

# Activity Diagrams

# Activity Diagrams

---

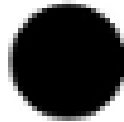
- Model business workflows
- Identify candidate use cases, through the examination of business workflows
- Identify pre- and post-conditions for use cases
- Model workflows between/within use cases
- Model complex workflows in operations on objects
- Model in detail complex activities in a high level activity diagram



# Symbols

---

## Start symbol



- Represents the beginning of a process or workflow in an activity diagram.
- It can be used by itself or with a note symbol that explains the starting point.



## Note symbol

- Allows the diagram creators or collaborators to communicate additional messages that don't fit within the diagram itself.
- Leave notes for added clarity and specification.

# Symbols

---

## Activity symbol

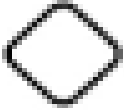


- Indicates the activities that make up a modelled process.
- These symbols, which include short descriptions within the shape, are the main building blocks of an activity diagram.



## Connector symbol

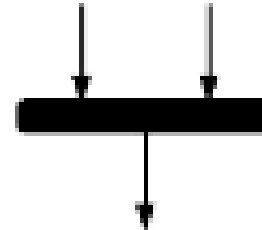
- Shows the directional flow, or control flow, of the activity.
- An incoming arrow starts a step of an activity.
- once the step is completed, the flow continues with the outgoing arrow.



# Symbols

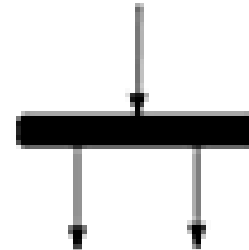
---

## Join symbol/ Synchronization bar



- Combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time.
- Represented with a thick vertical or horizontal line.

## Fork symbol

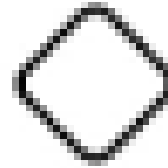


- Splits a single activity flow into two concurrent activities.
- Symbolized with multiple arrowed lines from a join.

# Symbols

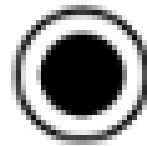
---

## Decision symbol



- Represents a decision and always has at least two paths branching out with condition text to allow users to view options.
- This symbol represents the branching or merging of various flows with the symbol acting as a frame or container.

## End symbol



- Marks the end state of an activity and represents the completion of all flows of a process.









The Start Point  
represents the  
EVENT that triggers  
the use case.



Label the Start Point to  
make the INTENT and  
TRIGGER of the use case  
explicit.

- Actor elects to AddCustomer

- Actor elects to AddCustomer



- Actor elects to AddCustomer

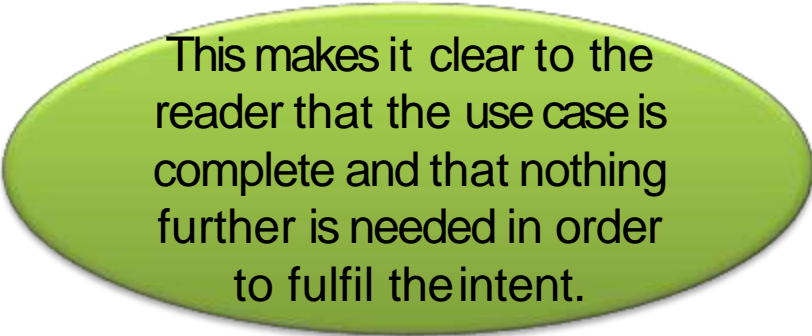


Label the End Point to  
EXPLICITLY confirm that  
the intent of the use case  
has been achieved.

● Actor elects to AddCustomer

◎ Customer added

- Actor elects to AddCustomer



This makes it clear to the reader that the use case is complete and that nothing further is needed in order to fulfil the intent.

- ⦿ Customer added

- Actor elects to AddCustomer



● **End of process**



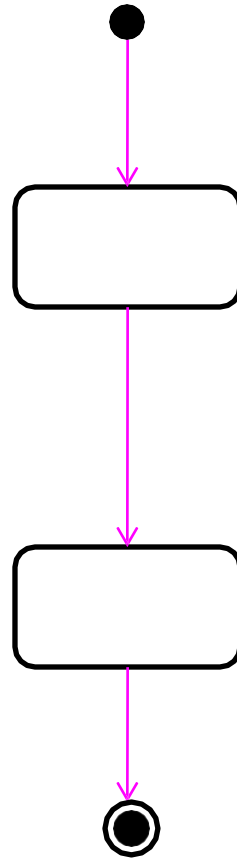
To reach the End  
Point...





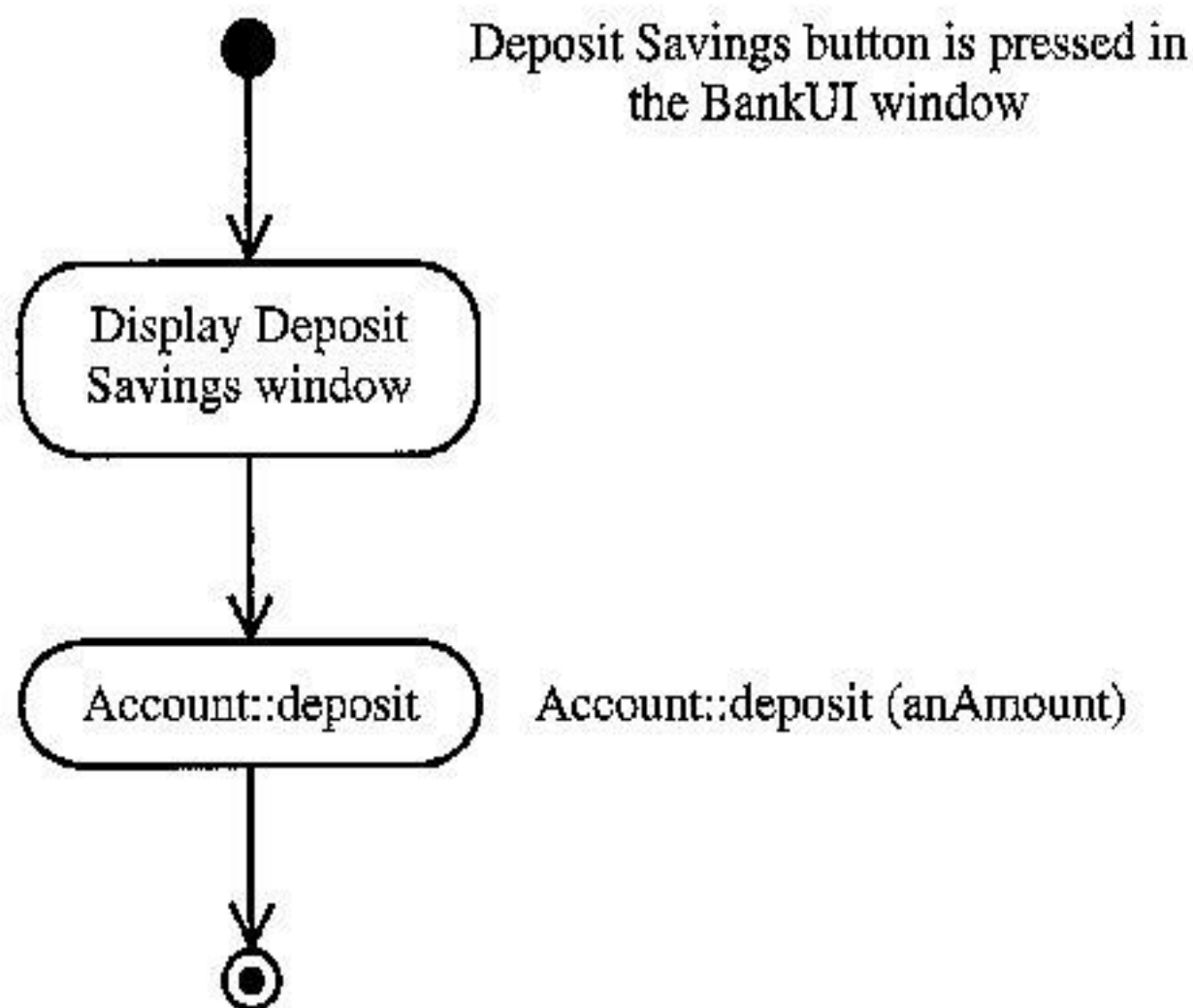
Link the steps with  
TRANSITIONS.

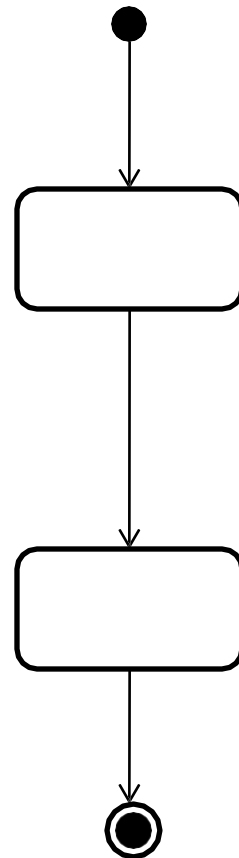




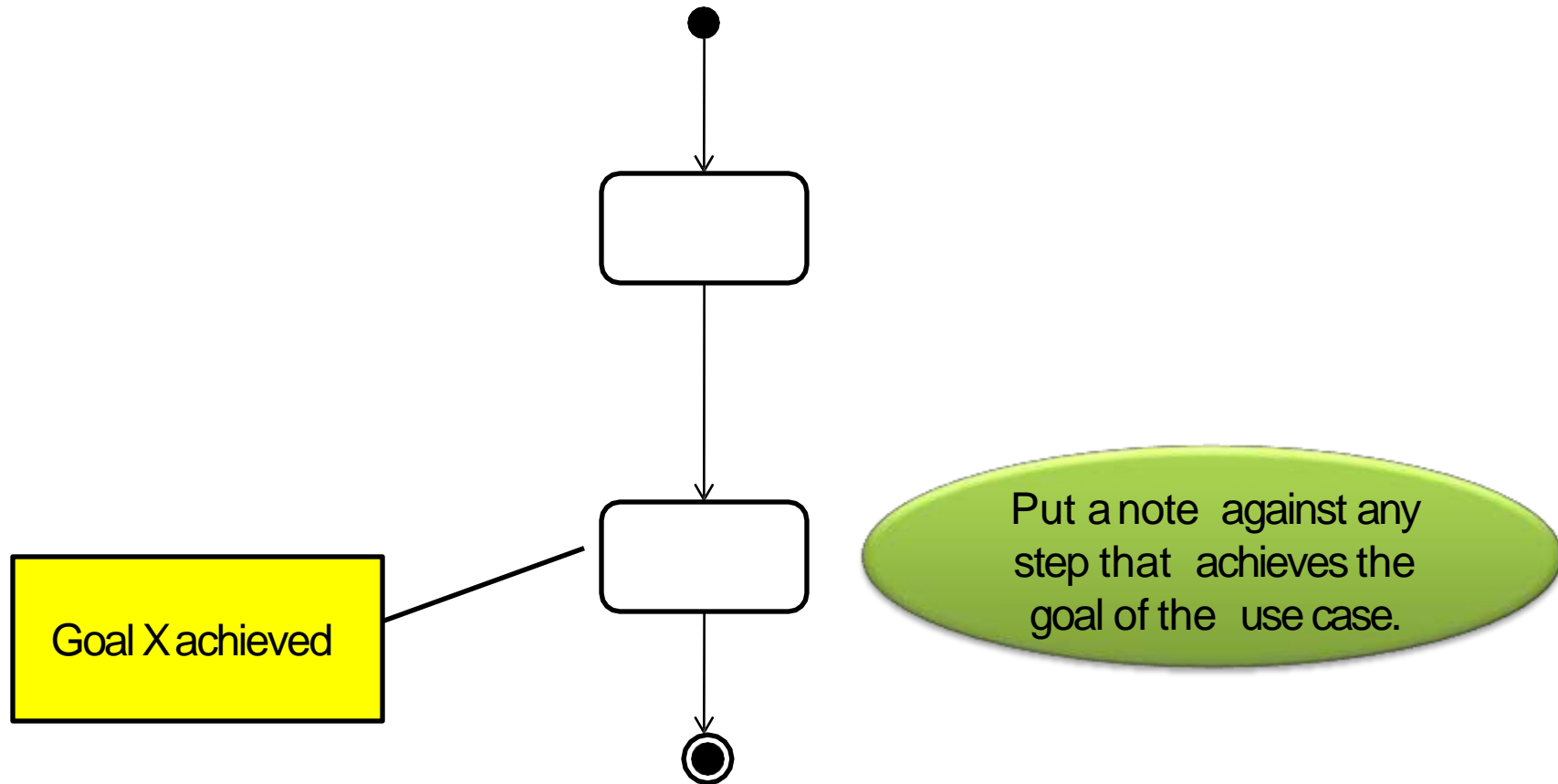
**FIGURE 12-24**

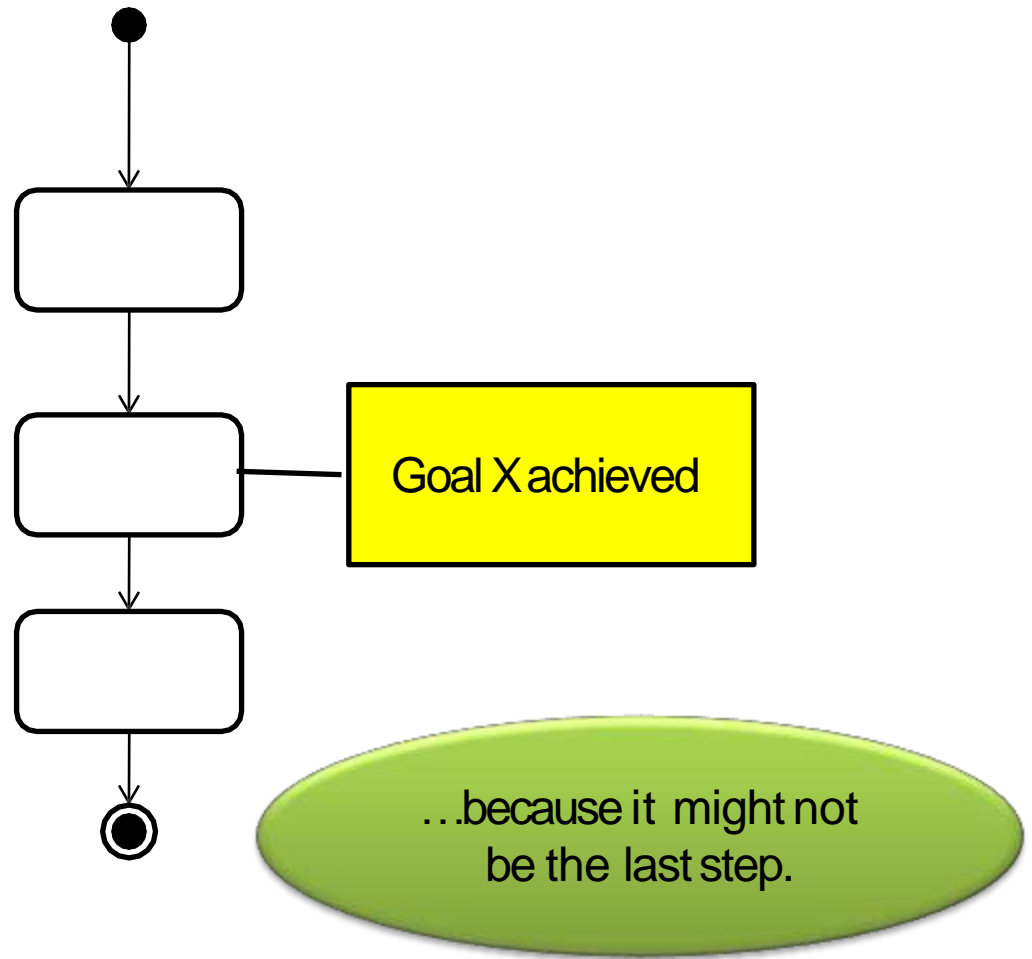
Activity diagram for processing a deposit to a savings account.





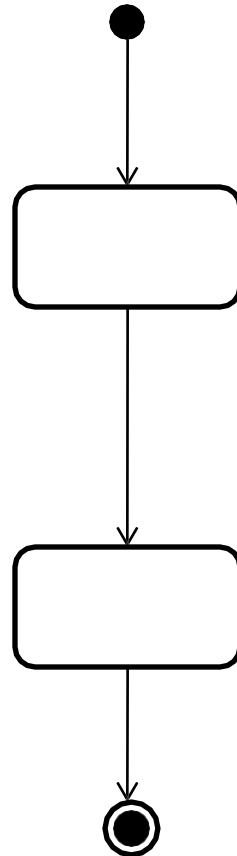
Transitions use arrow heads to show the direction of processflow.

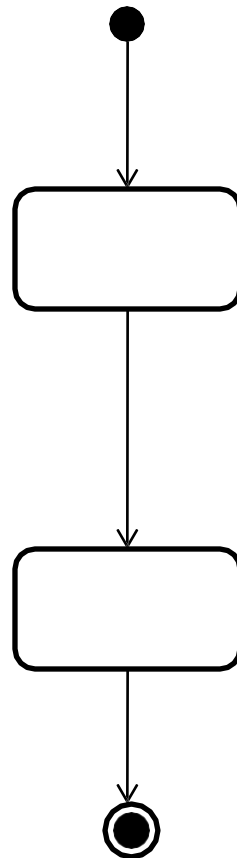






The most common  
route from the start  
point to the endpoint  
has many names.





But they effectively  
mean the same thing.

Combine any word on the  
left with any phrase on  
the right.

PRIMARY...

BASIC...

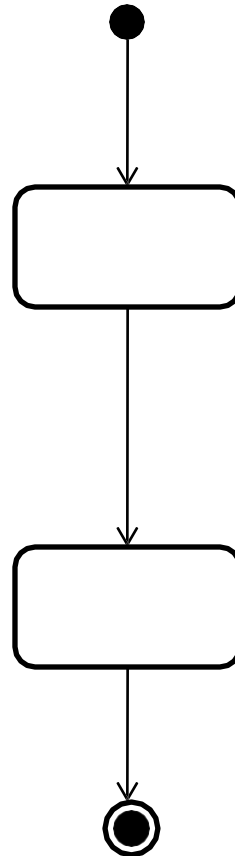
TYPICAL...

...PATH

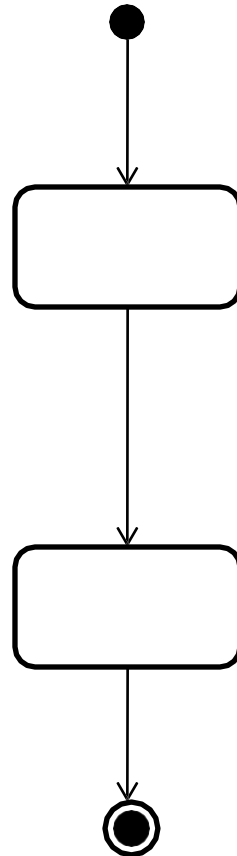
...FLOW

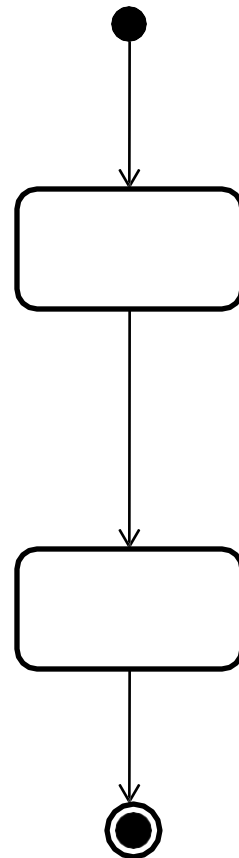
...COURSE OF EVENTS

...SCENARIO

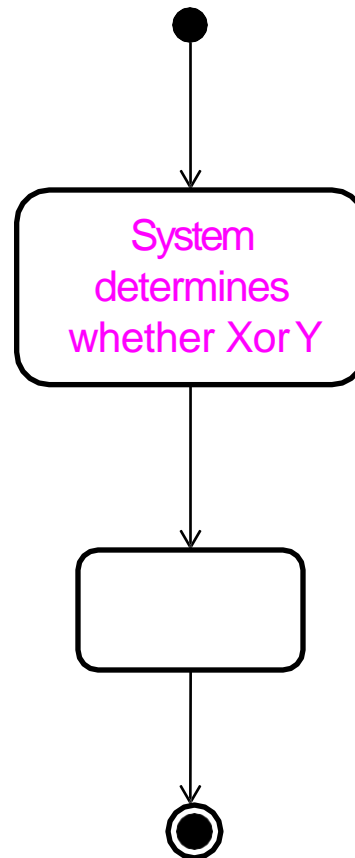


Often in a use case the  
System has to make a  
decision based on  
business rules...



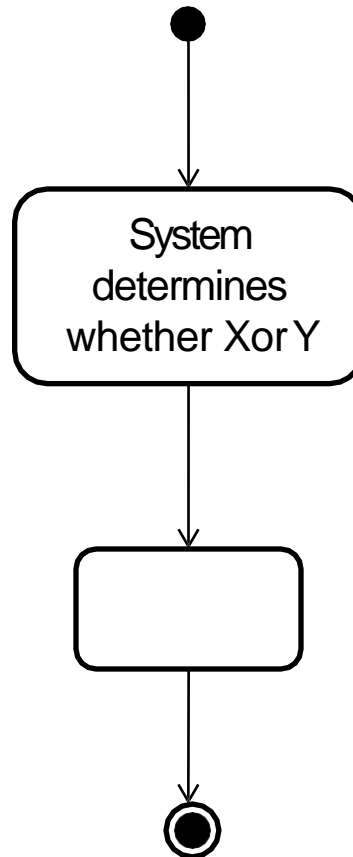


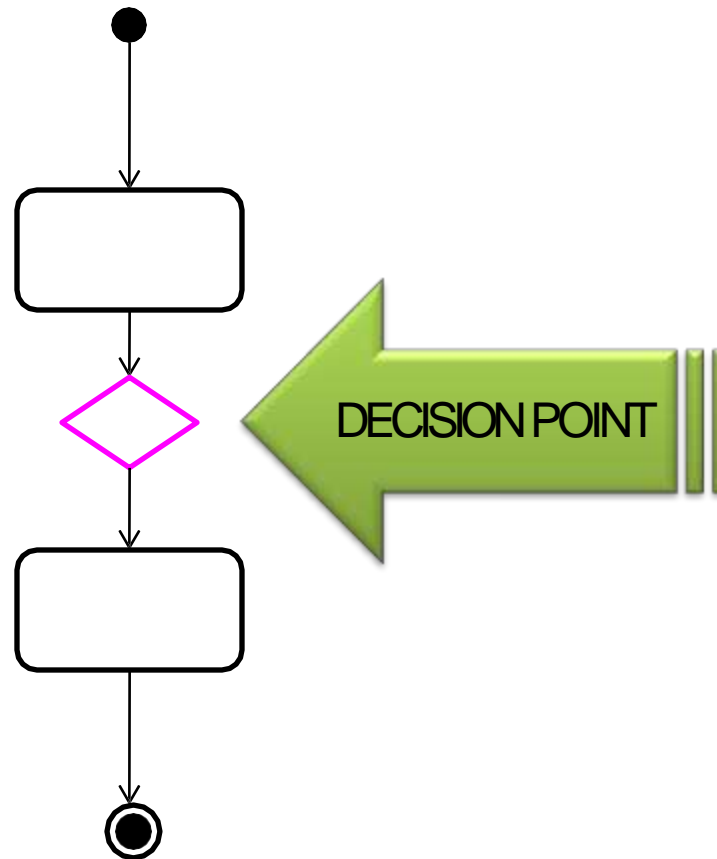
The actual decision  
takes place within a  
STEP



The actual decision  
takes place within a  
STEP

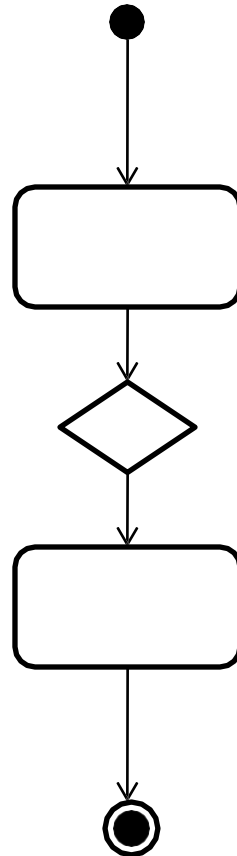
A DECISION POINT is then used to help the reader navigate the diagram.

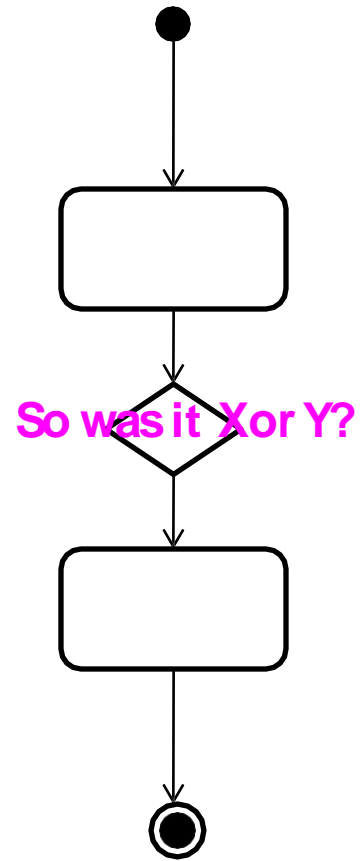


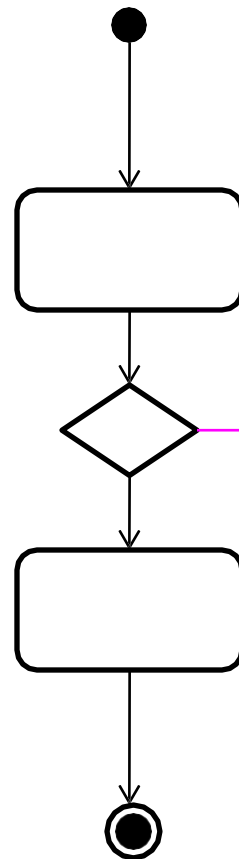




Decision Points contain  
text which describes  
the nature of the  
decision to be made.

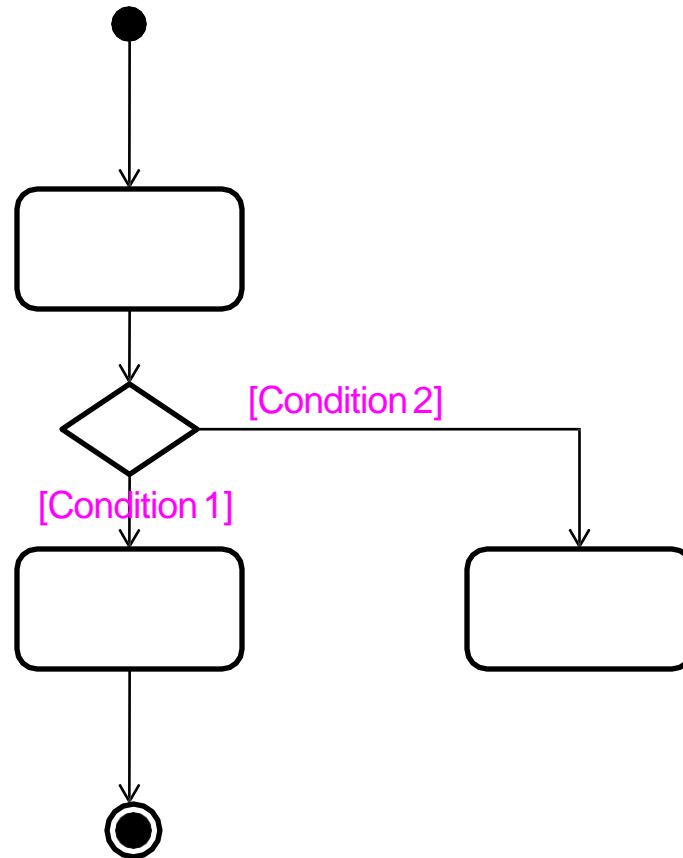




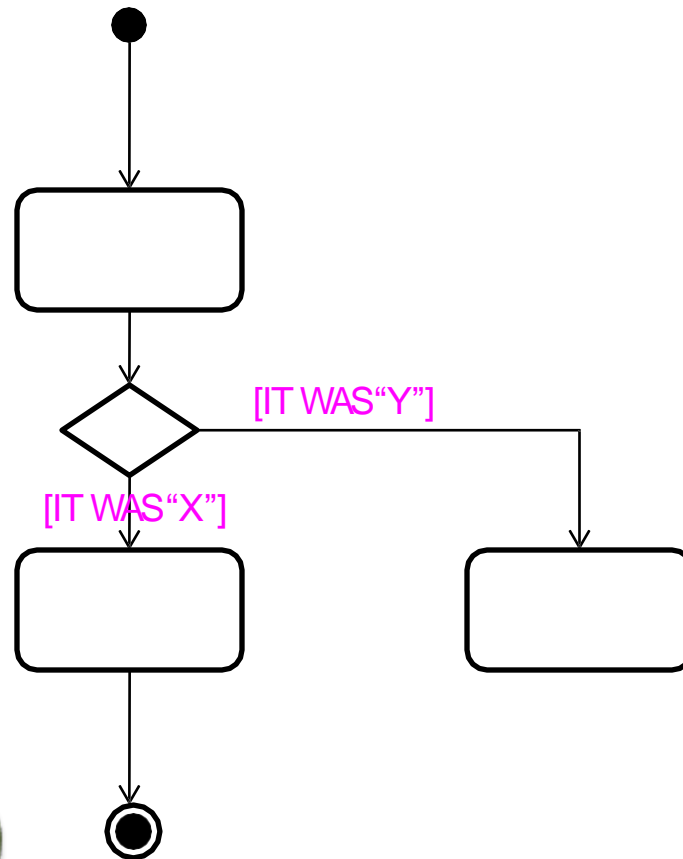


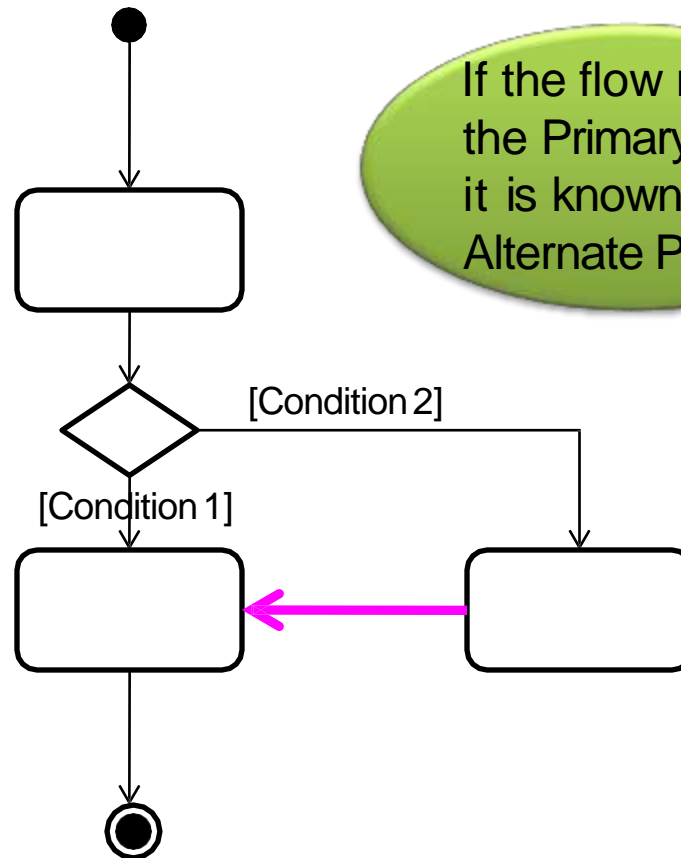
Decision points allow  
the flow to branch  
away from the Primary  
Path.

Transitions coming  
out of Decision  
Points must have a  
GUARD.



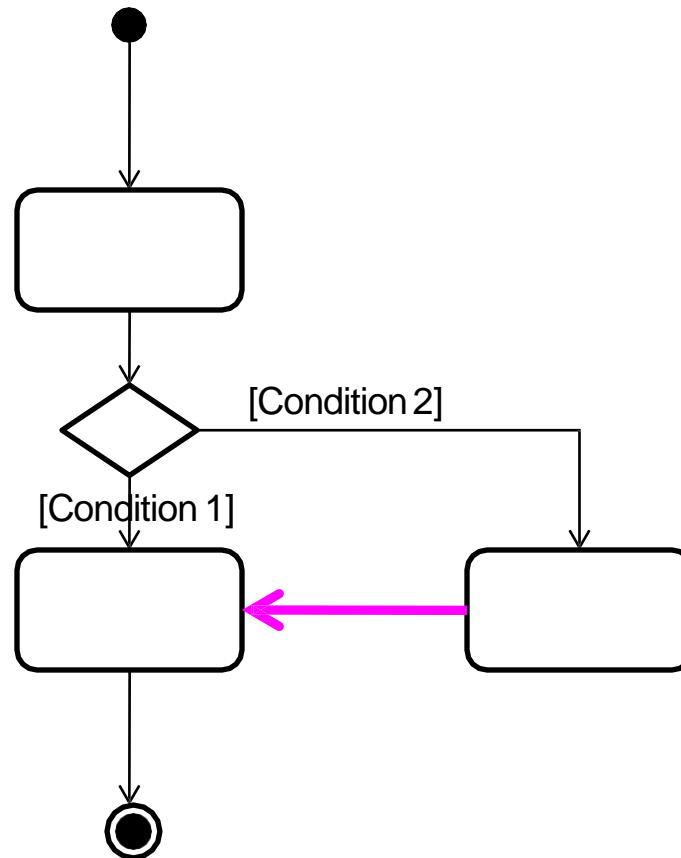
A Guard needs to explicitly describe a condition which must be true in order to proceed down that path.

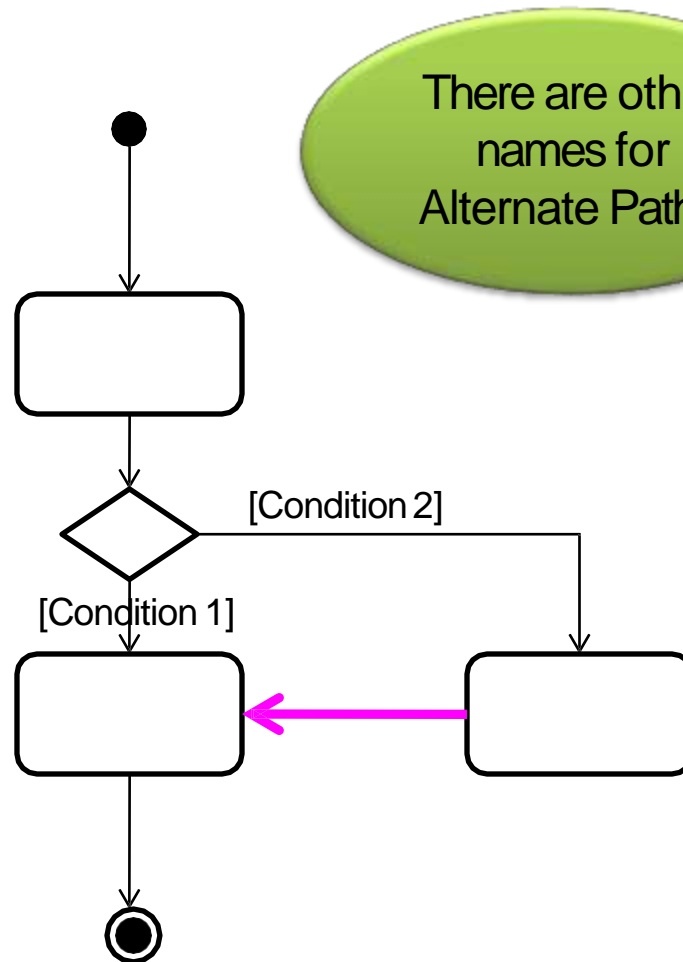




If the flow rejoins  
the Primary Path,  
it is known as an  
Alternate Path.

With Alternate  
Paths, the goal of  
the Use Case is  
still achieved.





There are other  
names for  
Alternate Paths.



Combine any word on the left with any phrase on the right.

ALTERNATE

...

ALTERNATIVE

...

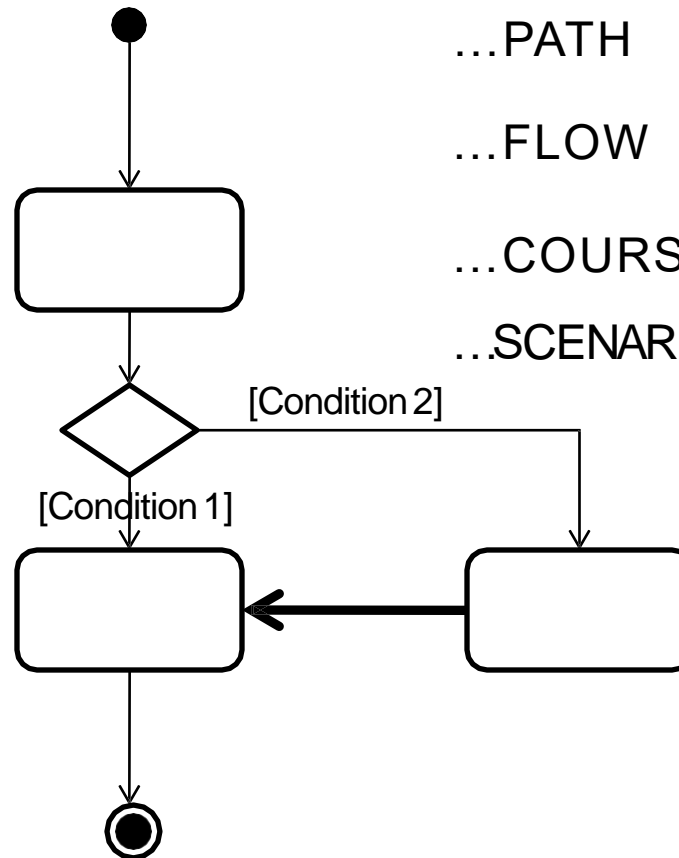
SECONDARY...

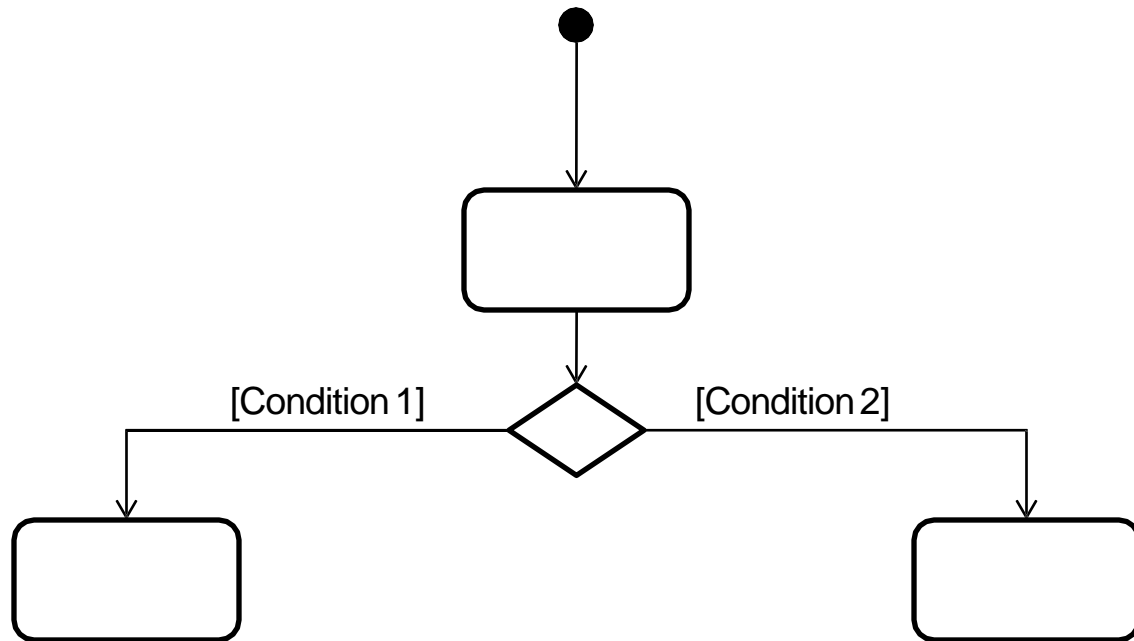
...PATH

...FLOW

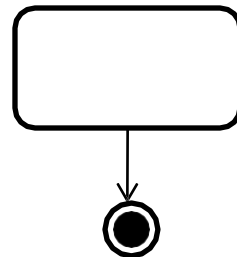
...COURSE OF EVENTS

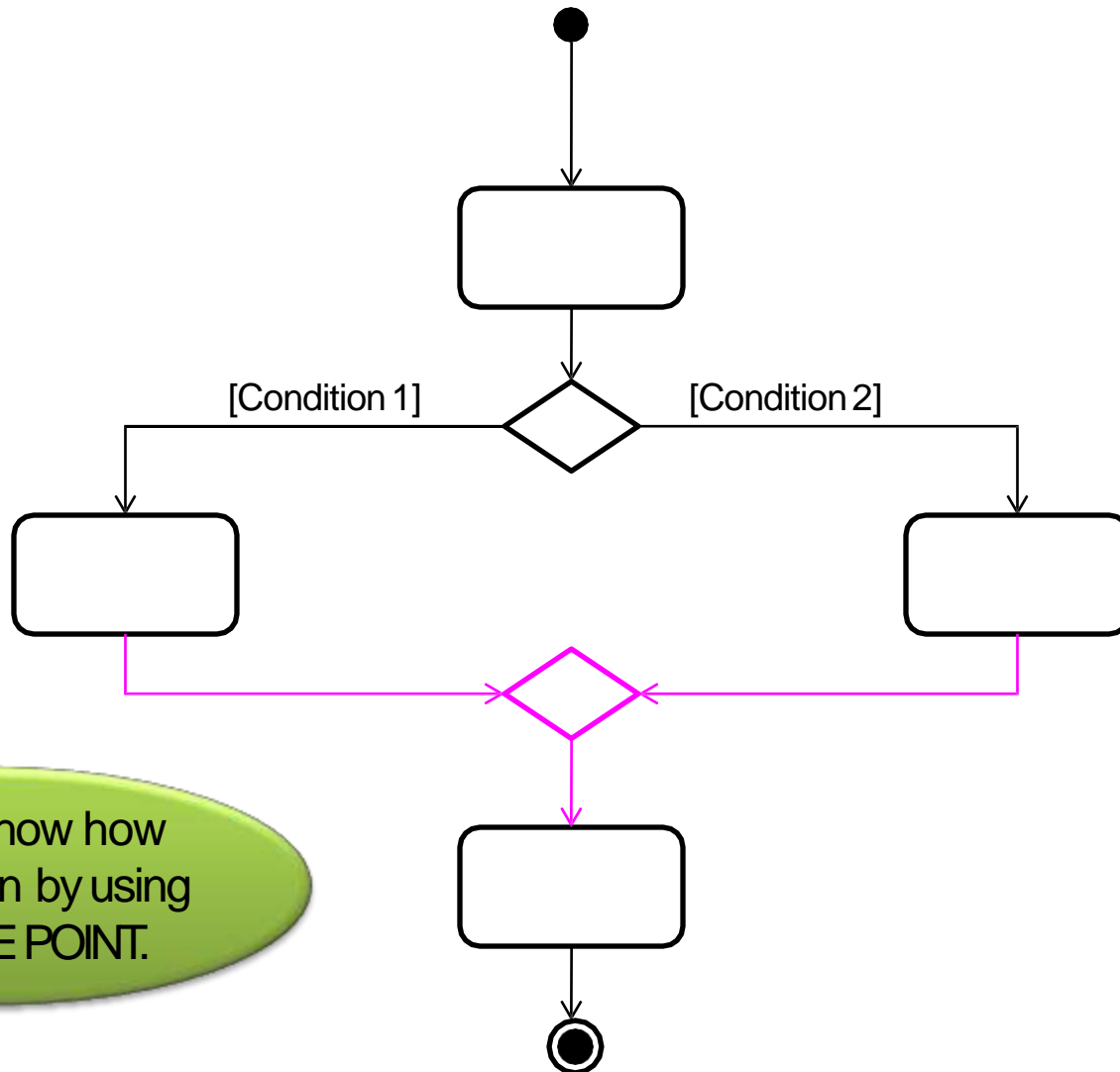
...SCENARIO



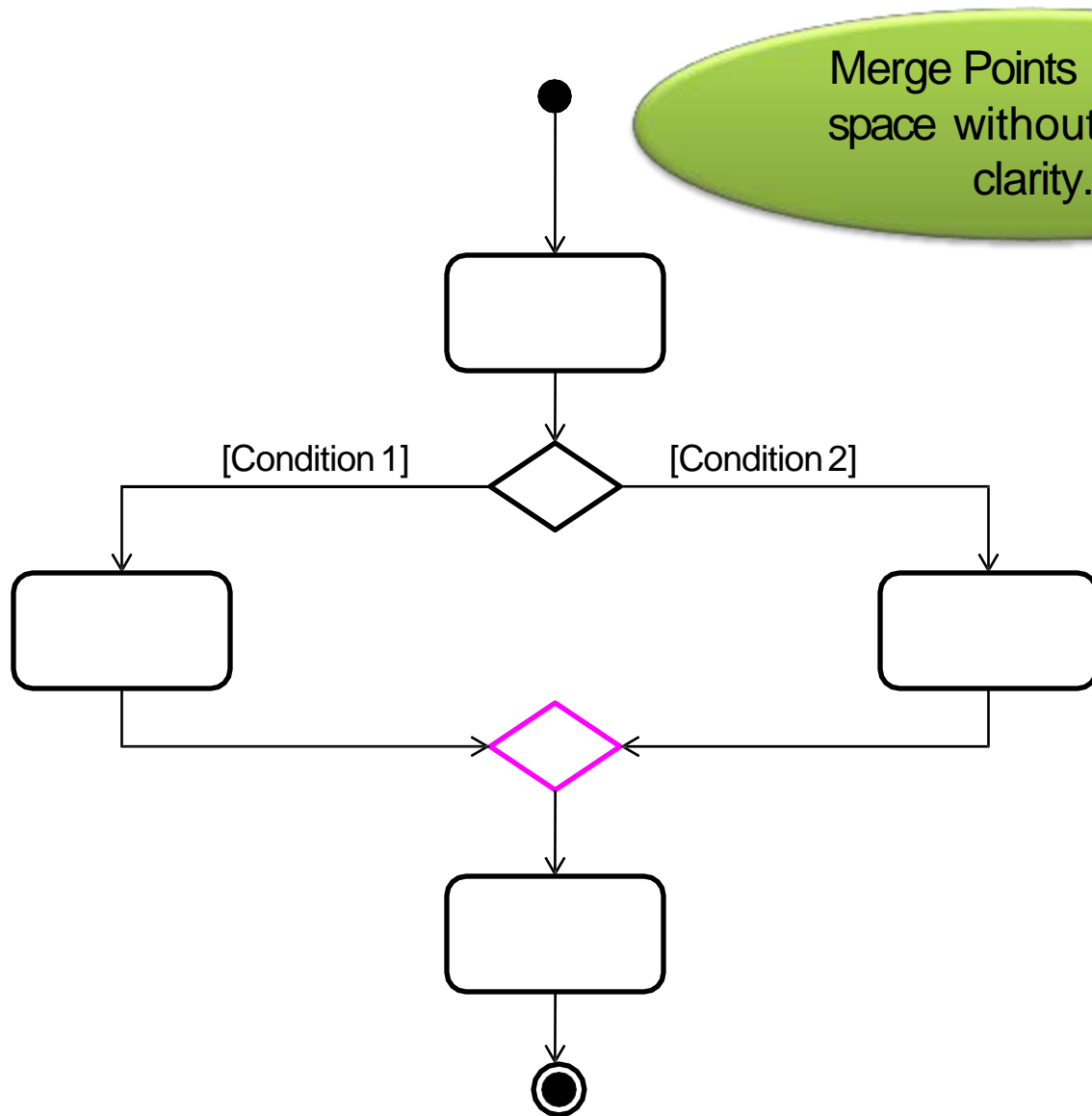


You can show how  
paths rejoin by using  
a MERGE POINT.



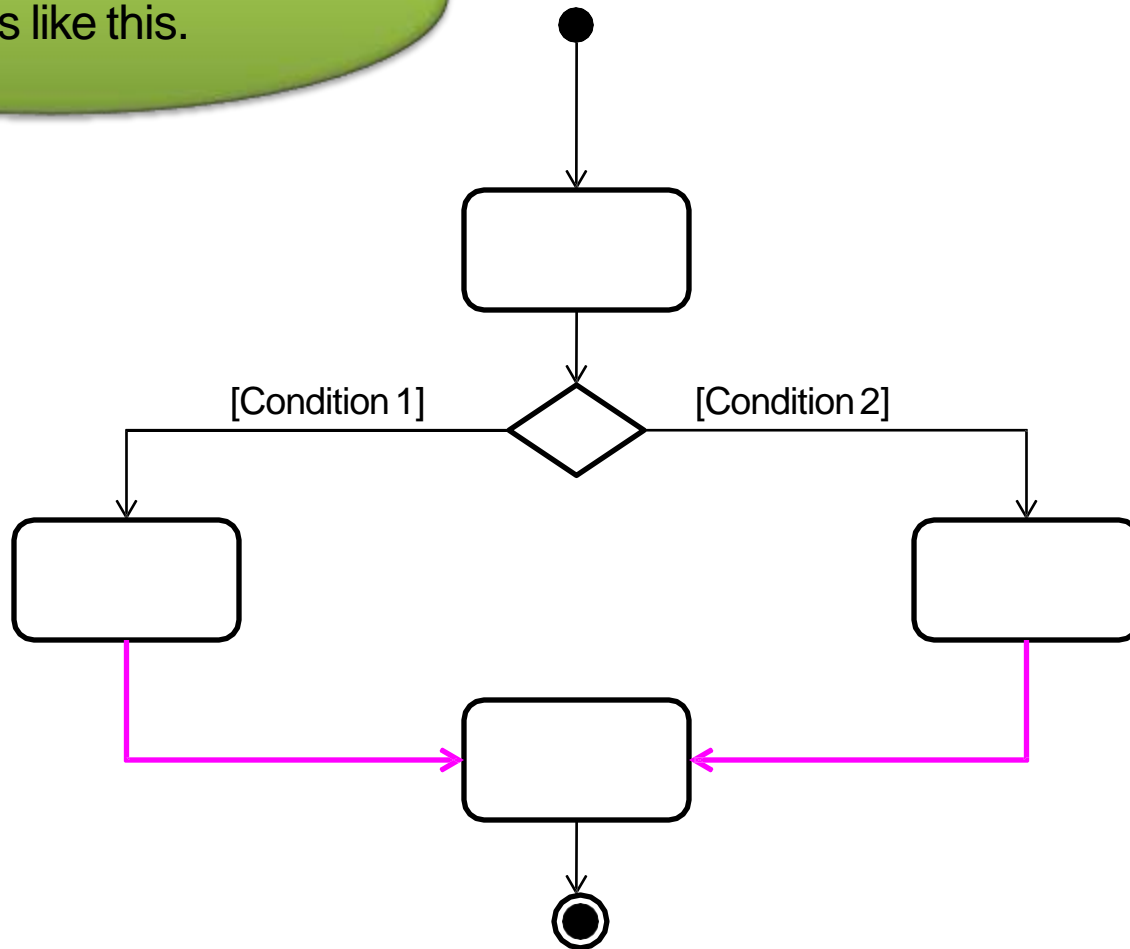


You can show how  
paths rejoin by using  
a MERGE POINT.

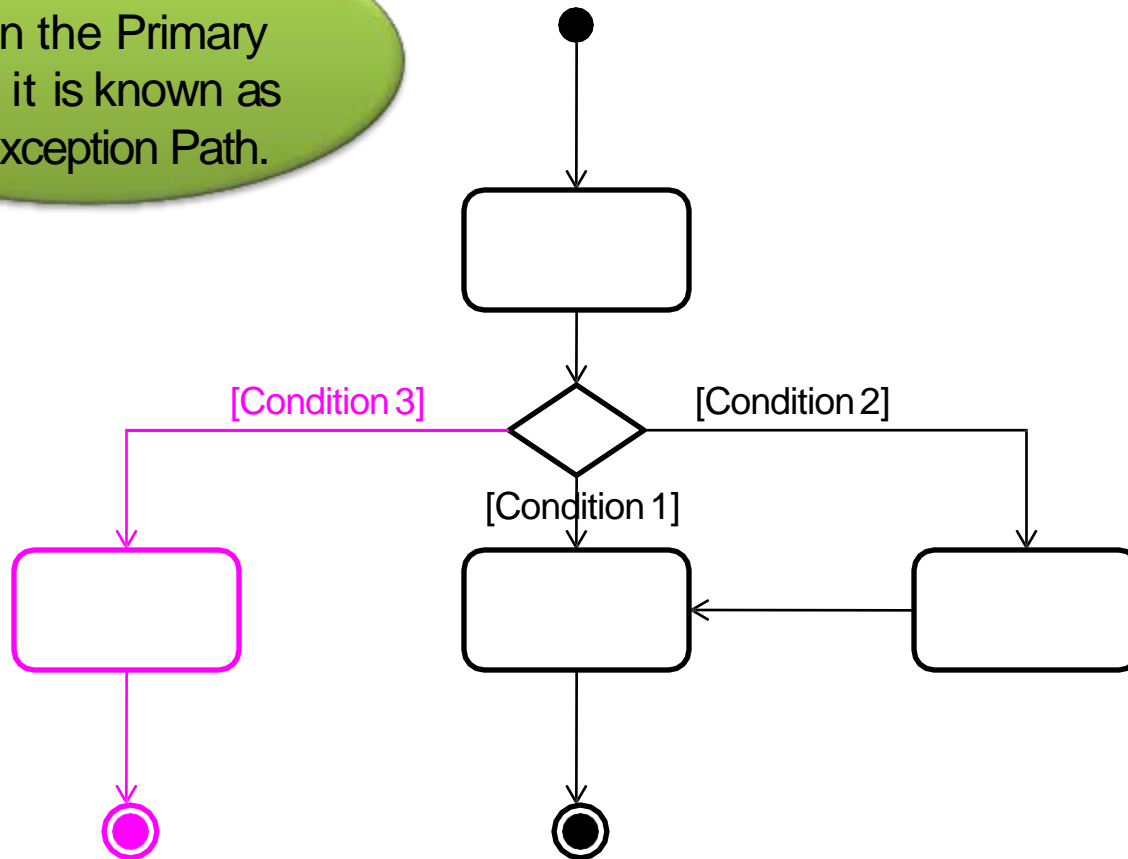


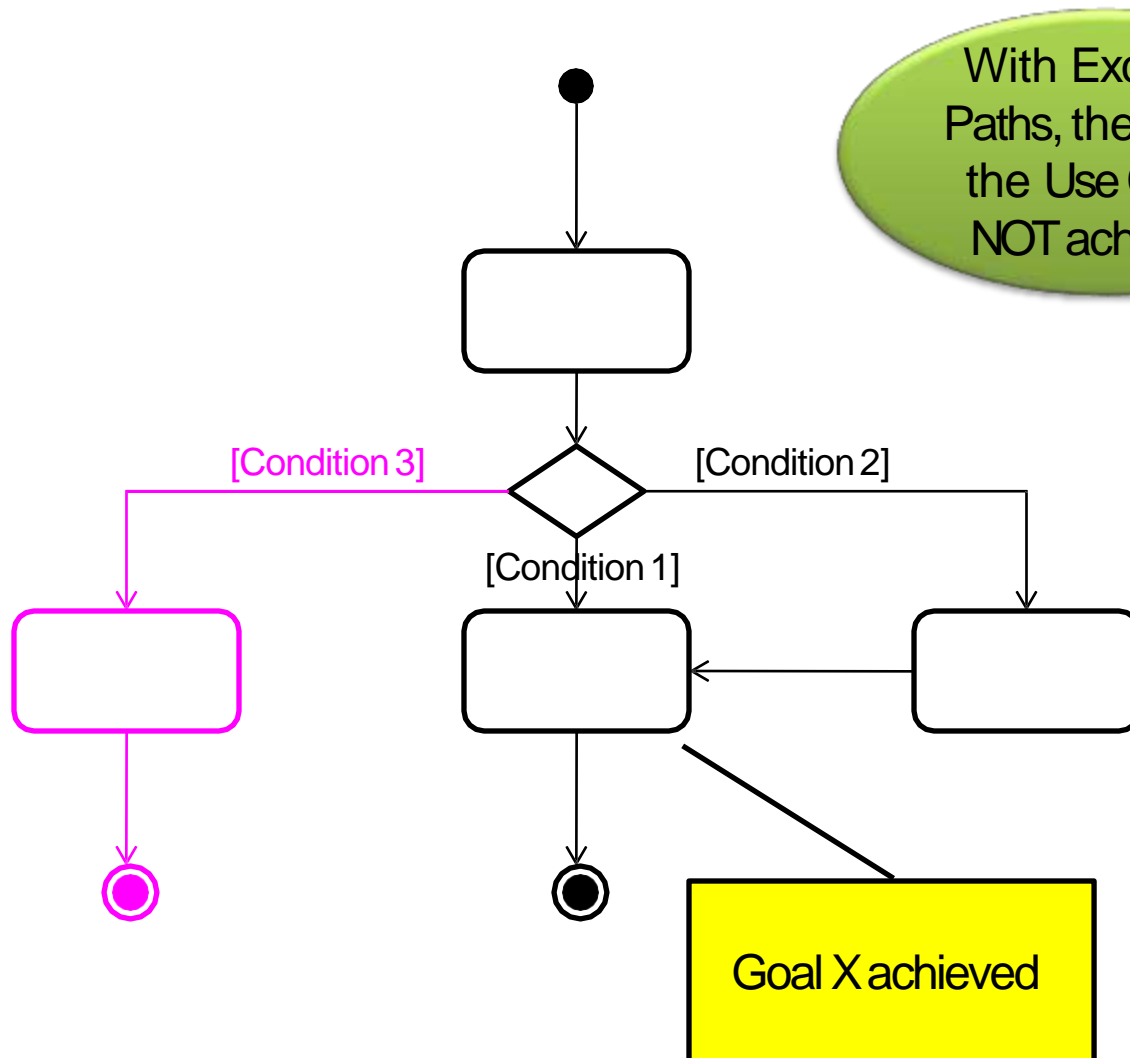
Merge Points take up space without adding clarity.

We can model merging paths like this.

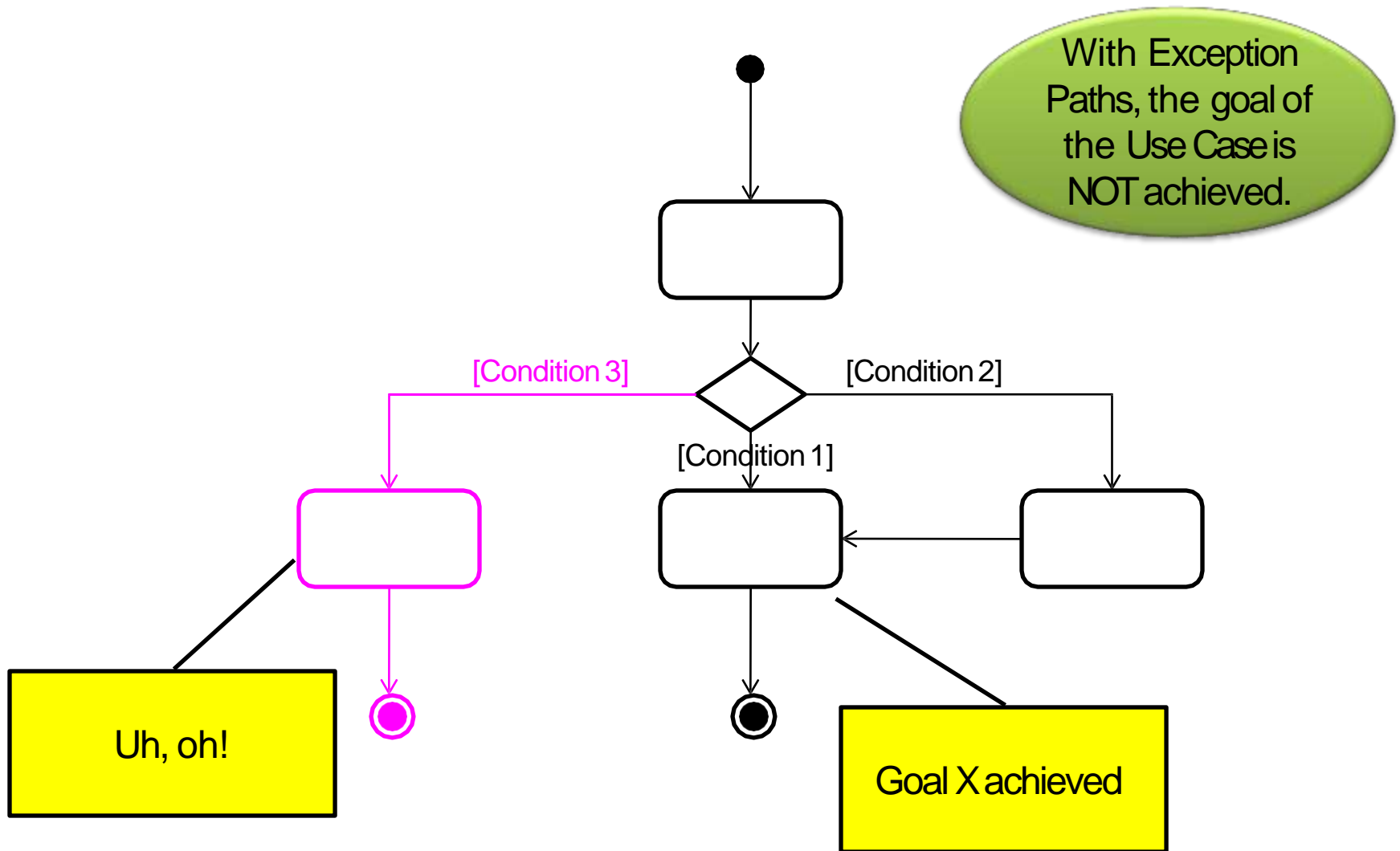


If the flow does NOT  
rejoin the Primary  
Path, it is known as  
an Exception Path.



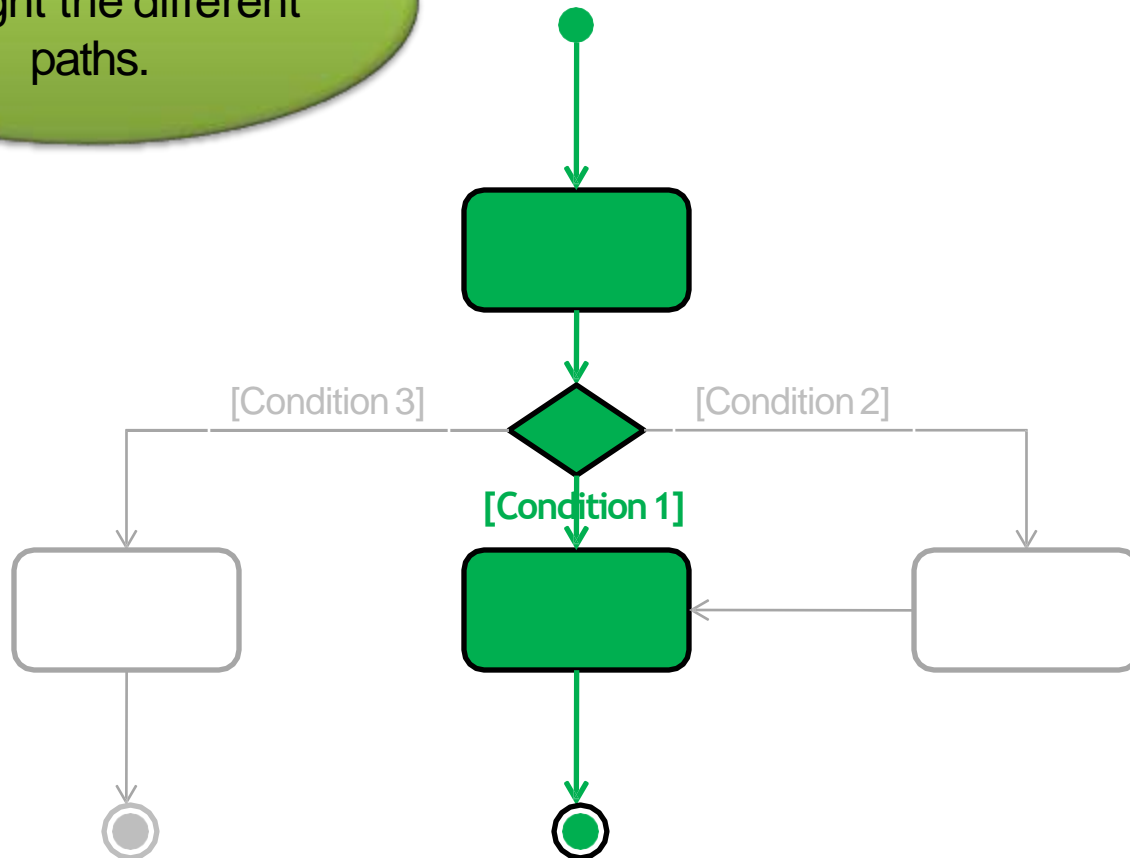


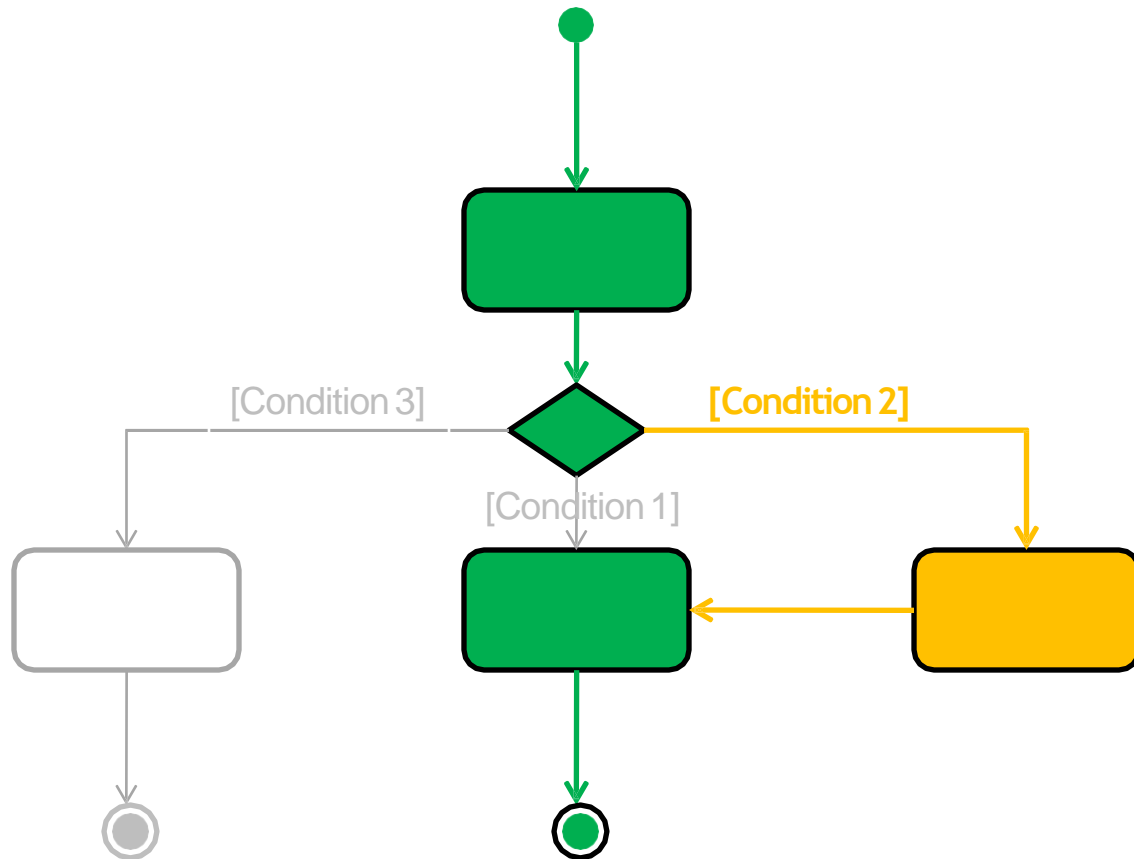
With Exception  
Paths, the goal of  
the Use Case is  
NOT achieved.

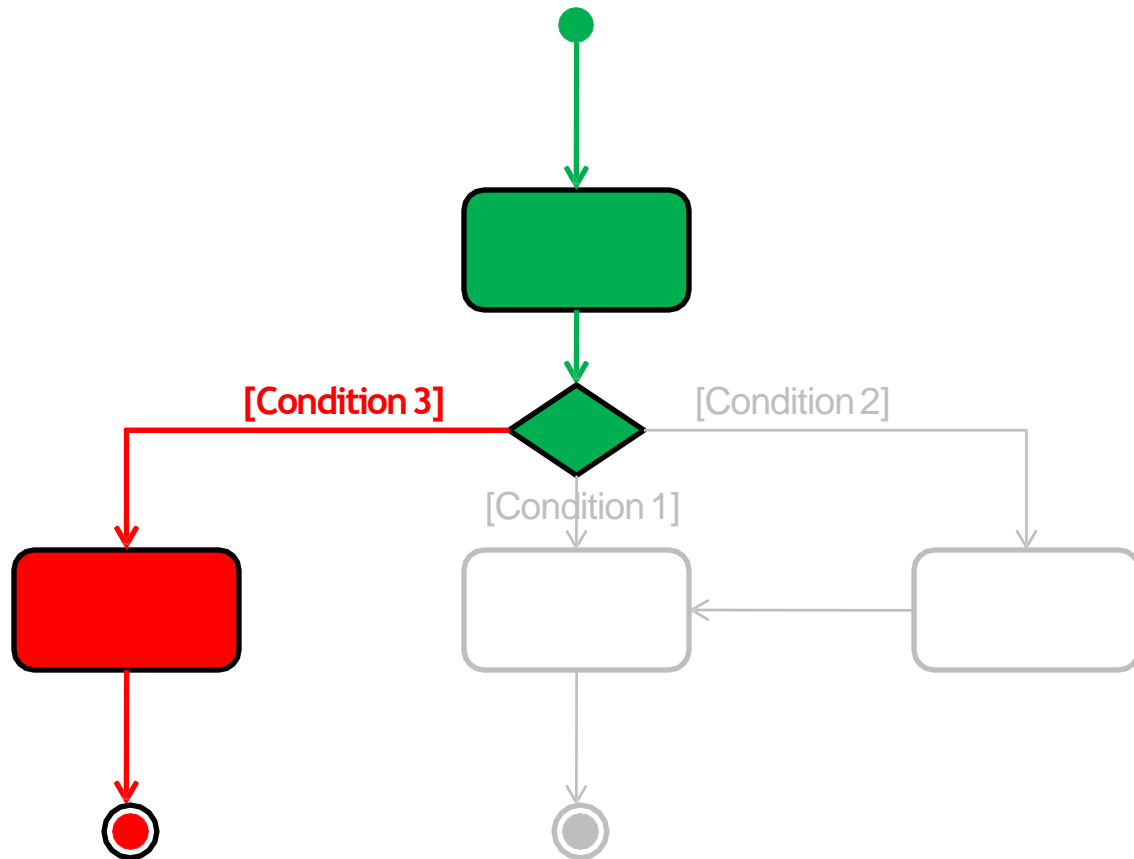




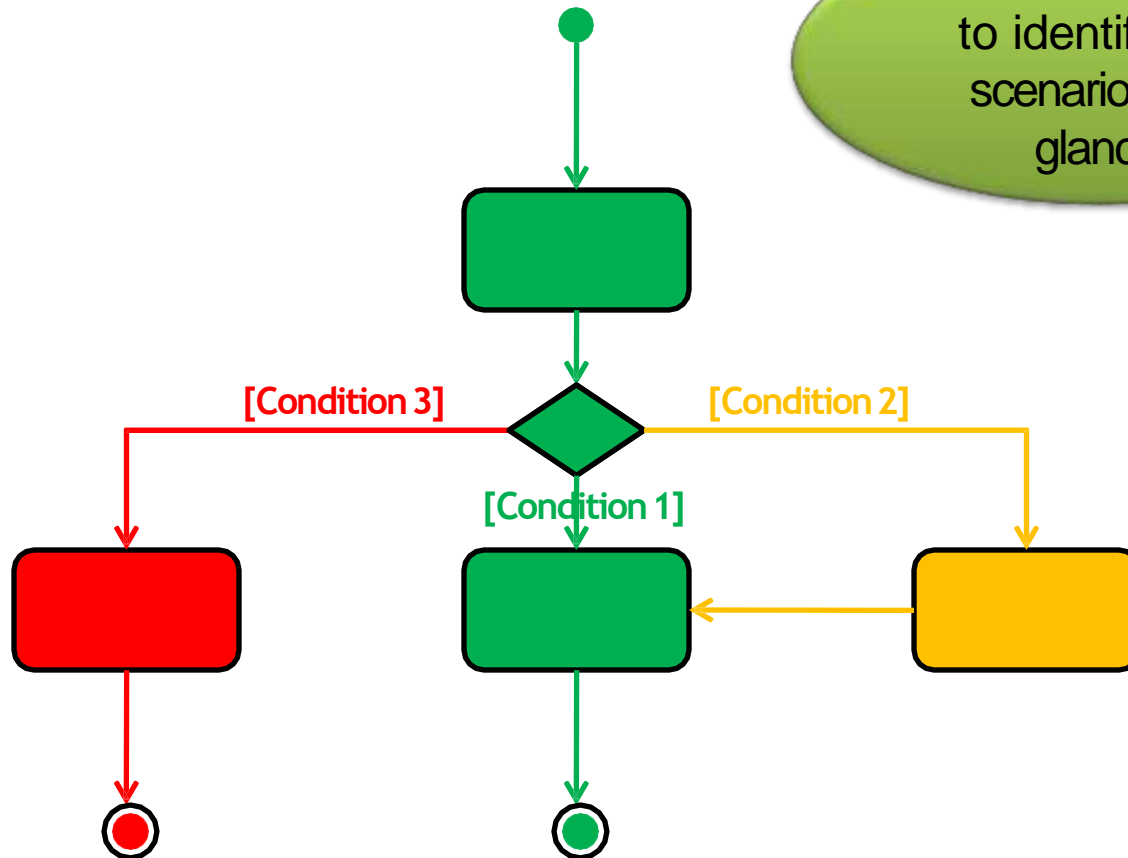
Different colours highlight the different paths.



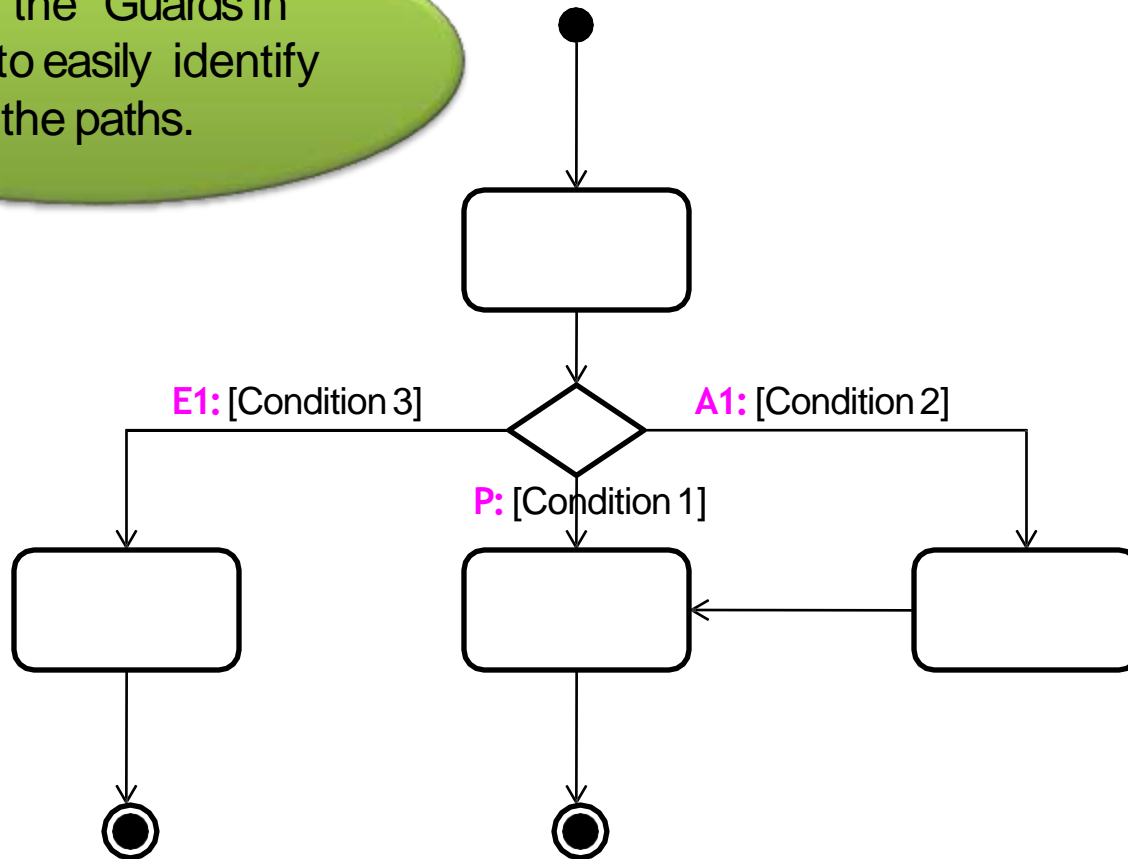




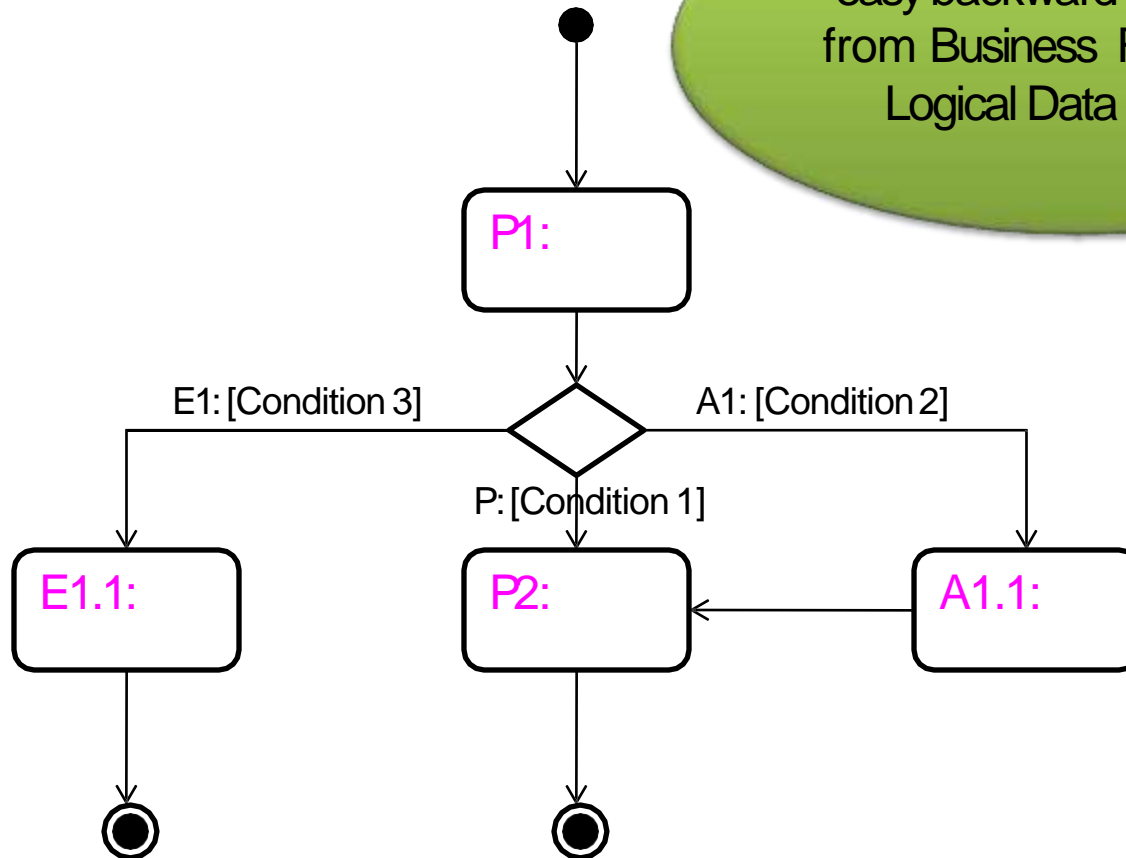
This makes it easy  
to identify test  
scenarios at a  
glance.



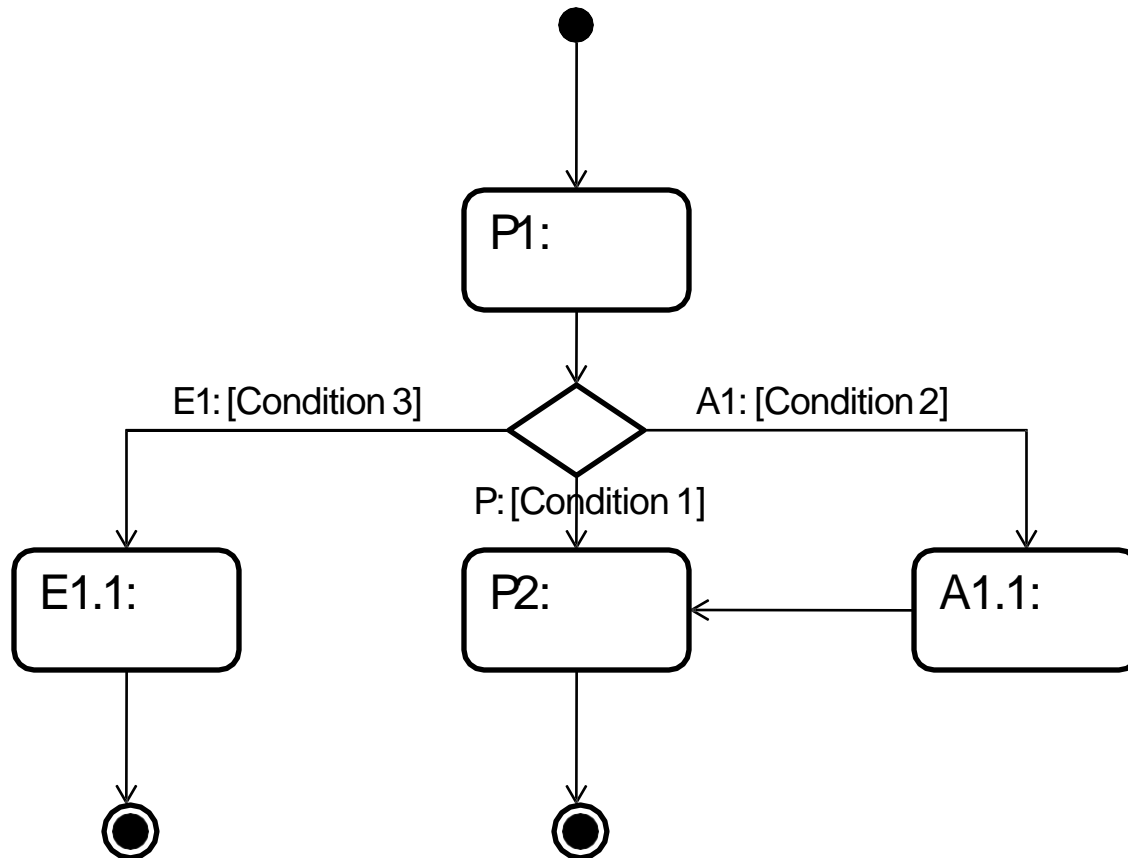
label the Guards in  
order to easily identify  
the paths.



And label the Steps for  
easy backward reference  
from Business Rules and a  
Logical Data Model.

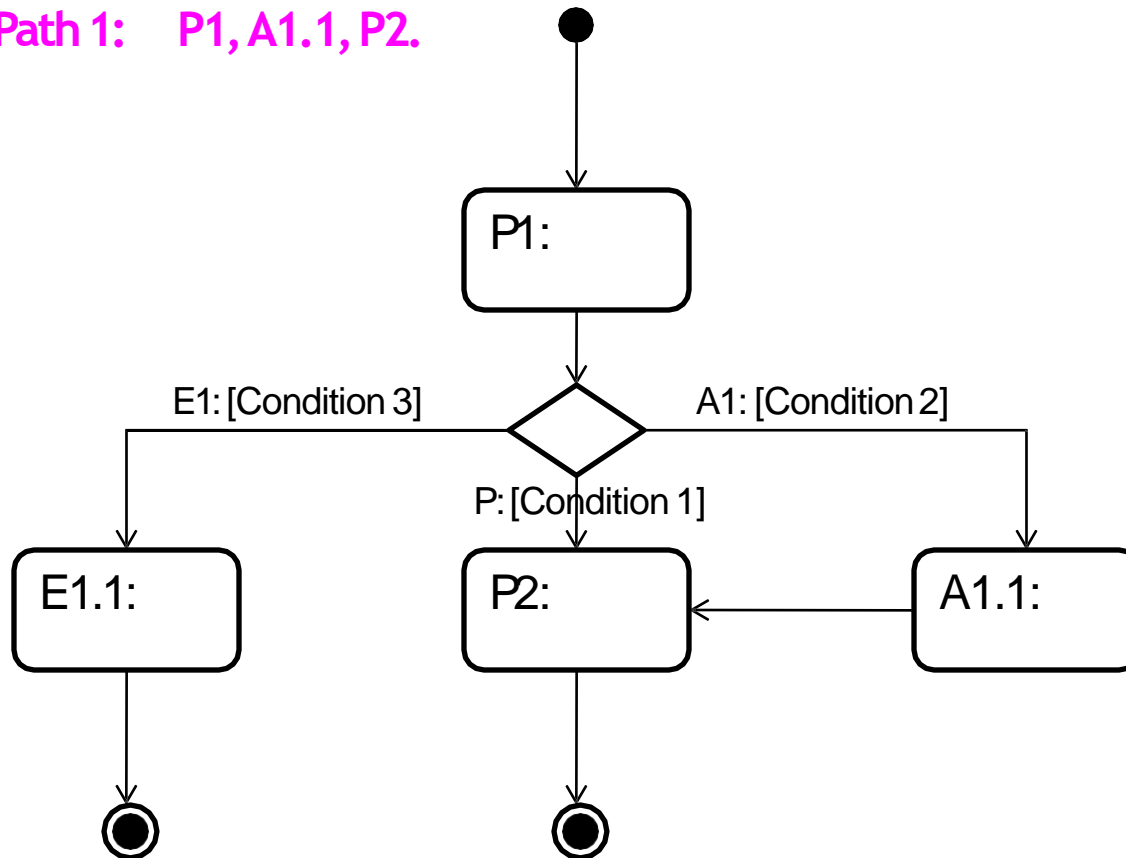


Primary Path: P1, P2.



Primary Path: P1, P2.

Alternate Path 1: P1, A1.1, P2.

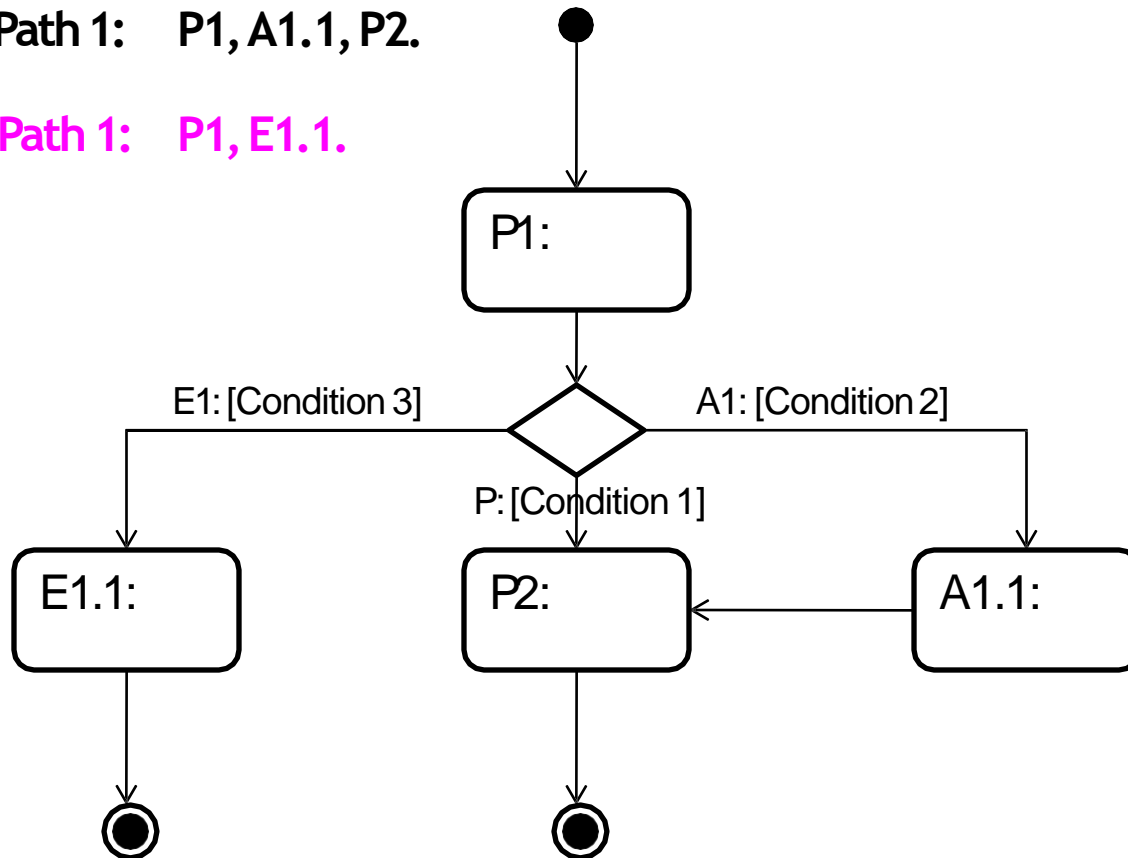




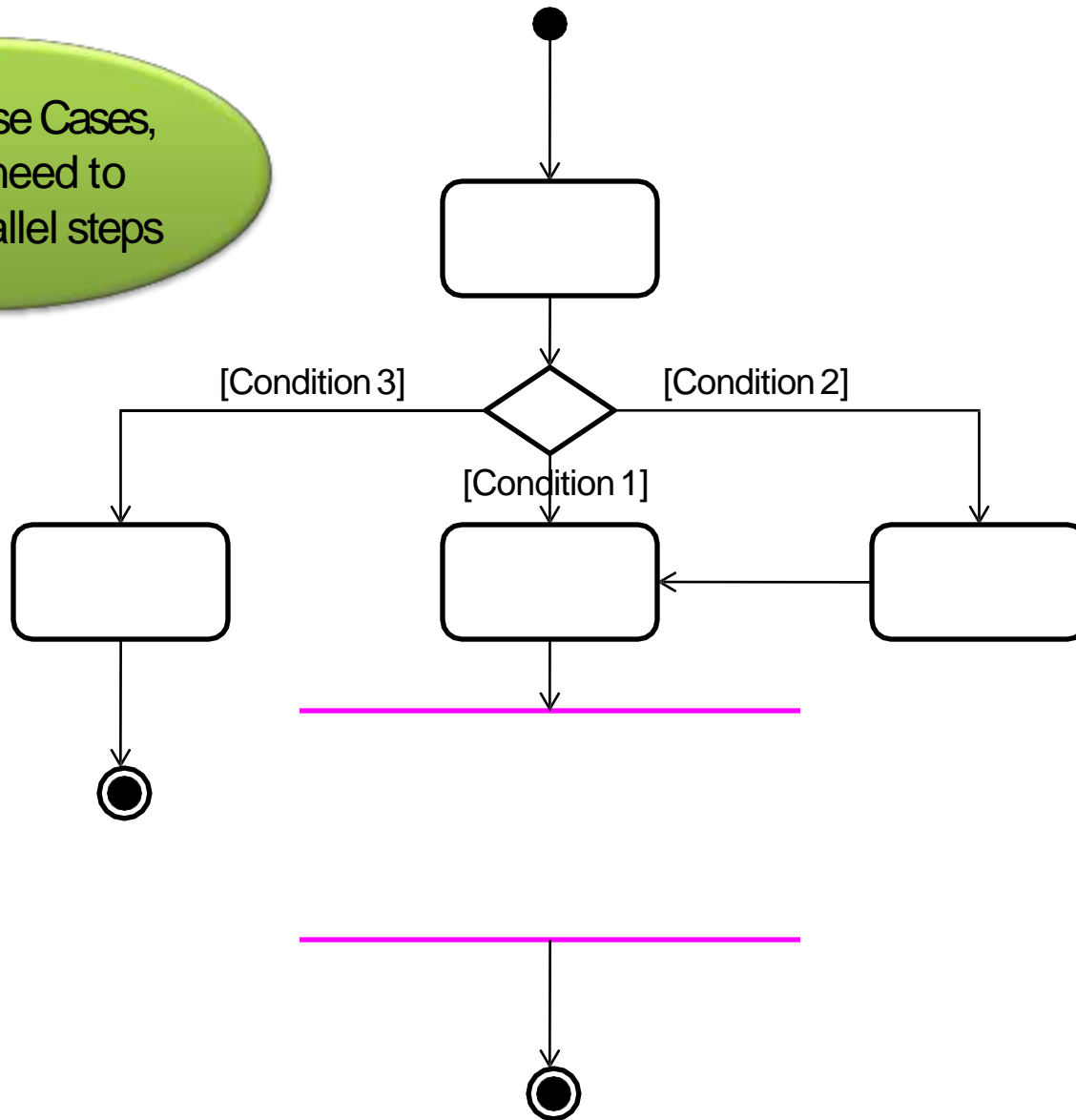
Primary Path: P1, P2.

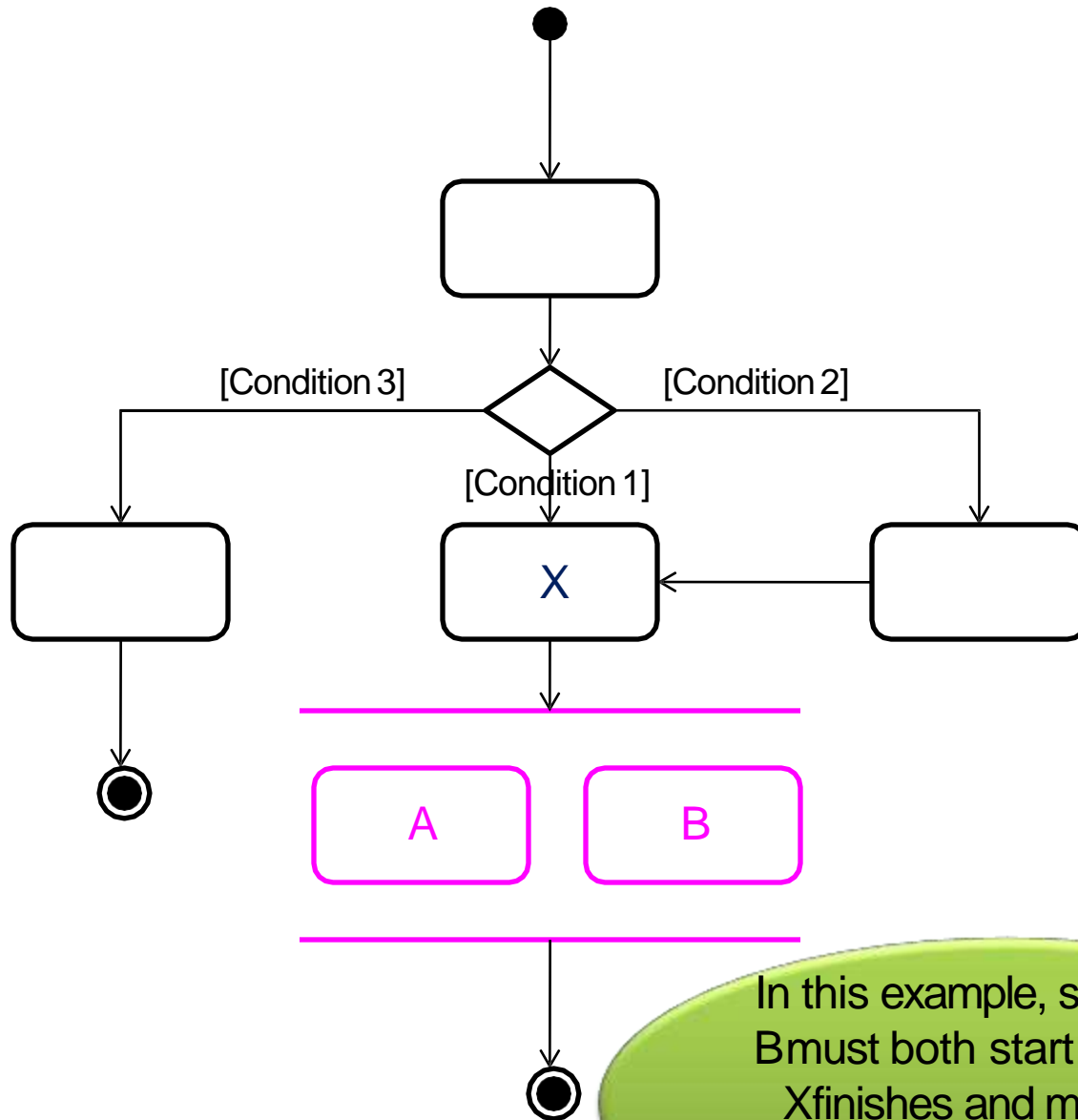
Alternate Path 1: P1, A1.1, P2.

Exception Path 1: P1, E1.1.

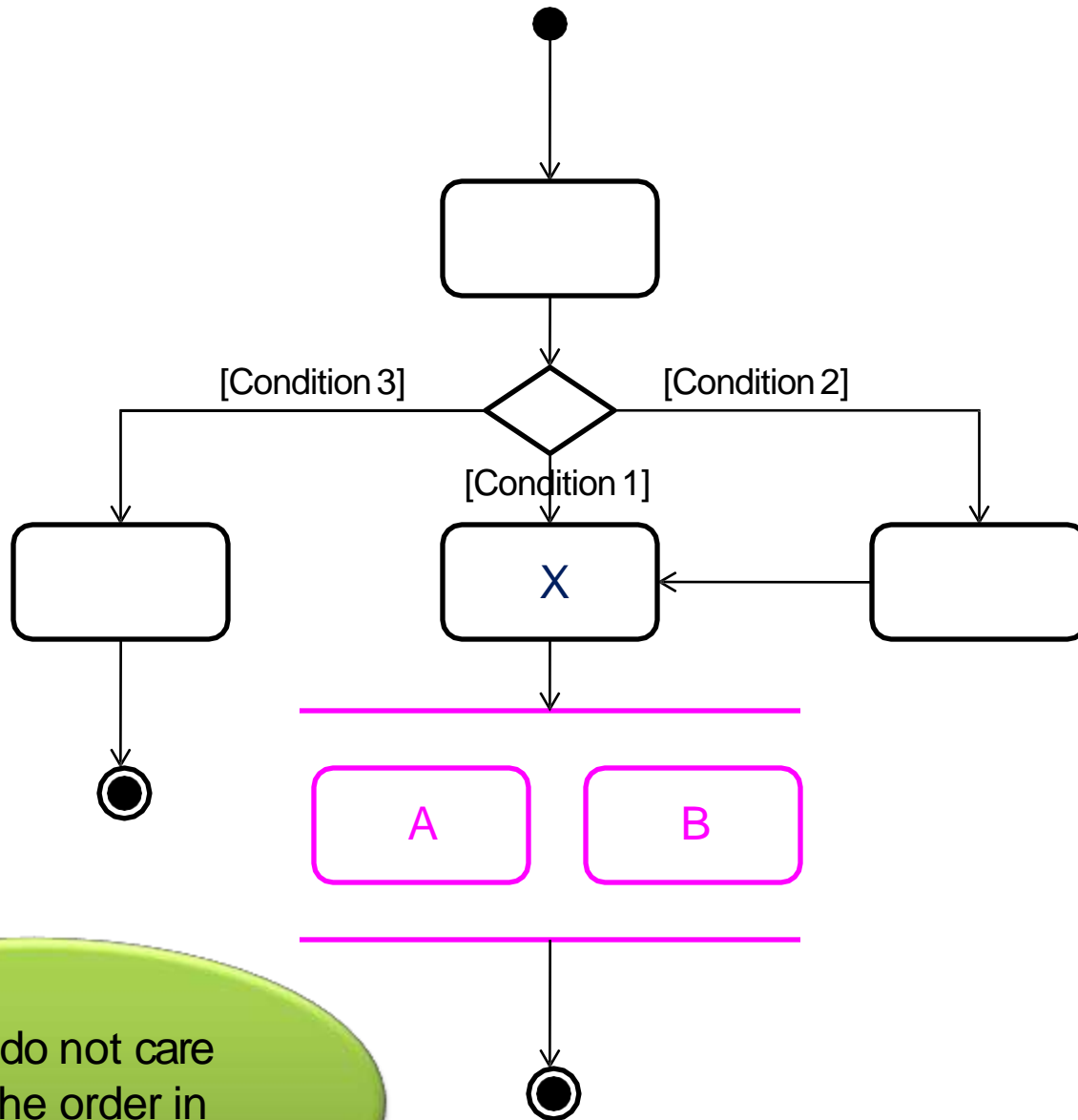


In some Use Cases,  
you will need to  
model parallel steps

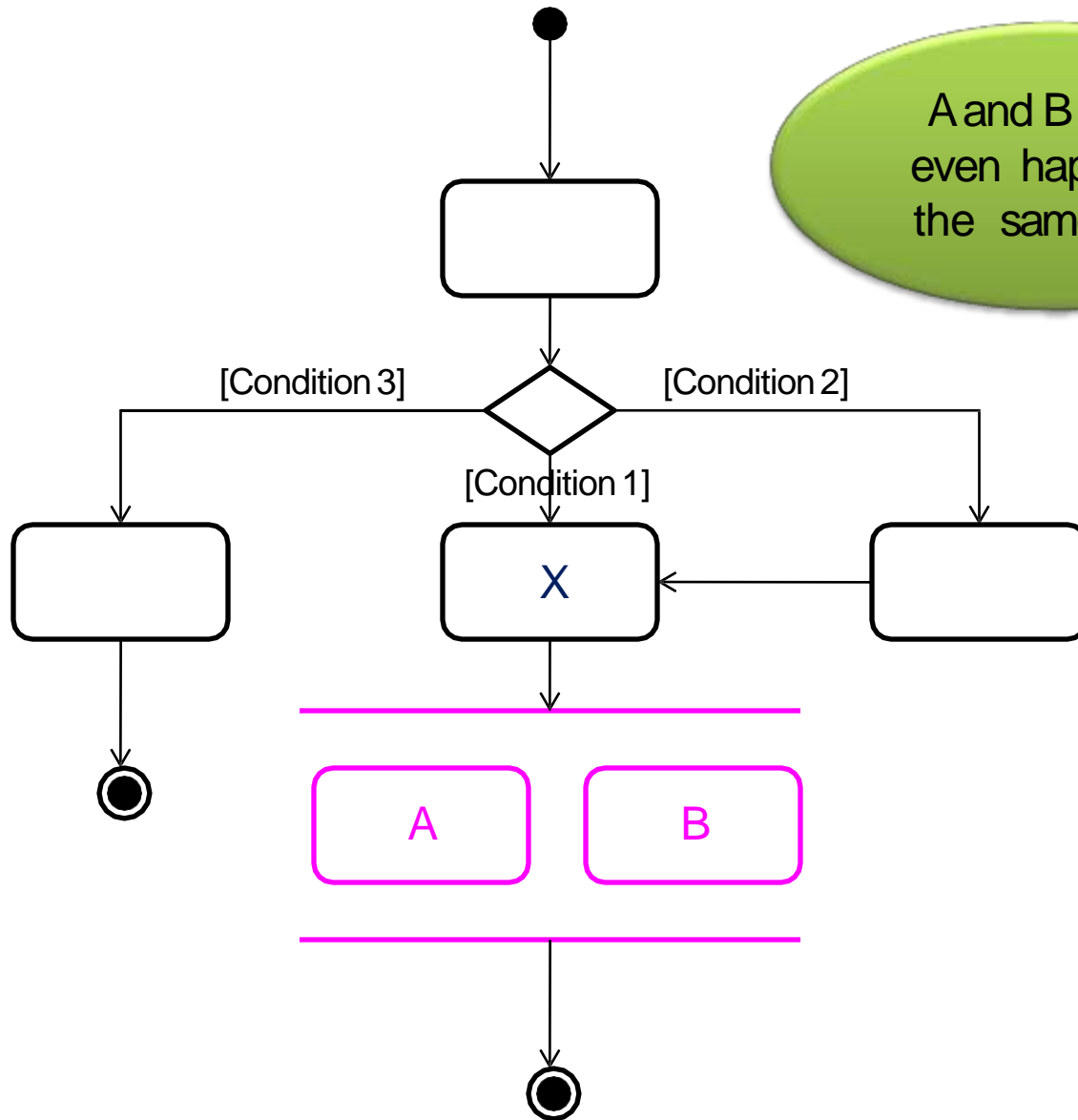




In this example, steps A and B must both start after step X finishes and must both finish before the Use Case ends.

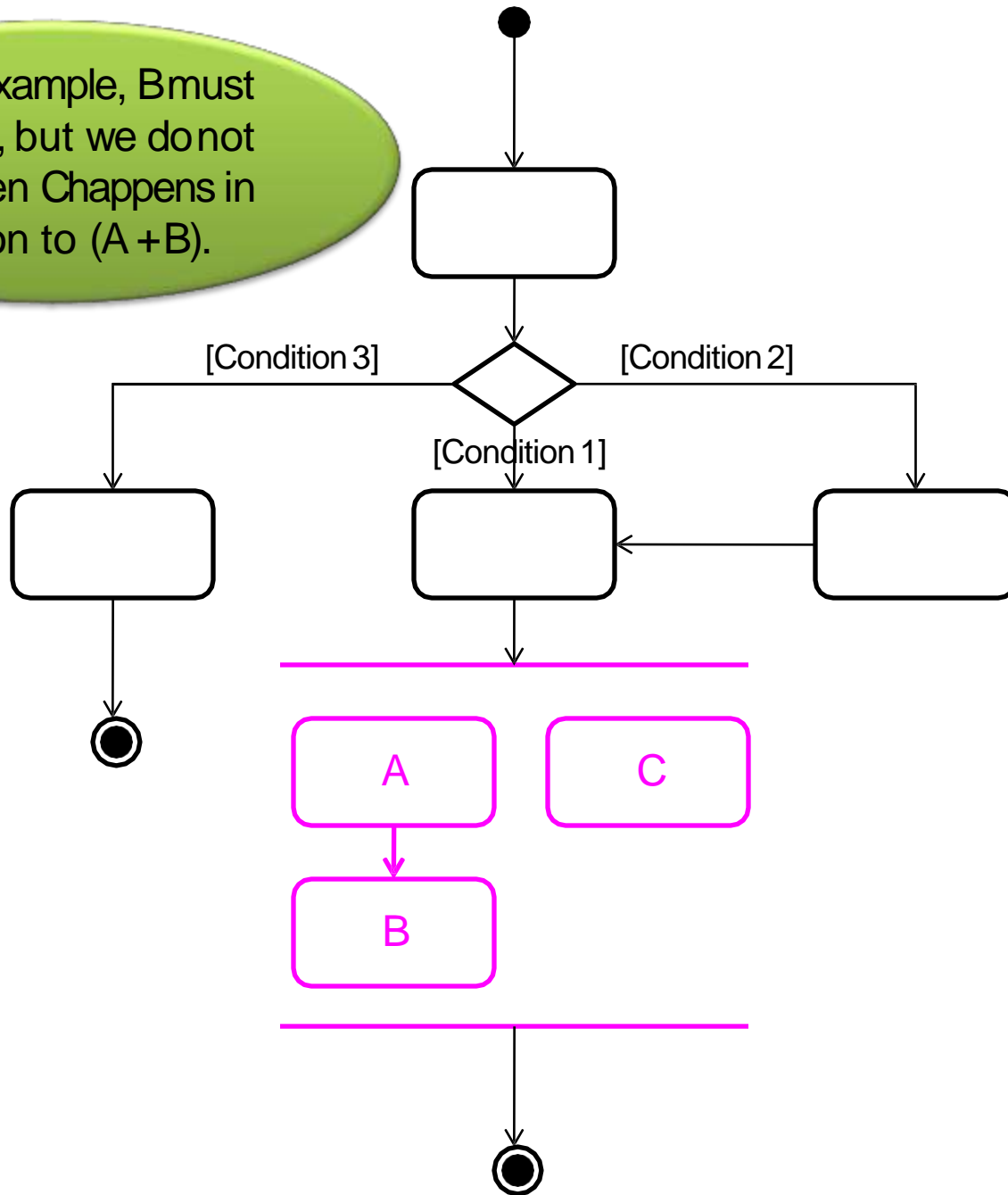


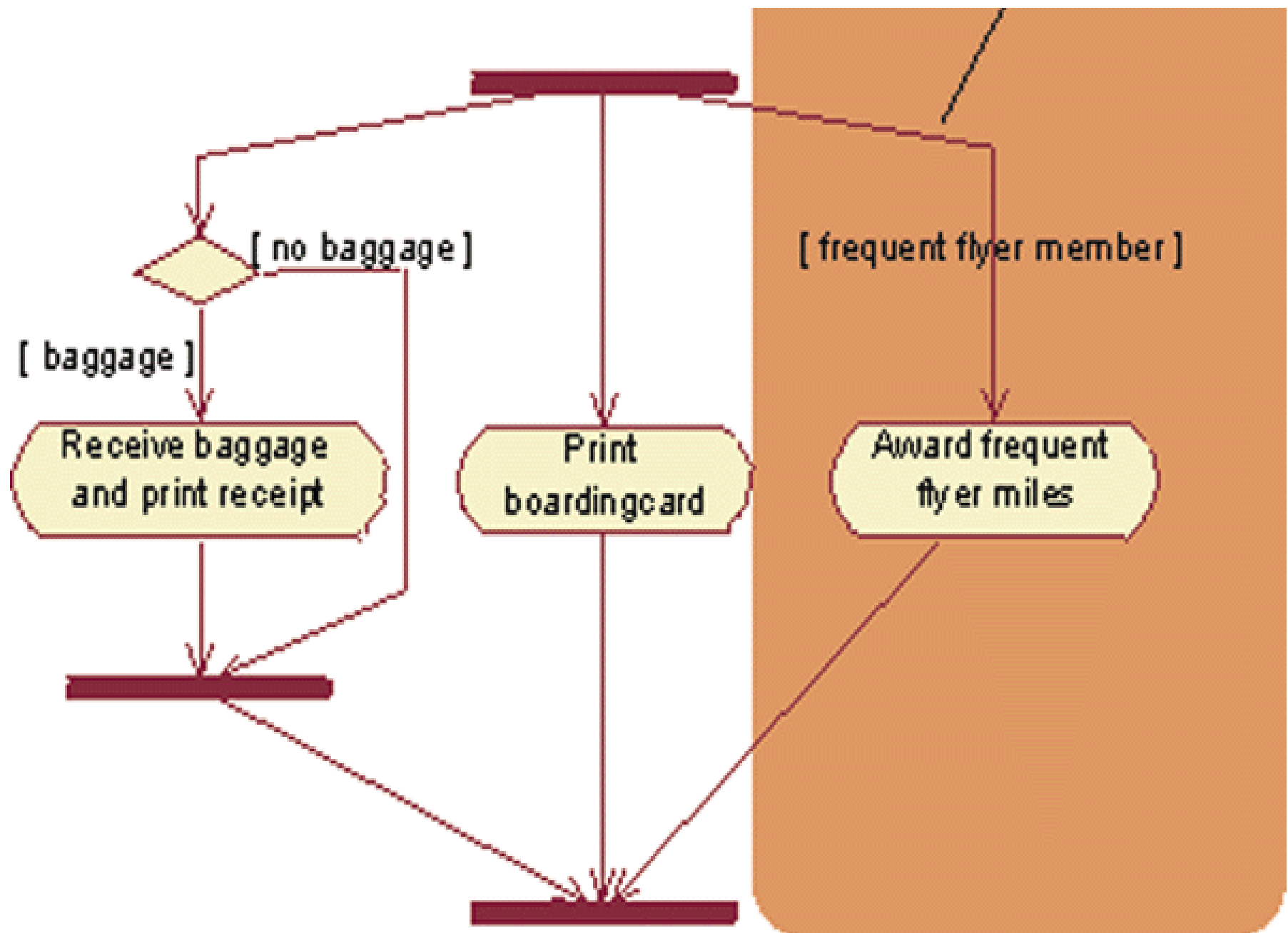
But we do not care  
about the order in  
which A and B happen.

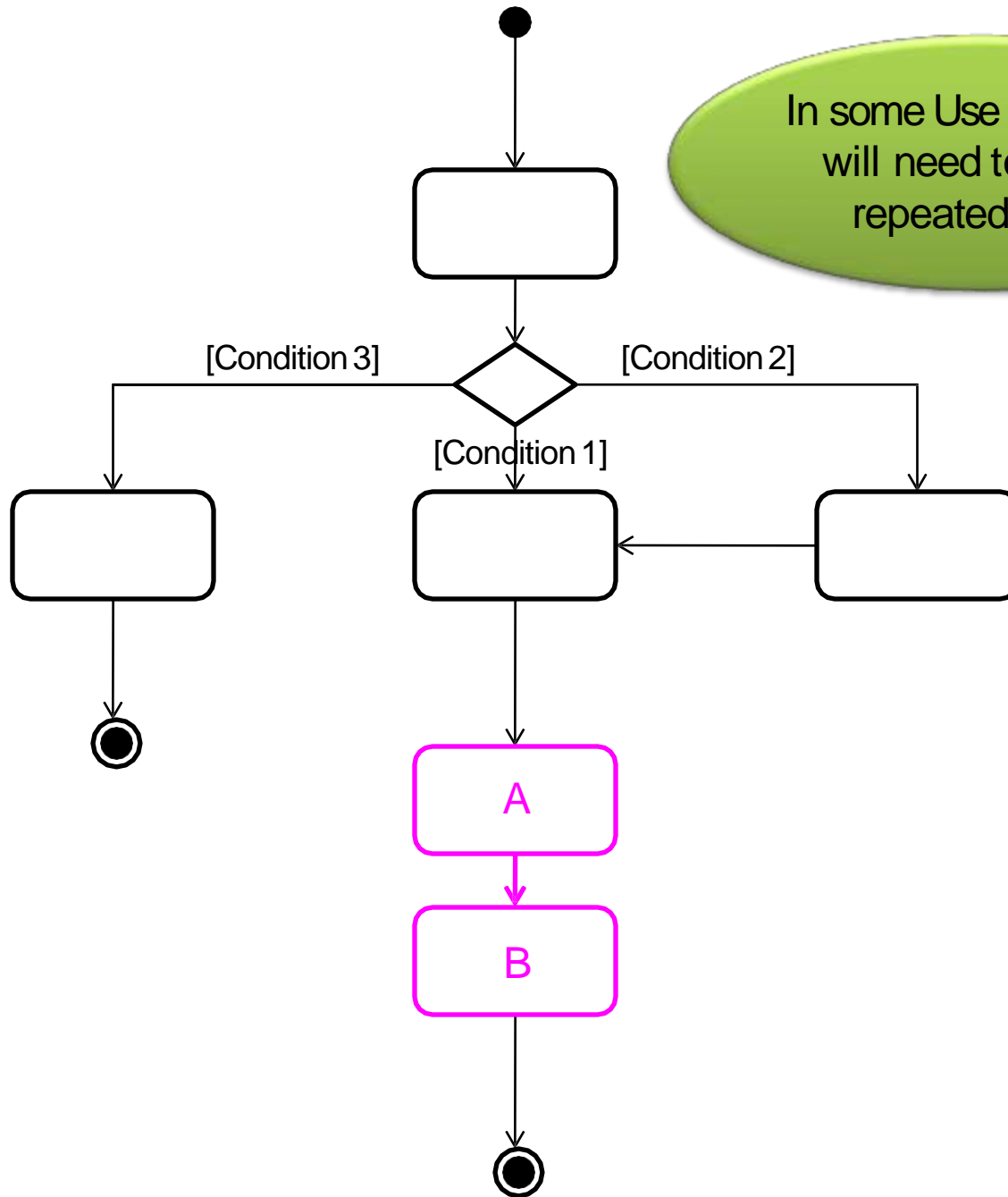


A and B could even happen at the same time.

In this example, B must follow A, but we do not care when C happens in relation to (A + B).

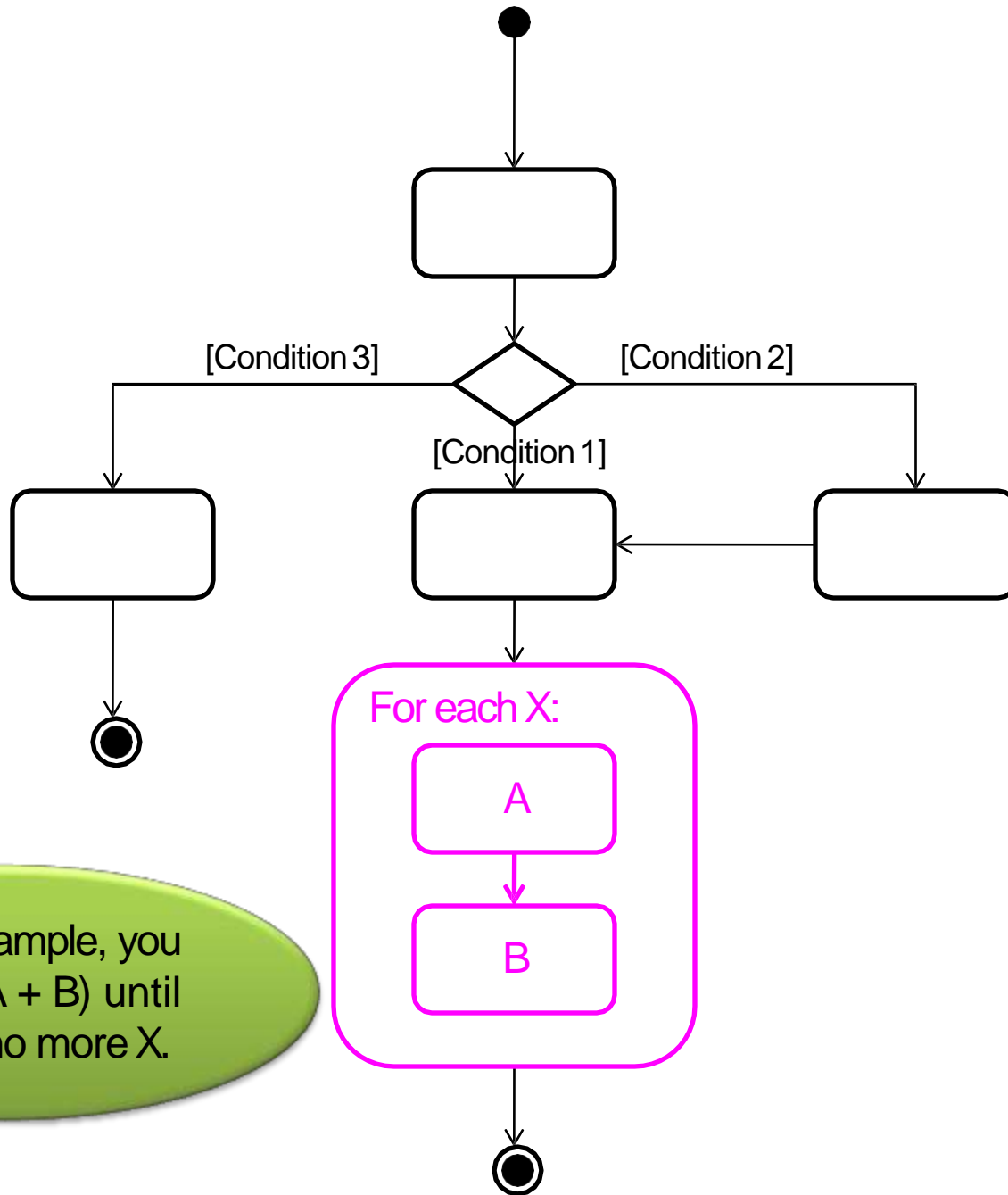




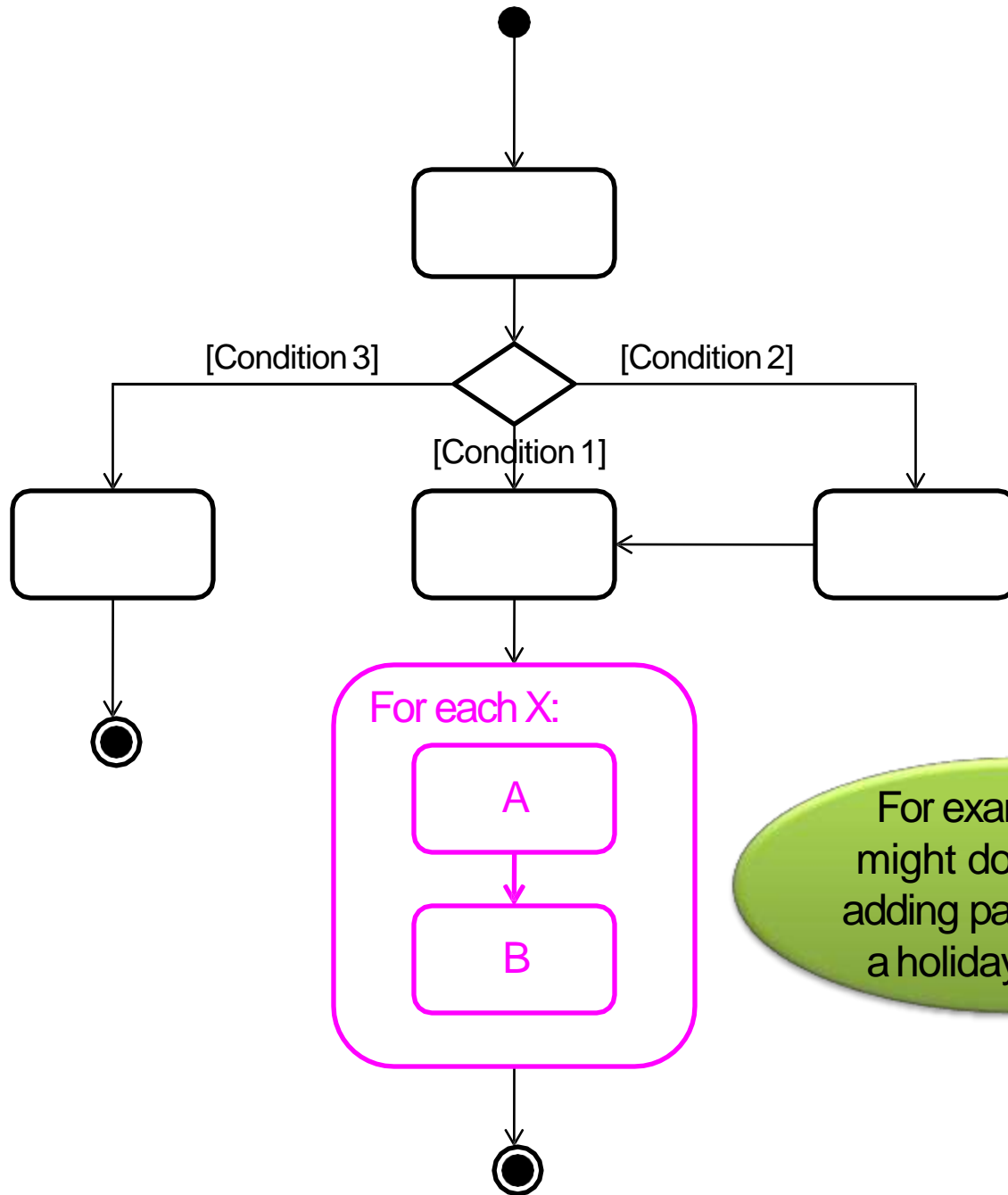


In some Use Cases, you will need to model repeated steps.





In this example, you repeat (A + B) until there is no more X.

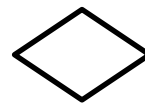


For example, you might do this when adding passengers to a holiday booking.

Let's review the shapes



START POINT



DECISION POINT



END POINT

[Condition]

GUARD



STEP



PARALLEL  
STEPS



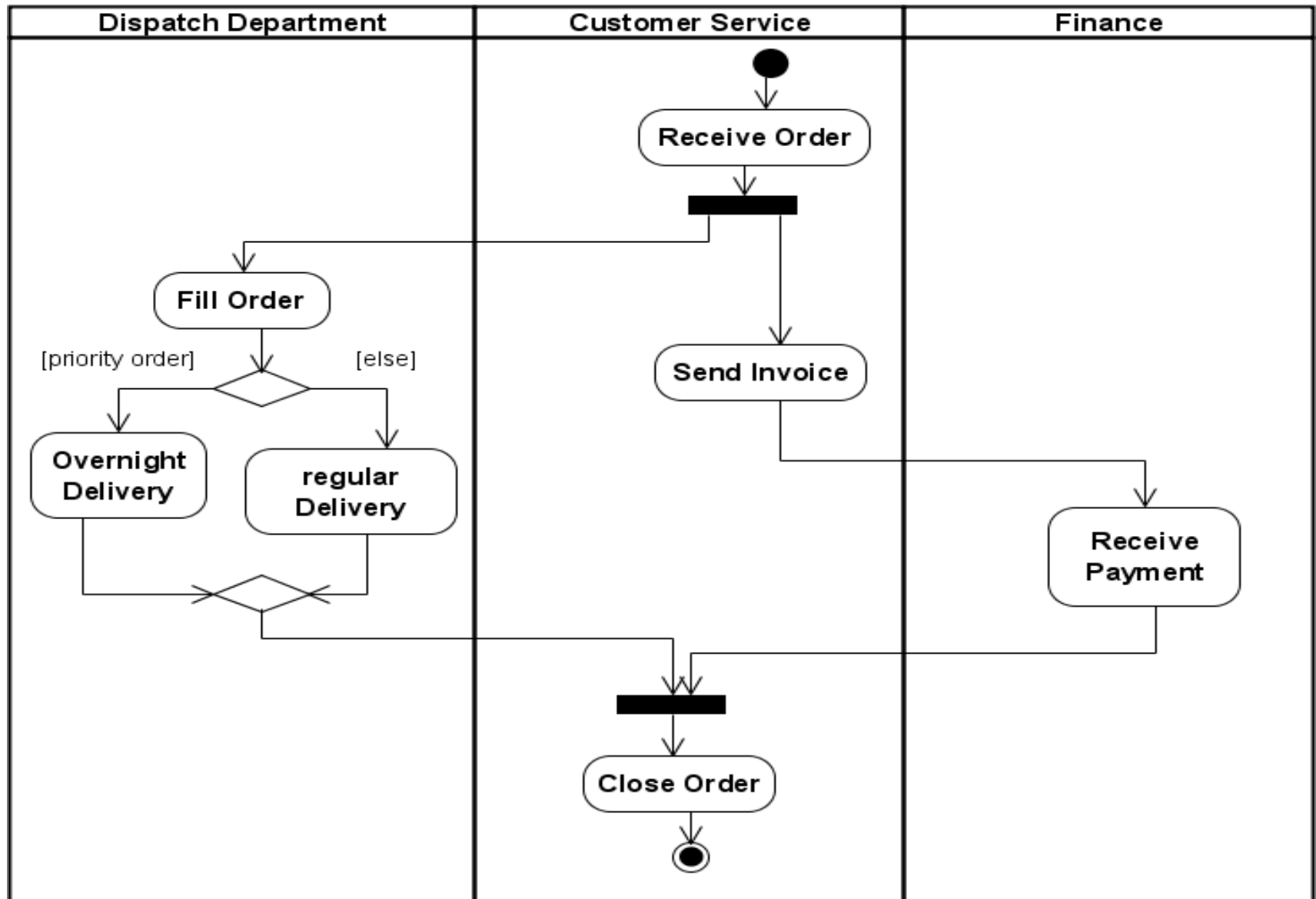
TRANSITION

For each X:

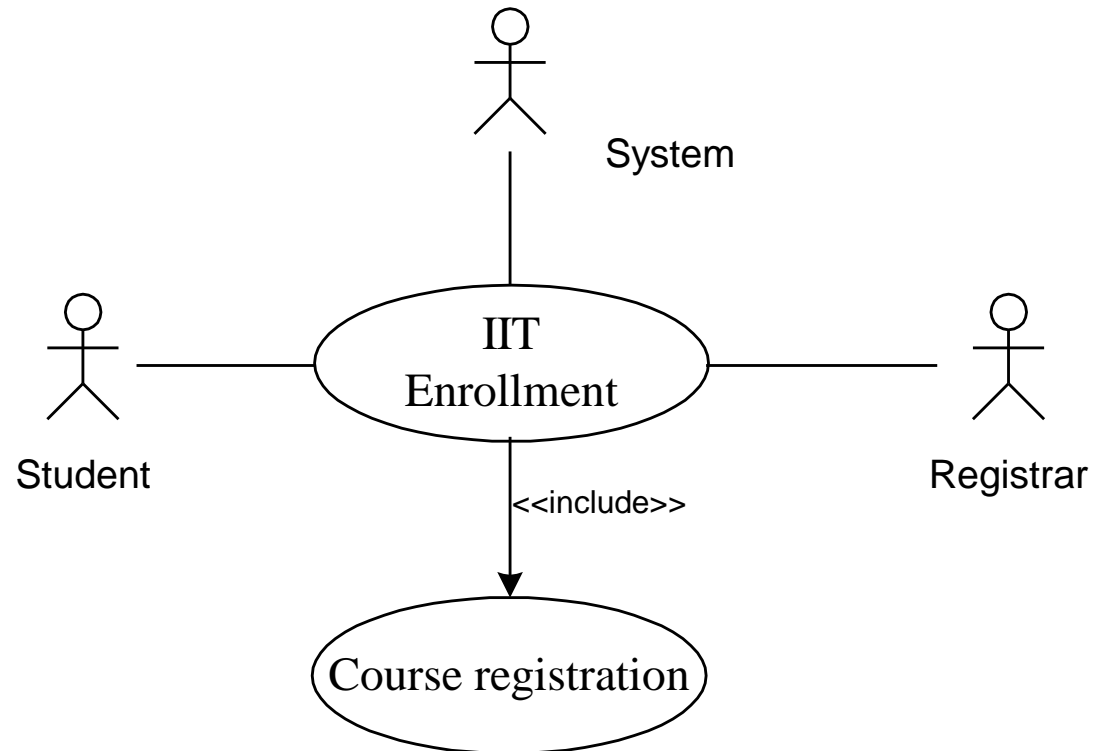
REPEATED STEPS

# Partitions / Swimlanes

- The contents of an activity diagram may be organized into *partitions* (**swimlanes**) using solid vertical lines.
- Indicate where activities take place.
- Partitions may be constructed on the basis of:
  - The class and actor doing the activity
  - Partitioning by class and actor can help to identify new associations that have not been documented in the class model
  - the use case the activity belongs to
  - Partitioning by use cases can help document how use cases interact



# Student Enrollment in IIT (SEIIT)

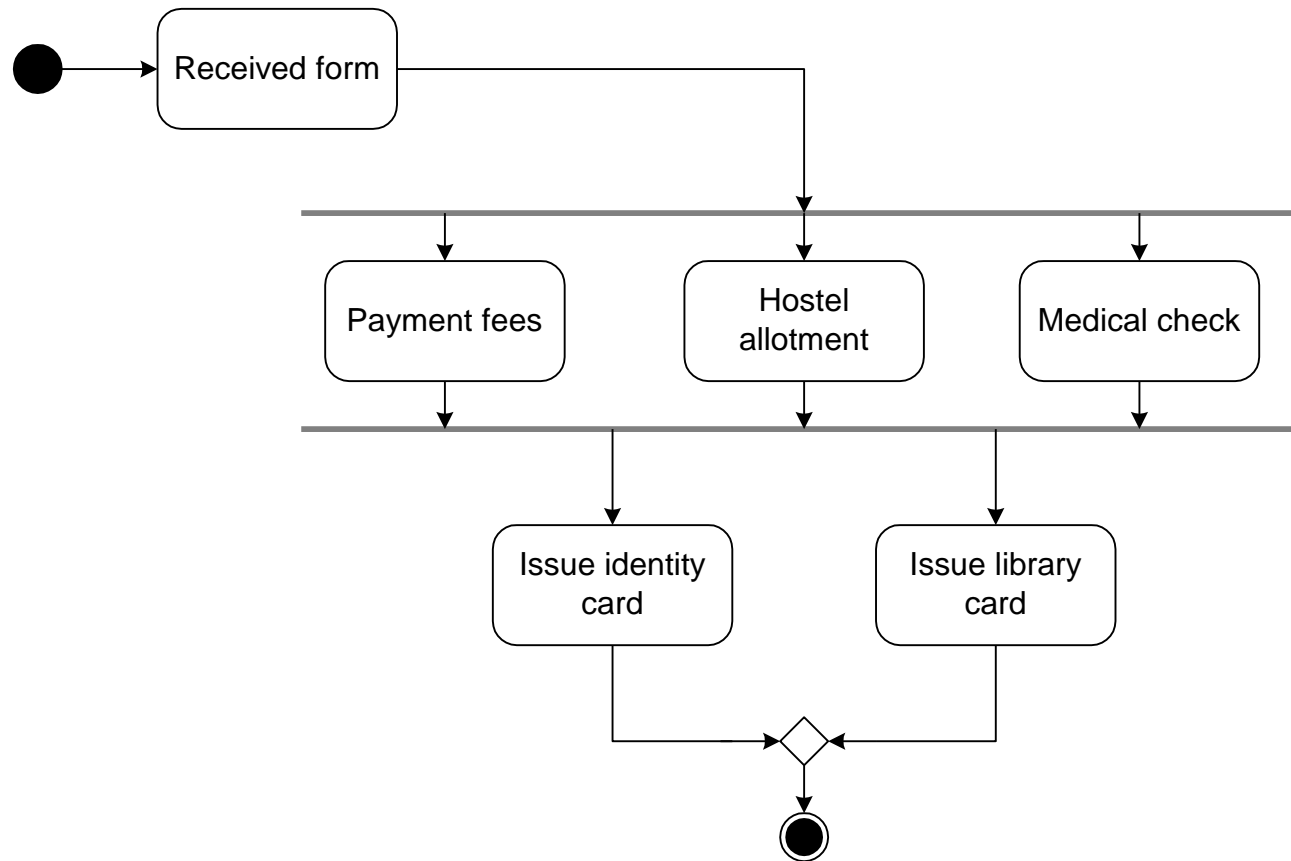


# SEIIT System

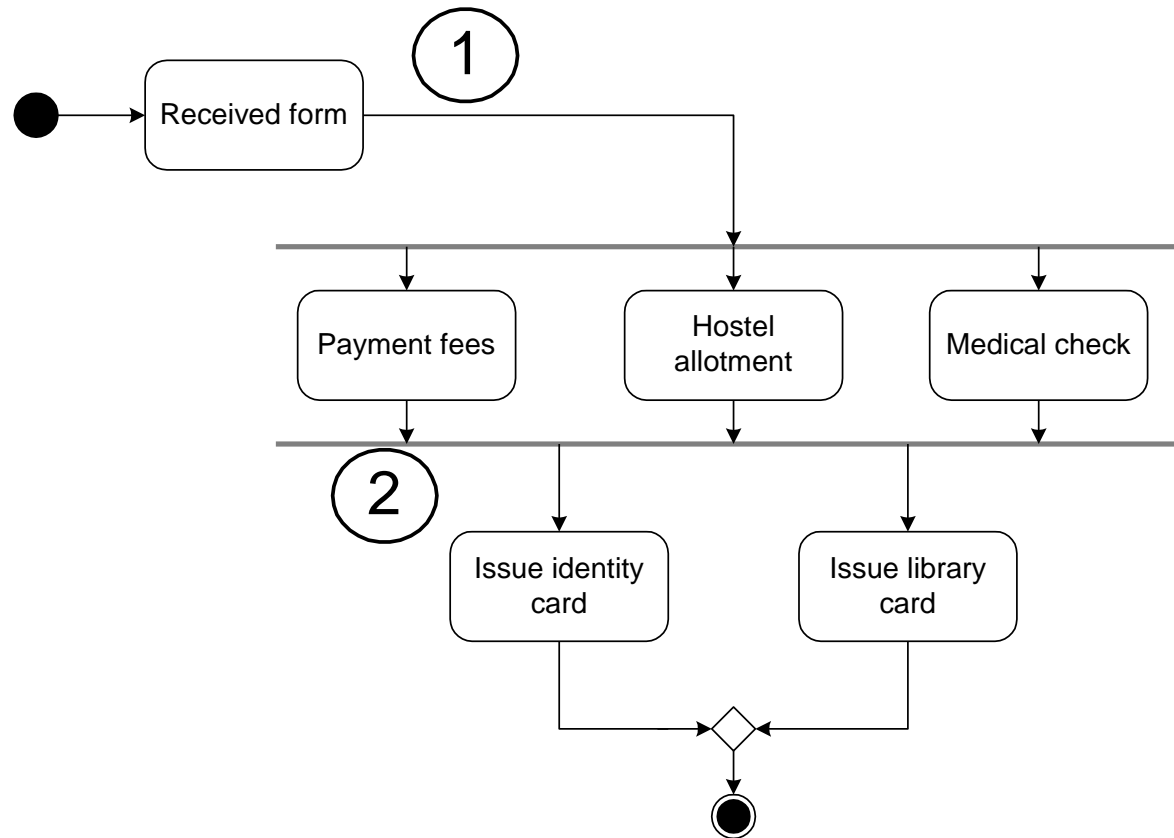
- Here different activities are:
  - Received enrollment form filled by the student
    - Registrar checks the form
    - Input data to the system
    - System authenticate the environment
  - Pay fees by the student
    - Registrar checks the amount to be remitted and prepare a bill
    - System acknowledge fee receipts and print receipt
  - Hostel allotment
    - Allot hostel
    - Receive hostel charge
    - Allot room
  - Medical check up
    - Create hostel record
    - Conduct medical bill
    - Enter record
  - Issue library card
  - Issue identity card



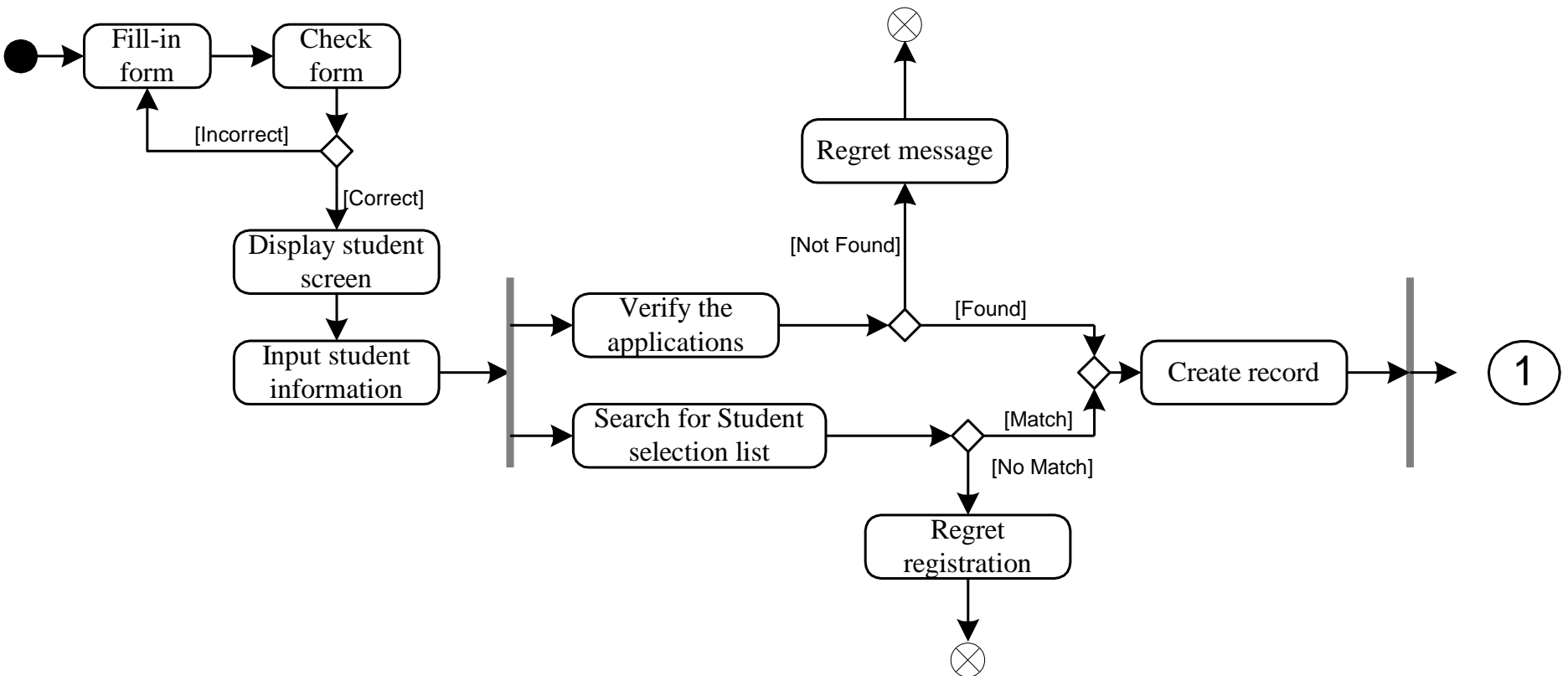
# Activity Diagram for the Use Case in SEIIT



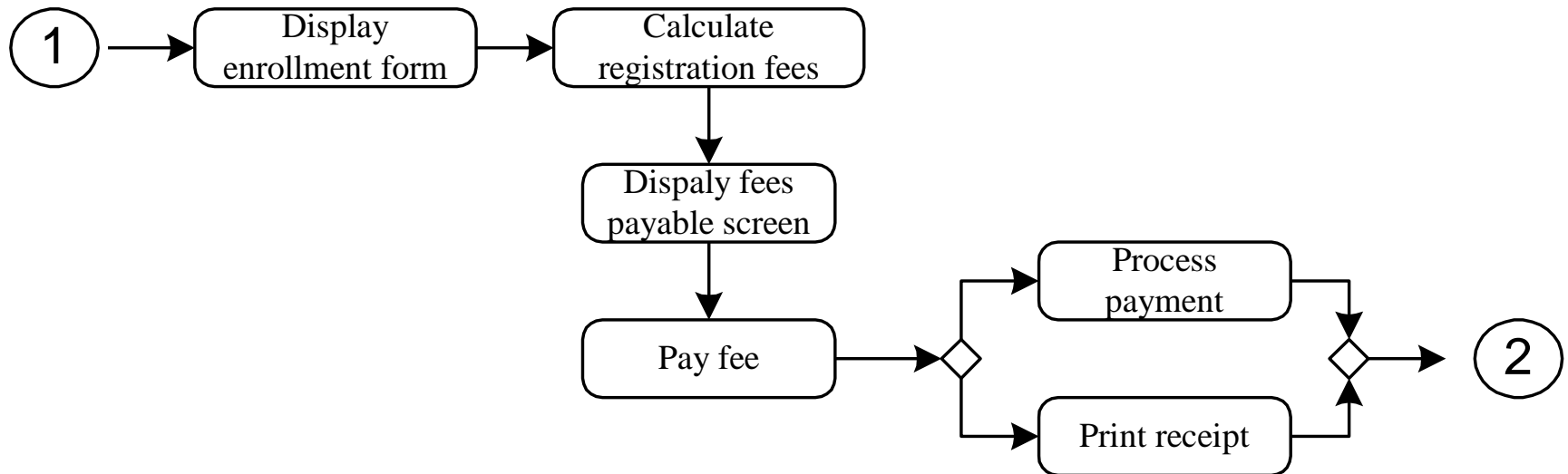
# Detailed Activity Diagram of SEIIT



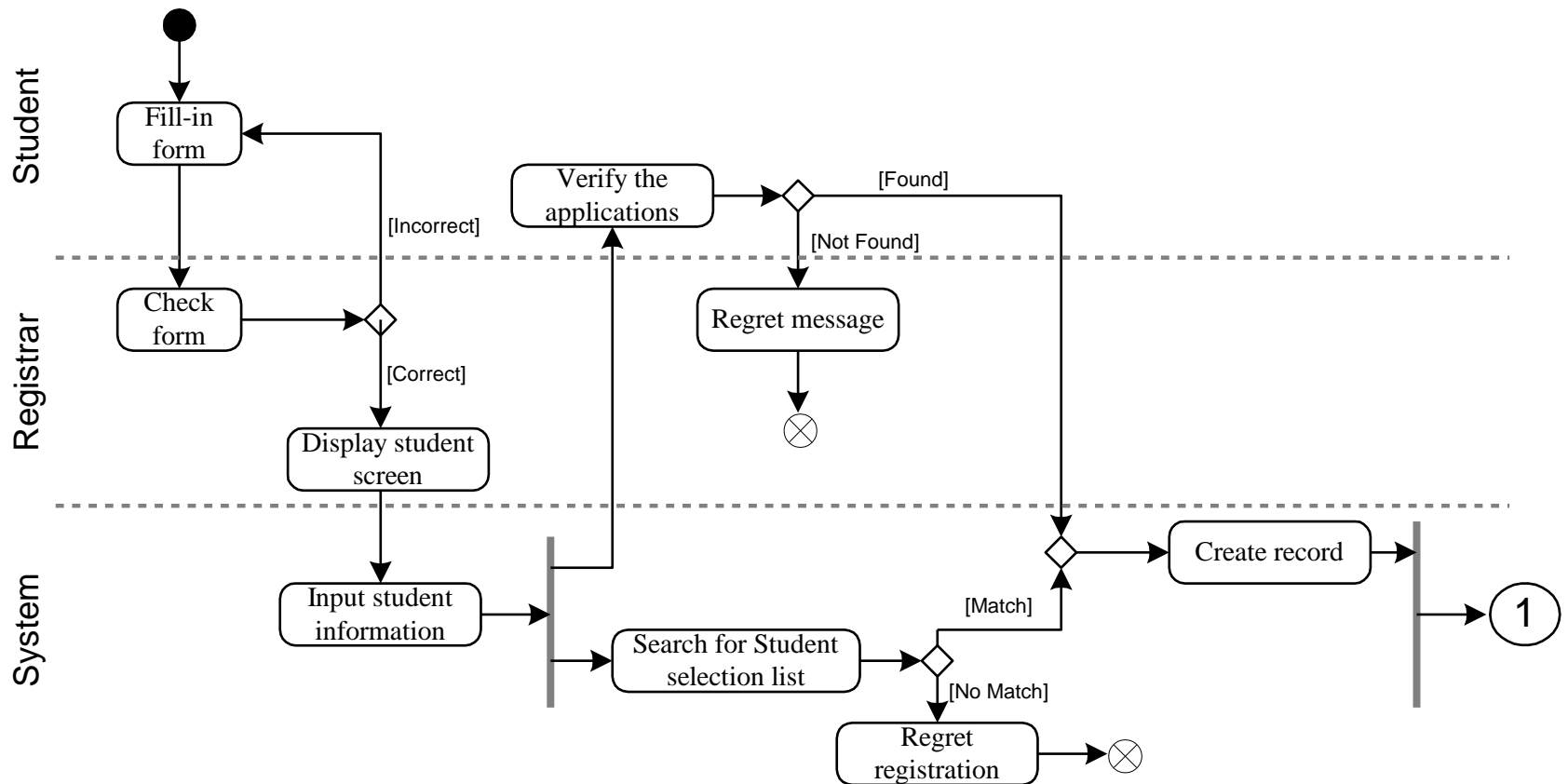
# Detailed Activity Diagram of SEIIT



# Detailed Activity Diagram of SEIIT



# Activity Diagram of SEIIT with Swim Lane



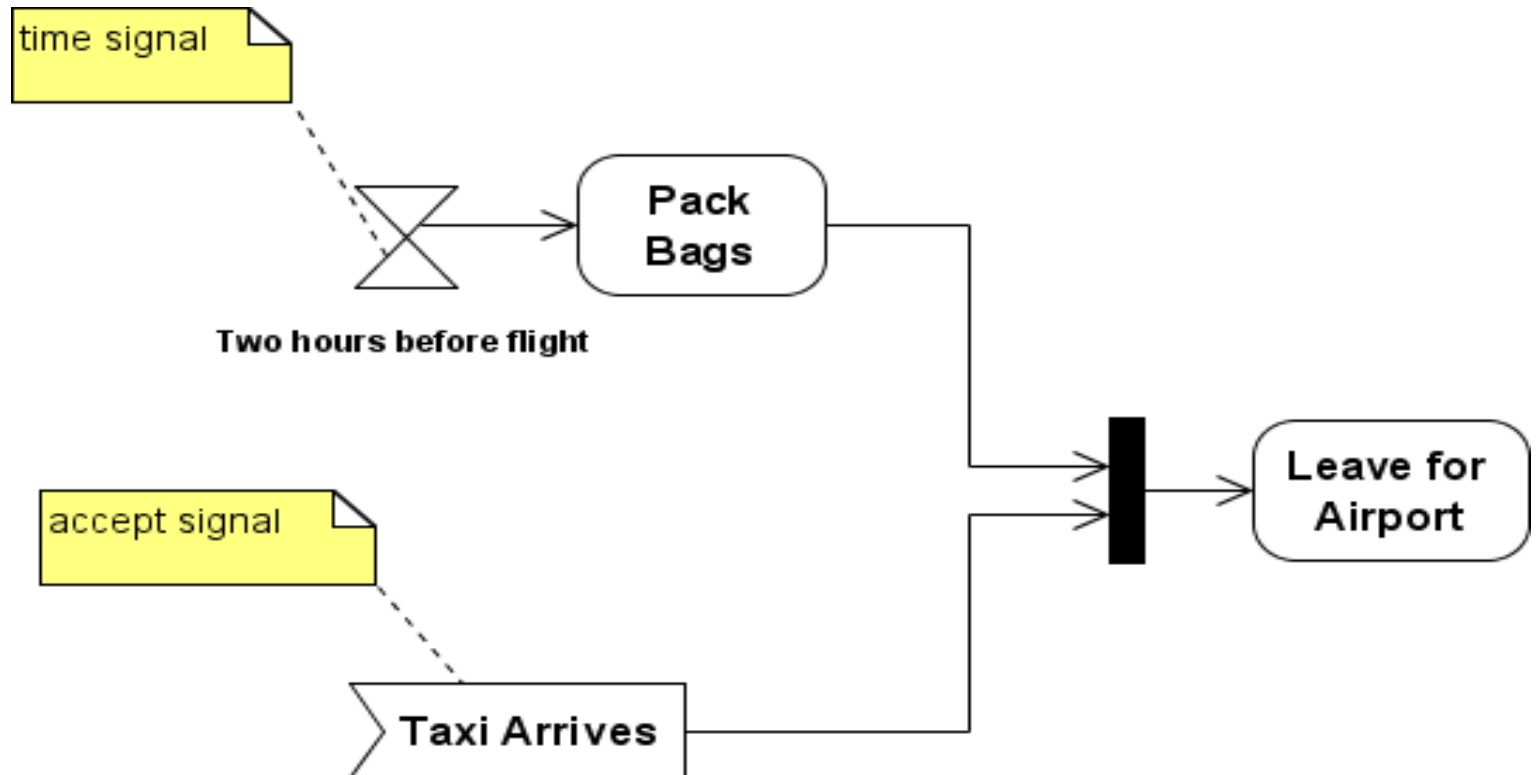
# Sending and Receiving Signals

- In activity diagrams, signals represent interactions with external participants
- Signals are messages that can be sent or received
- A receive signal has the effect of waking up an action in your activity diagram
- Send signals are signals sent to external participants

# Sending and Receiving Signals

- Note that combining send and receive signals results in behaviour similar to synchronous call, or a call that waits for a response.
- It is common to combine send and receive signals in activity diagrams, because you often need a response to the signal you sent.


# Signals on Activity Diagrams



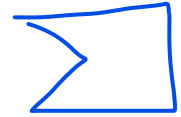


**Notify Customer** send signal action creates and sends  
Notify Customer signal

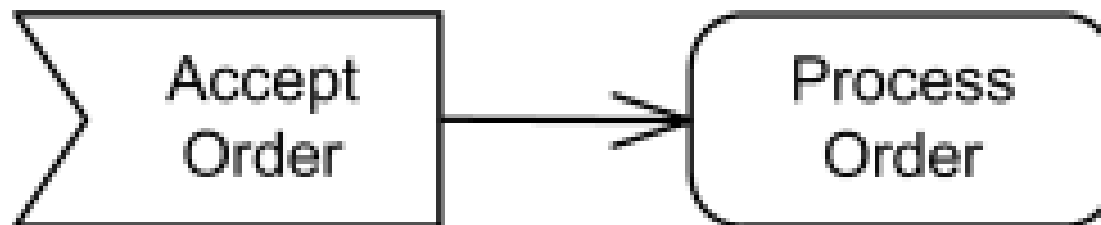


- Send signal action is an invocation action that creates a signal from its inputs, and transmits it to the specified target object, ~~where it may cause the firing of a state machine transition or the execution of an activity.~~
- When all the prerequisites of the action execution are satisfied, a signal is generated from the arguments and is transmitted to the identified target object.
- The sender of the signal (aka "requestor") continues execution immediately, without waiting for any response.
- Send signal action is notated as convex pentagon. 
- Note, that the name of the action corresponds to the name of signal class it sends.
- Target object is not specified with this notation.

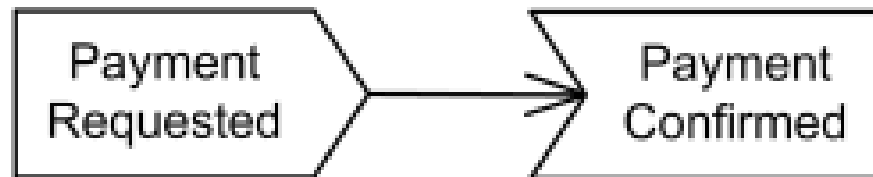
- **Accept event action** is notated with a concave pentagon.

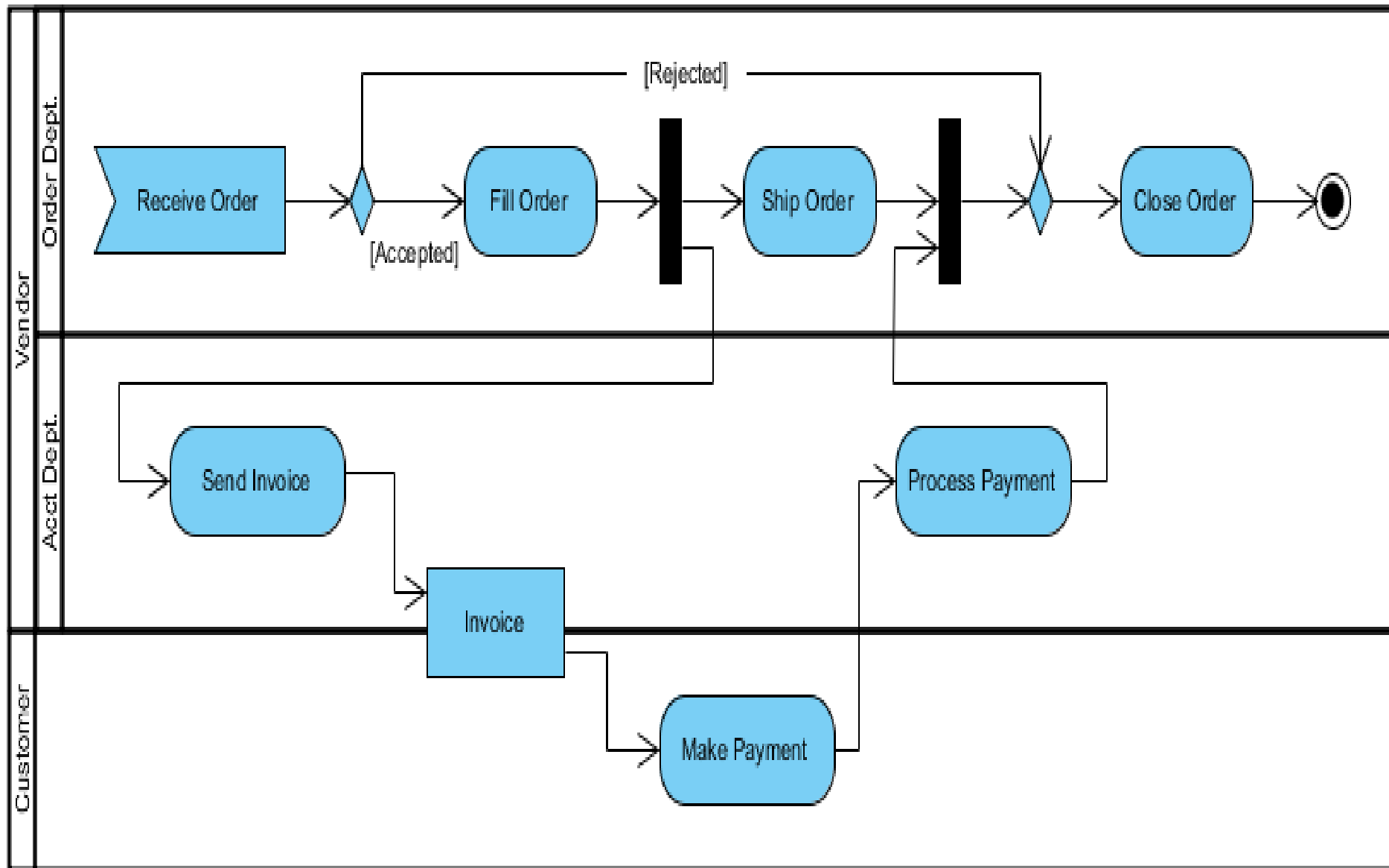


- If an accept event action has **no incoming edges**, then the action starts when the containing activity or structured node does, whichever most immediately contains the action.
- In addition, an accept event action with no incoming edges remains enabled after it accepts an event.
- It does not terminate 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.
- Payment Requested signal is sent.
- The activity then waits to receive Payment Confirmed signal.
- Acceptance of the Payment Confirmed is enabled only after the request for payment is sent; no confirmation is accepted until then.





# Pins

---

- **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.
- Item is **input pin** to the Add to Shopping Cart action.



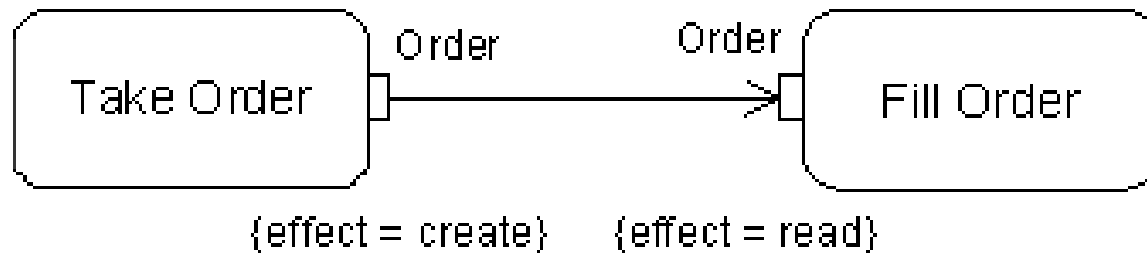
- Invoice is **output pin** from the Create Invoice action.



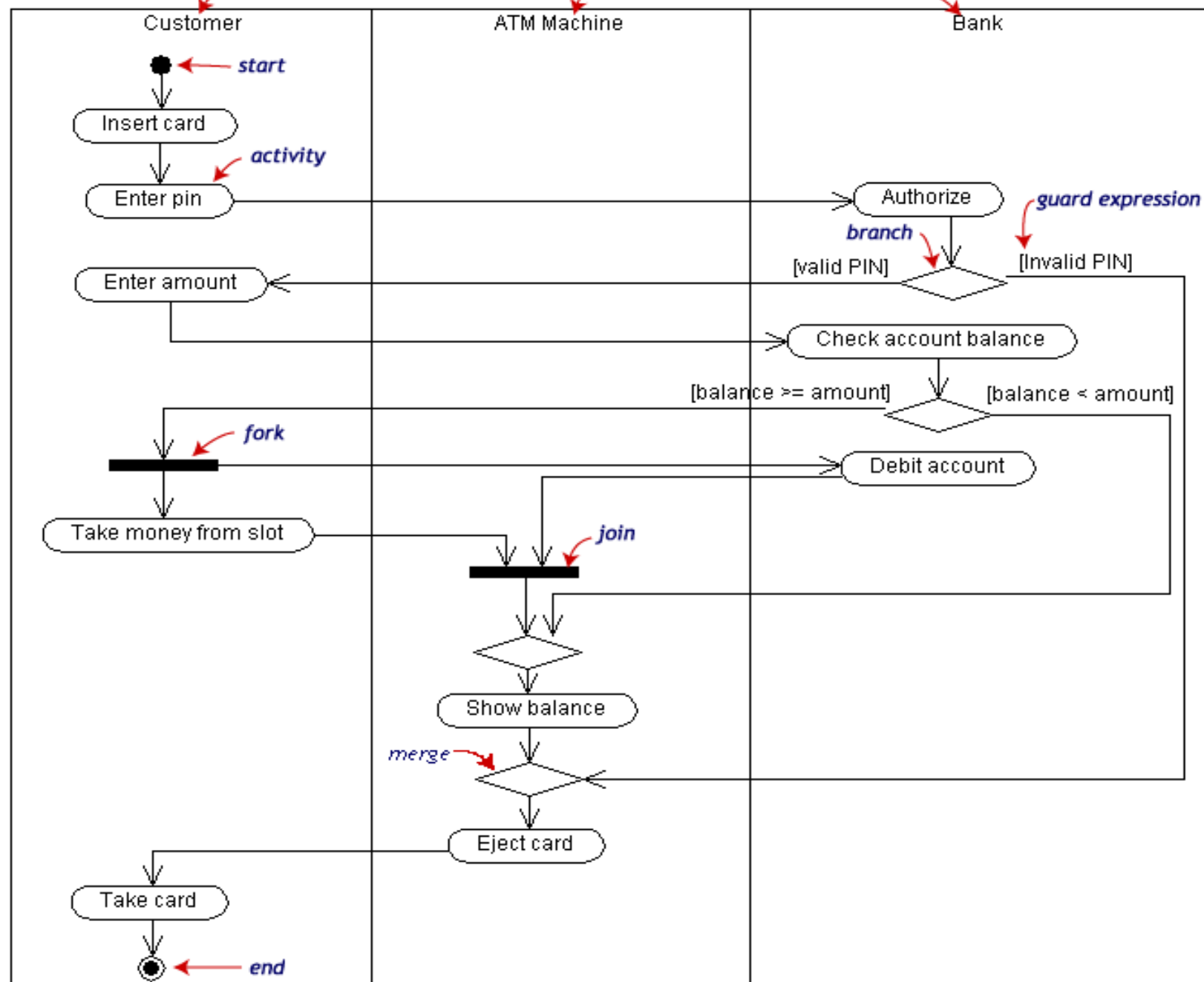
# Pins

---

- Pins can be notated with the effect that their actions have on objects that move through the pin.
- Effect is one of the four values create, read, update, or delete.
- The example indicates that Take Order creates an instance of Order and Fill Order reads it.
- The create effect is only possible on outputs, and the delete effect is only possible on inputs.

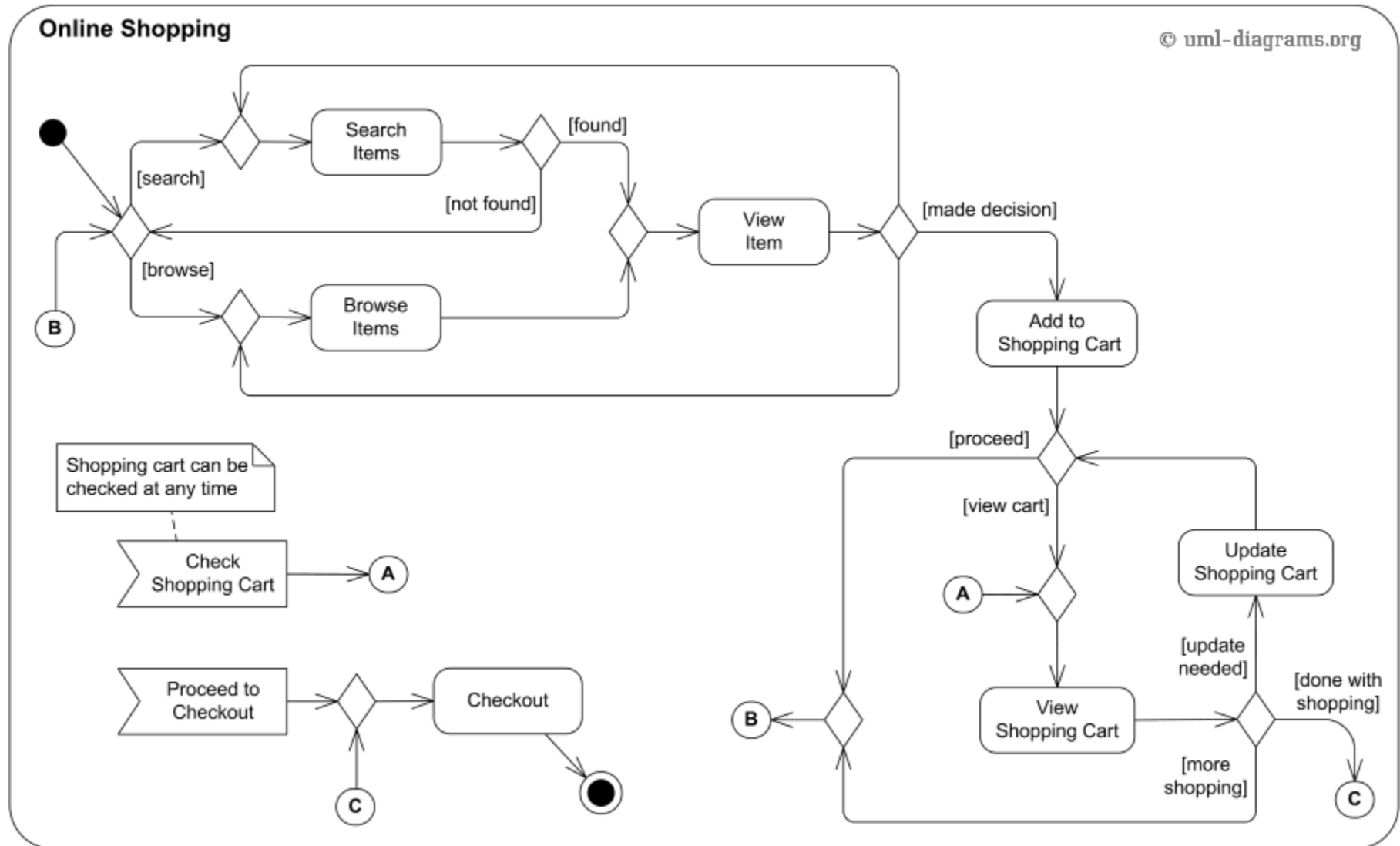


swimlane





# Activity diagram for online shopping



## **Problem statement (1)**

- Consider an inventory management system application for a computer spare parts store.
- The users of the system can add, remove or update spare parts information in the system.
- Spare parts information includes part no, part name, part type, part description and the company name. Any add, remove or update action require the users to be logged in.

## **Problem statement (2)**

- The users can search the inventory management system based on any of the information fields defined above.
- The search can be a simple keyword search or an advanced search with exact- phrase matching search, case sensitive search, etc.
- Manager is a special type of user who can update user information and can view the consolidated reports of available product inventory based on part name, type and company name.

## Problem statement (3)

- Administrator is a special type of user and can create other users.
- Administrators can also start and shut down the inventory management system.
- Draw an **Activity diagram** that depicts the relations between the various elements in this scenario.