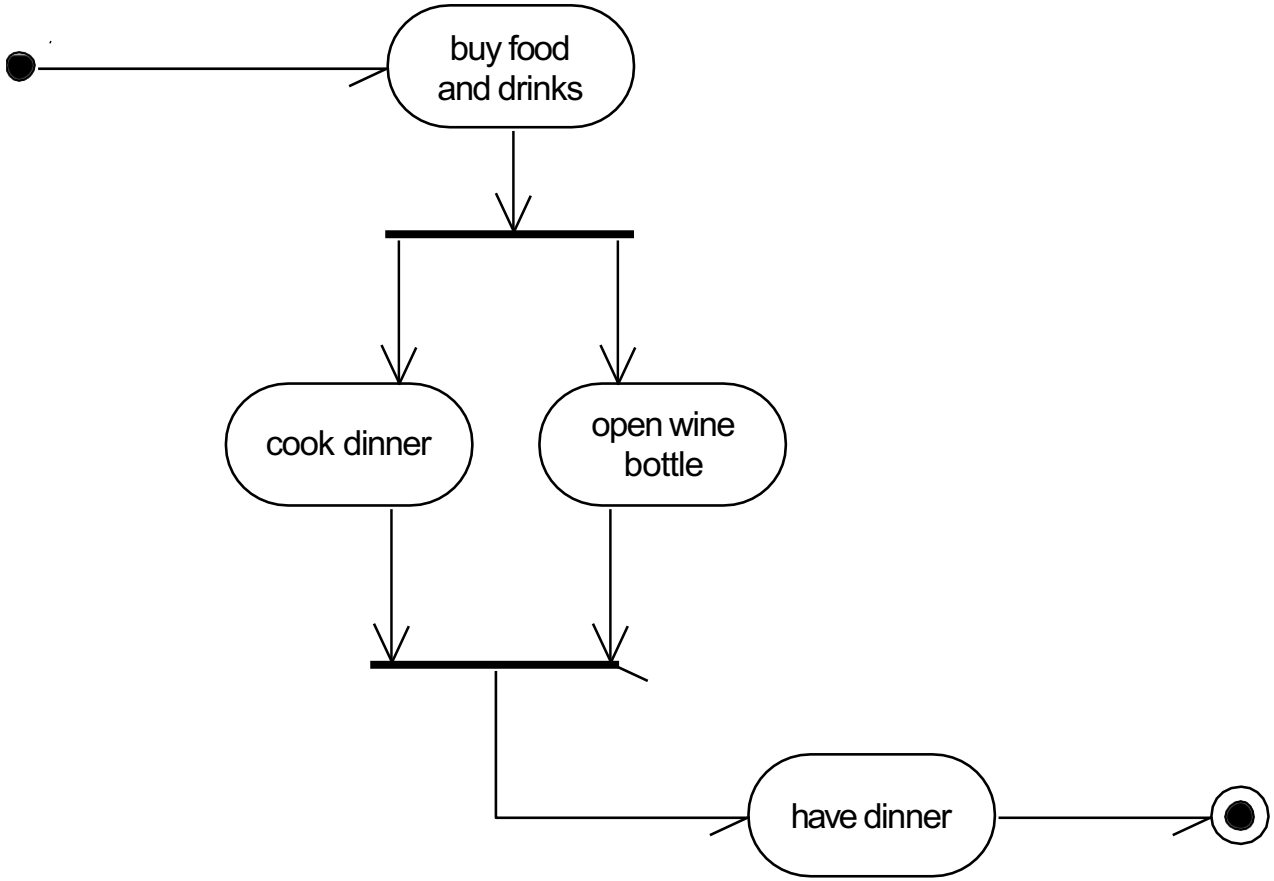
Fork and Join

- Forks and joins are used to showing activities that can occur at the **same time** (in **parallel**).
  - this does not mean that the activities ***must*** occur concurrently in the finished software system;
  - it means that the **order of execution of the activities can be whatever** is convenient for the implementation.





Fork and join example



Activity 4

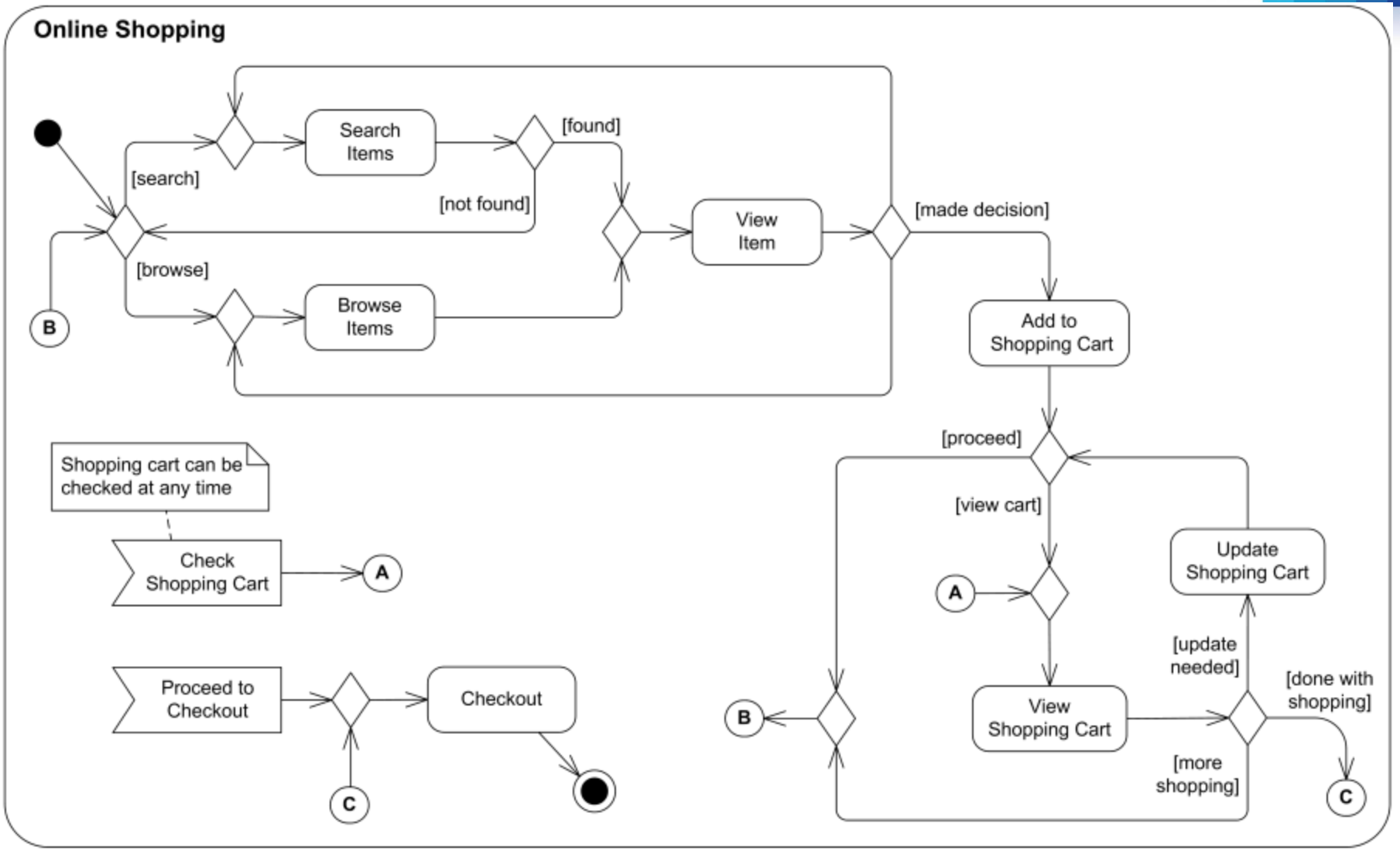
- Amara decided to go to the airport in a taxi. So the initial plan is changed as
  1. Pack the bags two hours before the flight
  2. Once the taxi arrives, leave for airport.

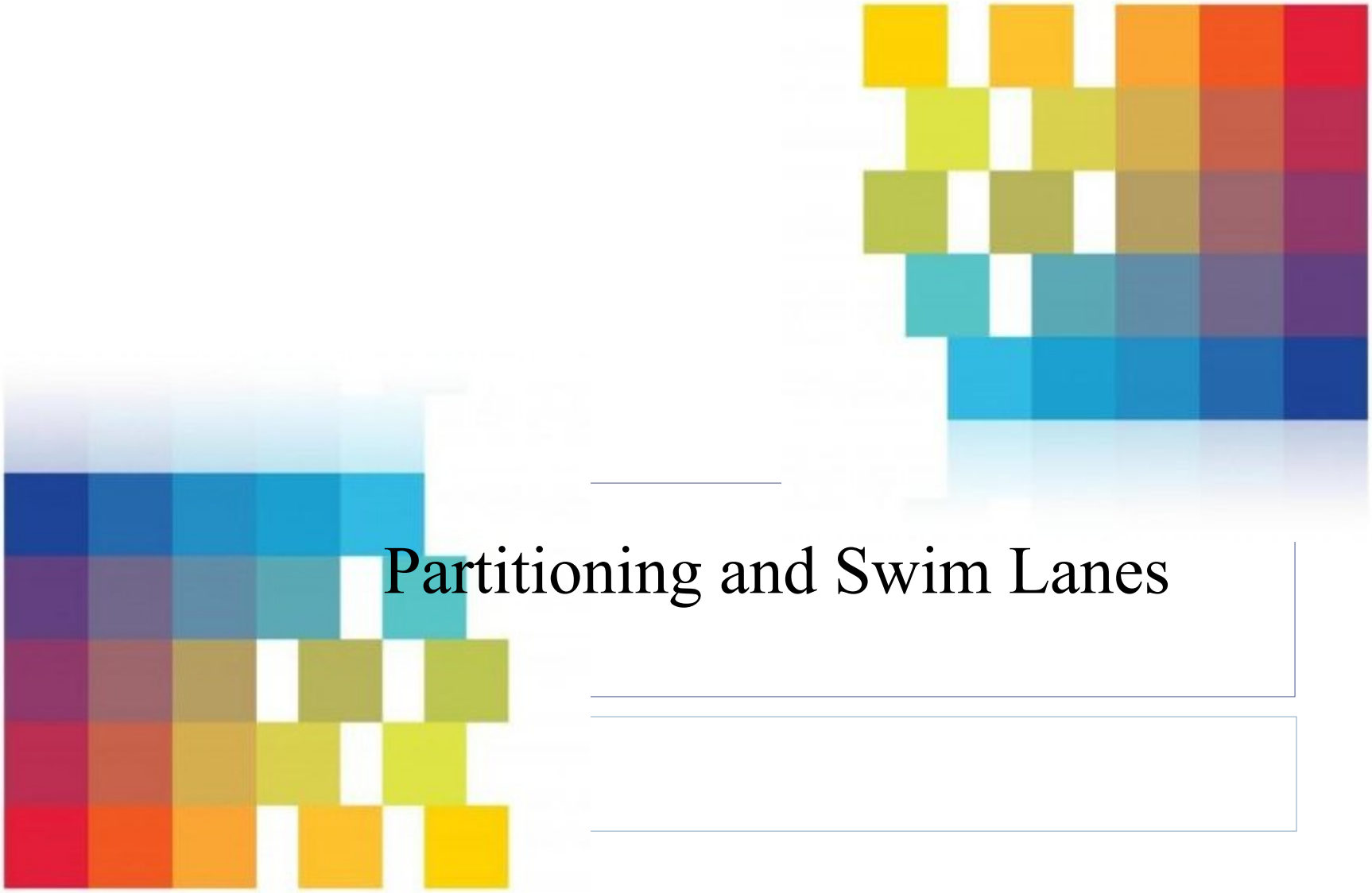
Draw the partial activity diagram for this scenario.

Iteration

- An **asterisk** inside an action state indicates that it may need to be **performed more than once:**
  - This notation is used to show iteration, without the unnecessary details of a loop construct.
- The next action state does not occur until the loop is finished.

Activity Diagram Example





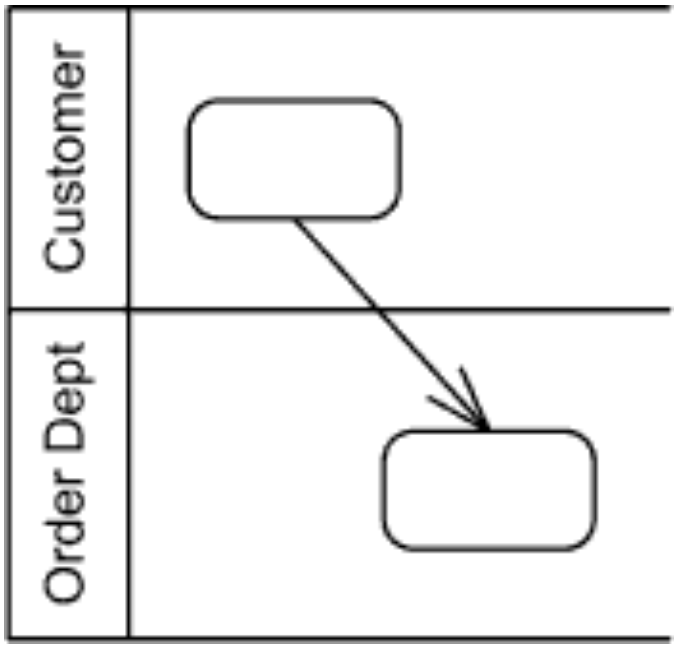
Partitioning and Swim Lanes

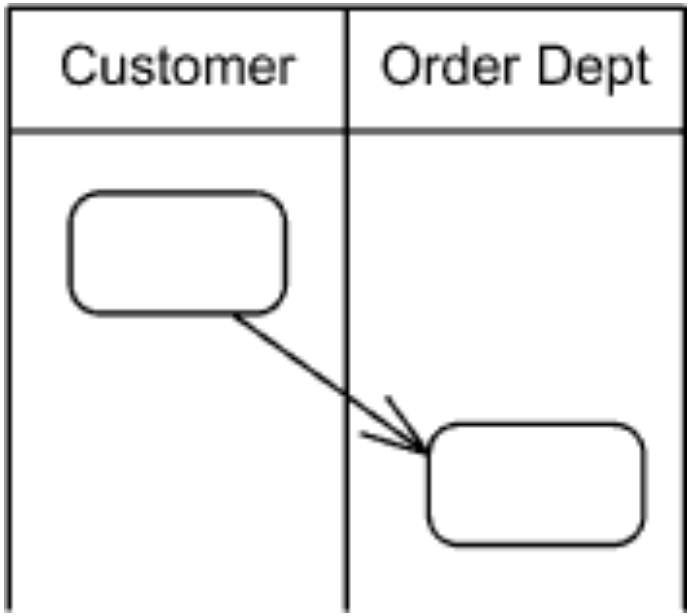
Partitioning

- An activity **partition** is an **activity group** for actions that have some common characteristic.
- Activity **partition** may be shown using a **swimlane** notation - with two, usually parallel lines, either horizontal or vertical, and a name labeling the partition in a box at one end.

Swim Lanes

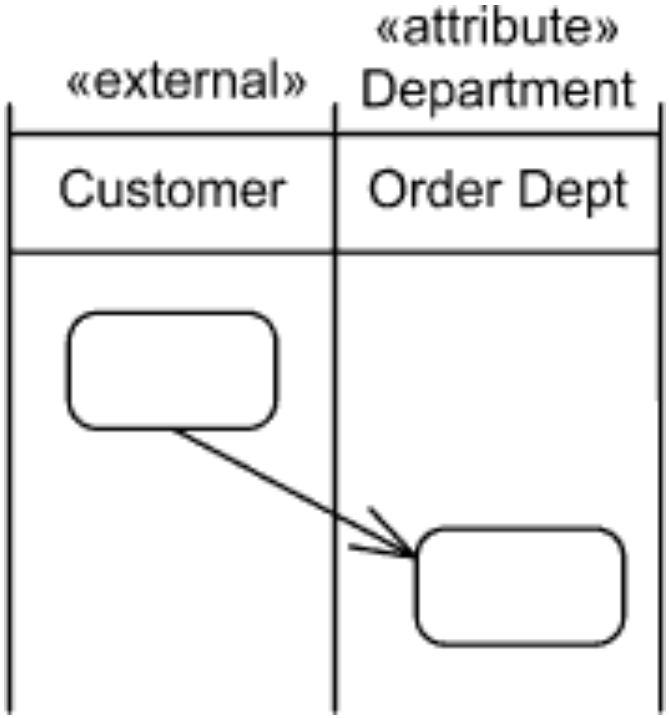
- Can be either horizontal or Vertical



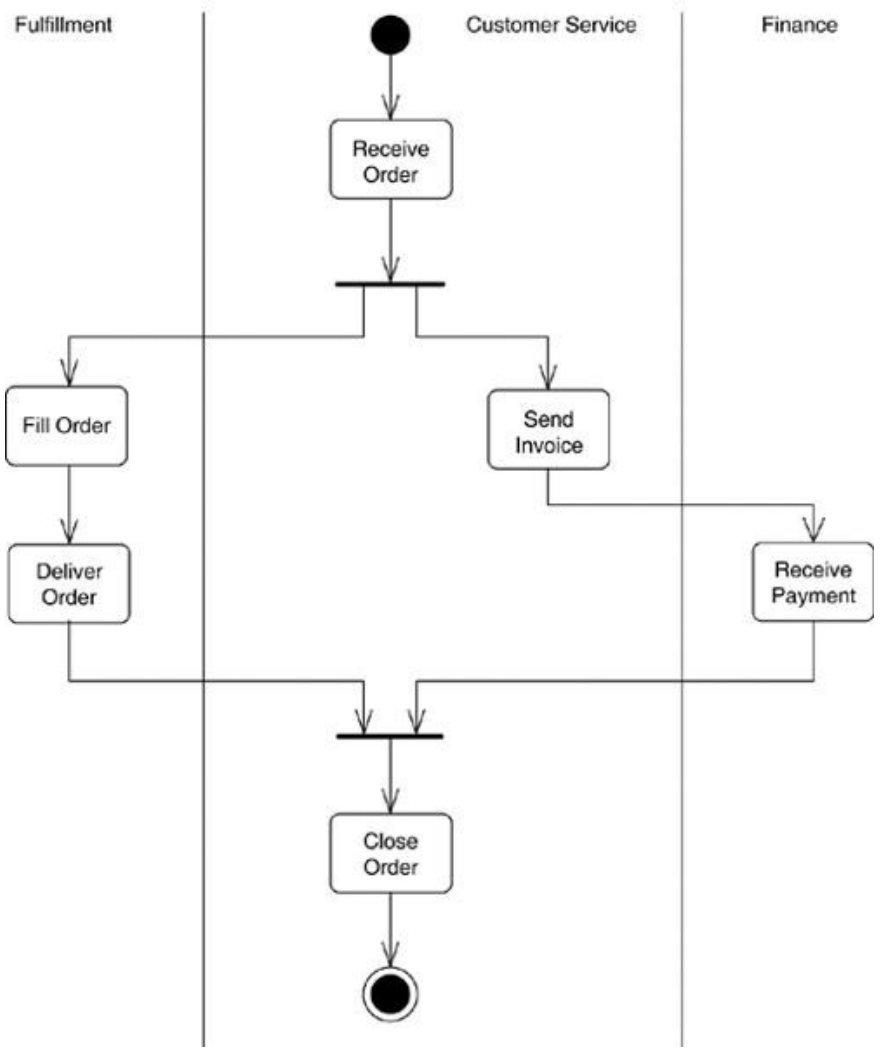


Sub Partitioning in Swim Lanes

- Order dept is a sub class under Department class



Example – Swim Lanes



Activity 5- Draw the activity diagram for the following with swim lanes

"Develop a software system for a client"

- Project Manager (**PM**) first gathers the requirements.
- UI Engineer (**UIE**) develops the prototype.
- PM shows the prototype to **client.**
- Till the client is satisfied, UIE modifies the prototype.
- Once the client is satisfied, signs off the prototype
- UIE develop the UI screens while Software Engineer (**SE**) develops the system.
- Once both (system and UI development) are done, SE integrates the system.
- PM delivers the system to client
- Client signs off the delivery.

Activity 6

- Draw an activity diagram for purchasing tickets through a tickets vending machine. (train/bus)
- Customer needs to buy a ticket. Ticket vending machine will request trip information from Customer. This information will include number and type of tickets, e.g. whether it is a monthly pass, one way or round ticket, route number, destination or zone number, etc.
- Based on the provided trip info ticket vending machine will calculate payment due and request payment options. Those options include payment by cash, or by credit or debit card. If payment by card was selected by Commuter, another actor, Bank will participate in the activity by authorizing the payment.
- After payment is complete, ticket will be dispensed to the Customer. Cash payment might result in some change due, so the change is dispensed to the Customer in this case. Ticket vending machine will show some "Thank You" screen at the end of the activity.

References

- http://www.uml-diagrams.org/
- UML Distilled: A Brief Guide to the Standard Object Modeling Language by Martin Fowler, Kendall Scott
- The Complete UML Training Course, Student Edition by Grady Booch, et al
- UML Explained by Kendall Scott