UML – Interaction and Activity

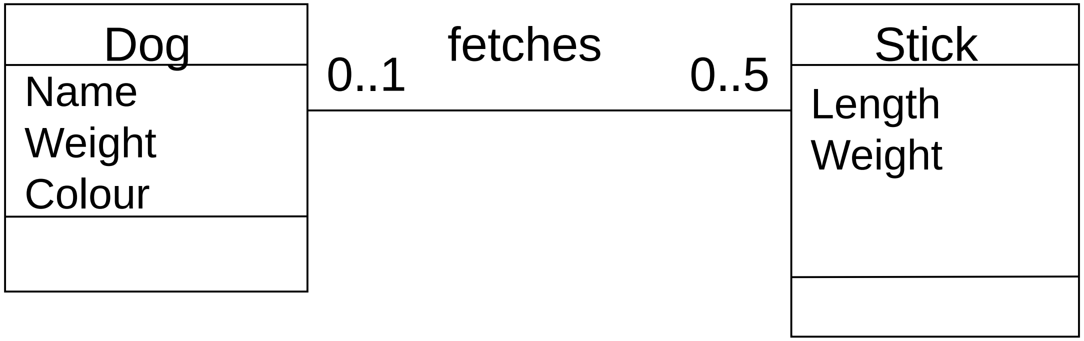
Lecture -UML

In this lecture we will look at UML diagrams to model behaviour of the system and complex logic.

- Interaction Diagrams: Sequence Diagrams
- Activity Diagrams

Motivating Example

- Consider the following example:



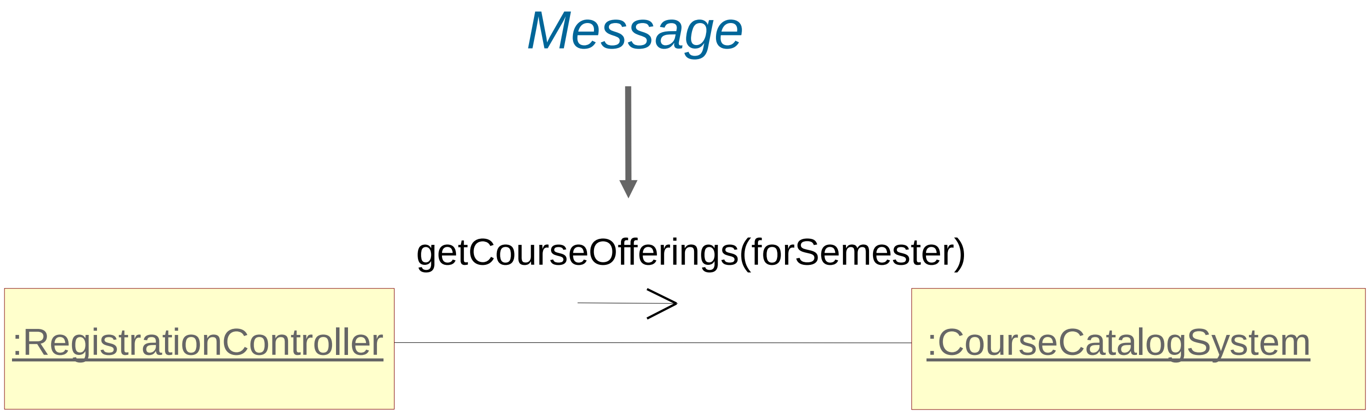
- This model tells us structurally what a dog is and what a stick is
- It also tells us the relationship between dog and stick
- But it does not tell us what happens when a dog fetches a stick!
  - How do we capture or model behaviour?

Objects Need to Collaborate

- Objects are useless unless they can collaborate to solve a problem.
  - Each object is responsible for its own behavior and status.
  - No one object can carry out every responsibility on its own.
- How do objects interact with each other?
  - They interact through messages.

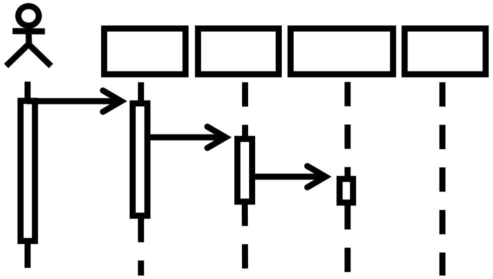
Objects Interact with Messages

- A message shows how one object asks another object to perform some activity.

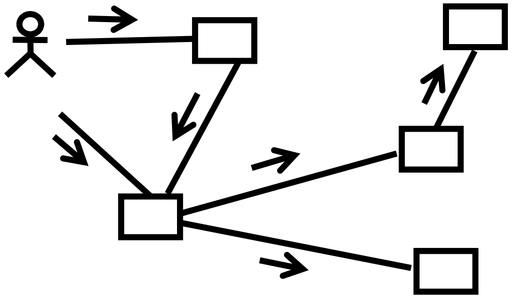


Interaction Diagrams

- Describe how groups of objects collaborate in some behaviour.
- Interaction diagrams show an interaction, consisting of a set of objects and their relationships, including message that may be dispatched among them.
- UML provides several forms of interaction diagram which include
  - Sequence Diagrams (time ordered)
  - Communication Diagrams (structurally ordered)



Sequence Diagrams

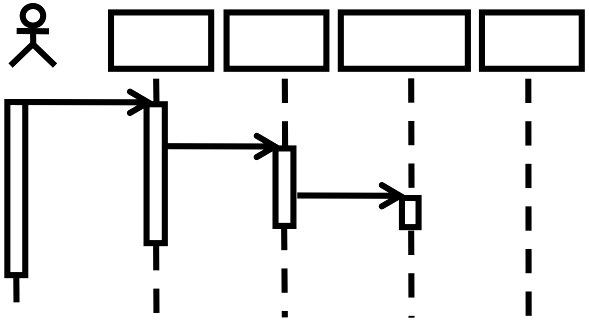


Communication Diagrams

Sequence Diagrams

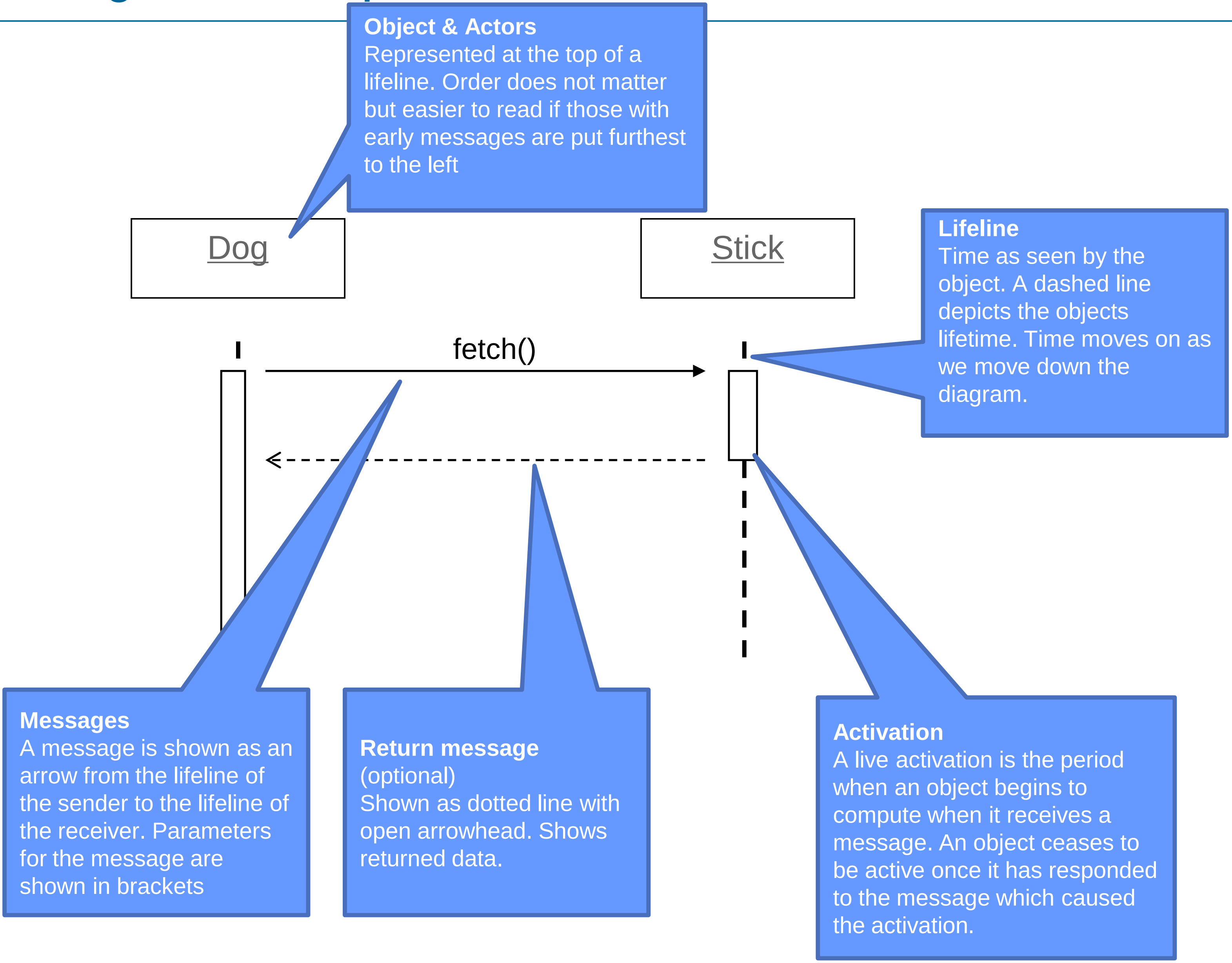
Sequence Diagrams

- A sequence diagram is an interaction diagram that emphasises the time ordering of messages.
- It typically captures the behaviour of a single scenario.
- The diagram shows a number of sample objects and the messages that are passed between these objects within the use case.
- It models objects-actors and objects-objects interactions



Sequence Diagram

Sequence Diagram Components



Object & Actors
Represented at the top of a lifeline. Order does not matter but easier to read if those with early messages are put furthest to the left

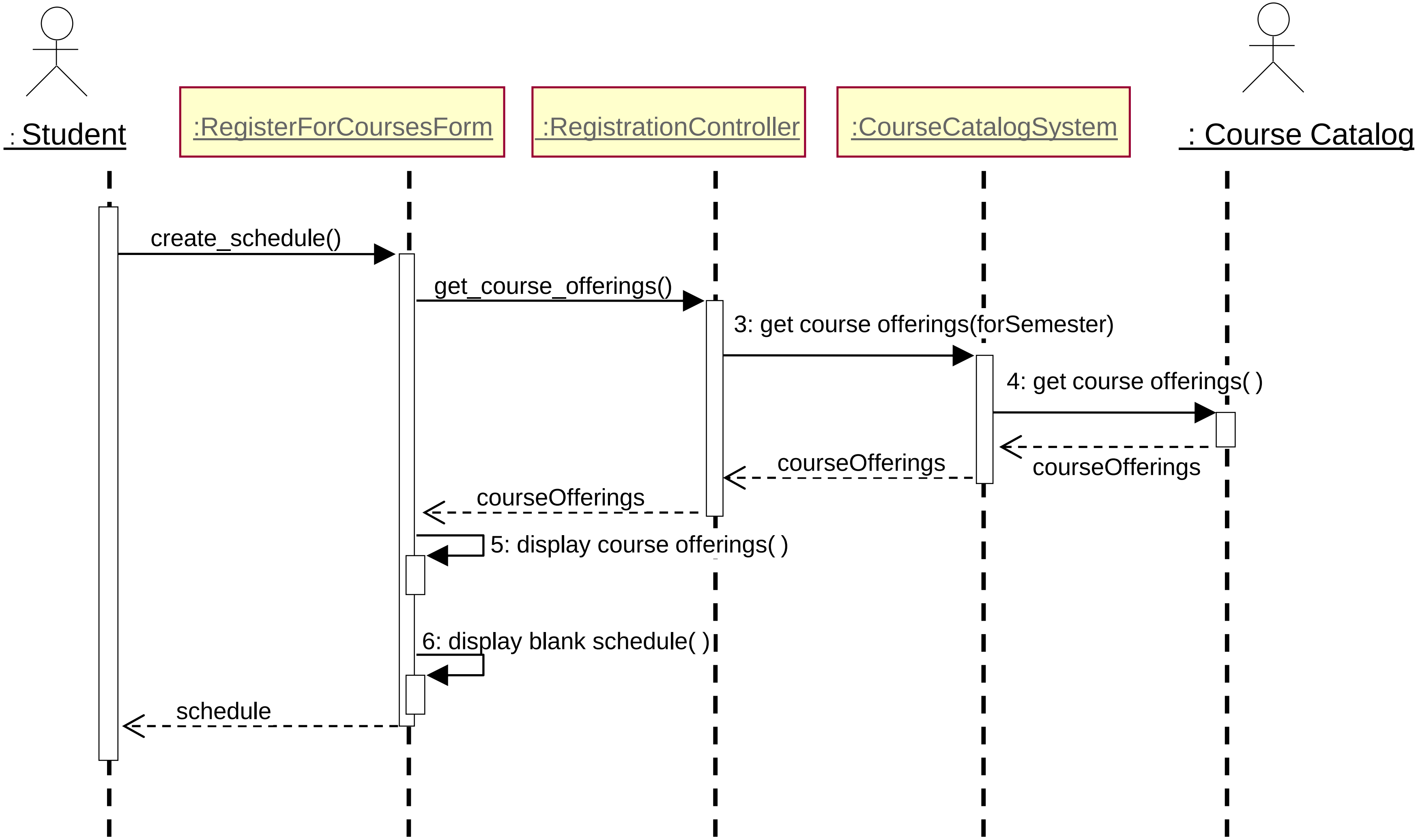
Lifeline
Time as seen by the object. A dashed line depicts the objects lifetime. Time moves on as we move down the diagram.

Messages
A message is shown as an arrow from the lifeline of the sender to the lifeline of the receiver. Parameters for the message are shown in brackets

Return message
(optional)
Shown as dotted line with open arrowhead. Shows returned data.

Activation
A live activation is the period when an object begins to compute when it receives a message. An object ceases to be active once it has responded to the message which caused the activation.

Example: Sequence Diagram

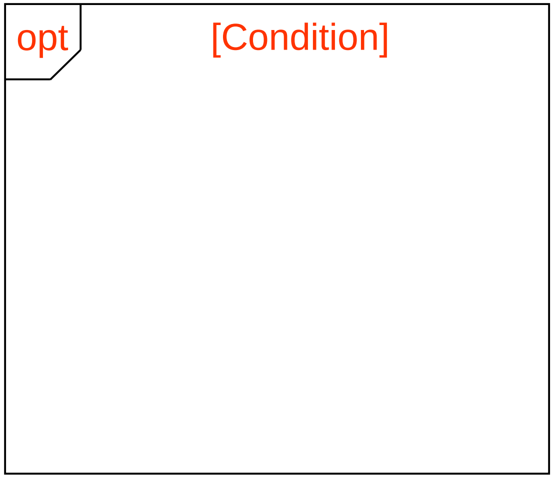


Control

- You may think of control as a token that is passed among objects.
- Only the object with the token may do something at that point in time.
- Other objects must wait until they get passed the token or receive the token back before they may continue with their activity.

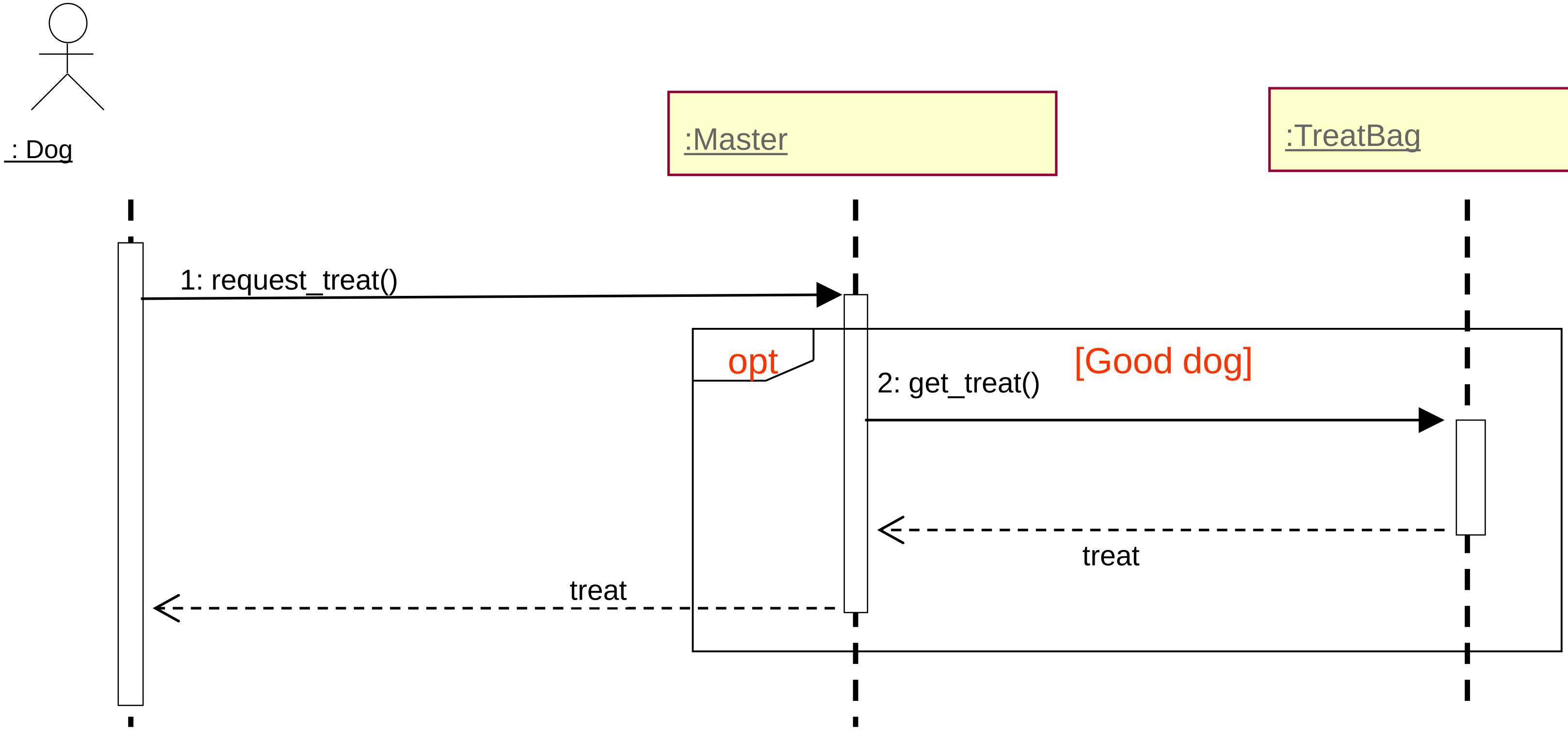
Conditional Behaviour

- A message or larger behaviour may be guarded by a condition.
- The behaviour is only executed if the guard evaluates to true.
- In sequence diagrams we enclose the conditional behaviour inside a rectangle labelled “opt” – short for optional.



Example

Model the following: A dog may request a treat from its master. The master may or may not give the treat, depending on whether the dog is a good dog.



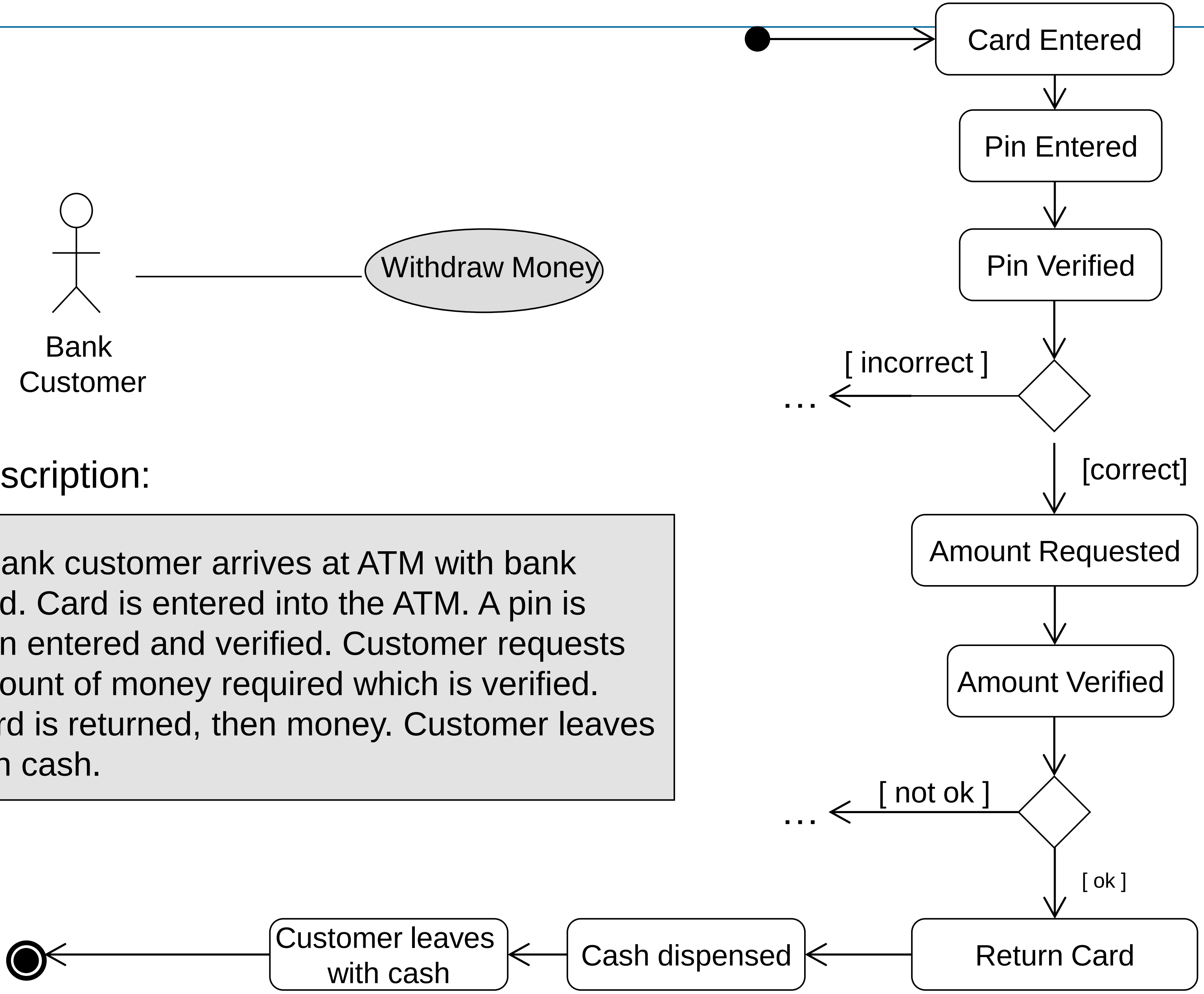
Activity Diagrams

Activity Diagrams

- Used to model a system function represented by a use case.
  - Activity diagrams allow us to describe the text more formally and in unambiguous detail.

- Can be used to describe procedural logic (including business processes and workflows) and how operations are carried out in the code.

Simple Example: compare to Sequence diagram

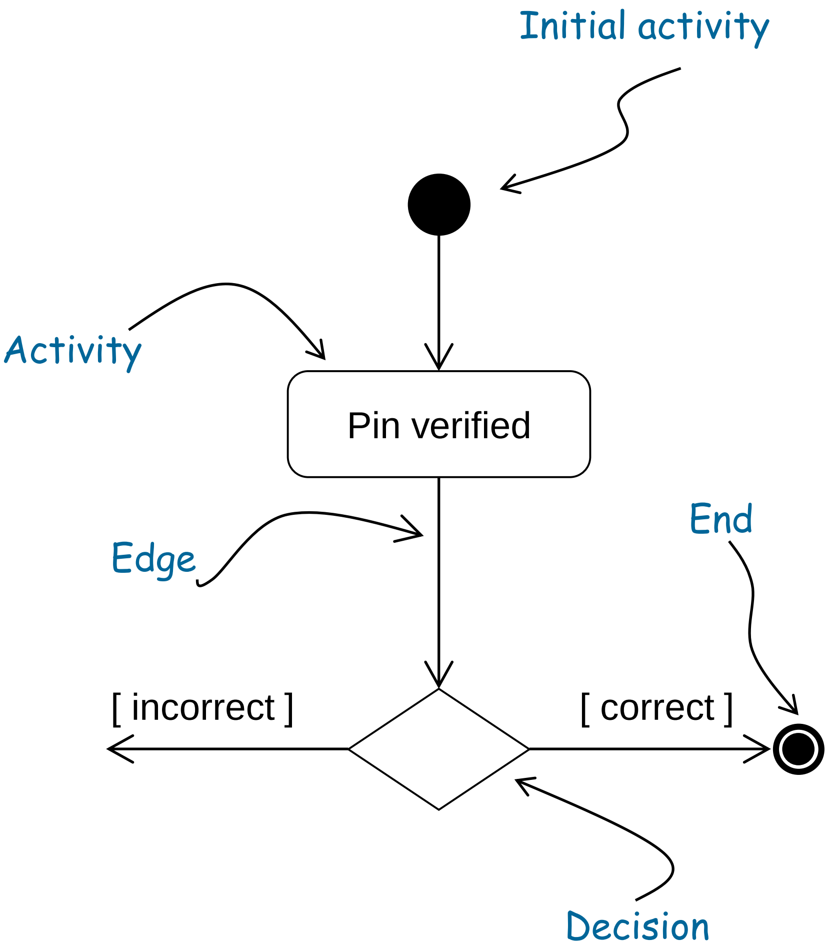


Description:

A bank customer arrives at ATM with bank card. Card is entered into the ATM. A pin is then entered and verified. Customer requests amount of money required which is verified. Card is returned, then money. Customer leaves with cash.

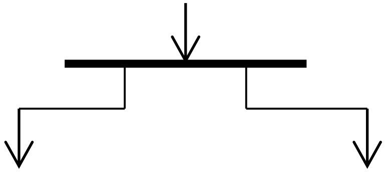
Syntax

- *Initial Activity*: This shows the starting point or first activity of the flow.
- *Activity*:
  - The task
  - Can be described in action language
- *Decisions*:
  - Symbol: Diamond
  - Options written on arrows emerging from diamond
  - Options in square brackets.
- *End:*
  - End of activity

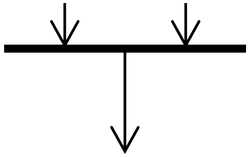


More Syntax

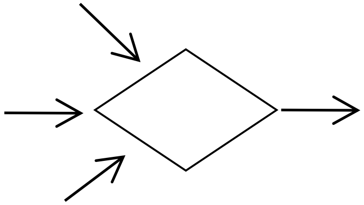
- Fork
  - Splits incoming flows into multiple concurrent flows



- Join
  - Synchronises multiple incoming flows

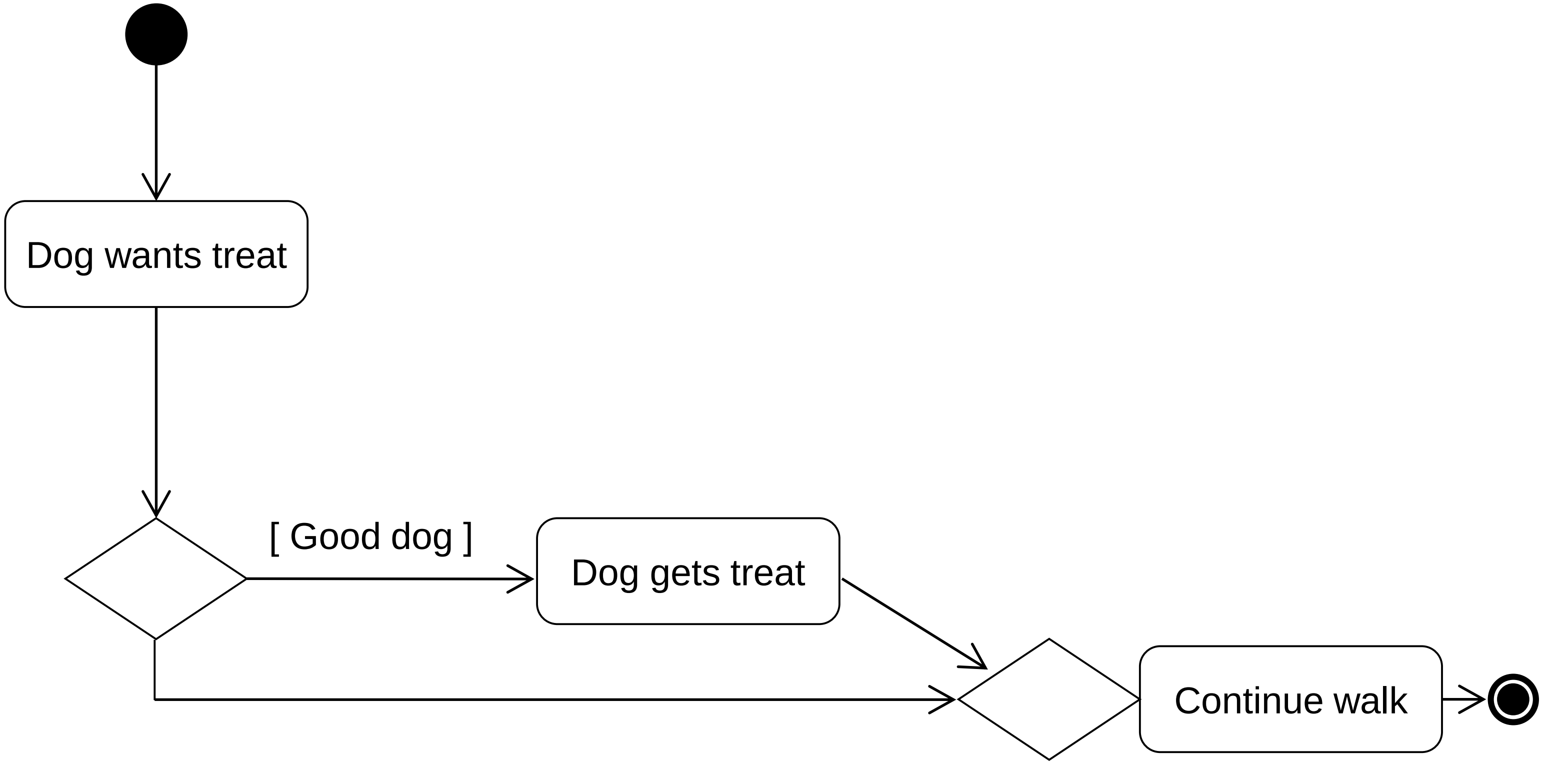


- Merge
  - Brings together different flows
  - Example use: Join after Decision would wait for both flows

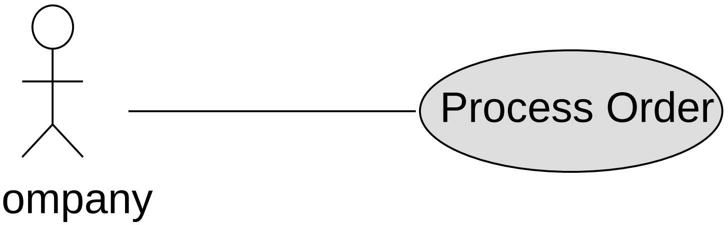


Example

Model the following: A dog and its master are on a walk. The dog may request a treat from its master. The master may or may not give the treat, depending on whether the dog is a good dog. The dog and master continue the walk in either case.

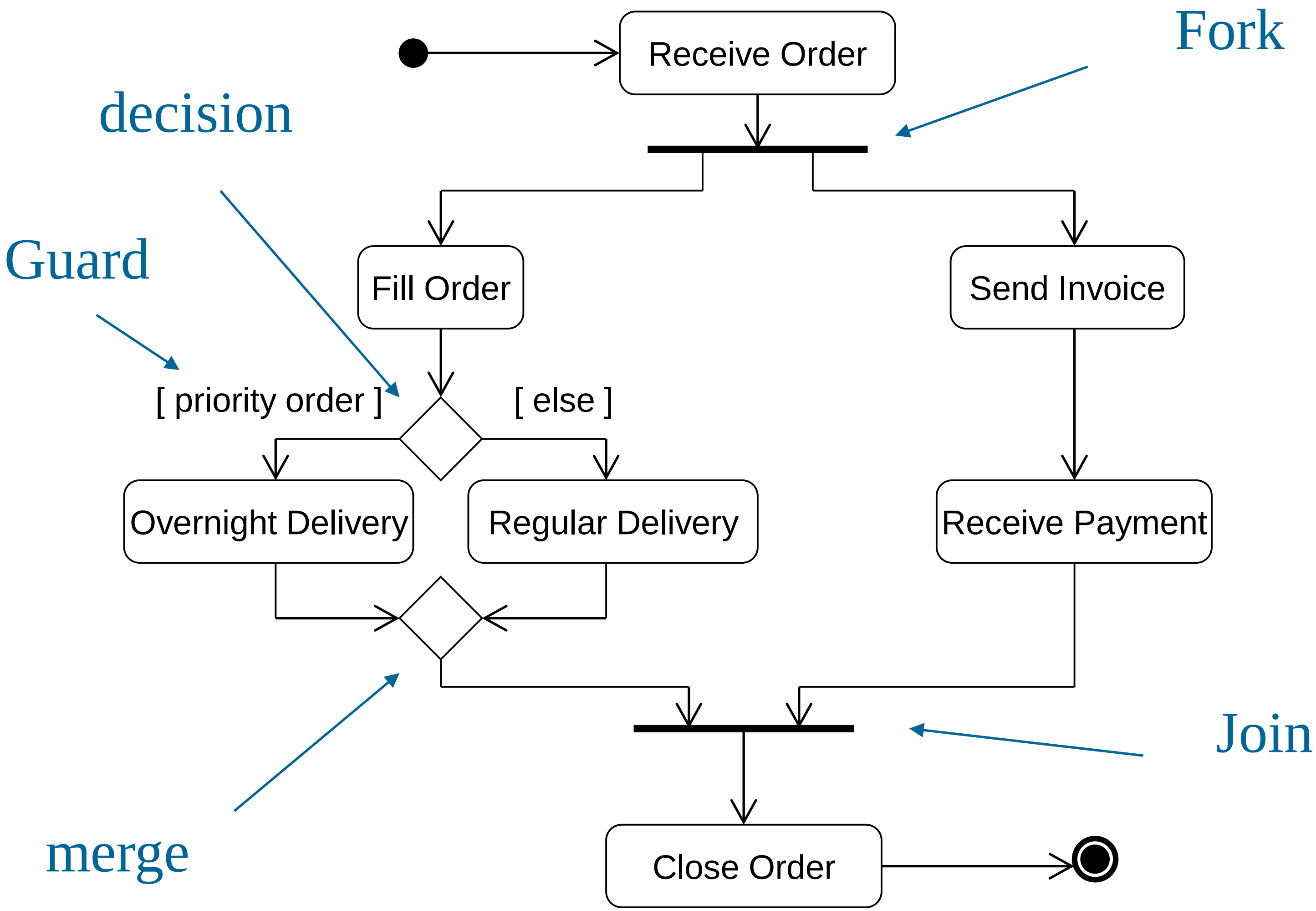


Order Processing System

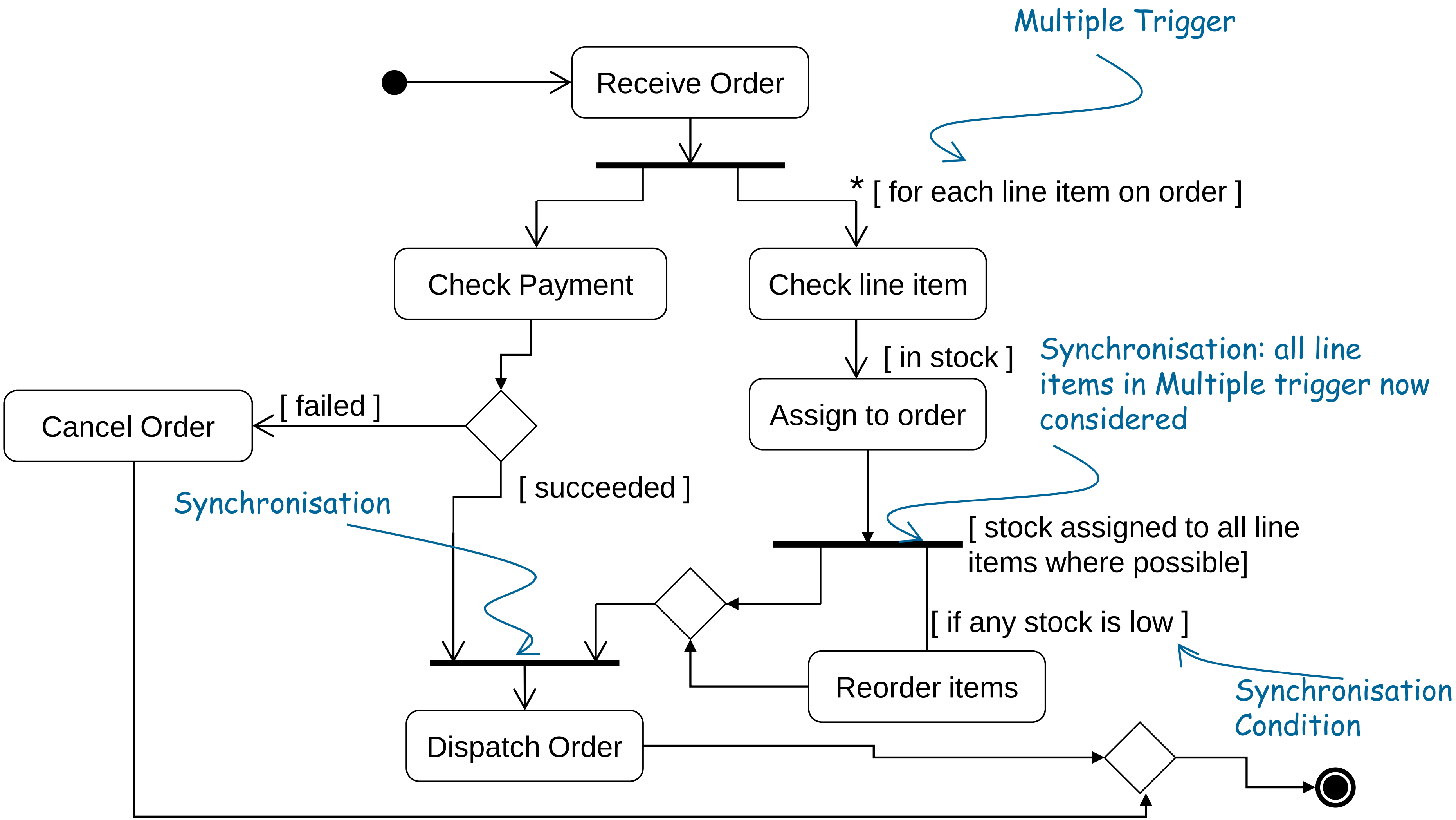


When an order is received, the system will fill the order with the goods. If it is a priority order, it is set for overnight delivery. Otherwise, the order is set for regular delivery. While this is being done, the system will send an invoice and await payment. If the payment is received, the order is filled and the delivery method assigned then the order is completed and may be closed.

Simple Example



A More Complex Activity Diagram



Conclusion

- Sequence diagrams: Temporal view on interactions
  - Skilful identification of message passing is very important in object oriented design.
  - Determining appropriate messages often occurs during the creation of interaction diagrams (communication and sequence diagrams).
  - The effort spent on interaction design should absorb a significant percentage of the design phase of a project.
- Activity diagrams: Model system function
  - Describe use-cases in detail

References and recommended reading

- Learning UML 2.0, O’Reilly Media.
- https://www.uml-diagrams.org/activity-diagrams.html (last accessed 6 February 2018)