LECTURE 3

UML ACTIVITY DIAGRAMS

TOPICS

Software modeling review

UML Activity Diagrams

- Activities
- Actions
- Control Flow
- Object Flow
- Decision and Merge
- Fork and Join
- Conditional thread
- Partition
- Signal
- Interruptible region
- Expansion region

Domain and software process modeling using Activity Diagrams

Generating an activity diagram from a user story

SOFTWARE MODELING

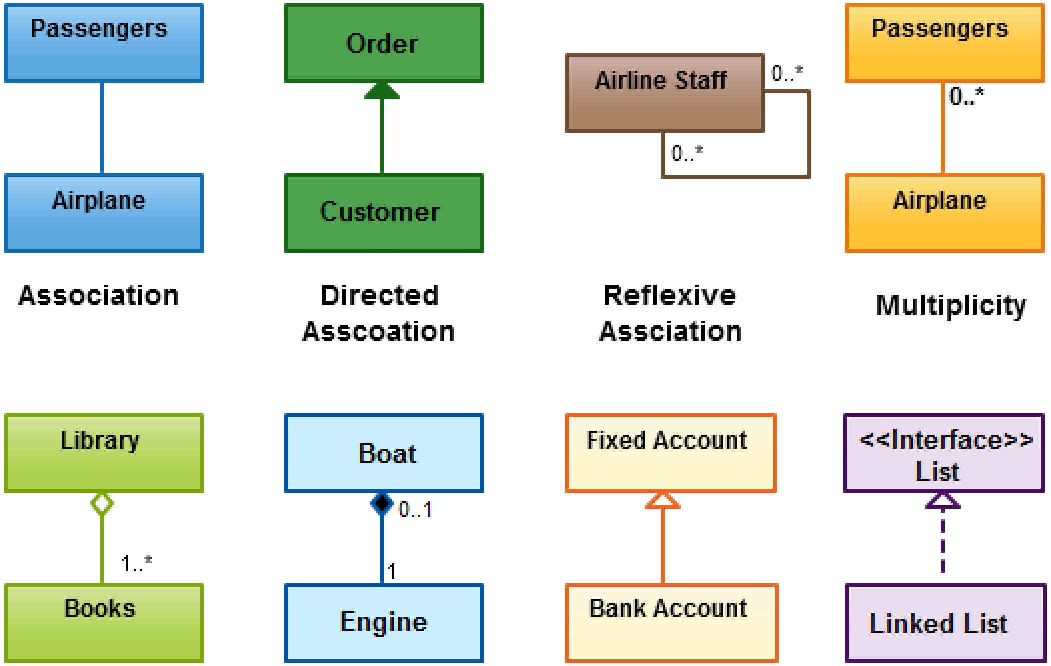
UML defines thirteen basic diagram types, divided into two general sets:

- Structural Models
- Behavioral Models

Structural Models define the static architecture of a system

- They are used to model the “things” that make up a system – the classes, objects, interfaces and physical components
- In addition they are used to model the relationships and dependencies between these elements

REVIEW OF UML RELATIONSHIPS



Passengers

Airplane

Association

Order

Customer

Directed Asscoation

Airline Staff

0..*

0..*

Reflexive Assciation

Passengers

0..*

Airplane

Multiplicity

Library

1..*

Books

Aggregation

Boat

0..1

1

Engine

Composition

Fixed Account

Bank Account

Inheritance

<<Interface>> List

Linked List

Realization

EVENTS IN UML

In UML, you can model four kinds of events:

- Signals: object sent **asynchronously** by one object and received by another
- Calls: method calls between objects (usually synchronous)
- Passage of time
- Change in state

Events may be external or internal

- External events are those that pass between the system and its actors (e.g. pushing a button on a GUI)
- Internal events are those passed among the objects that live inside the system (e.g. IO exception)

UML ACTIVITY DIAGRAMS

Activity diagrams show the workflow of a process from start to finish

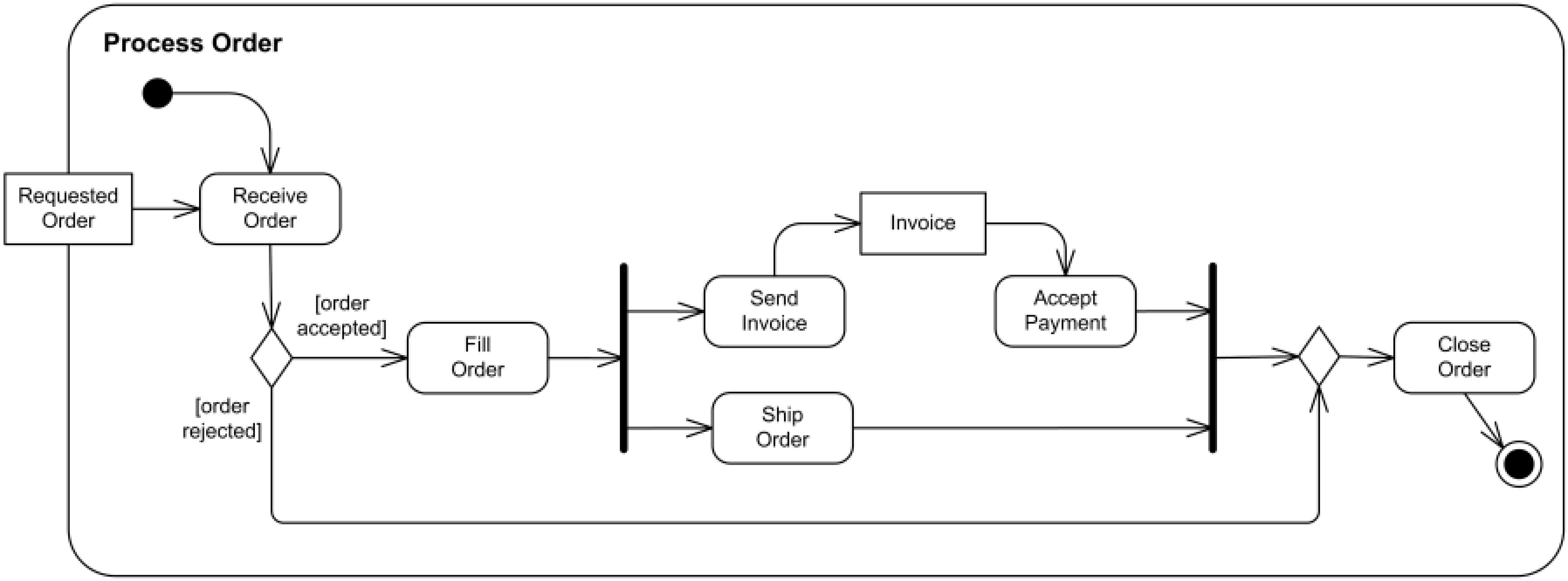
- Detail the many decision paths that exist in the progression of events contained in the activity

Very useful when parallel processing may occur in the execution of some activities

UML ACTIVITY DIAGRAMS

An example of an activity diagram is shown below

(We will come back to that diagram)

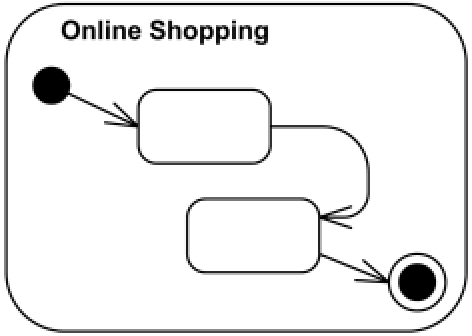


ACTIVITY

Activity is parameterized behavior represented as coordinated flow of actions

- It takes time
- Similar to a state, where the criterion for leaving the sate is the completion of the activity

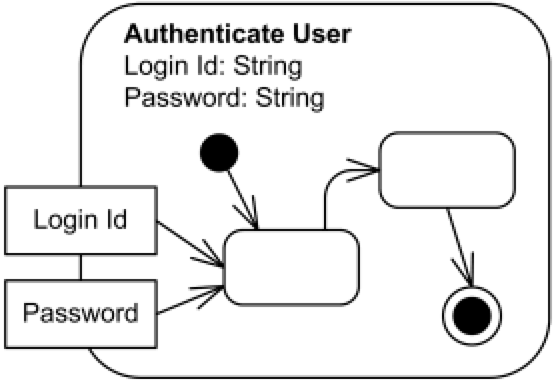
Shown as a round-cornered rectangle enclosing all the actions and control flows



ACTIVITY PARAMETERS

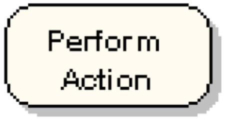
Activity parameters are displayed on the border and listed below the activity name as:

parameter-name: parameter-type

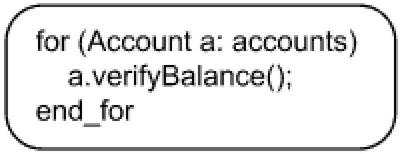


ACTIONS

An action represents a single step within an activity



Action could be expressed in some application-dependent action language



```
for (Account a: accounts)
   a.verifyBalance();
end_for
```

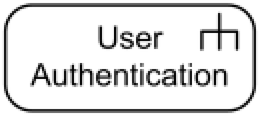
There are four ways in which an action can be triggered

1. As soon as the activity starts
2. During lifetime of the activity
3. In response to an event
4. Just before the activity completes

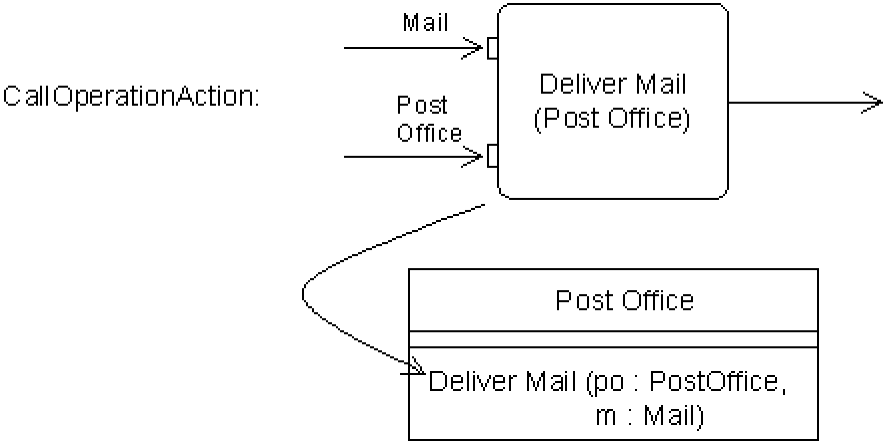
CALL ACTIONS

Call Activity Action: allows us to call any predefined activity

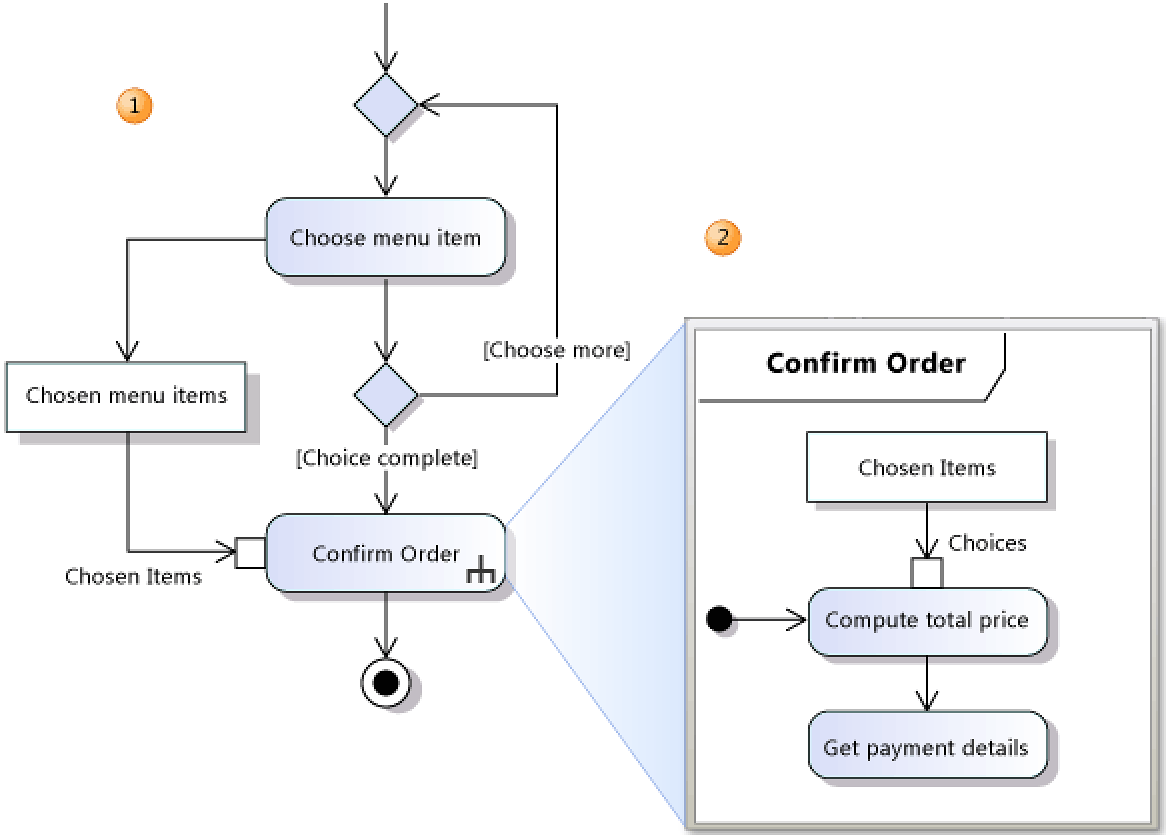
- This will avoid redundant definitions of activities



Call Operation Action: calls a behavior of a structural element (operation of a class)

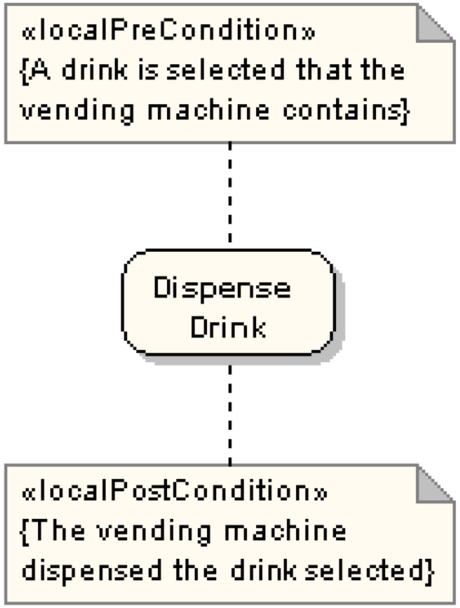

CallOperationAction:
Mail
Post Office
Deliver Mail (Post Office)
Post Office
Deliver Mail (po : PostOffice, m : Mail)

CALL ACTIONS



CONSTRAINS

Pre- and Post condition Constraints can be attached to actions



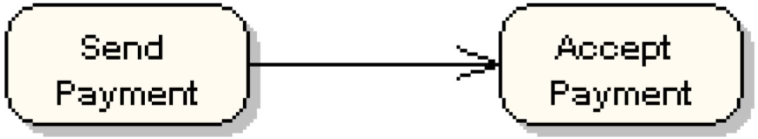
«localPreCondition»
{A drink is selected that the vending machine contains}

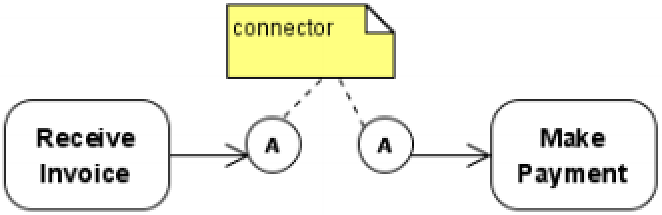
«localPostCondition»
{The vending machine dispensed the drink selected}

CONTROL FLOW

Shows the flow of control from one action to the next

- Its notation is a line with an arrowhead.

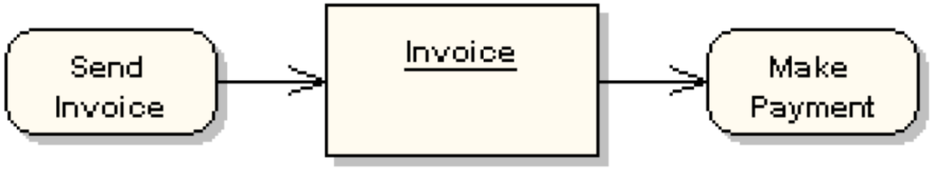




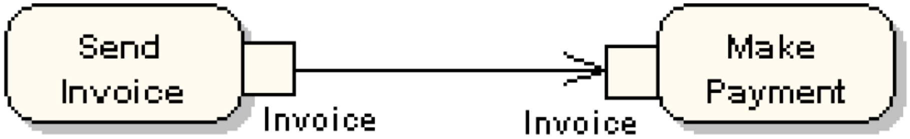
OBJECTS FLOW

An object flow is a path along which objects or data can pass

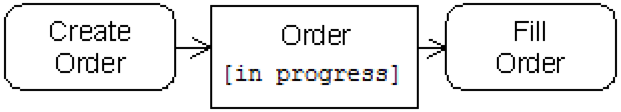
- An object is shown as a rectangle



A short hand for the above notation

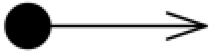


You can also show the state of the object being passed (shown in brackets below the object's name)



INITIAL AND FINAL NODES

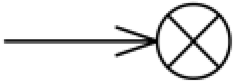
Initial node is a control node at which flow starts when the activity is invoked



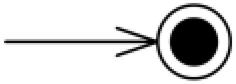
Activities may have more than one initial node

- In this case, invoking the activity starts multiple flows, one at each initial node

Flow Final node is a control final node that terminates a flow



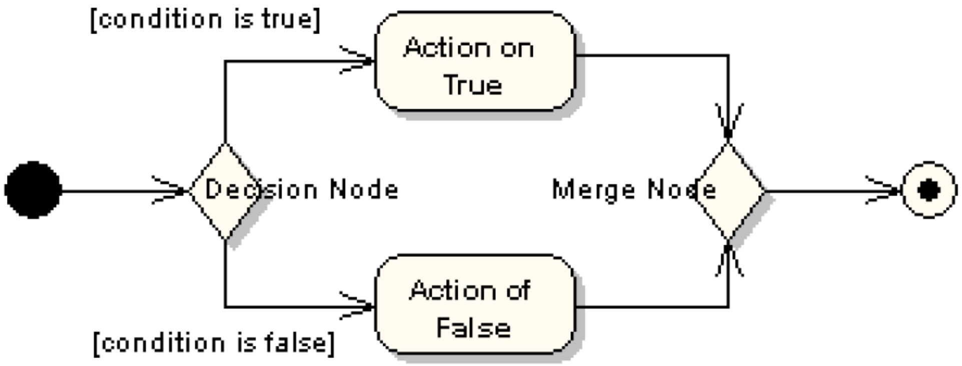
Activity Final node is a control final node that stops all flows in an activity



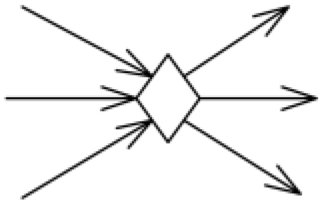
DECISION AND MERGE NODES

Decision nodes and merge nodes have the same notation: a diamond shape

The control flows coming away from a decision node will have guard conditions



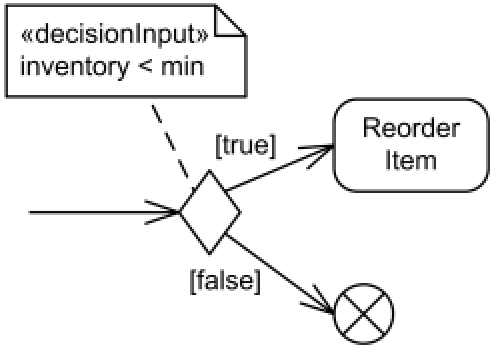
Merge node and decision node combined



DECISION NODES

Decision can have decision input behavior specified

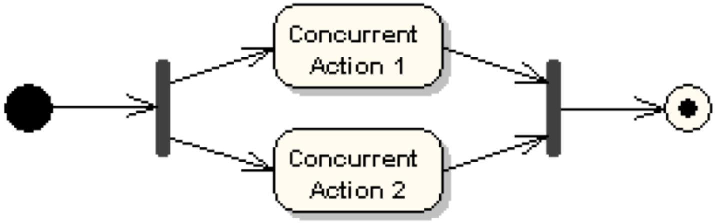
- Decision input behaviors were introduced in UML to avoid redundant recalculations in guards
- It is specified by the keyword «decisionInput» and a condition is placed in a note symbol



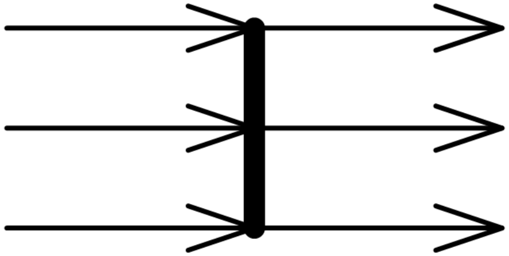
FORK AND JOIN NODES

Forks and joins have the same notation: either a horizontal or vertical bar

- They indicate the start and end of concurrent threads of control
- Join synchronizes multiple inflows and produces a single outflow
- The outflow from a join cannot execute until all inflows have been received

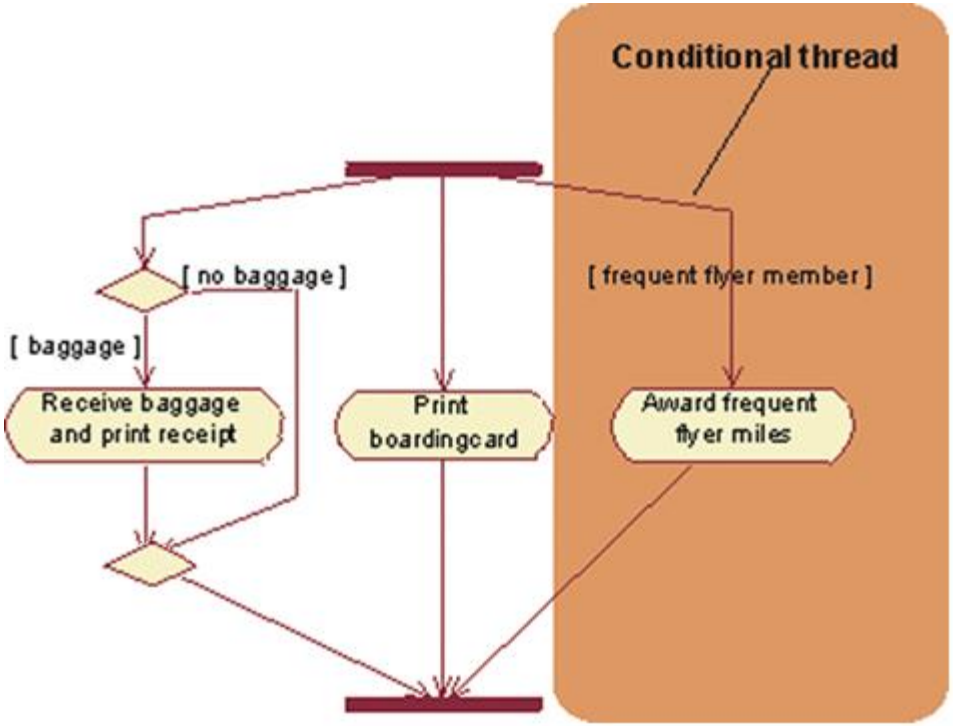


Join and fork node combined

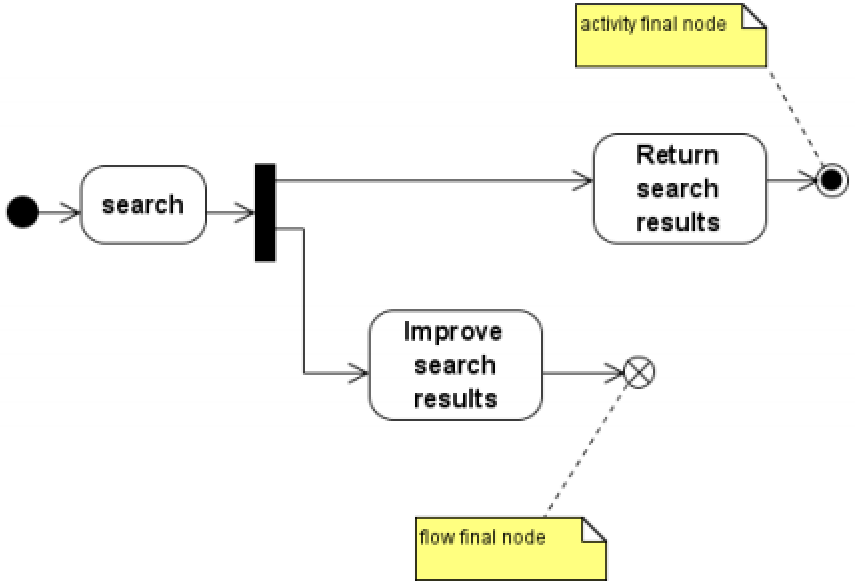


CONDITIONAL THREAD

Guard conditions can be used to show that one of a set of concurrent threads is conditional


Conditional thread
[no baggage]
[baggage]
Receive baggage and print receipt
Print boardingcard
[frequent flyer member]
Award frequent flyer miles

INITIAL AND FINAL NODES

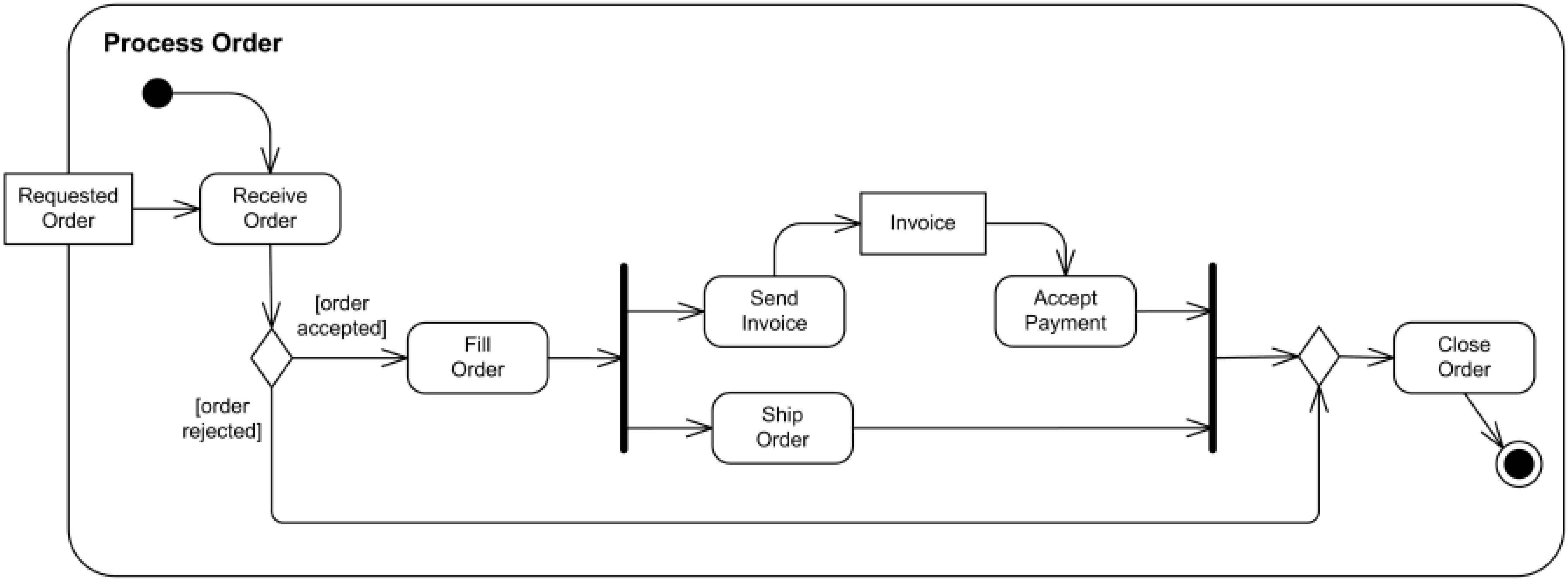


Warning: be careful when using a flow final node after a fork

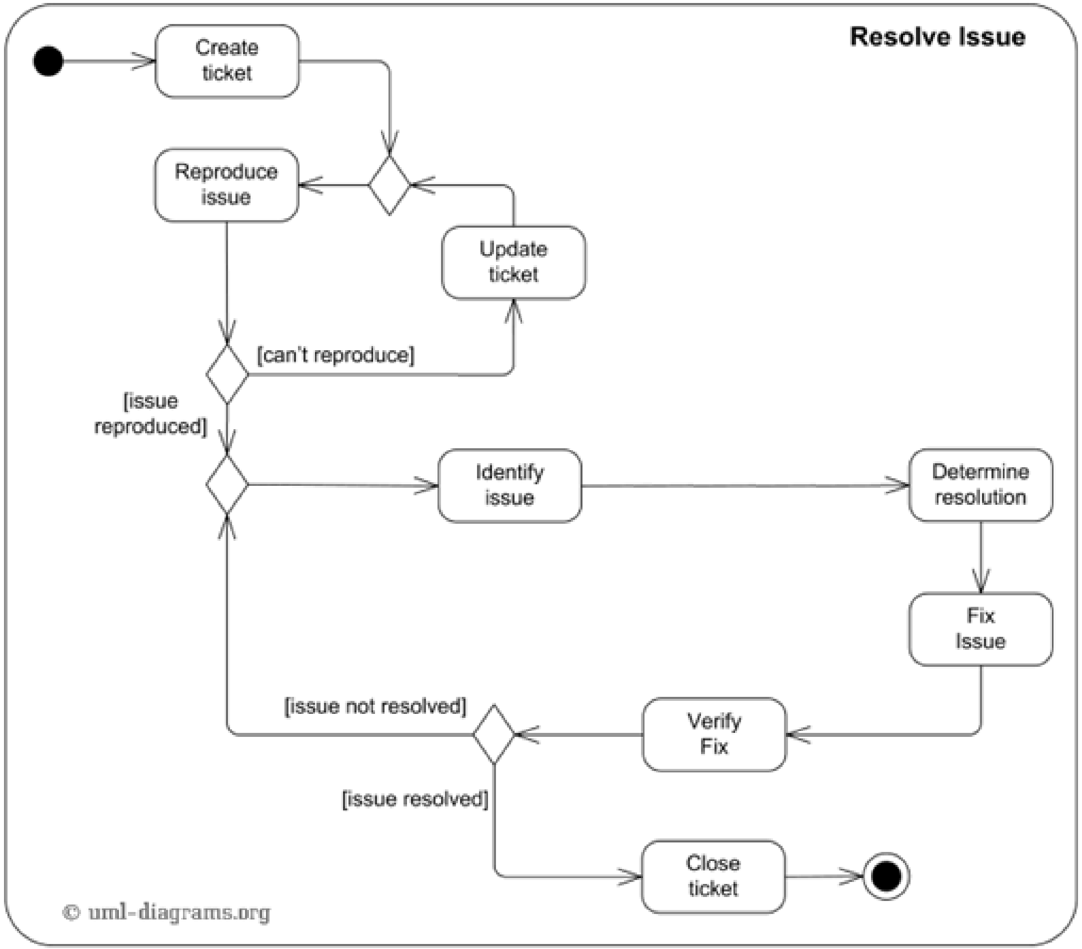
- As soon as the activity final node is reached, all other actions in the activity (including the ones before the final flow node) terminate

UML ACTIVITY DIAGRAMS

Coming back to our initial example



ISSUE HANDLING IN SOFTWARE PROJECTS

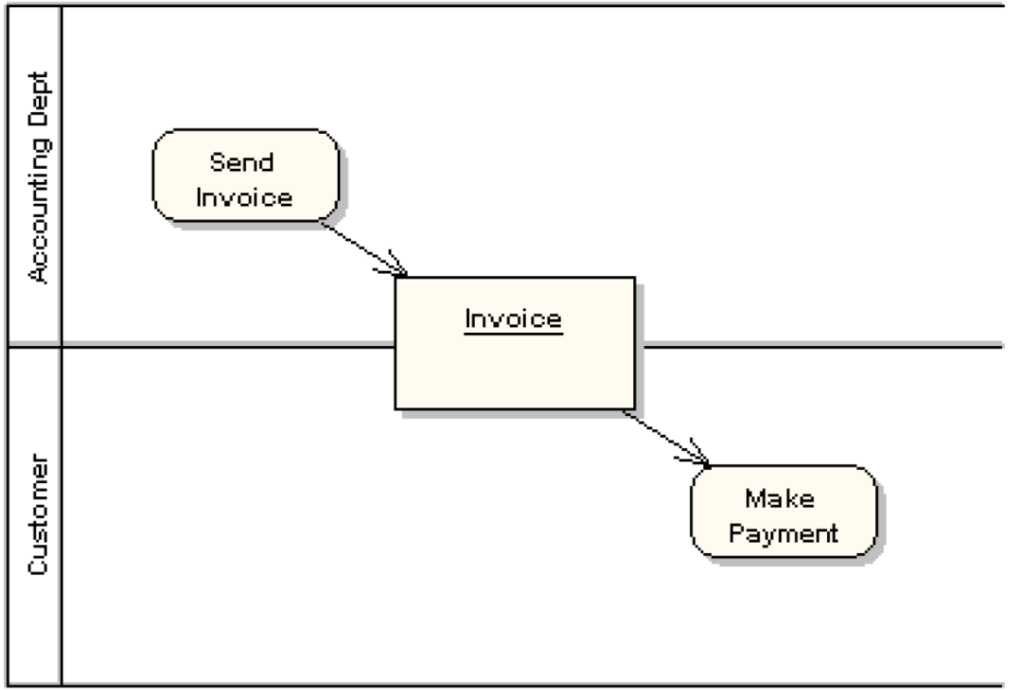


Courtesy of uml-diagrams.org

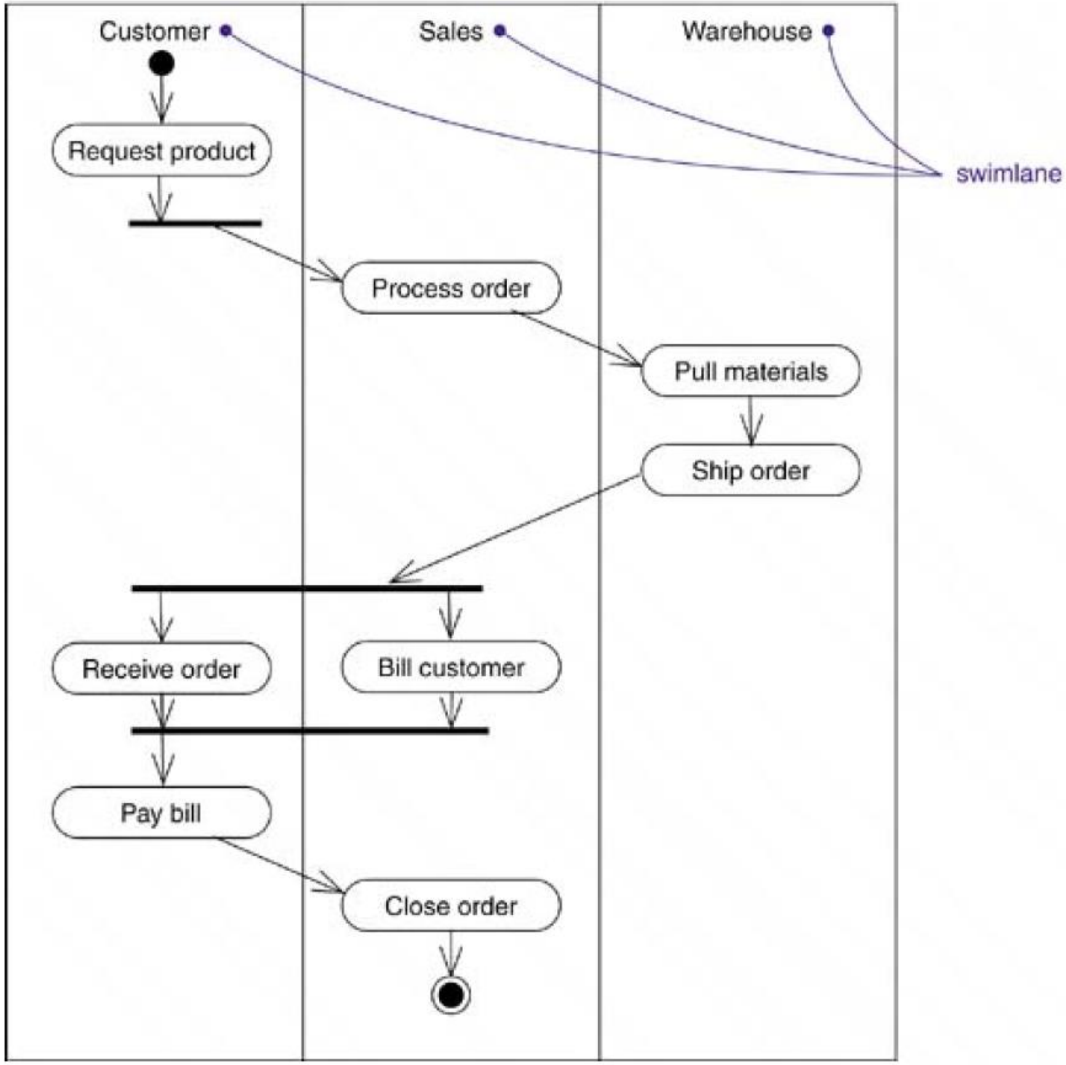
PARTITION

Shown as horizontal or vertical swim lane

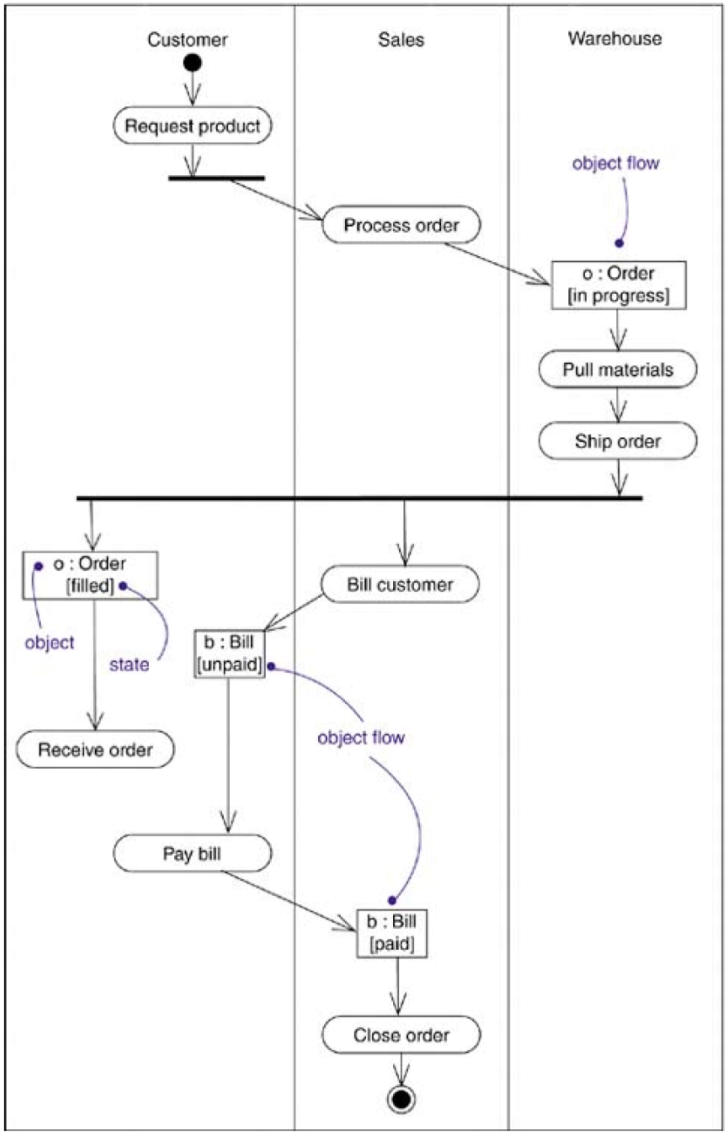
- Represents a group of actions that have some common characteristic



PARTITION


Customer
Sales
Warehouse
swimlane
Request product
Process order
Pull materials
Ship order
Receive order
Bill customer
Pay bill
Close order

PARTITION EXAMPLE WITH OBJECT FLOW



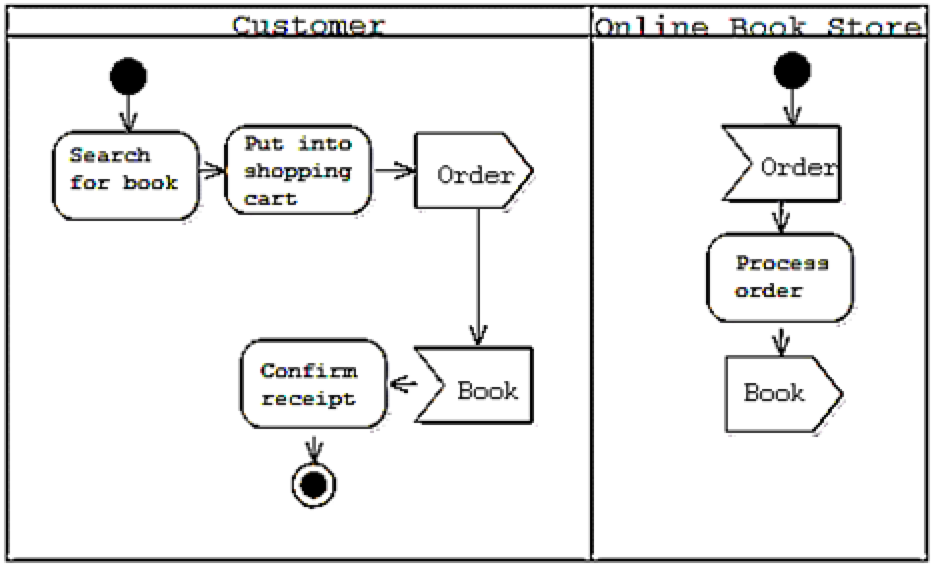
SEND AND RECEIVE SIGNALS AND TIME EVENTS

Control flows or object flows connect actions

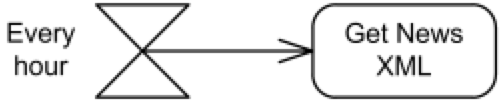
- They define "synchronous" processes where the flow is determined through an ordered sequence of steps

Through the use of signals, processes can be uncoupled

- Achieve asynchronous communication



Accept time event action generates an output every hour

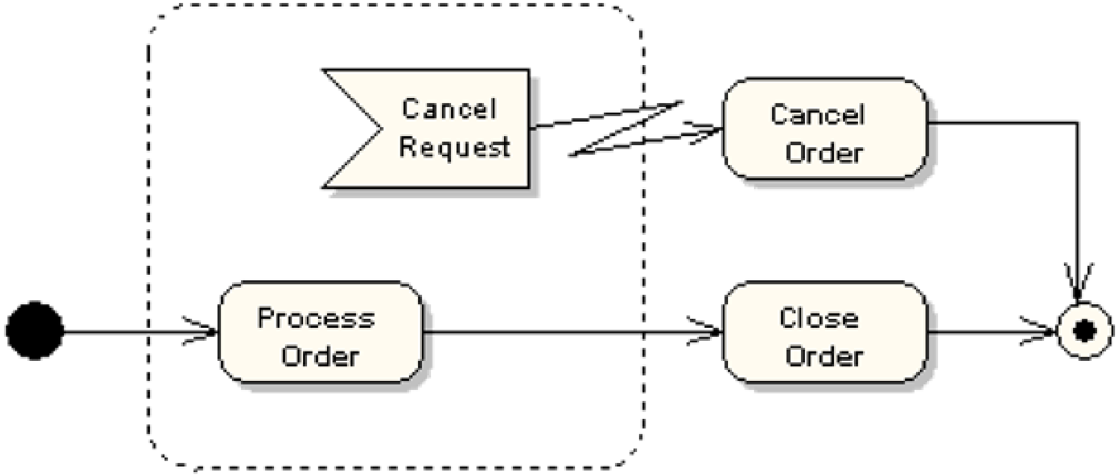


INTERRUPTIBLE ACTIVITY REGION

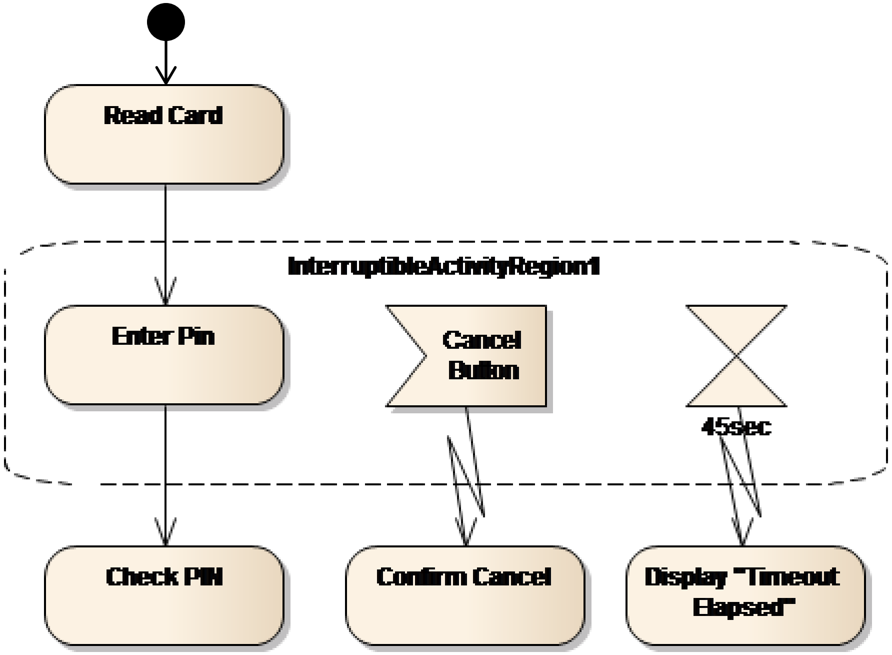
Surrounds a group of actions that can be interrupted

Example below:

- “Process Order” action will execute until completion, when it will pass control to the “Close Order” action, unless a “Cancel Request” interrupt is received, which will pass control to the “Cancel Order” action.



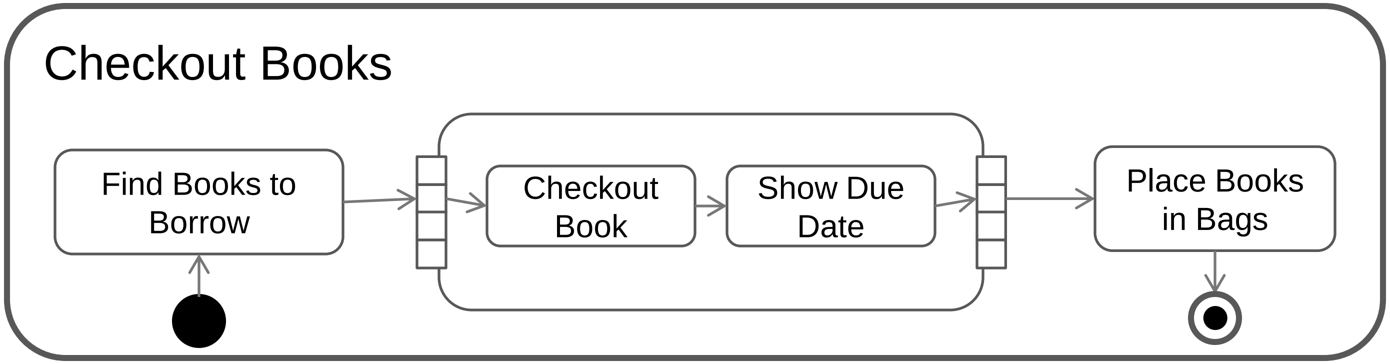
INTERRUPTIBLE ACTIVITY REGION



EXPANSION REGION

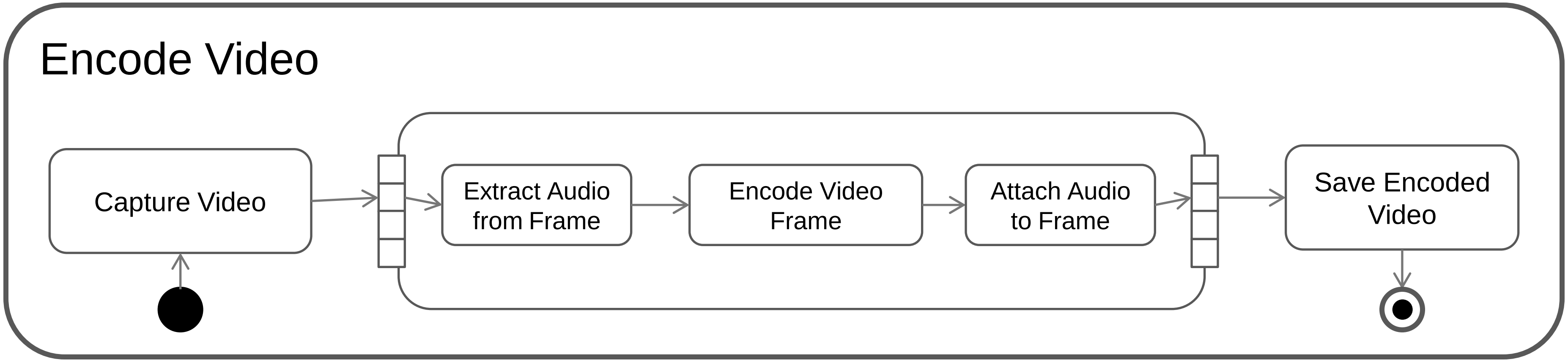
An expansion region is an activity region that executes multiple times to consume all elements of an input collection

Example of books checkout at a library modeled using an expansion region:



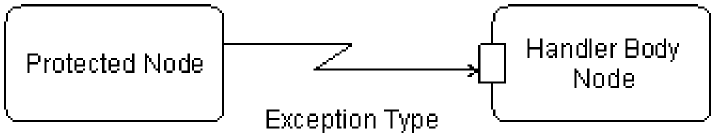
EXPANSION REGION

Another example: Encoding Video



EXCEPTION HANDLERS

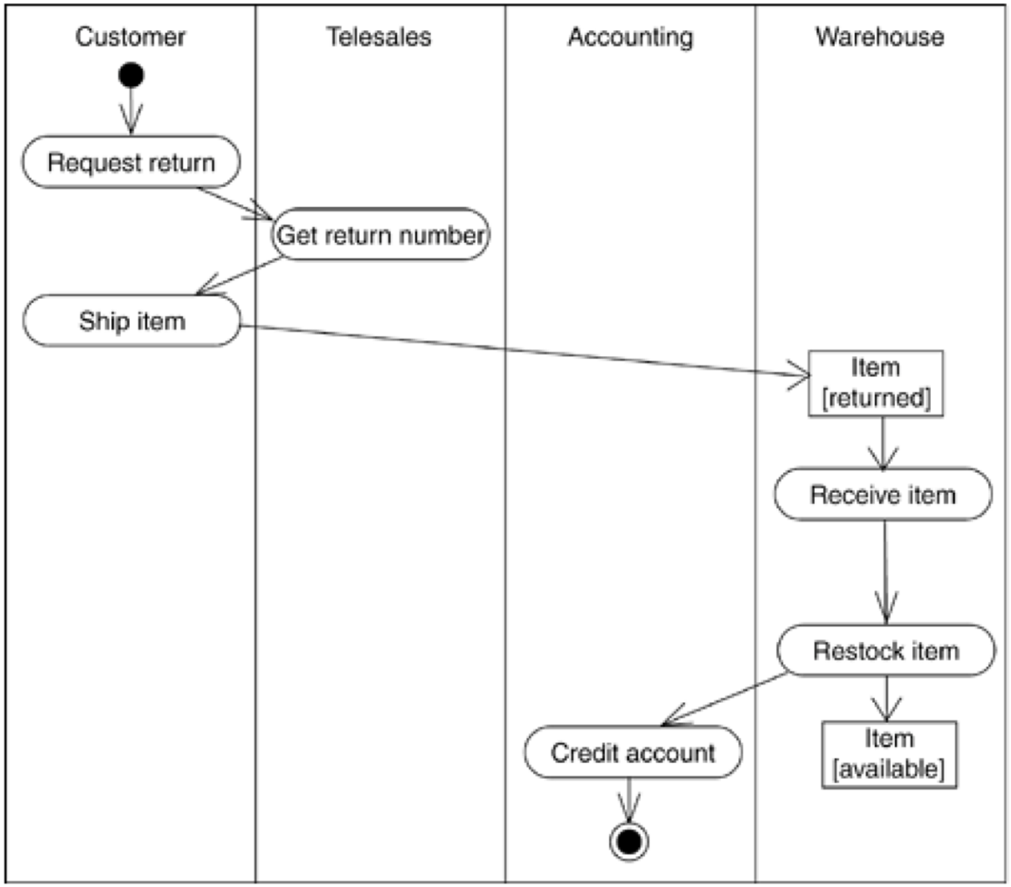
An exception handler is an element that specifies what to execute in case the specified exception occurs during the execution of the protected node



In Java

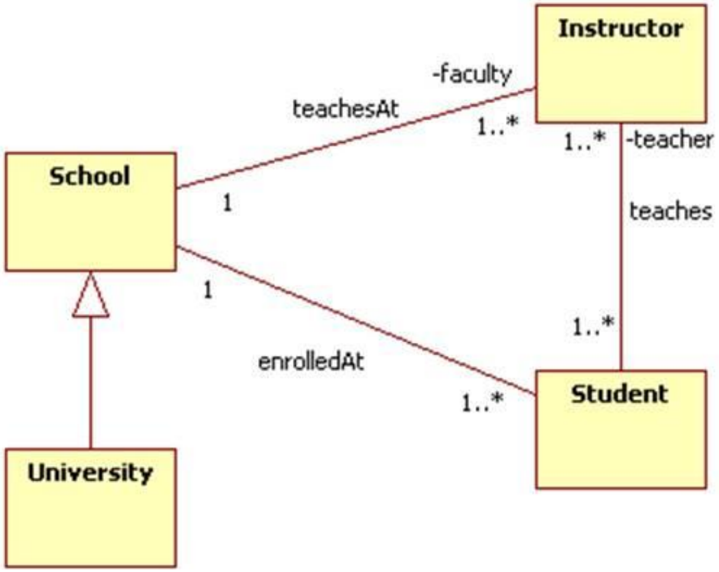
- “Try block” corresponds to “Protected Node”
- “Catch block” corresponds to the “Handler Body Node”

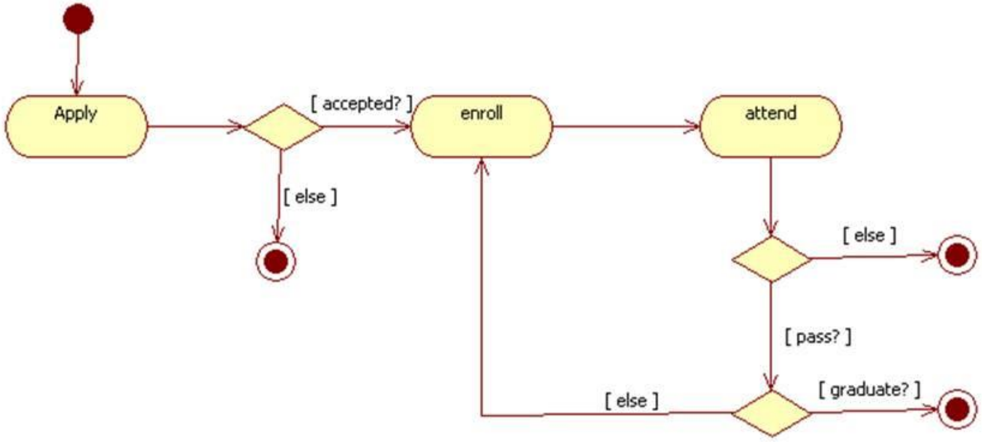
ACTIVITY DIAGRAMS TO MODEL A WORKFLOW



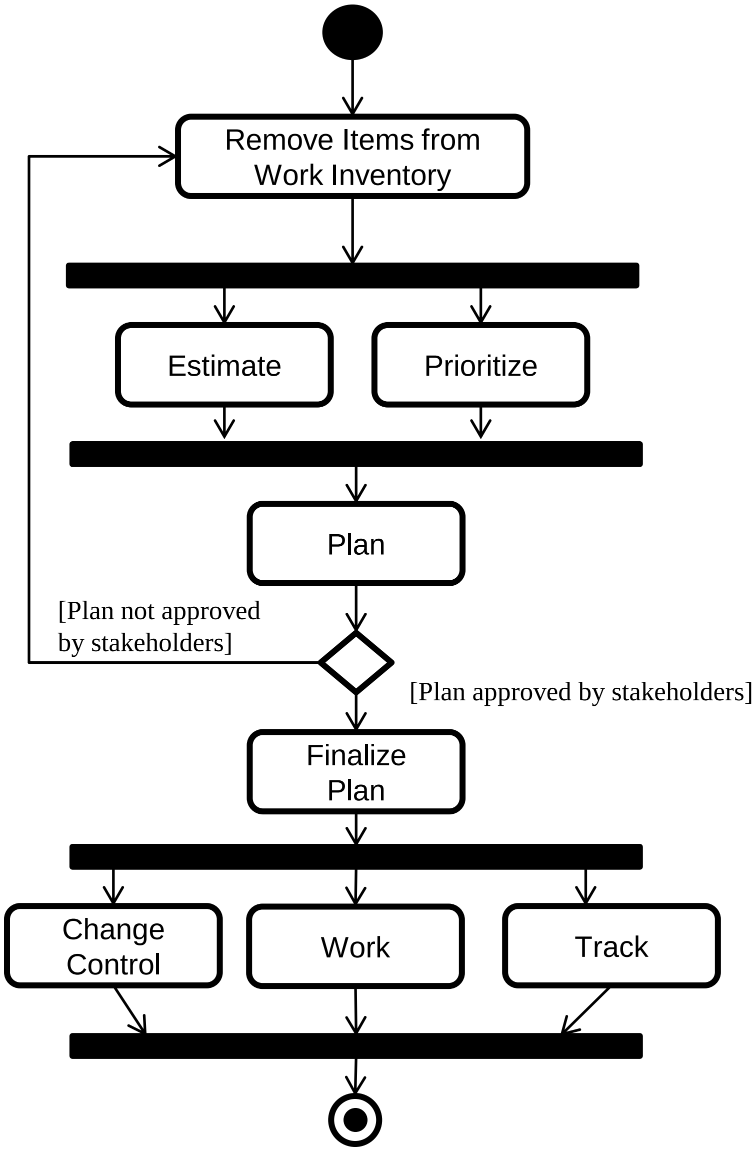
Returning a purchased item

ACTIVITY DIAGRAMS USED IN DOMAIN MODELING

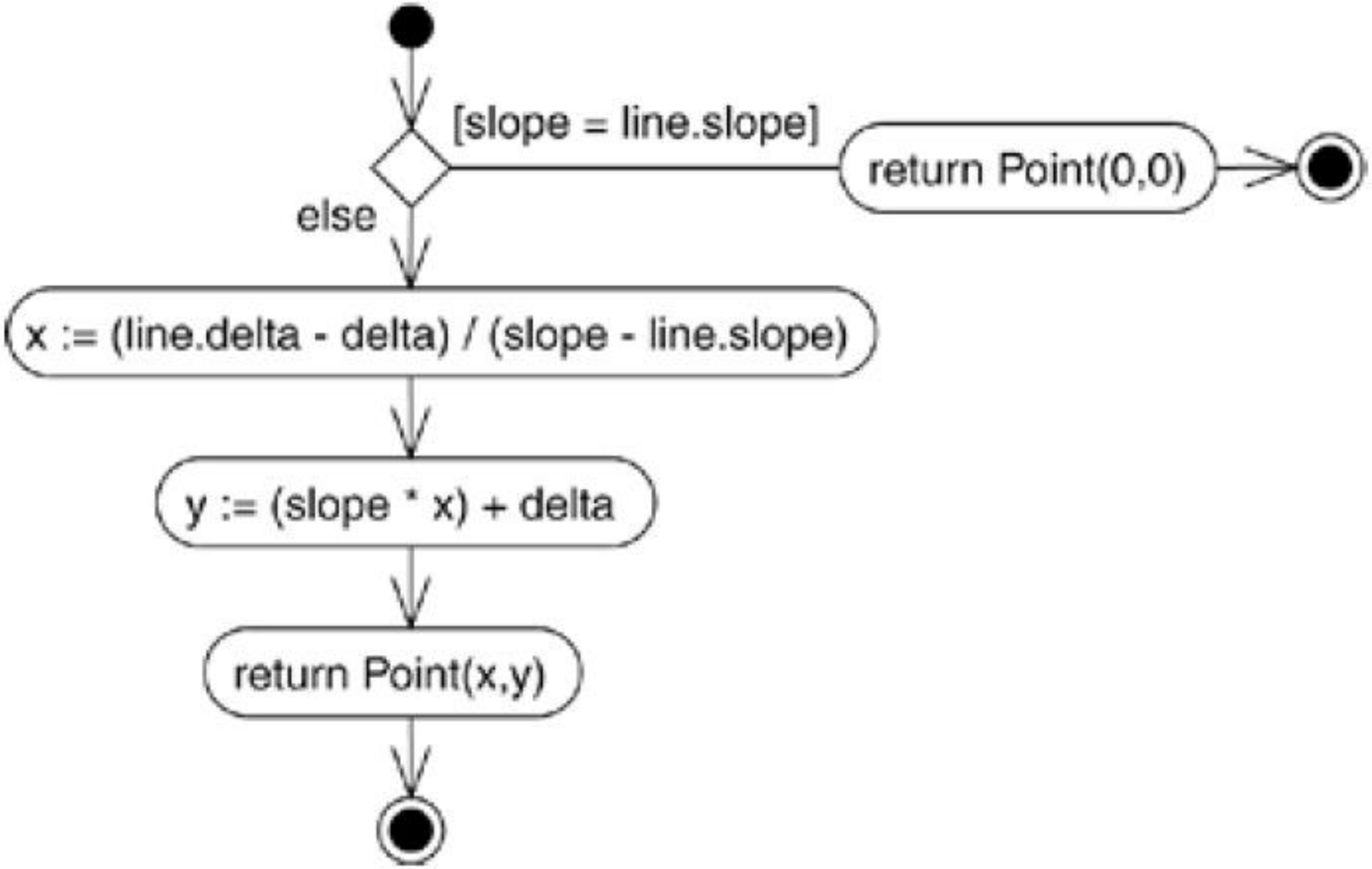




MODELING A SOFTWARE PROCESS USING AN ACTIVITY DIAGRAM


Remove Items from Work Inventory
Estimate
Prioritize
Plan
[Plan not approved by stakeholders]
[Plan approved by stakeholders]
Finalize Plan
Change Control
Work
Track

ACTIVITY DIAGRAMS TO MODEL AN OPERATION



[slope = line.slope]

return Point(0,0)

else

x := (line.delta - delta) / (slope - line.slope)

y := (slope * x) + delta

return Point(x,y)

HOW TO CONSTRUCT ACTIVITY DIAGRAMS

1. Find system Actors, Classes and use cases
2. Identify key scenarios of system use cases
3. Combine the scenarios to produce comprehensive workflows described using activity diagrams
4. Where significant object behaviour is triggered by a workflow, add object flows to the diagrams
5. Where workflows cross technology boundaries, use swim lanes to map the activities
6. Refine complicated high level activities

WHEN TO USE ACTIVITY DIAGRAMS

Do use them for

- Analysing Use Cases
- Understanding workflow across many Use Cases
- Dealing with multi-threaded applications

Do not use them

- To see how objects collaborate
- To see how an object behaves over its lifetime

GENERATING A UML ACTIVITY DIAGRAM FROM A USER STORY

A UML activity diagram can provide a visual representation of a user story for all stakeholders

- Complex user stories (high risk) can be further elaborated with a UML activity diagram
- Allows us to identify any misunderstandings between stakeholders
- Remember: the earlier these misunderstandings are cleared up, the less costly they are

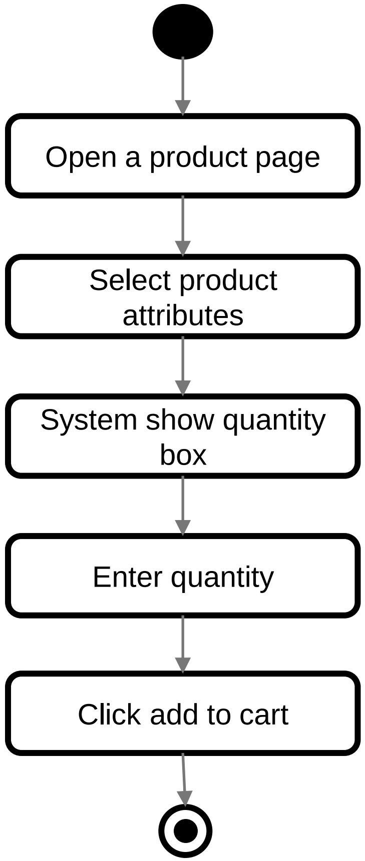
GENERATING A UML ACTIVITY DIAGRAM FROM A USER STORY

Let's consider this example user story for a typical e-commerce system:

User can add an item to shopping cart

GENERATING A UML ACTIVITY DIAGRAM FROM A USER STORY

Start with a simple diagram that enumerates the necessary steps


Open a product page

Select product attributes

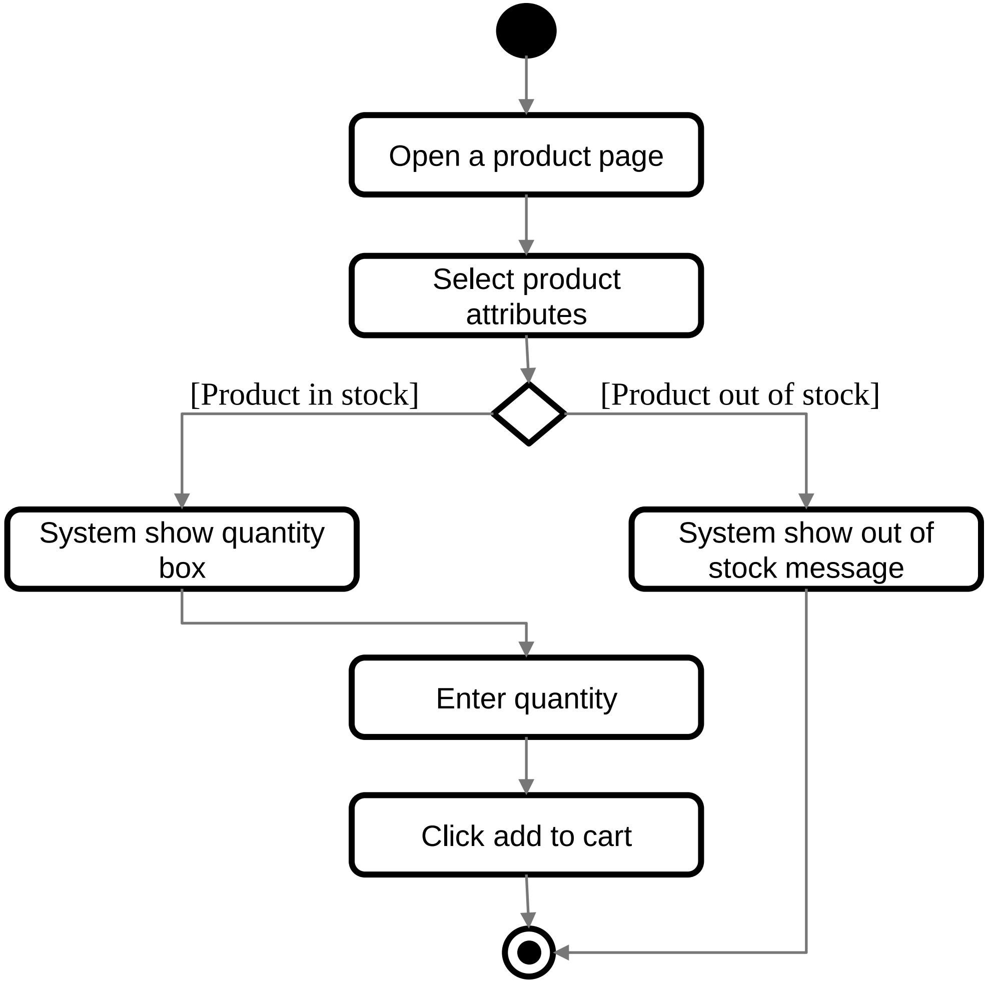
System show quantity box

Enter quantity

Click add to cart

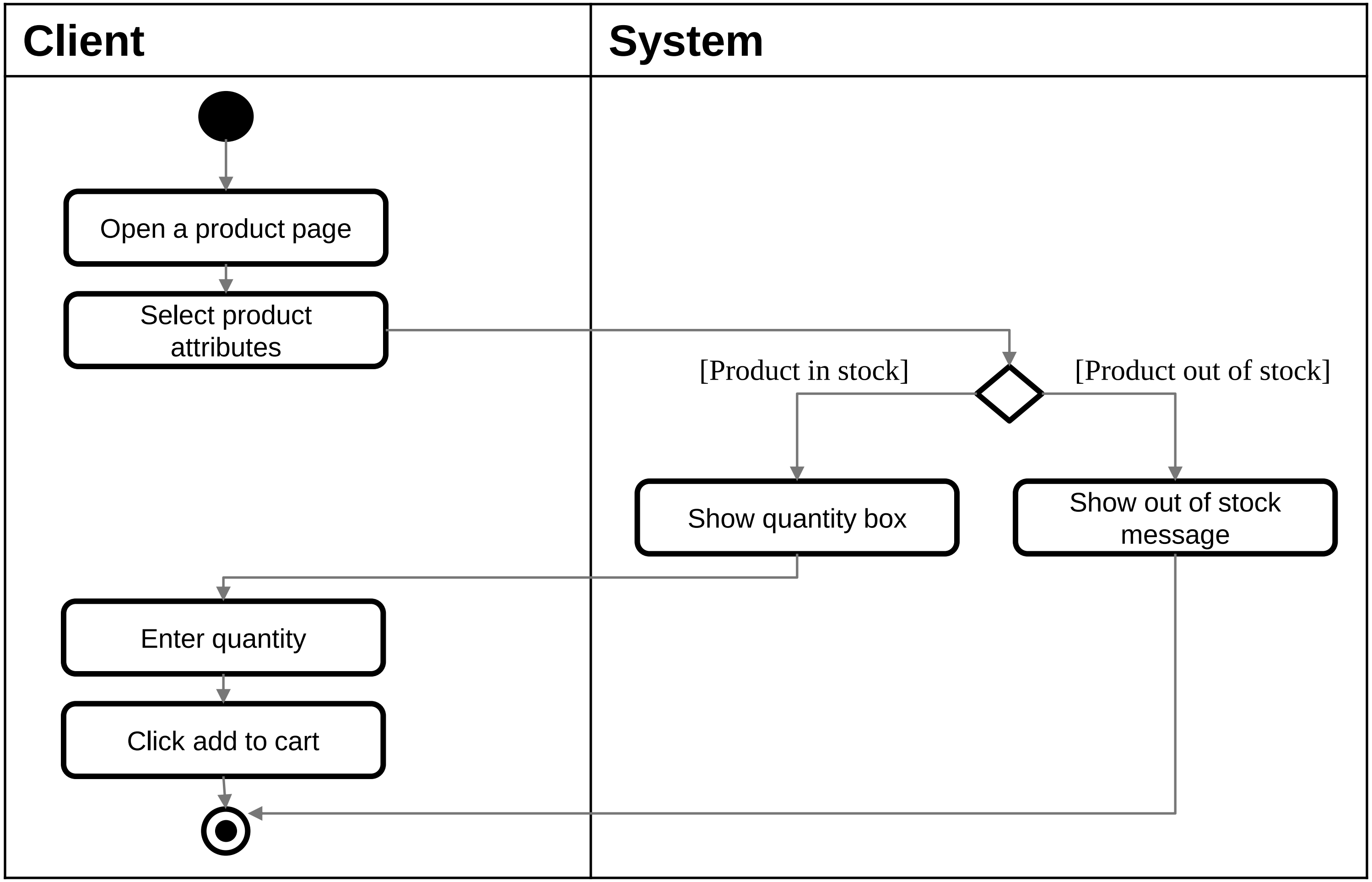
GENERATING A UML ACTIVITY DIAGRAM FROM A USER STORY

Refine model to add alternate sequences



GENERATING A UML ACTIVITY DIAGRAM FROM A USER STORY

Add partitions (swim lanes) to clarify who is doing what…



THANK YOU!

QUESTIONS?