Object-Oriented Design

Lecturer: Raman Ramsin

Lecture 13:

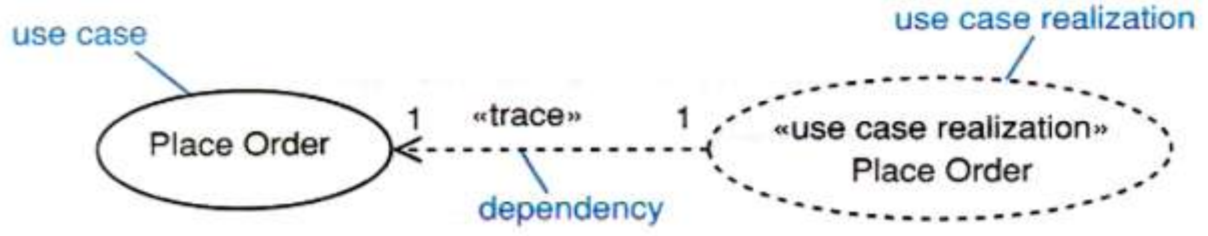
Use Case Realizations – Part 1

Analysis Workflow: *Analyze a Use Case*

- The *analysis workflow* consists of the following activities:
  - Architectural analysis
  - **Analyze a use case**
    - **Outputs:**
      - **analysis classes**
      - **use case realizations**
  - Analyze a class
  - Analyze a package

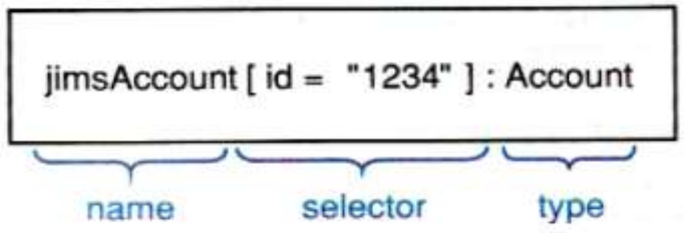
Use Case Realizations

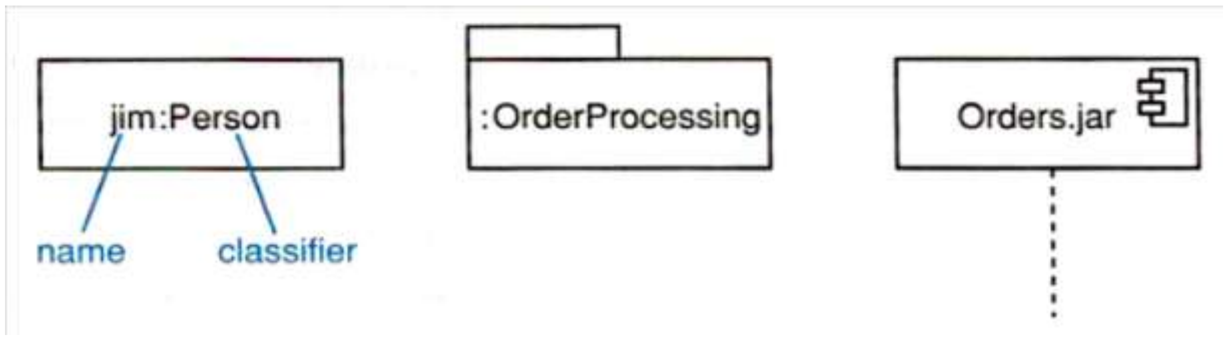
- Use case realizations show how instances of analysis classes interact to realize the functional requirements specified by a use case.
- Each use case realization realizes exactly one use case.
- Use case realizations consist of:
  - analysis class diagrams - these should "tell a story" about one (or more) use cases;
  - interaction diagrams - these demonstrate how objects interact to realize the use case behavior;
  - special requirements - you always uncover new requirements during use case realization and you need to record these;
  - use case refinement - you may need to change a use case as you begin to realize it.



Lifelines

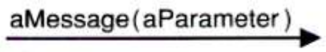
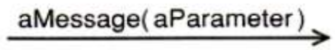
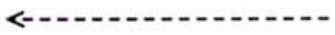
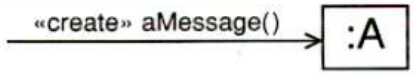
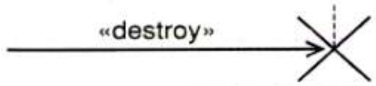
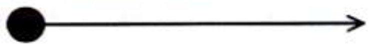
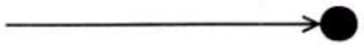
- A lifeline represents a participant in an interaction - how an instance of a classifier participates in the interaction.
  - Each lifeline has an optional name, a type, and an optional selector.
  - Each lifeline is drawn with the same icon as its type.
  - Underline the name, type, and selector to show actual instances .





Messages

- A message represents a specific kind of communication between two lifelines in an interaction.

Syntax	Name	Semantics
	Synchronous message	The sender waits for the receiver to return from executing the message
	Asynchronous message	The sender sends the message and continues executing – it does *not* wait for a return from the receiver
	Message return	The receiver of an earlier message returns focus of control to the sender of that message
	Object creation	The sender creates an instance of the classifier specified by the receiver
	Object destruction	The sender destroys the receiver If its lifeline has a tail, this is terminated with an X
	Found message	The sender of the message is outside the scope of the interaction Use this when you want to show a message receipt, but don't want to show where it came from
	Lost message	The message never reaches its destination May be used to indicate error conditions in which messages are lost

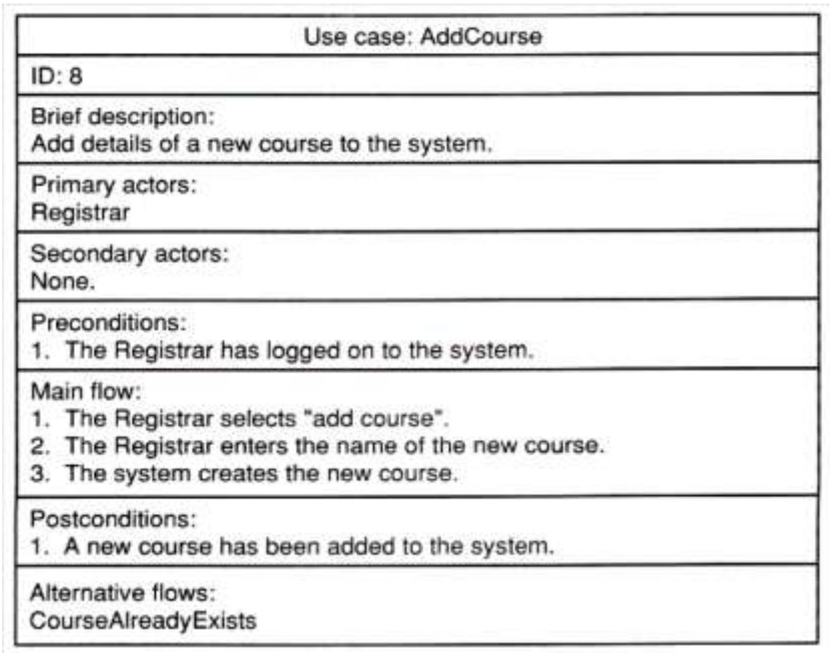
Interaction Diagrams

- **Sequence diagrams** - emphasize time-ordered sequence of message sends.
- **Communication diagrams** - emphasize structural relationships between objects.
- **Interaction overview diagrams** - emphasize relationships between interactions.
- **Timing diagrams** - emphasize real-time aspects of interactions.

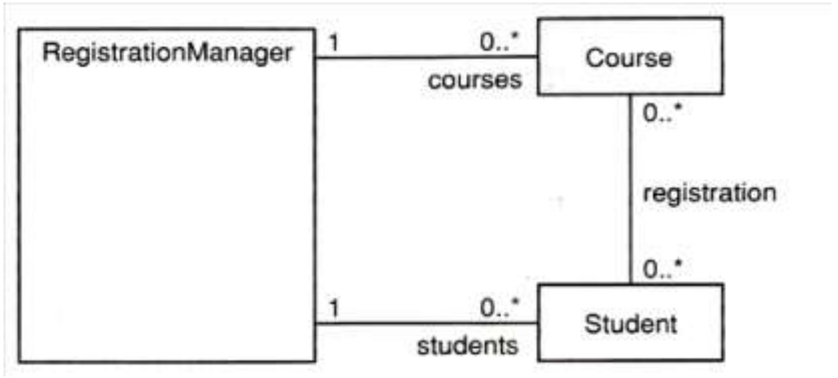
Sequence Diagrams: General Notation

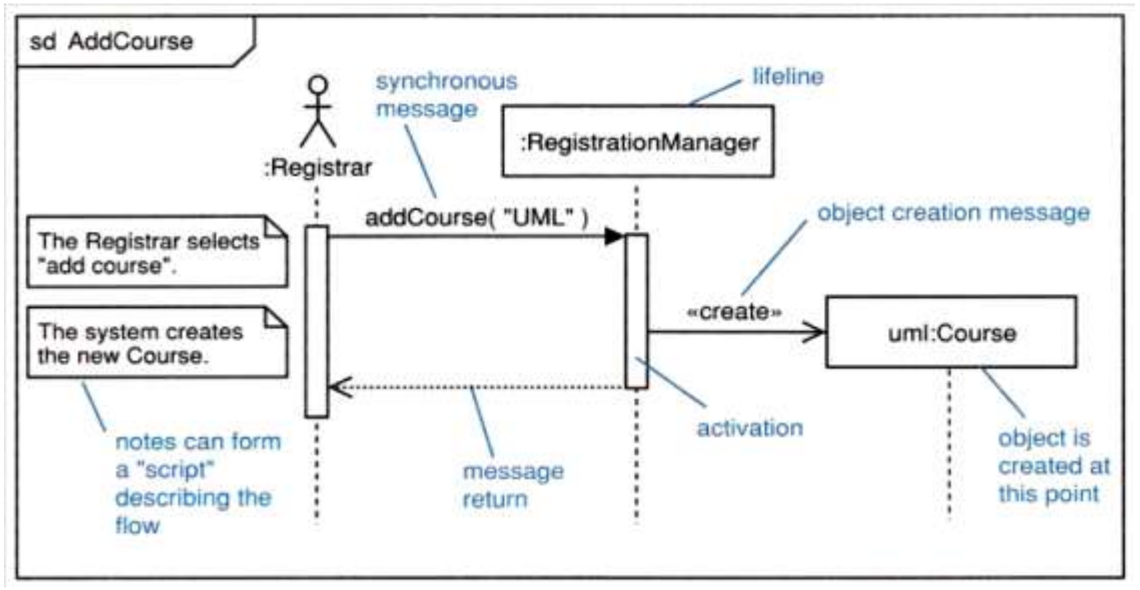
- Time runs top to bottom.
- Lifelines run left to right:
  - lifelines have dashed vertical tails that indicate the duration of the lifeline;
  - lifelines may have activations to indicate when the lifeline has focus of control;
  - organize lifelines to minimize the number of crossing lines.
- Place explanatory scripts down the left-hand side of the sequence diagram.
- State invariants - place state symbols on the lifeline at the appropriate points.
- Constraints - place constraints in {} on or near lifelines.

Sequence Diagrams: Realization Example 1

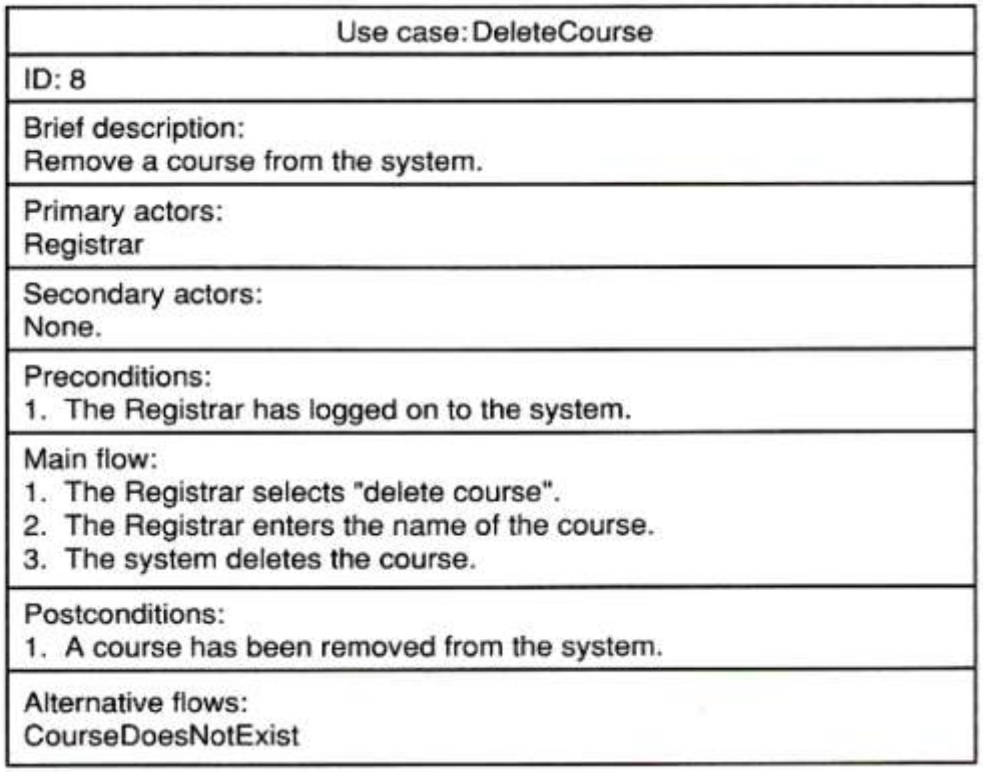


Use case: AddCourse
ID: 8
Brief description: Add details of a new course to the system.
Primary actors: Registrar
Secondary actors: None.
Preconditions: 1. The Registrar has logged on to the system.
Main flow: 1. The Registrar selects "add course". 2. The Registrar enters the name of the new course. 3. The system creates the new course.
Postconditions: 1. A new course has been added to the system.
Alternative flows: CourseAlreadyExists

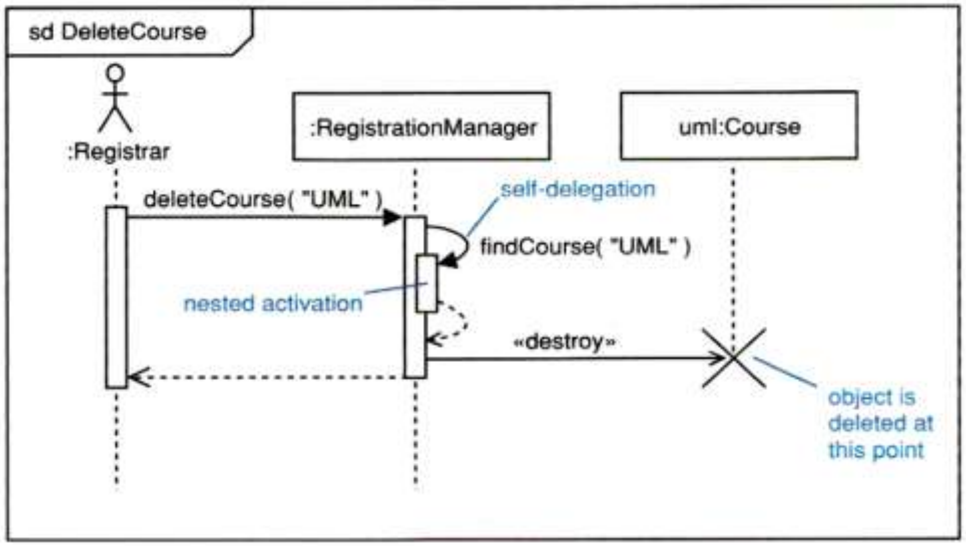




Sequence Diagrams: Realization Example 2

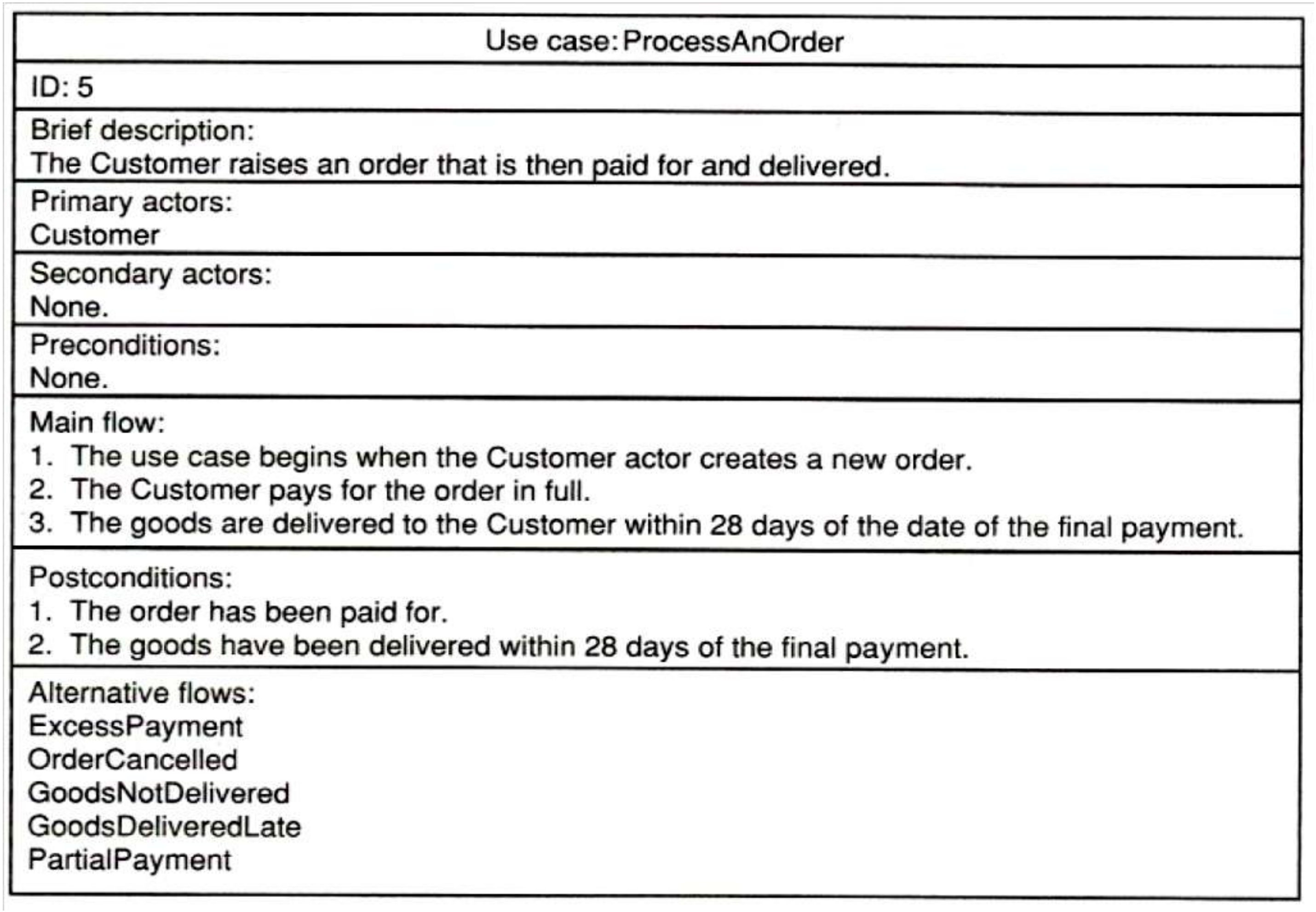


Use case: DeleteCourse
ID: 8
Brief description: Remove a course from the system.
Primary actors: Registrar
Secondary actors: None.
Preconditions: 1. The Registrar has logged on to the system.
Main flow: 1. The Registrar selects "delete course". 2. The Registrar enters the name of the course. 3. The system deletes the course.
Postconditions: 1. A course has been removed from the system.
Alternative flows: CourseDoesNotExist



Sequence Diagrams: Realization Example 3

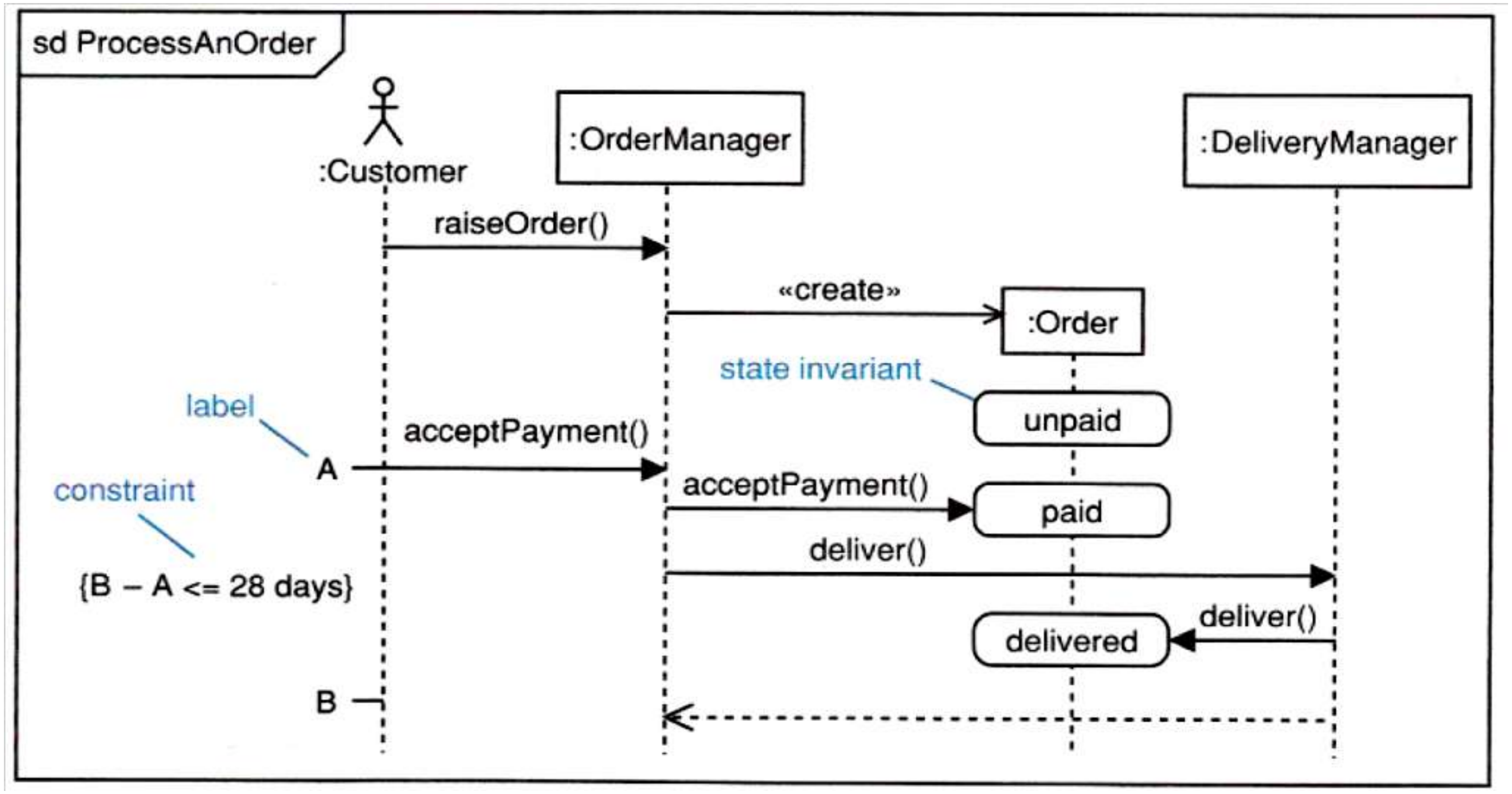
Use Case



Use case: ProcessAnOrder
ID: 5
Brief description: The Customer raises an order that is then paid for and delivered.
Primary actors: Customer
Secondary actors: None.
Preconditions: None.
Main flow: 1. The use case begins when the Customer actor creates a new order. 2. The Customer pays for the order in full. 3. The goods are delivered to the Customer within 28 days of the date of the final payment.
Postconditions: 1. The order has been paid for. 2. The goods have been delivered within 28 days of the final payment.
Alternative flows: ExcessPayment OrderCancelled GoodsNotDelivered GoodsDeliveredLate PartialPayment

Sequence Diagrams: Realization Example 3

Sequence Diagram



Reference

- Arlow, J., Neustadt, I., *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*, 2nd Ed. Addison-Wesley, 2005.