Unified Modeling Language

Activity Diagram and Sequence Diagram

Mojtaba Shahin

Week #7: Lecture - Part #1

Topics covered

- **Part 1**
  - Activity Diagram
- Part 2
  - Sequence Diagram

Notes and Acknowledgements

- Slides/images come from the following main sources:
  - Arlow, J., Neustadt, I., UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd Ed. Addison-Wesley, 2005.
  - UML Distilled, Martin Fowler
  - Object-oriented Design course by Raman Ramsin, Sharif University of Technology, Iran, http://sharif.edu/~ramsin/index_files/undergradcourse_OOD.htm
  - Sebastian Rodriguez, Software Engineering Fundamentals for IT (2110), RMIT University, Course Materials on RMIT Canvas
  - Schaum's Outlines UML (2nd edition)
  - Melina Vidoni, Software Engineering Fundamentals (2050), RMIT University, Course Materials on RMIT Canvas
  - Halil Ali , Software Engineering Fundamentals (Semester 1, 2020), RMIT University, Course Materials on RMIT Canvas
  - Ian Sommerville, Software Engineering, 10th Edition, 2015.

Classification of UML 2.2 Diagrams*

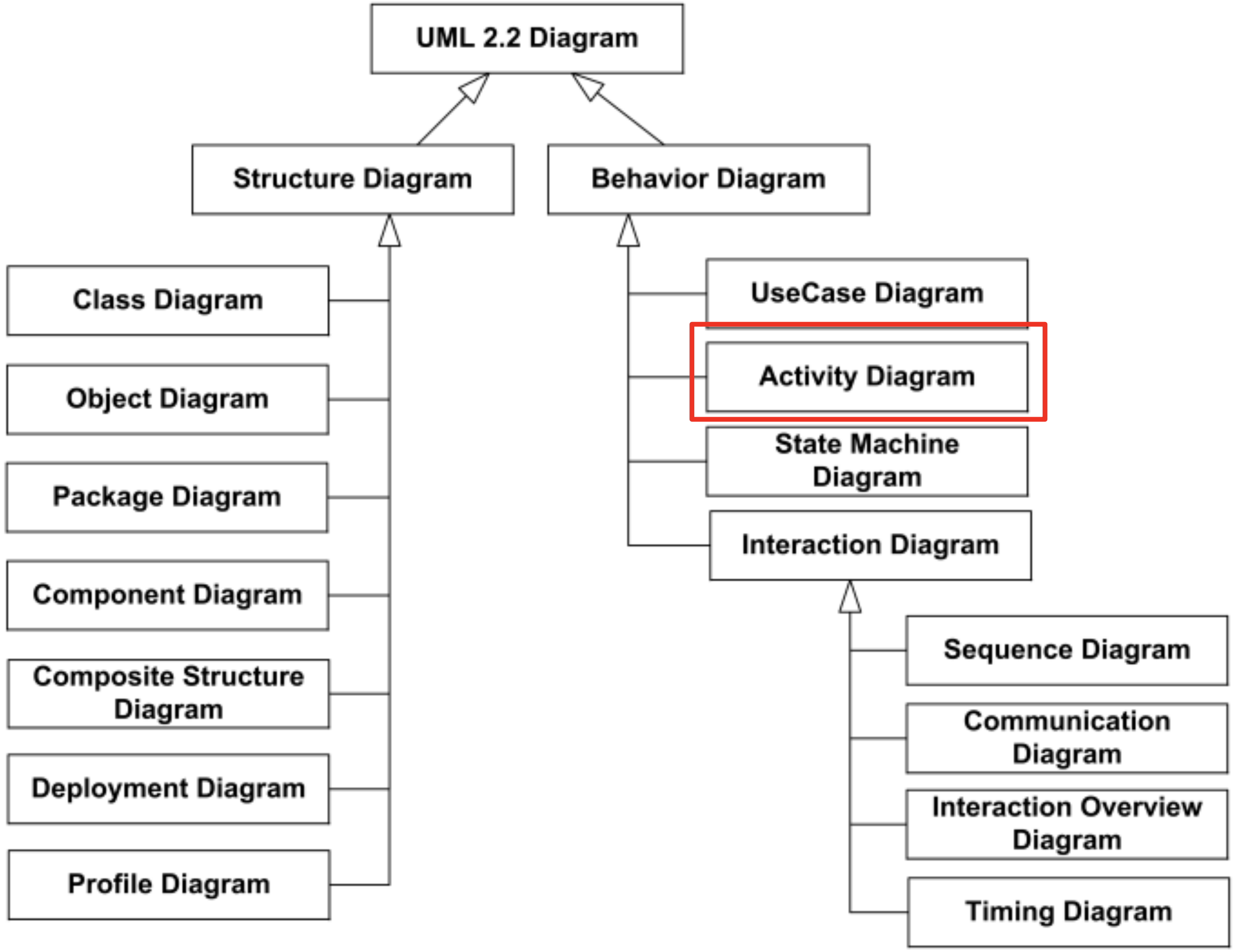
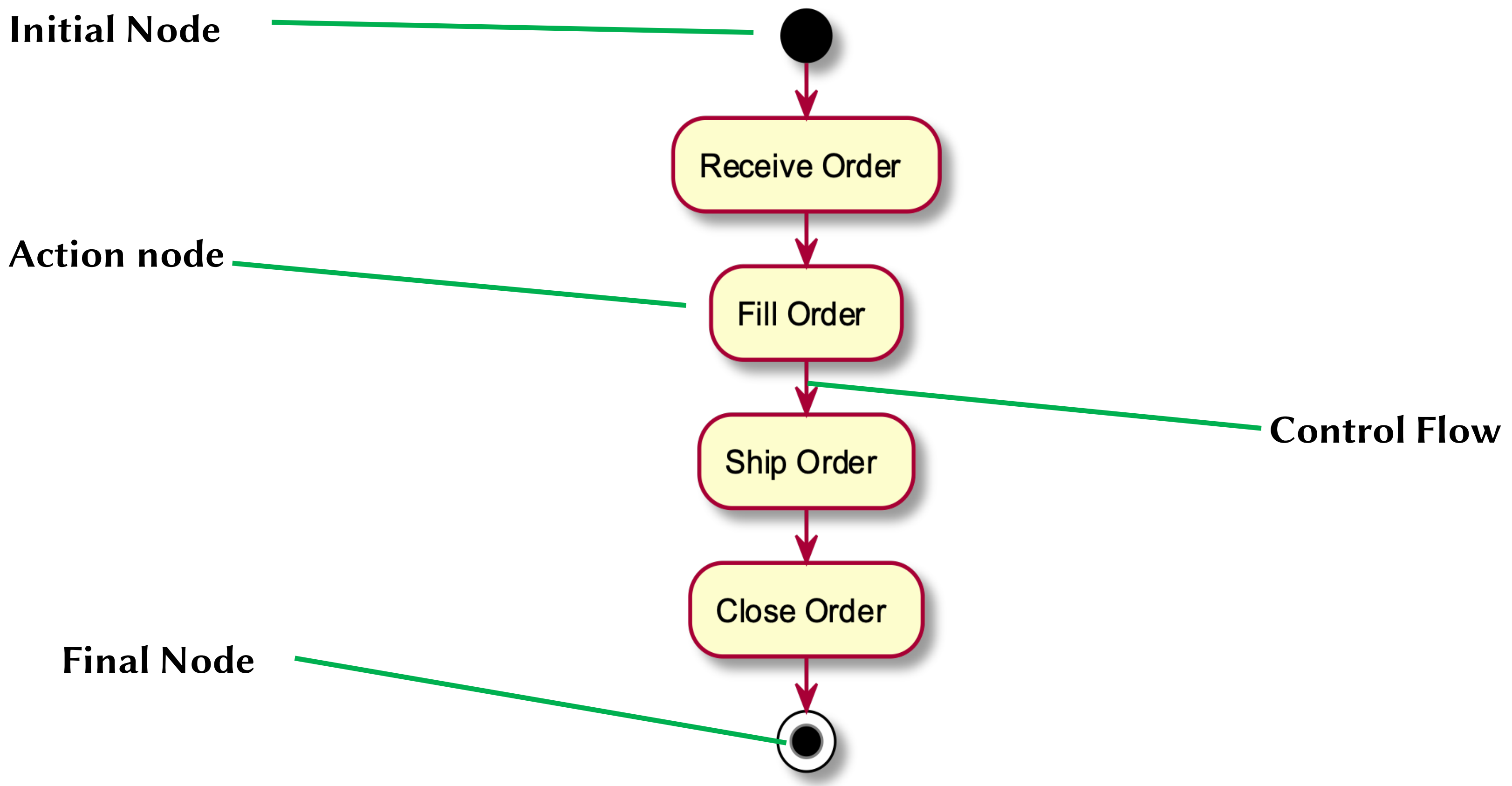


Image Source: https://www.uml-diagrams.org/uml-22-diagrams.html

Activity Diagrams

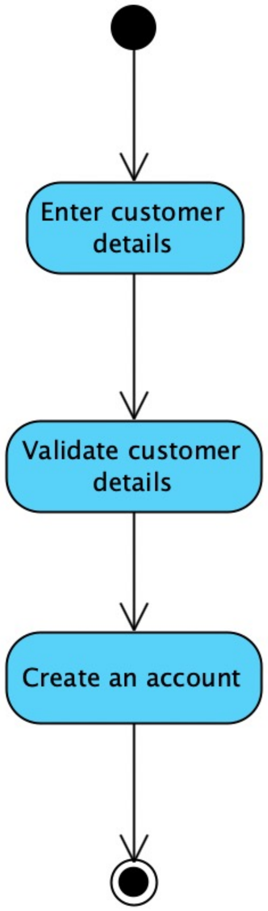
- Activity diagrams are object-oriented flowcharts to describe procedural logic, business process, and workflow.
- Activities are networks of **nodes** connected by **edges**.
  - **Categories of nodes:**
    - **action nodes** - atomic units of work within the activity;
    - **control nodes** - control the flow through the activity;
    - **object nodes** - represent objects used in the activity.
  - **Categories of edges**:
    - **control flows** - represent the flow of control though the activity;
    - **object flows** - represent the flow of objects through the activity.

Activities: Example



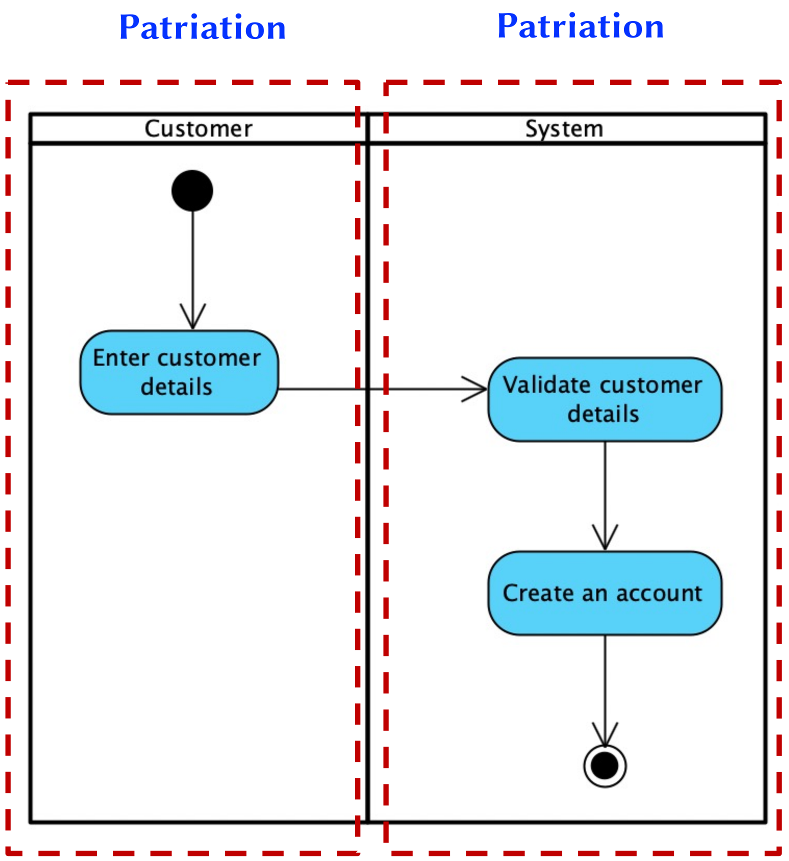
Activity Diagram and Use Case Modeling

Use Case: Create New Customer Account
ID: 5
Brief description This system creates a new account for the Customer.
Primary actors Customer
Secondary actors None
Preconditions None
Main flow 1. The use case starts when the Customer selects "**New Customer Account**" 2. The systems asks the Customer to enter their details comprising email address, password, and password again for confirmation. 3. The system validates the Customer details. 4. The System creates a new account for the Customer.
Postconditions A new account has been created for the Customer.
Alternative flows None

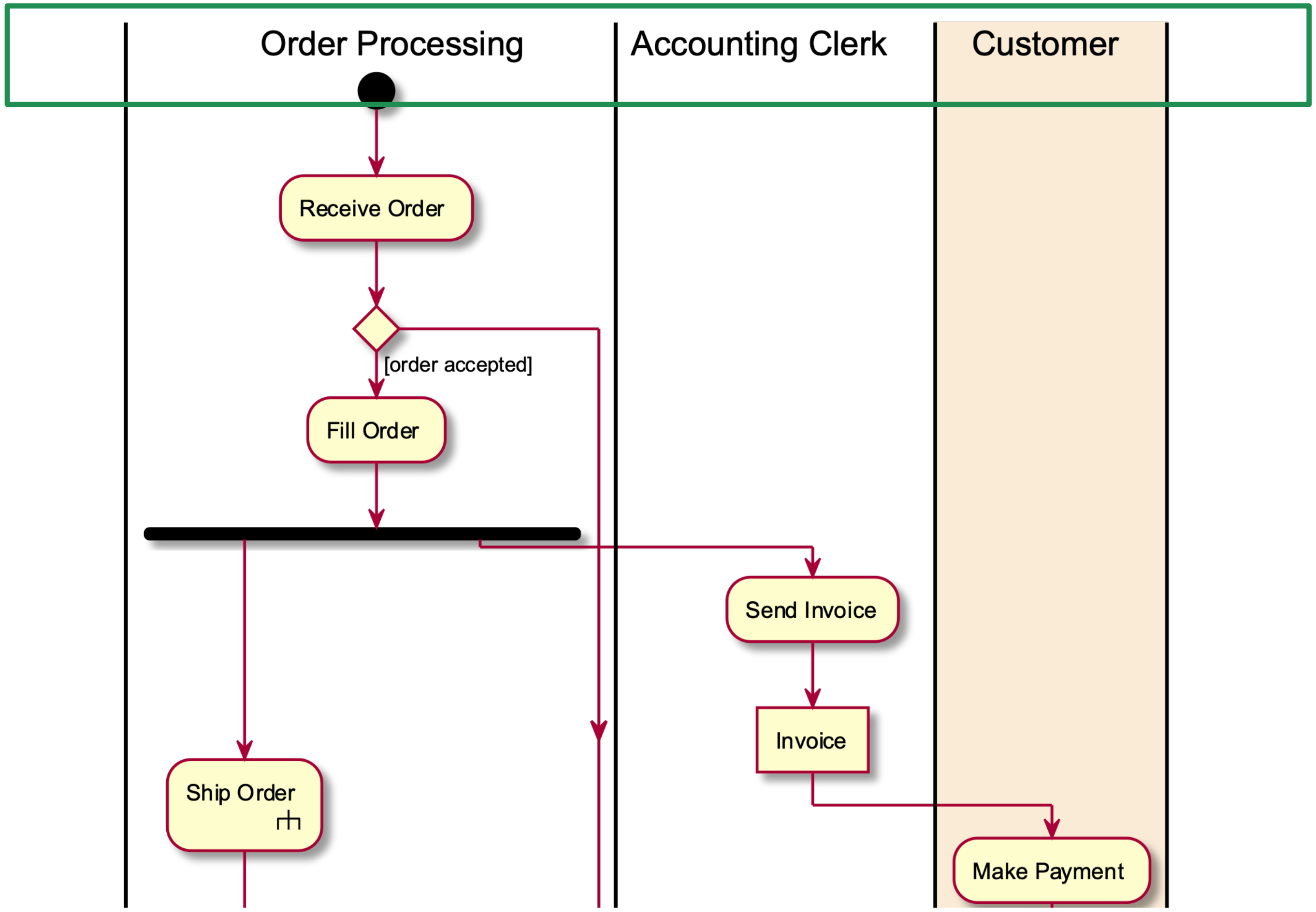


Activity Partitions/SwimLanes

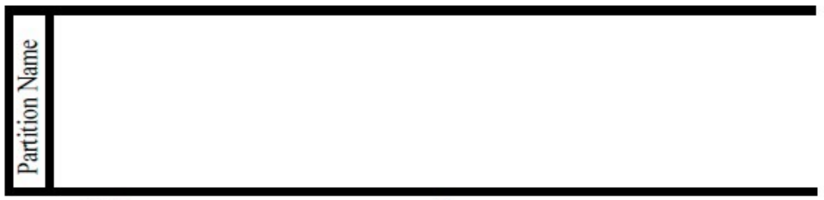
- The activity diagrams so far show the actions within a workflow, but not **who does them.**
- In **programming**, this means that the diagram does not convey which class is responsible for each action.
- In **business process modeling**, this does not convey which part of an organization carries out which action.
- We can show who is responsible for the actions by using "swimlanes" ("partitions") - all the actions within one are done by one entity/role
- Swimlanes are any grouping you think makes sense



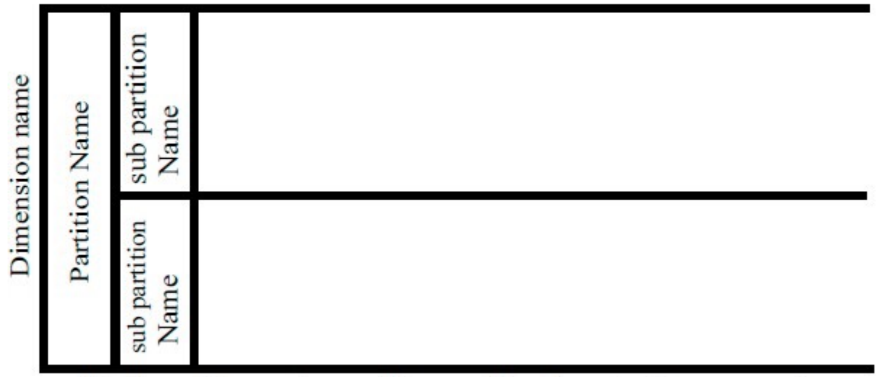
Activity Partitions/Swim Lanes: Example



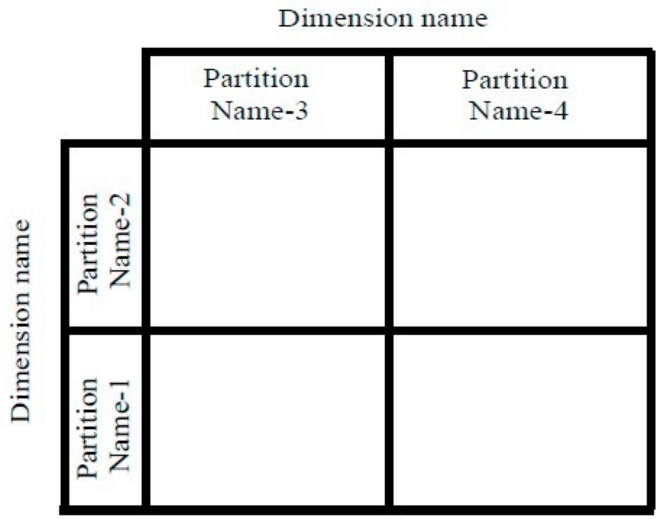
Activity Partitions/Swim Lanes



a) Partition using a swimlane notation



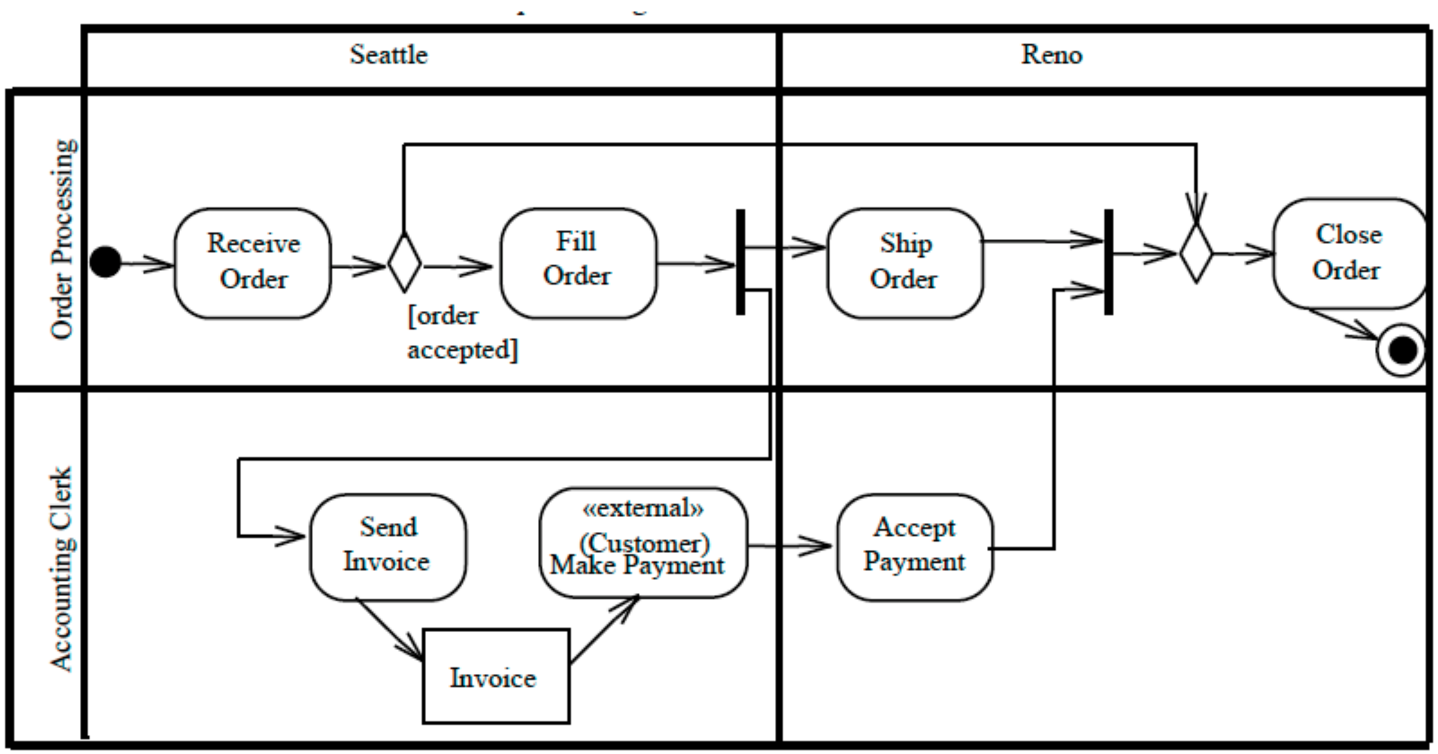
b) Partition using a hierarchical swimlane notation



c) Partition using a multidimensional hierarchical swimlane notation

UML Standard 2.5.1

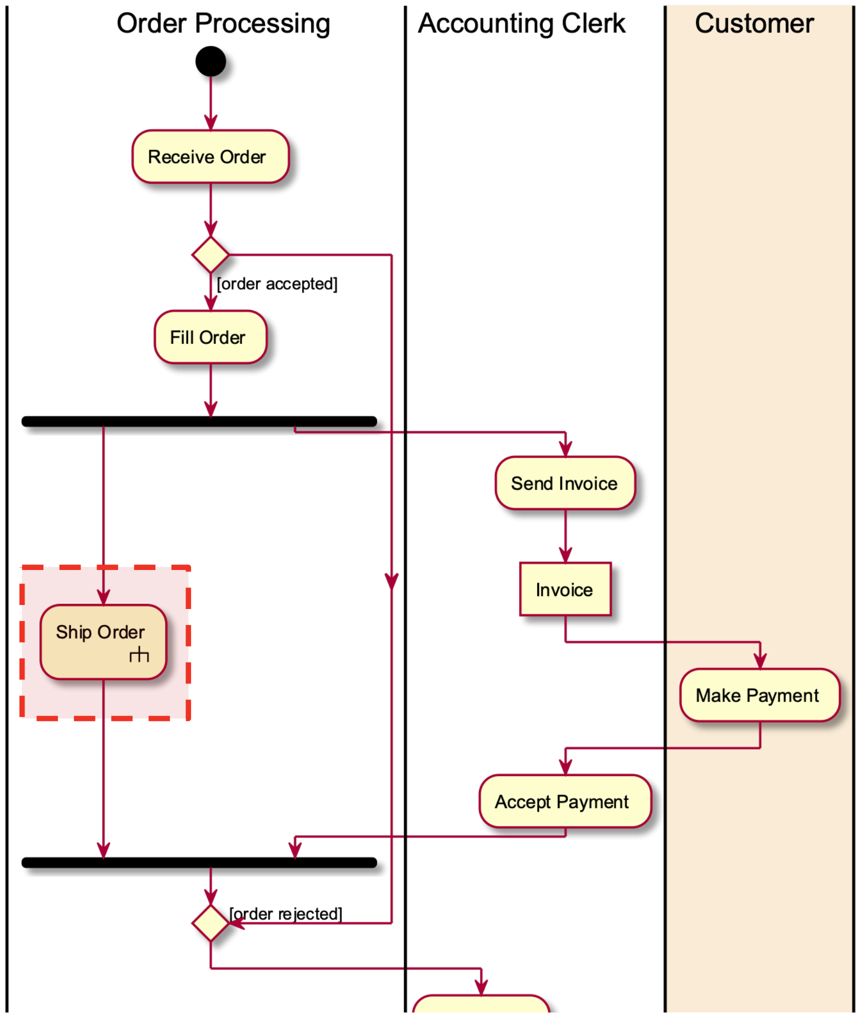
Activity Partitions/Swim Lanes: Example



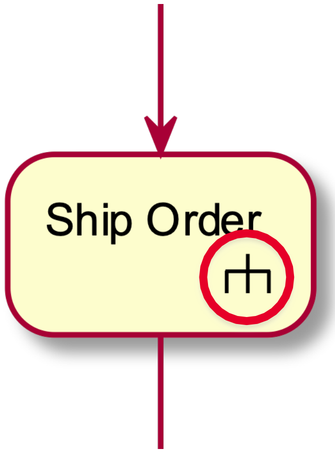
UML Standard 2.5.1

Actions with Sub Activities

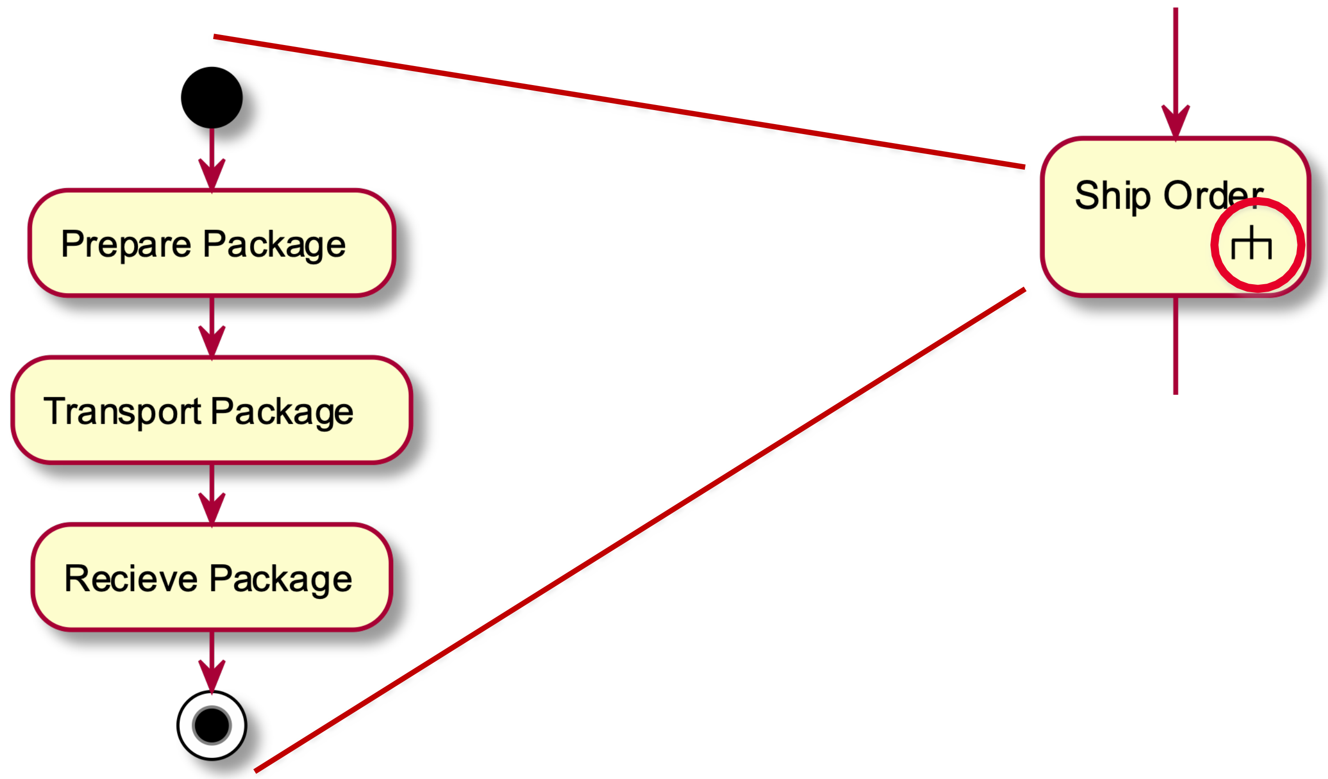
- An activity diagram can contain actions with **sub activities**
- Such actions are indicated with a **rake symbol**



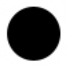
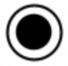
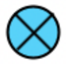
Call an activity - use the rake symbol



Call an activity - use the rake symbol

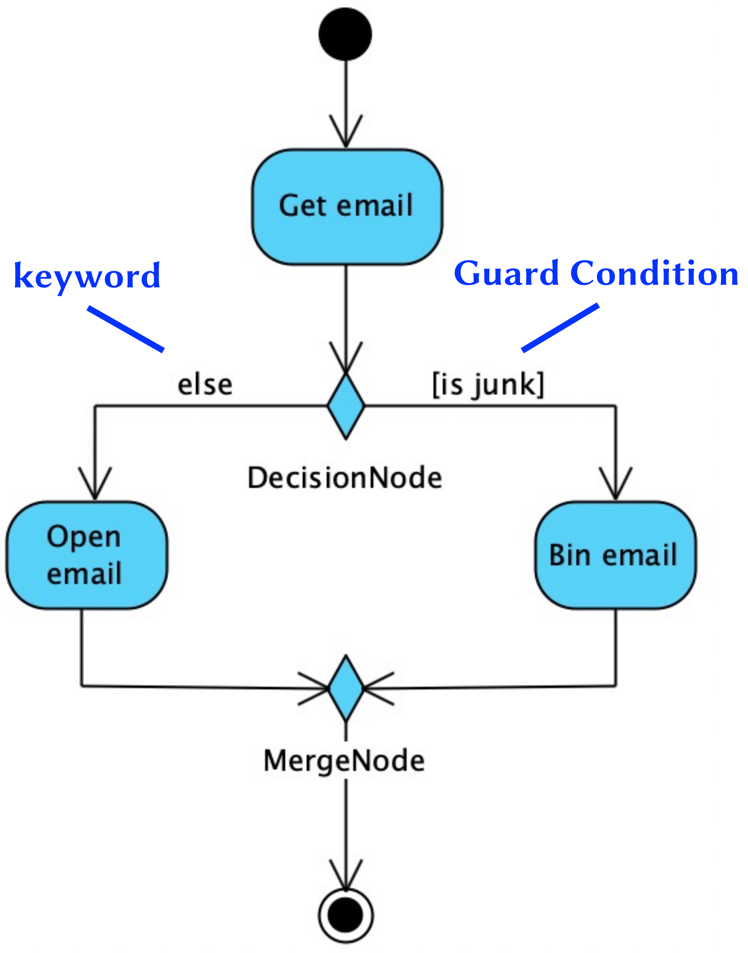


Control Nodes: Initial, Final Activity, and Flow Final Nodes

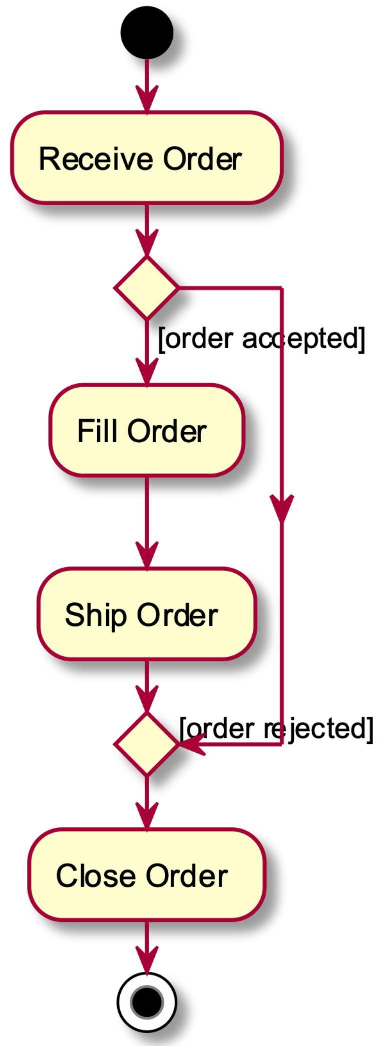
- **Initial Node** 
  - Indicates where the flow starts when an activity is invoked.
- **Final Activity Node** 
  - Terminates an activity
- **Flow Final Activity Node** 
  - Terminates a specific flow within an activity
    - Other flows are unaffected.

Control Nodes : Decision and Merge

- A **decision**, aka branch, has **a single incoming flow** and **several guarded out-bound flows**.
- Guards are Boolean conditions that
  - must not overlap (otherwise the diagram would be ambiguous)
  - must cover all possibilities
- A **merge** has **multiple input flows** and **a single output**.
  - A merge marks the end of conditional behavior started by a decision.

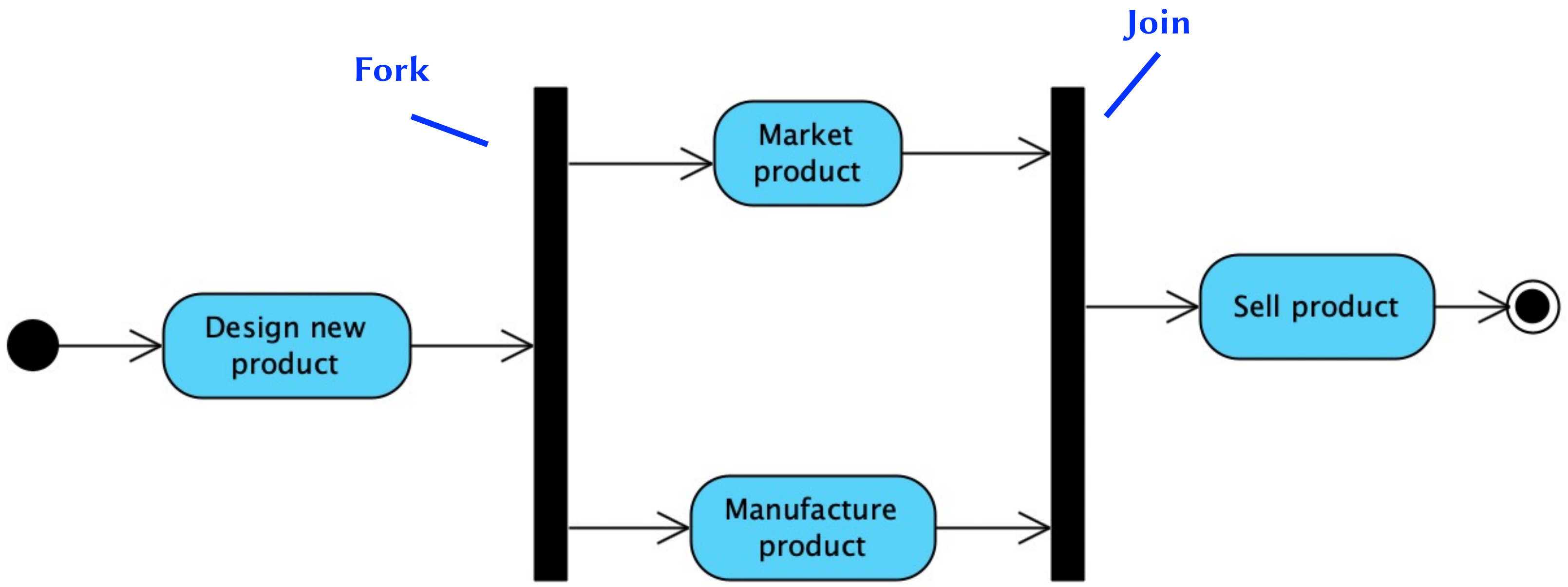


Decision and Merge - Example



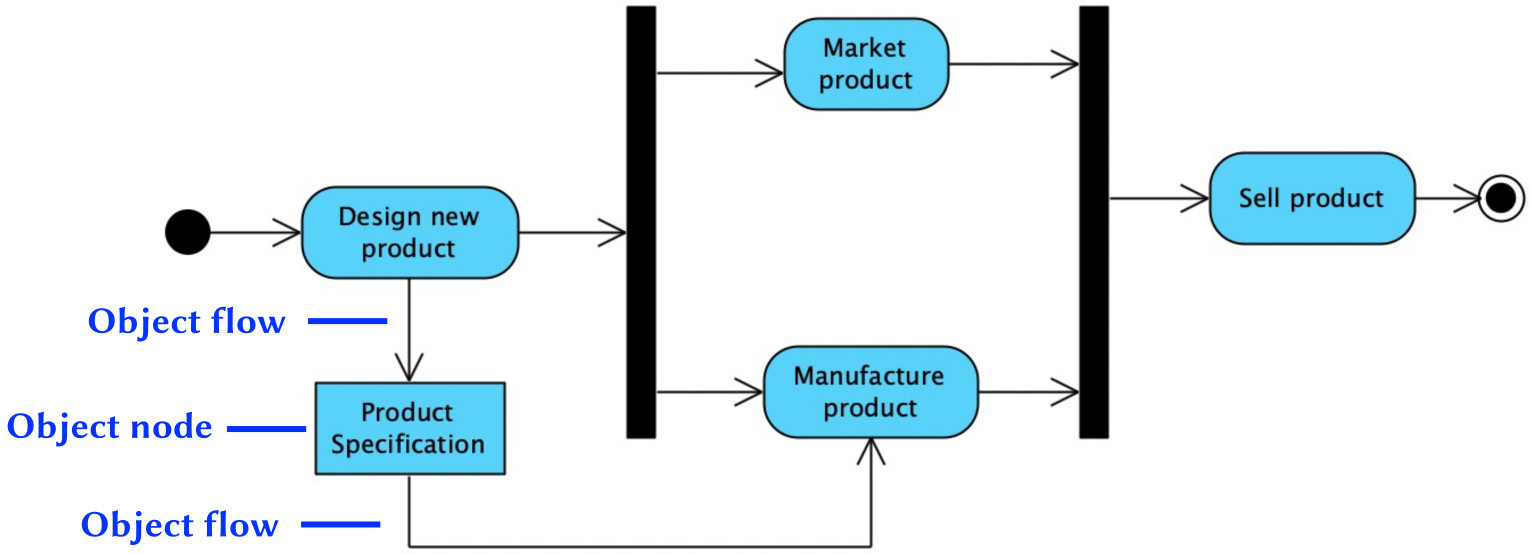
Control Nodes : Fork and Join

- **Fork** is used to model parallel actions.
- **Join** is used to synchronize parallel actions.
  - With a join, the outgoing flow is taken only when all the incoming flows reach the join.

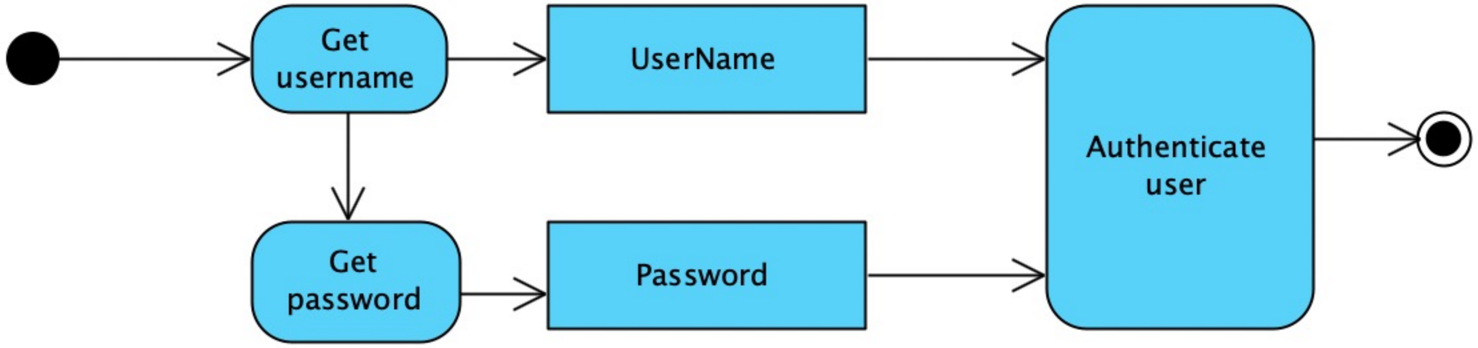


Object Nodes

- Activities may need to **exchange information**
- **Objects** are used to hold value-containing object tokens during the course of the execution of an Activity - UML Standard 2.5.1
- Input and output edges of an Object are called **object flows**



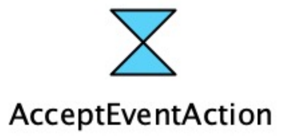
Object Nodes: Example



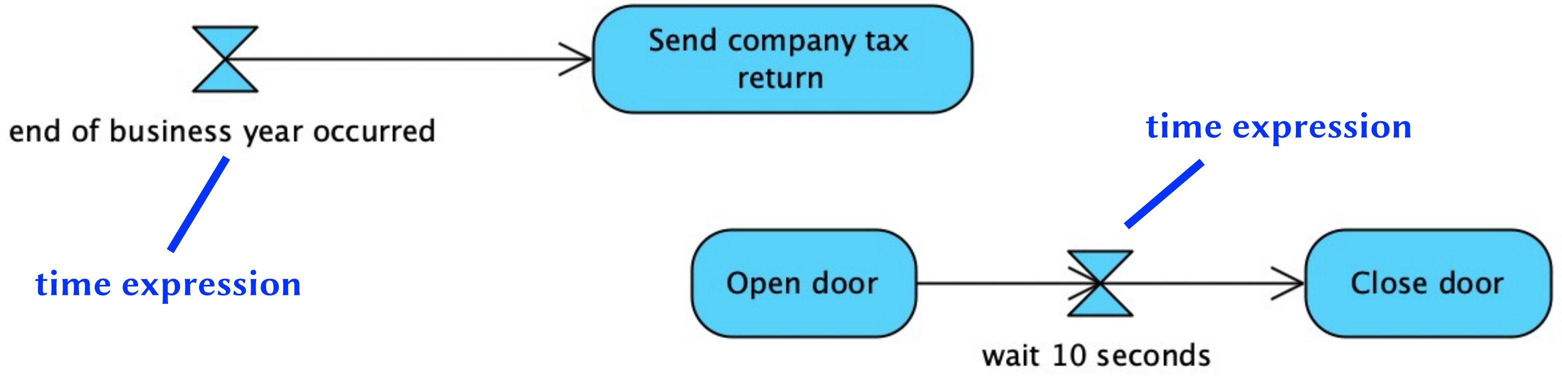
Signals

- Actions can also respond to or generate signals.
- A signal indicates that the activity receives **an event from an outside process**.

Time Signal



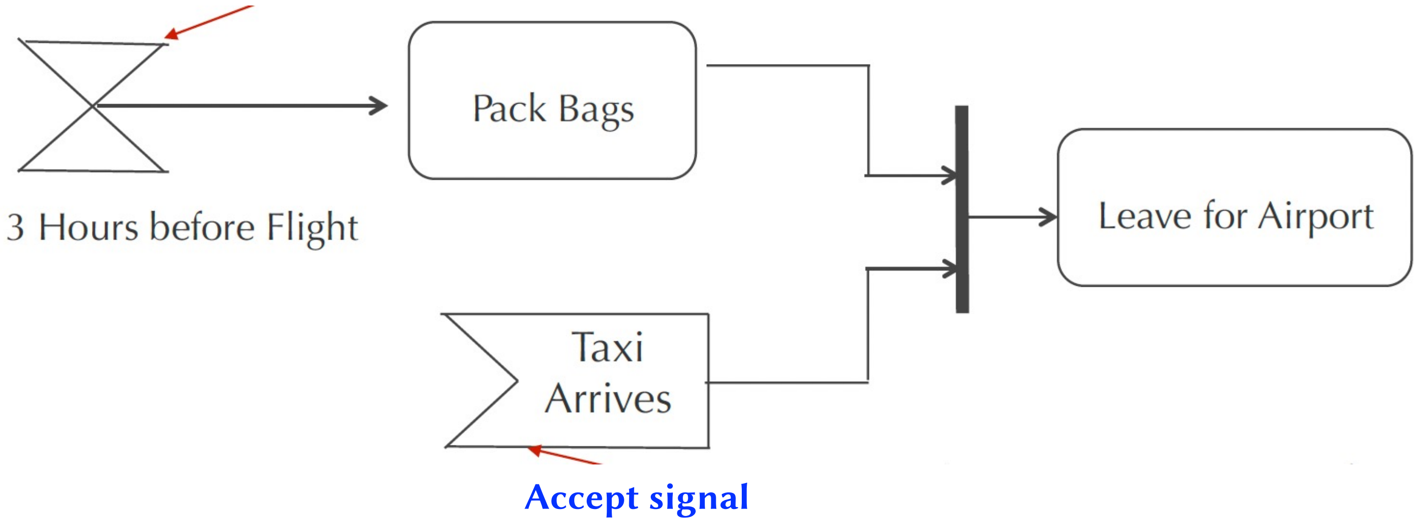
- A **time signal** occurs because of the passage of time.
- Such signals might execute when its **time expression** is true:
  - –an event in time (e.g., end of business year);
  - –a point in time (e.g., on 11/03/1960);
  - –a duration (e.g., wait 10 seconds).



Accept Signal

AcceptEventAction

- **Accept signal** waits for the occurrence of a specific event.



References

- Arlow, J., Neustadt, I., UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd Ed. Addison-Wesley, 2005.
- UML Distilled, Martin Fowler
- Object-oriented Design course by Raman Ramsin, Sharif University of Technology, Iran, http://sharif.edu/~ramsin/index_files/undergradcourse_OOD.htm
- Sebastian Rodriguez, Software Engineering Fundamentals for IT (2110), RMIT University, Course Materials on RMIT Canvas
- Schaum’s Outlines UML (2nd edition)
- Melina Vidoni, Software Engineering Fundamentals (2050), RMIT University, Course Materials on RMIT Canvas
- Halil Ali , Software Engineering Fundamentals (Semester 1, 2020), RMIT University, Course Materials on RMIT Canvas
- Ian Sommerville, Software Engineering, 10th Edition, 2015.

Thanks!

Mojtaba Shahin

mojtaba.shahin@rmit.edu.au