SYSTEM ANALYSIS AND DESIGN

- Nguyen Thanh Binh, Nguyen Quang Vu, Le Viet Truong, Vo Van Luong, Nguyen Ngoc Huyen Tran
- Faculty of Computer Science

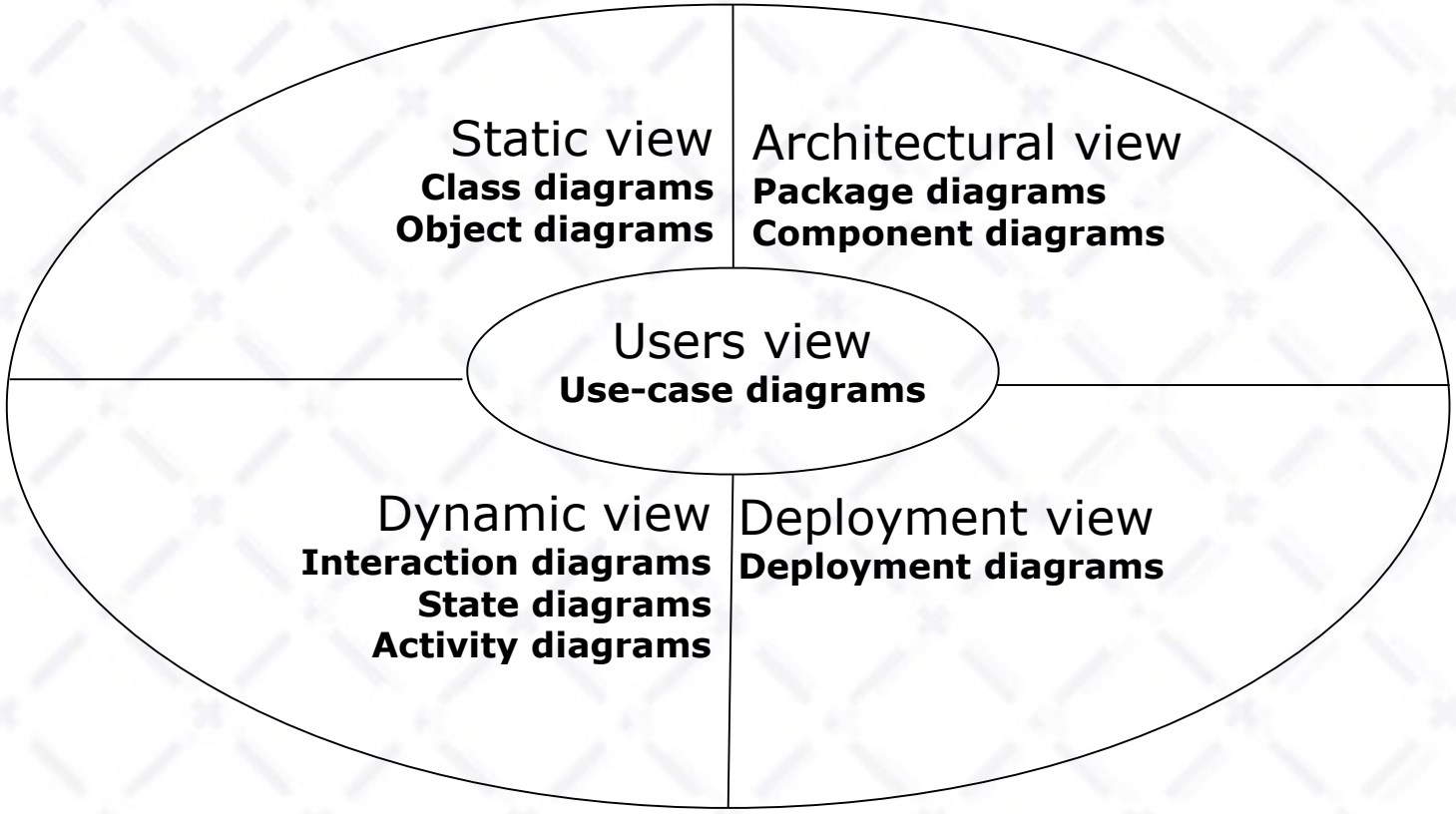
- Chapter 6

Dynamic behavioral modeling

Dynamic behavioral modeling

- Activity diagrams
- State diagrams
- Interaction diagrams

Views


Static view
Class diagrams
Object diagrams
Architectural view
Package diagrams
Component diagrams
Users view
Use-case diagrams
Dynamic view
Interaction diagrams
State diagrams
Activity diagrams
Deployment view
Deployment diagrams

Main Activities of Software Development

Requirements Gathering

Define requirement specification

Analysis

Define the conceptual model

Design

Design the solution / software plan

Implementation

Code the system based on the design

Integration and Test

Prove that the system meets the requirements

Deployment

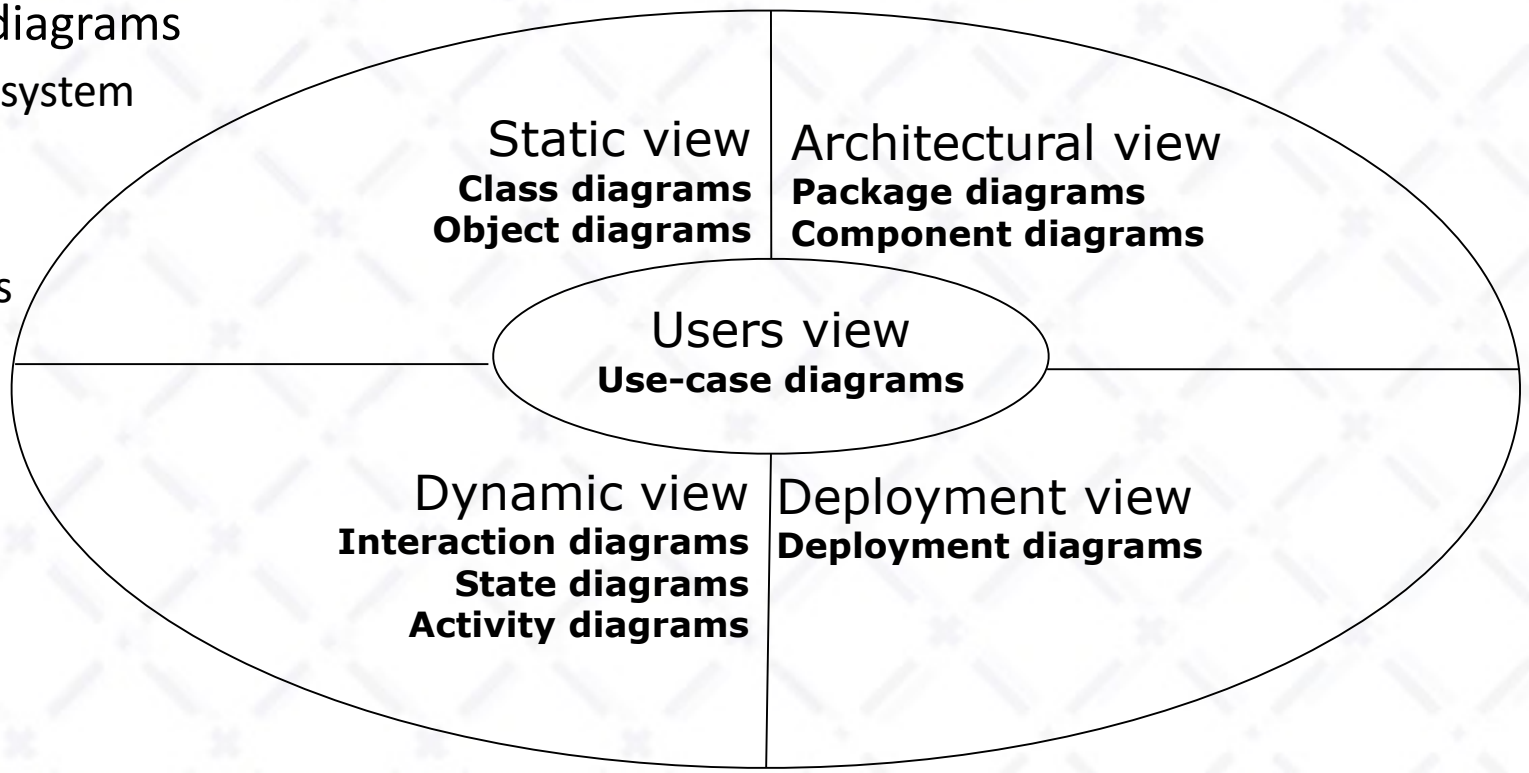
Installation and training

Maintenance

Post-install review
Support docs
Active support

Dynamic behavioral modeling

- Describing what happens during the execution of the system
  - The behavior of the objects
- Dynamic behavior modeling allows to complete the information in the static diagram
  - How the elements in the static diagrams
    - provide the functionality of the system
    - change their states
    - communicate with each other
    - cooperate to perform their tasks



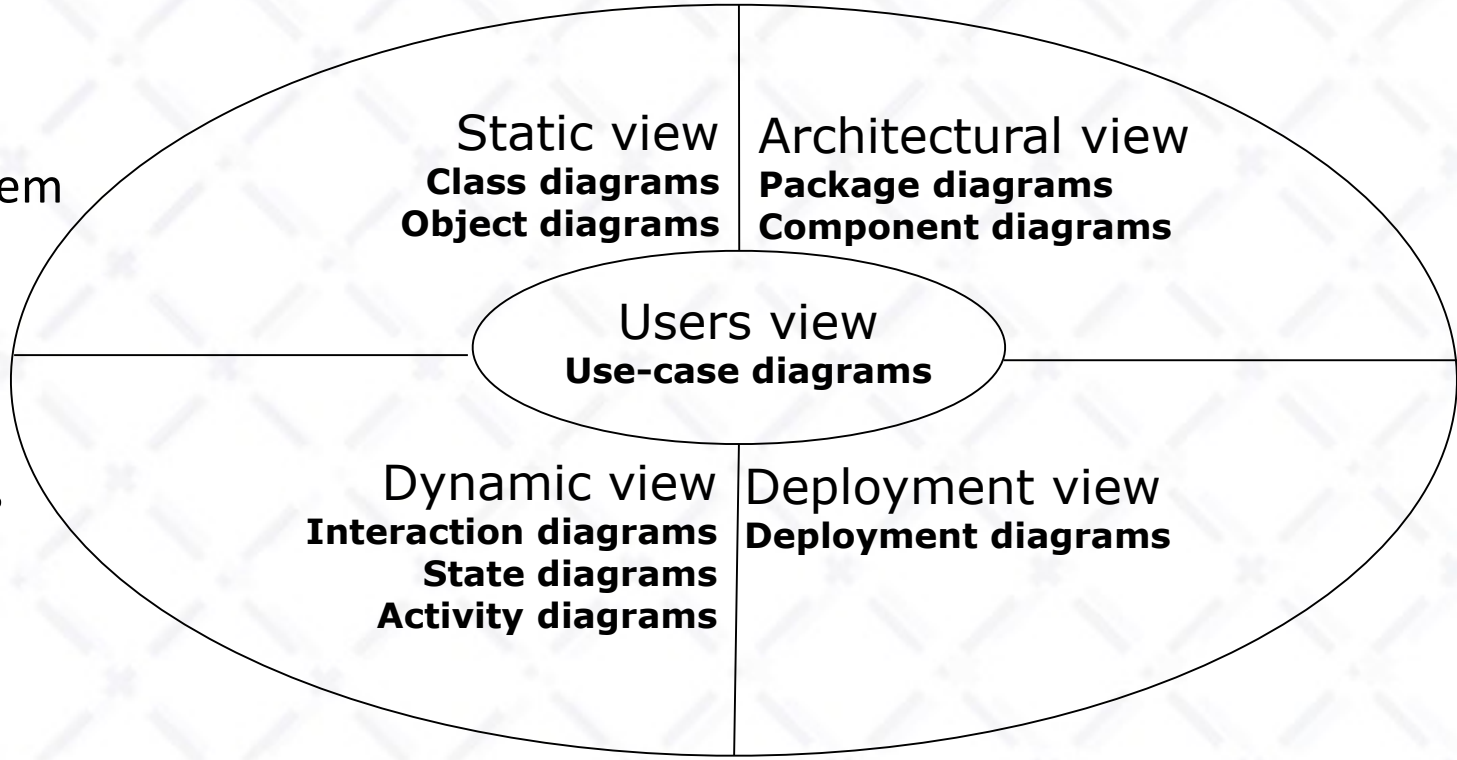
Dynamic behavioral modeling

- Difficulties
  - modeling of the dynamic behaviors of a complex system is always difficult
    - Too many features
    - Too many paths
  - The collaborations between objects are complicated
  - It is not easy to allocate and carry out responsibilities

- Suggestions
  - Focusing on the communication of one dynamic aspect of the system to better master the complexity
  - modeling only the essential elements
  - Providing a detail suitable for each level of abstraction

Dynamic behavioral modeling

- Diagrams
  - Activity diagrams
    - High level dynamic behavior
    - Performing objectives of the system
  - State diagrams
    - Internal behavior of the system
  - Interaction diagrams
    - Communication between objects
      - Sequence diagrams
      - Communication diagrams



Activity diagrams

- Allowing to determine the dynamic behavior of the system from **one or several use-cases**
- Being useful to model the flows of treatment in the system
- modeling the control flow and also the data stream

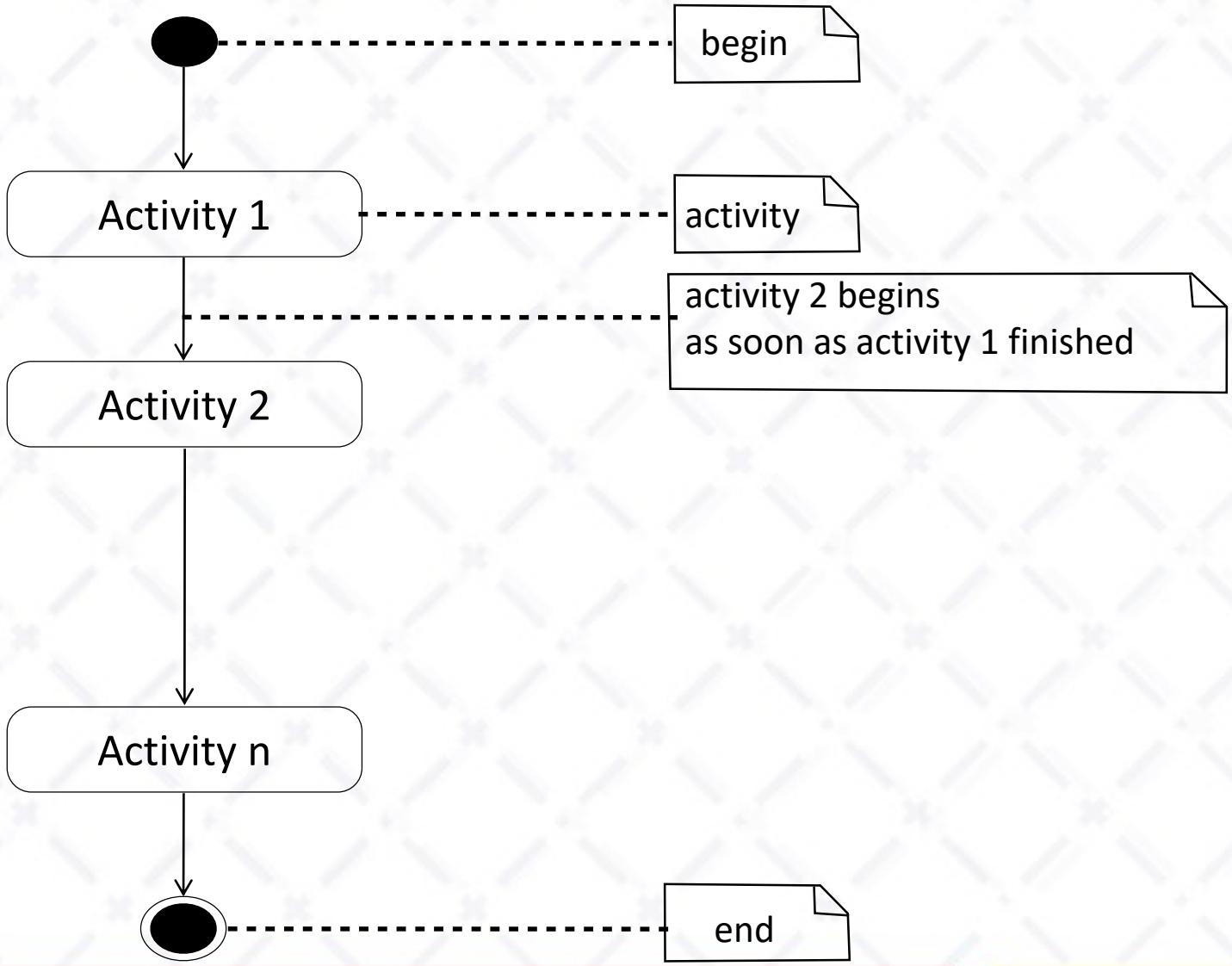
- An activity diagram includes
  - the activities carried out by the system and the actors
  - the order in which these activities are carried out
  - the possible dependencies between activities.

Activity diagrams

- An activity corresponds to a high-level task in the system
- Distinction between the activities and operations in the static structure
  - Activities are carried out by the system or the actors
  - Operations are related to classes
  - In general, activities do not correspond to operations
- Activity diagrams are generally built before (design) class diagrams
  - Activity diagrams are used to determine which operations to add to class diagrams

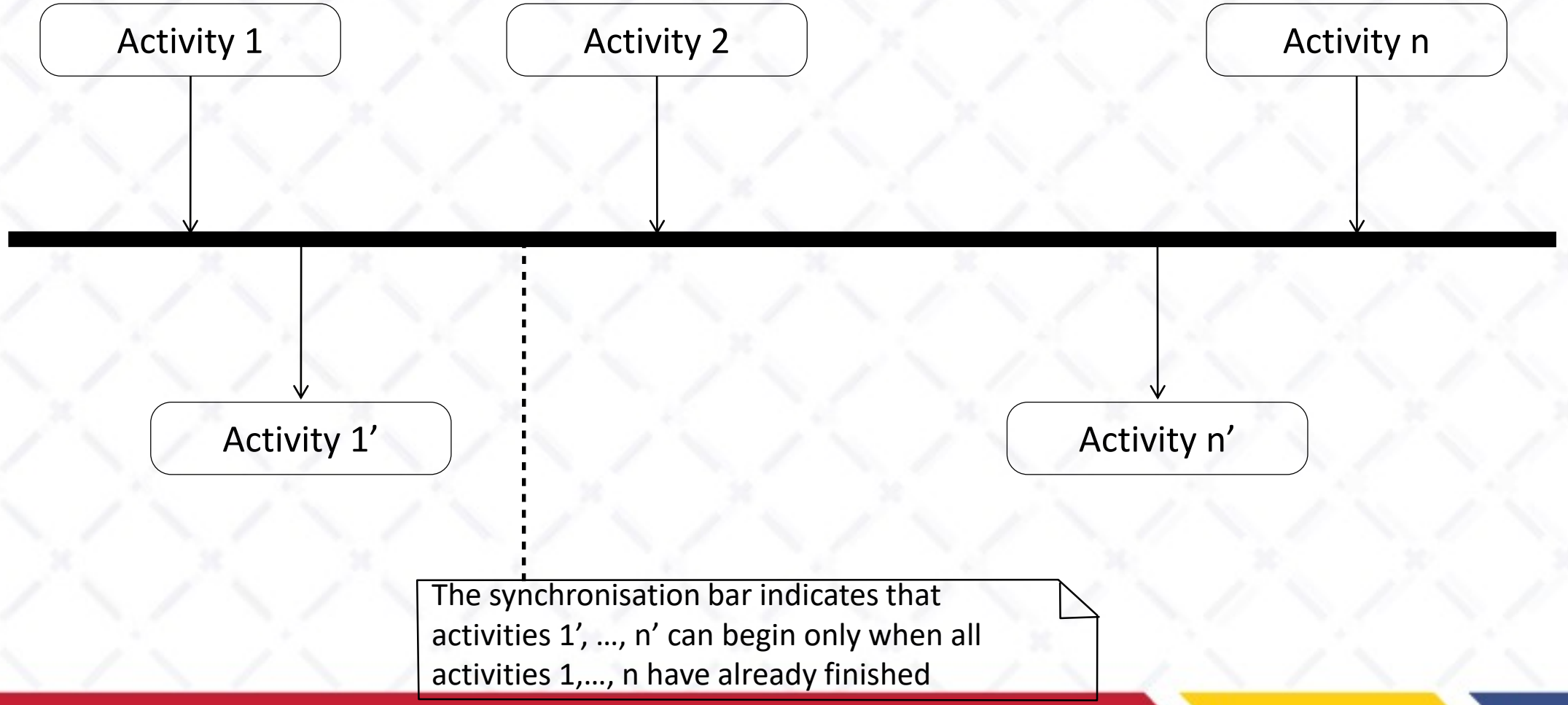
Activity diagrams

- Notation



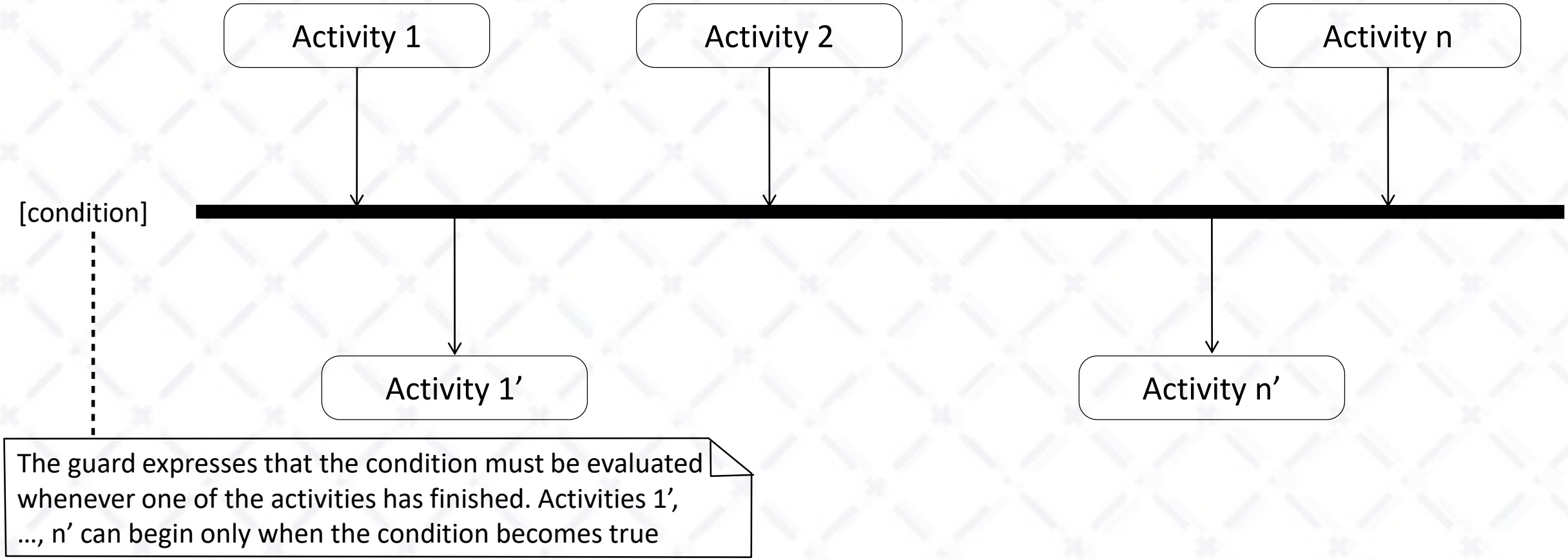
Activity diagrams

- Synchronisation of activities



Activity diagram

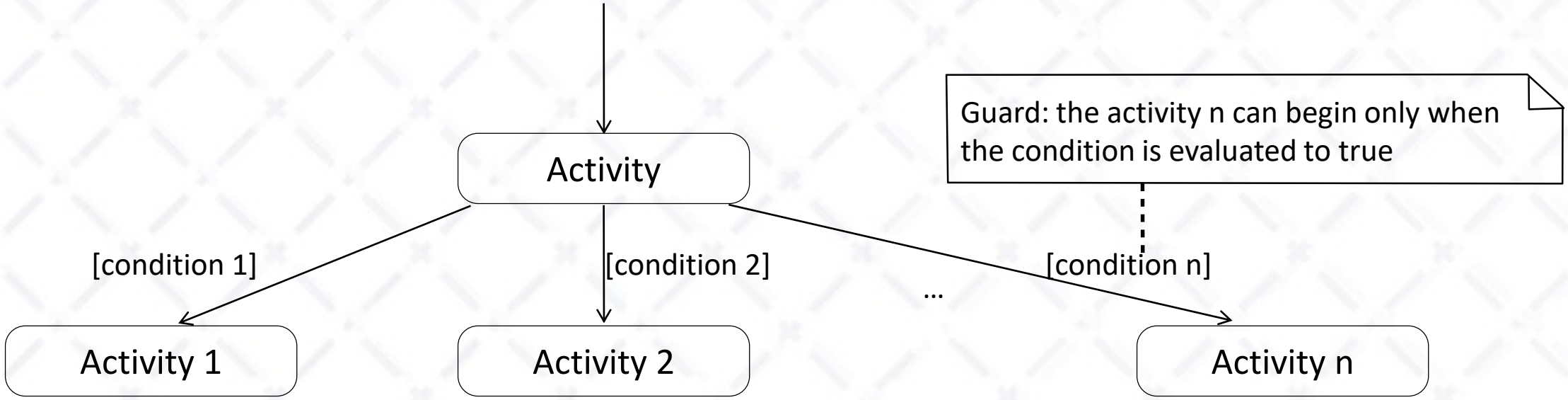
- Synchronisation guarded by a condition



- The absence of guard can be considered as a special guard. This one becomes true when all the activities at the entrance to the synchronisation bar have finished

Activity diagrams

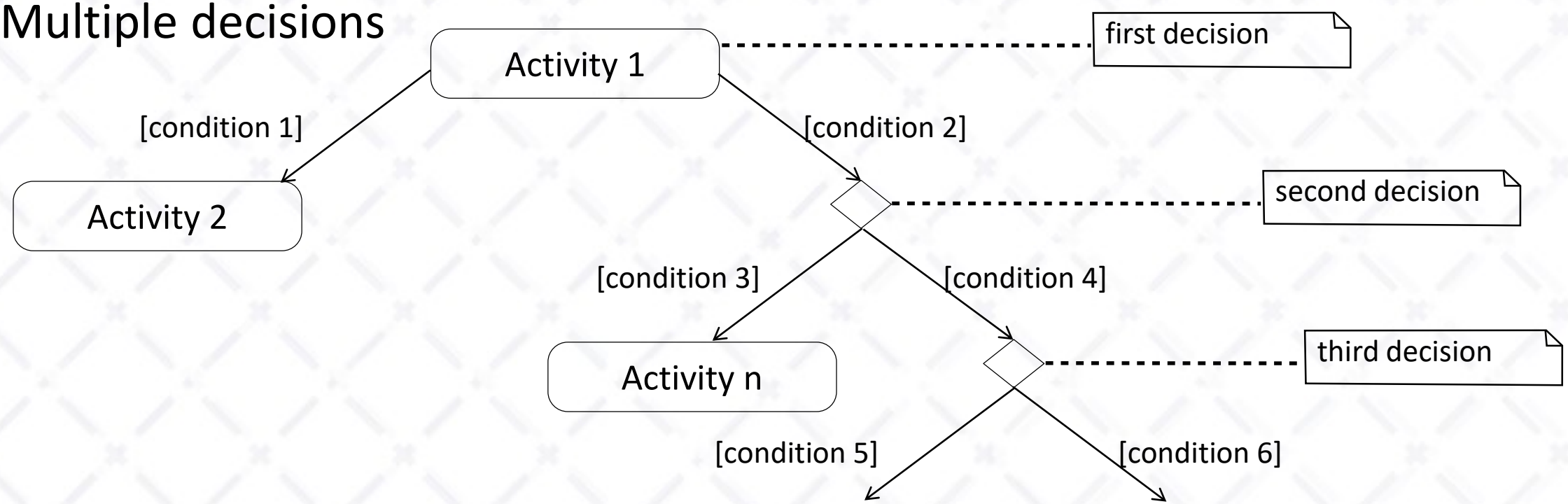
- Decision



- The transition guards coming out of the same activity should be mutually exclusive

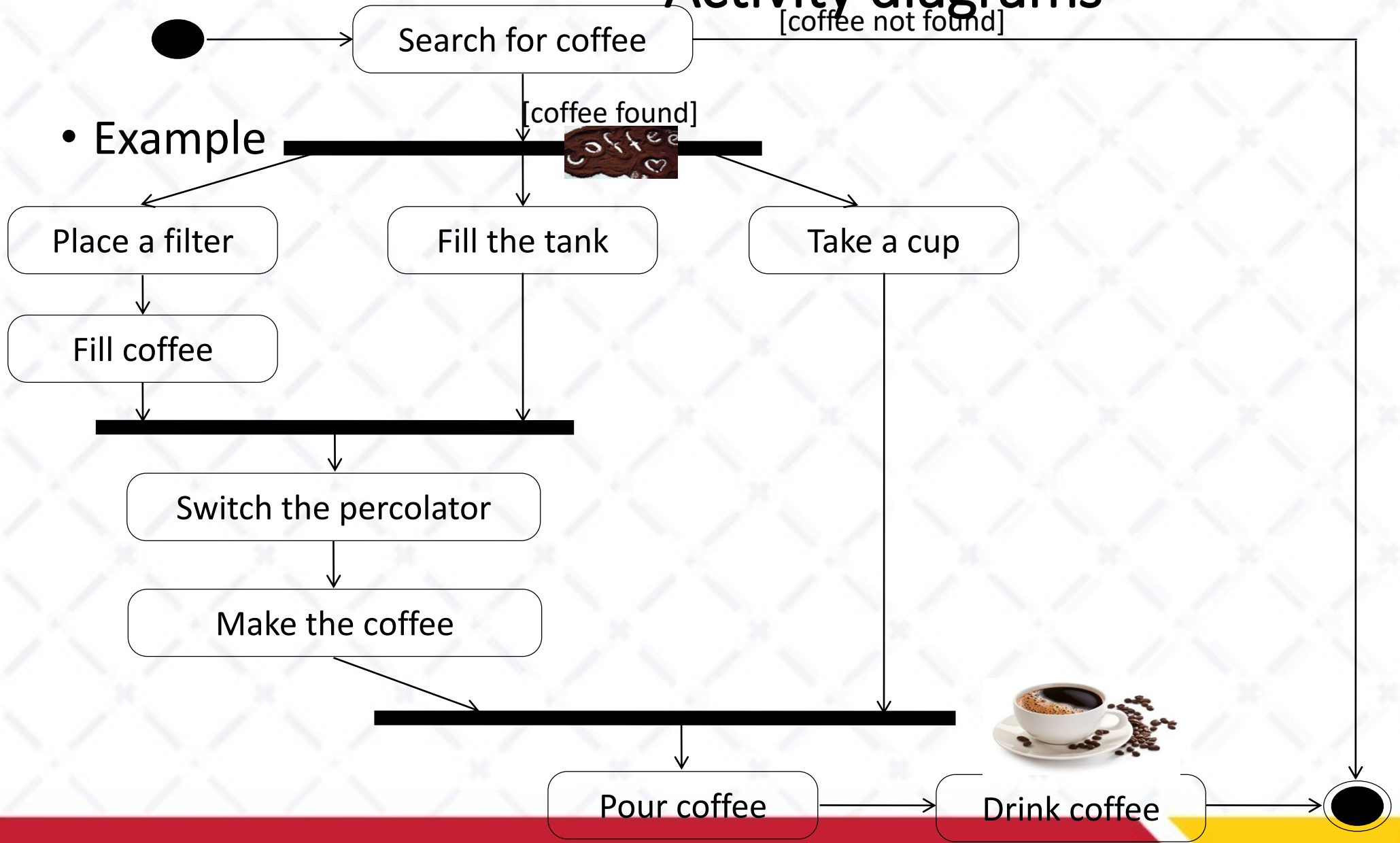
Activity diagrams

- Multiple decisions



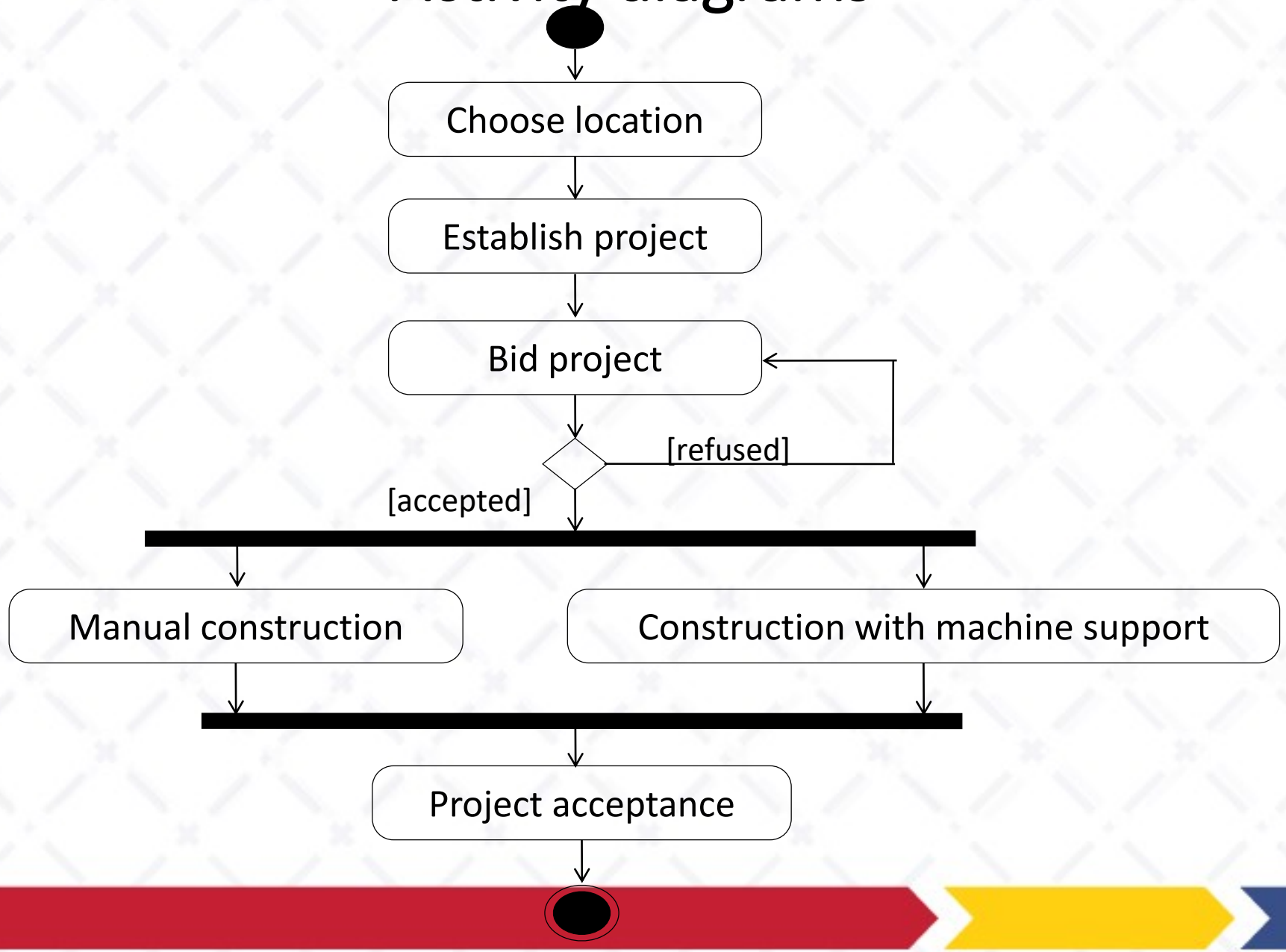
Activity diagrams

- Example



Activity diagrams

- Example

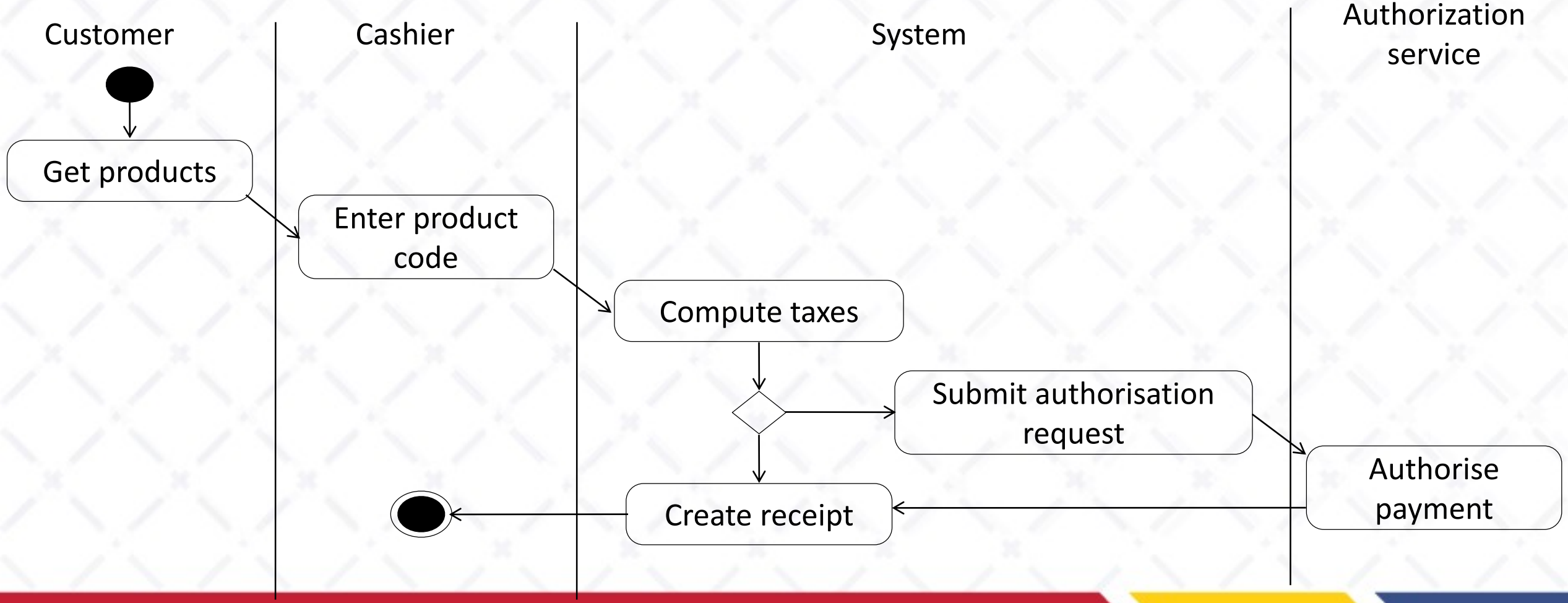


Activity diagram

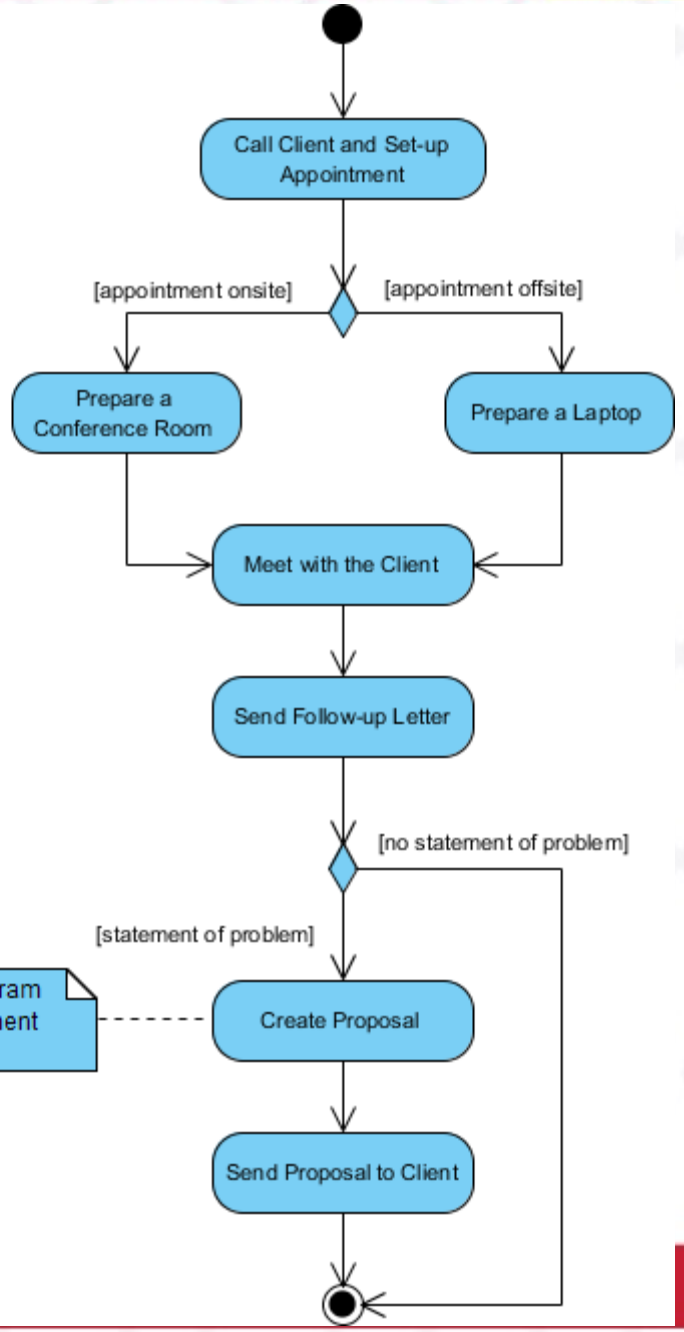
- **Swimlane** helps to clarify on the activity diagrams the actors or components of the system that perform different activities
- Example: Activity diagram for "**sale**" use-case

Activity diagram

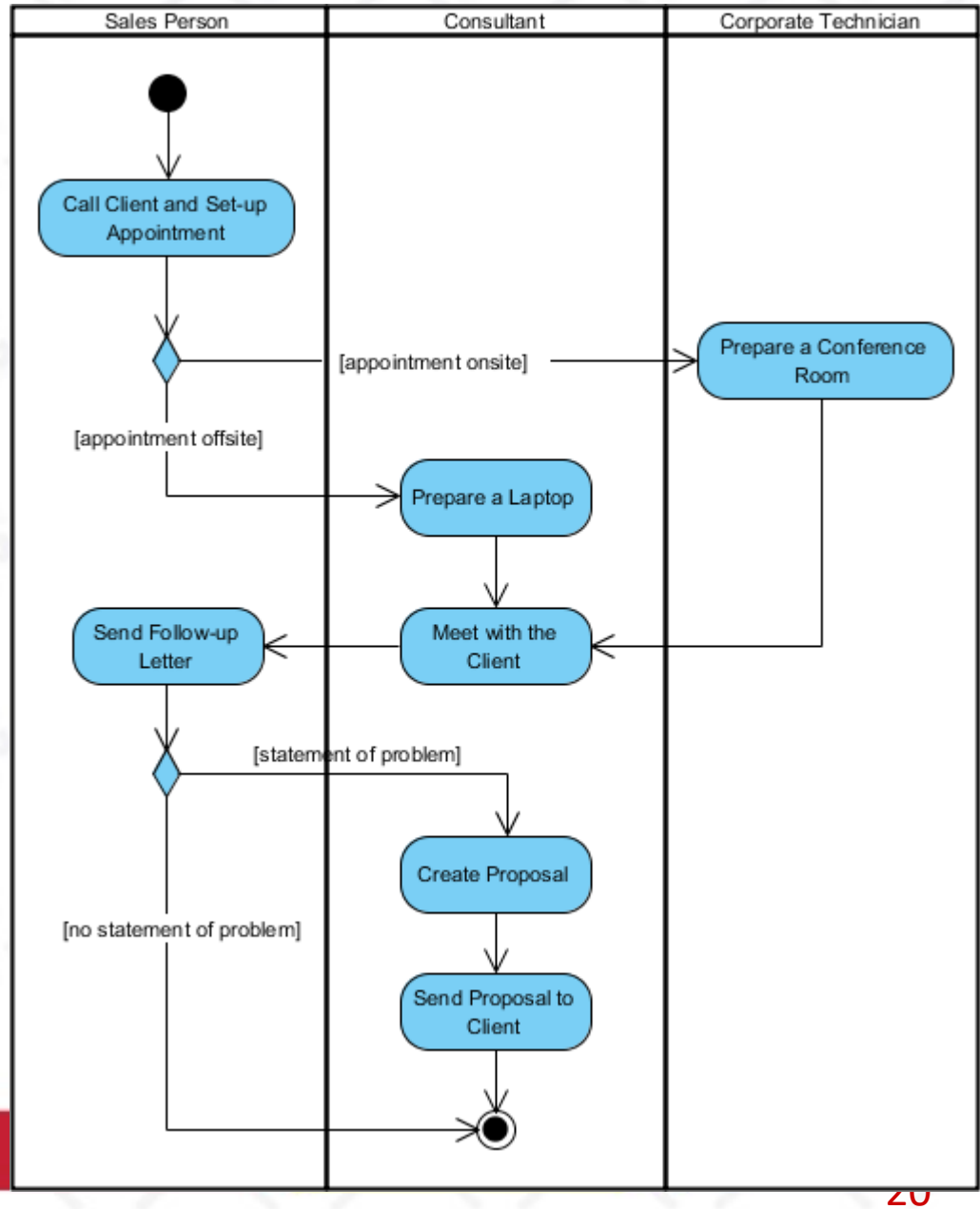
- Example: Activity diagram for "**purchase products**" use-case



Activity diagram



This figures describes the business process for meeting a new client using an activity Diagram without swinlane/with swinlane.



State Diagrams

- State diagrams
  - are **finite state automata**
  - allow to model the dynamic behavior of a Communication or a class
  - focus on the behavior of objects, ordered by events
  - are especially used for modeling reactive systems

State diagrams

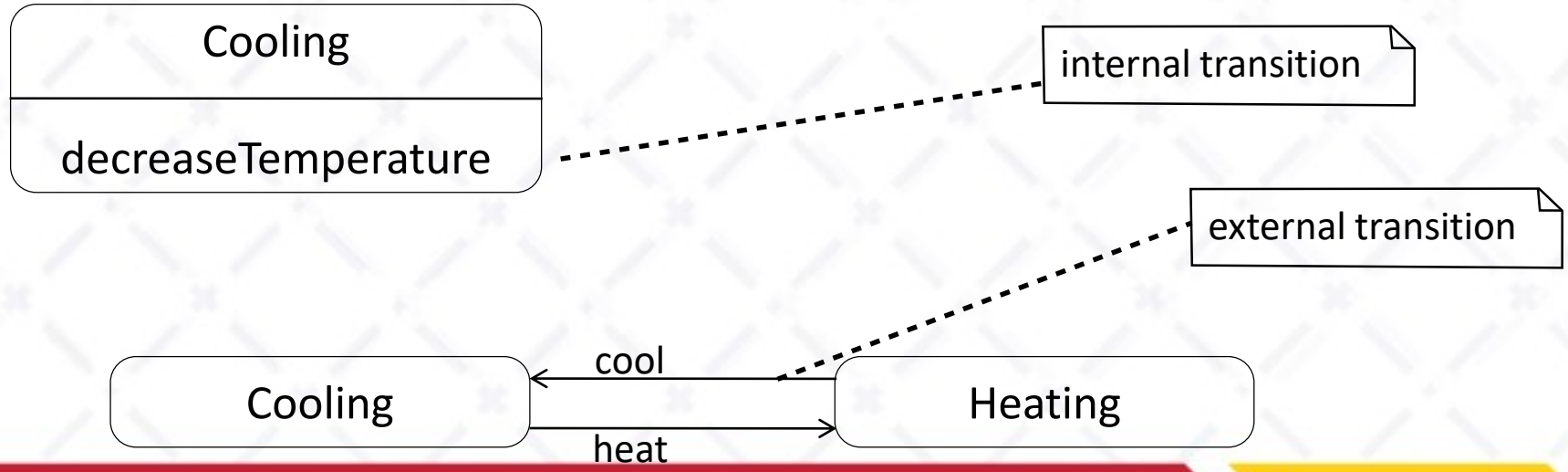
- State diagrams describe the behavior of a system, part of a system or an object in the system
  - Each system or object has a **state** at a given time
  - In a given state, the system behaves in a specific manner to respond to the coming events
  - The **events** trigger state changes
- Specifically, a state diagram models the changes of states of a system/object in response to events
- A state diagram includes
  - **State**: state of a system/object at a given time
  - **Transition**: allows to switch a state to another
  - **Event**: activates the transition

State diagrams

- State
  - Represents a situation of a system/object at an instance
  - System/object remains in a state for a while. Meanwhile, it can
    - perform certain **activities**
    - wait until an **event** occurs
  - Notation

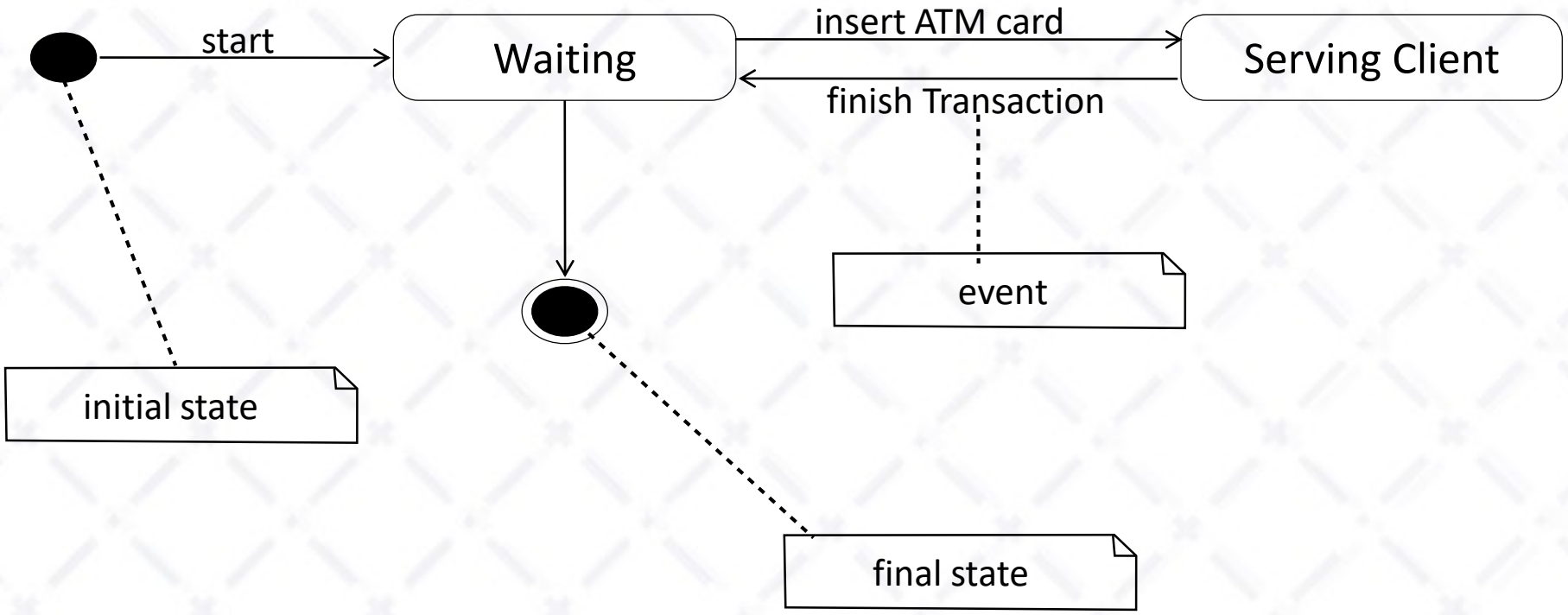
State diagrams

- Transitions
  - Transitions are related to actions that can be performed by the system/object associated with the diagram
  - Two transition types
    - **Internal transitions** to a state that react to an event without changing the current state of the system or object
    - The **transitions between states** or **external transitions**, which express a change in state
  - Example: States of an air conditioner



State diagrams

- Example: Describing the states of an ATM machine



State diagrams

- Event
  - Events of a transition have the following general form

    Event [guard] / action

    - Event: the event name leading to the transition
    - Guard: the condition must be satisfied in order to overcome the transition
    - Action: the operation performed when crossing the transition

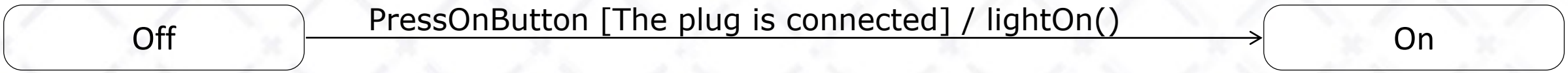
  - Remark: some of these elements may be omitted

State diagrams

- Event
  - Example: states of a heater

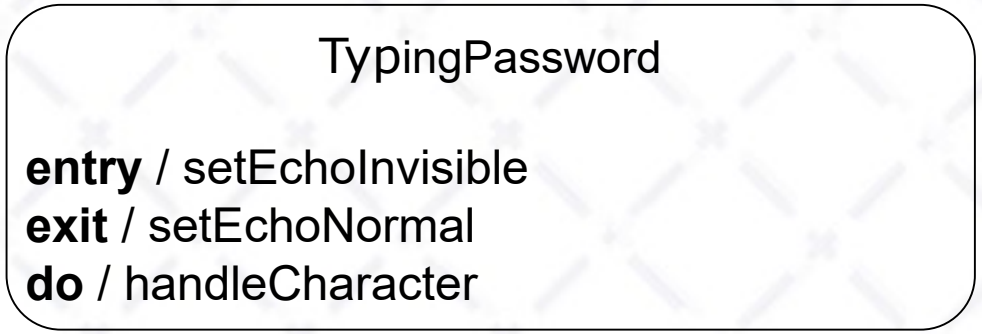
State diagrams

- Example: State of a lightbulb



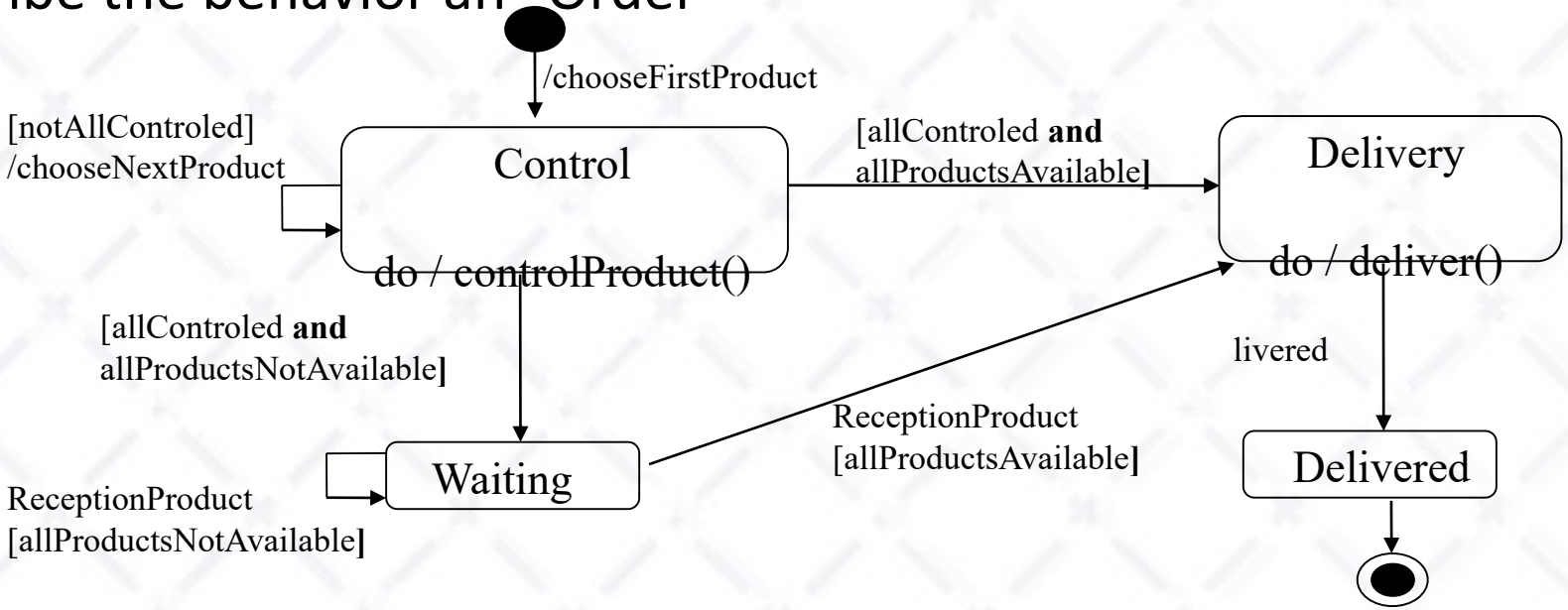
State diagrams

- Three **special events** associated with state transition
  - **entry**: allows to specify an action to be performed when entering the state
  - **exit**: allows to specify an action to be performed when going out of a state
  - **do**: allows to specify an action to be performed while the system/object is in the state
- Example



State diagrams

- Example
  - Describe the behavior an “Order”

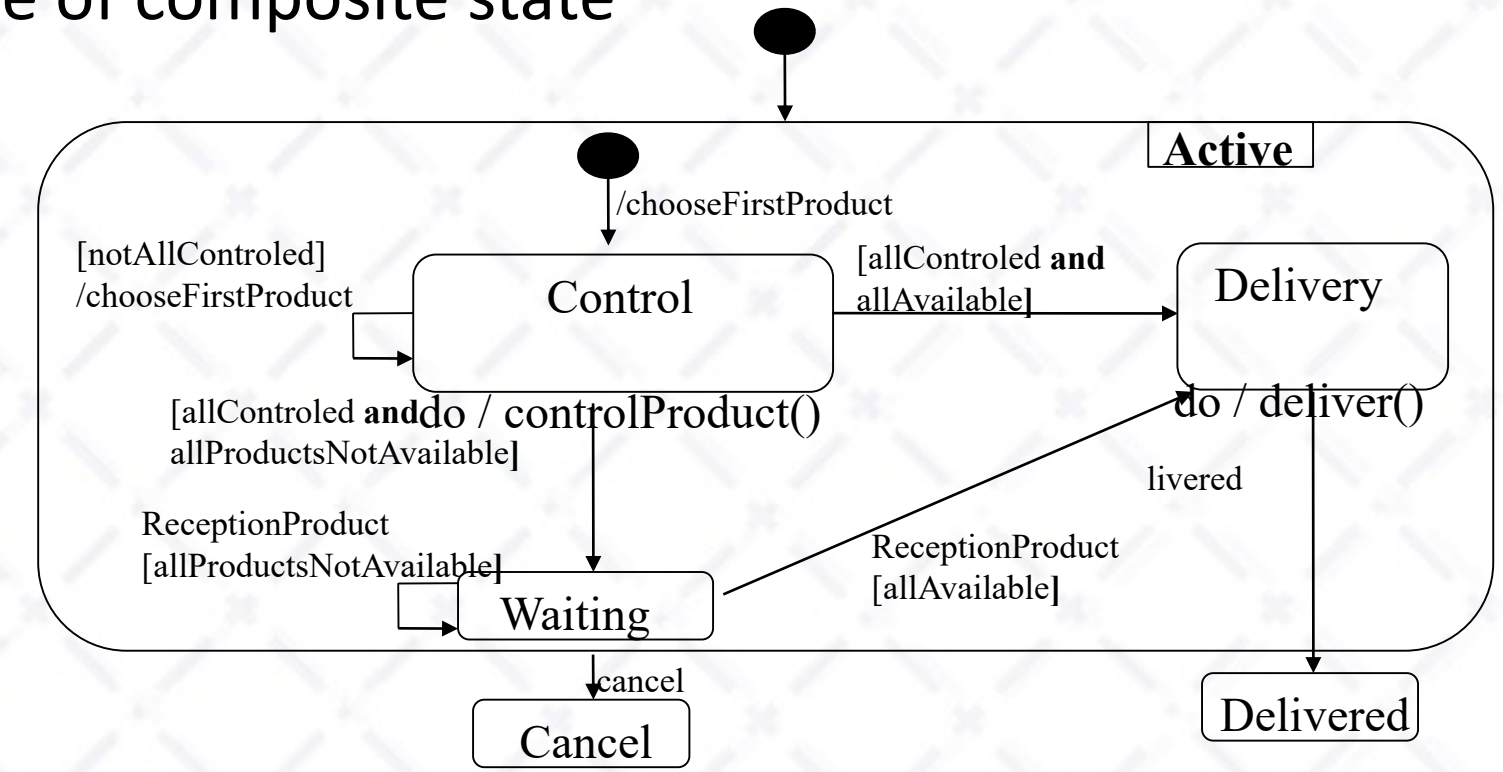


State diagrams

- Composite state
  - Several states and the transitions between these states can be combined into a composite state
  - Principles
    - The composite state has an initial state
    - The **transition to the composite state** is immediately followed by its initial state
    - The **transition from the composite state** may be originated from any of its belonging states

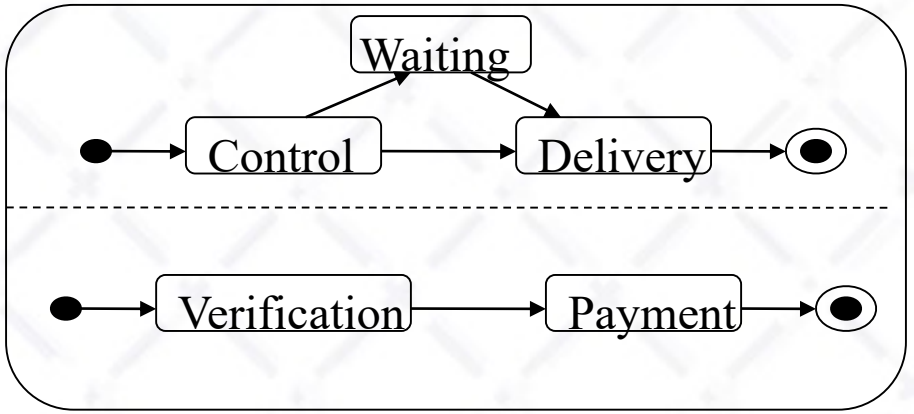
State diagrams

- Example of composite state



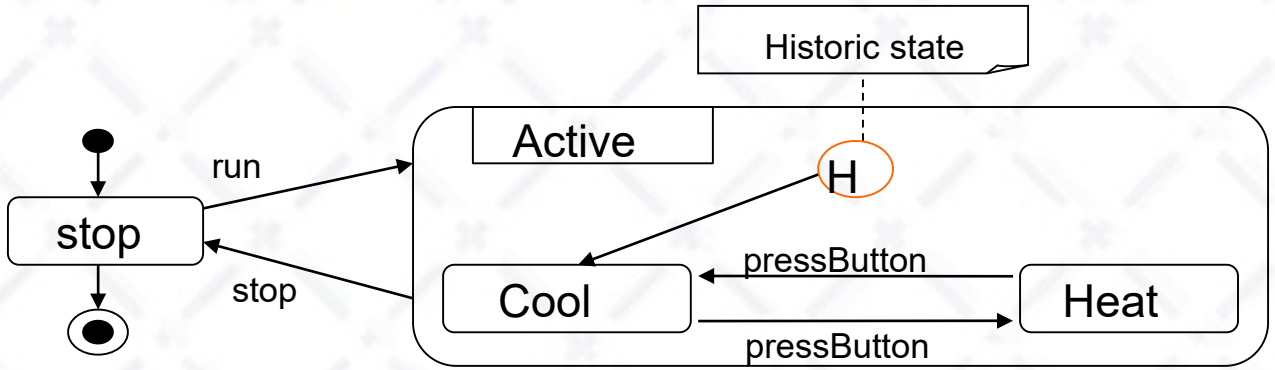
State diagrams

- Parallelism
  - Defining concurrent state within a composite state
  - Several states may exist simultaneously within a composite state
  - Example
    - Simultaneous processing of an order and its payment



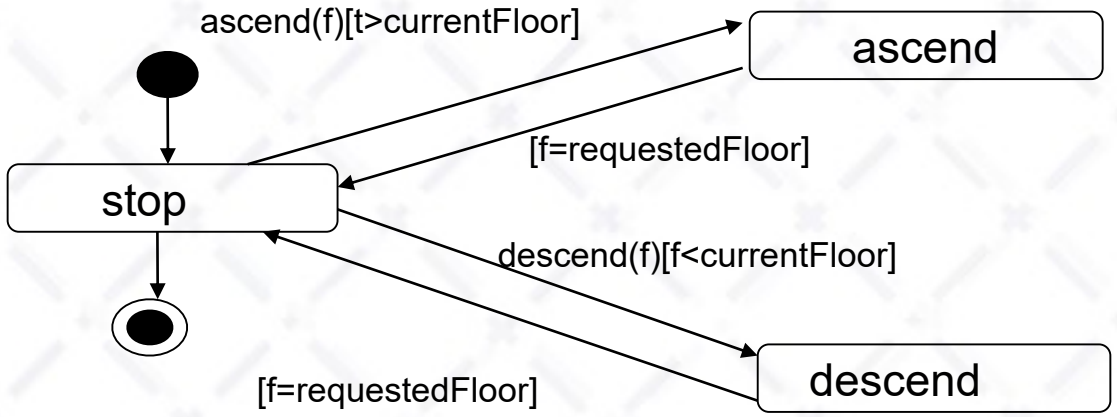
State diagrams

- Historic state
  - Allowing to memorize the current state when exiting a composite state



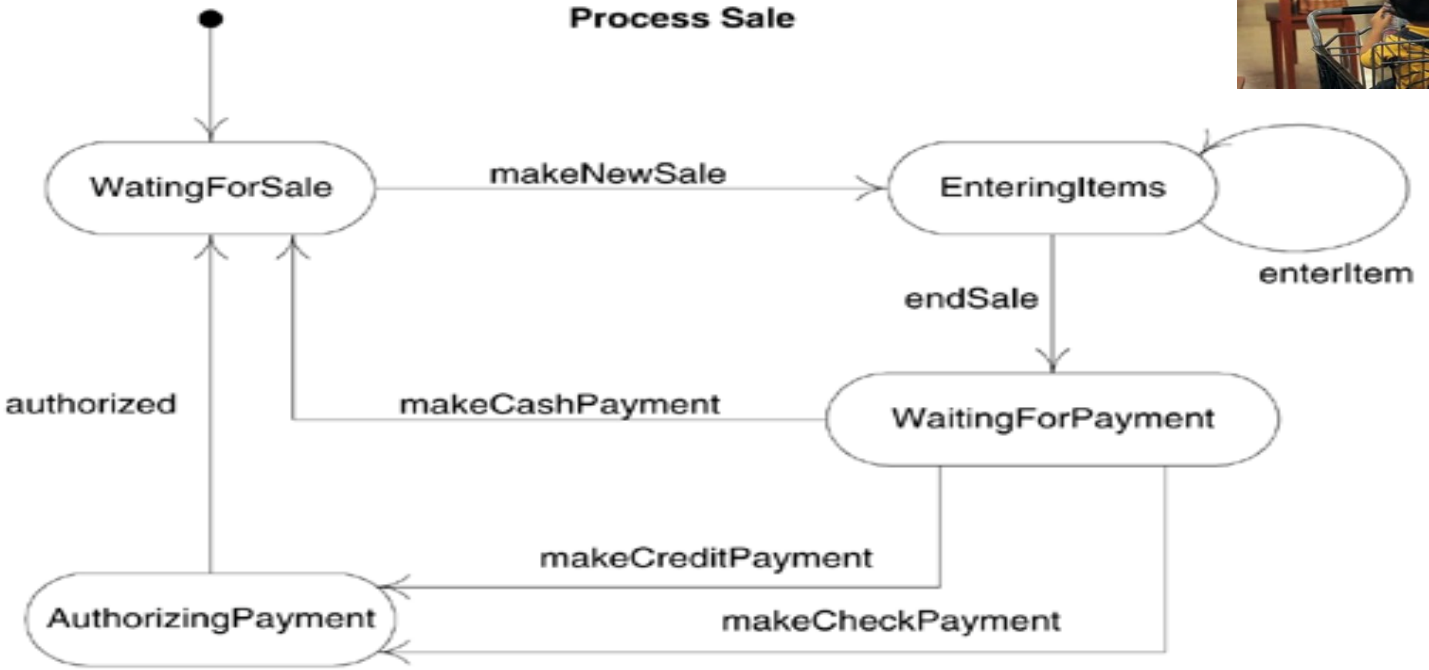
State diagrams

- Example
  - modeling an elevator's states



State diagram

- Example
  - Modelling the cash register system



Interaction diagrams

- Interaction diagrams are used to model the dynamic aspects of the system
  - An **interaction diagram is associated with a task** performed by the system or its components
  - Interaction diagrams are determined/built based on activity diagrams and use-case diagrams
  - An interaction diagram generally corresponds to **a use-case** or a functionality
  - The interaction diagram shows how objects and actors communicate together to achieve the task
- Specifically, an interaction diagram allows to describe in detail **the algorithms** in the system
- Interaction diagrams can be subsequently used in **the implementation of class methods**

Interaction diagrams

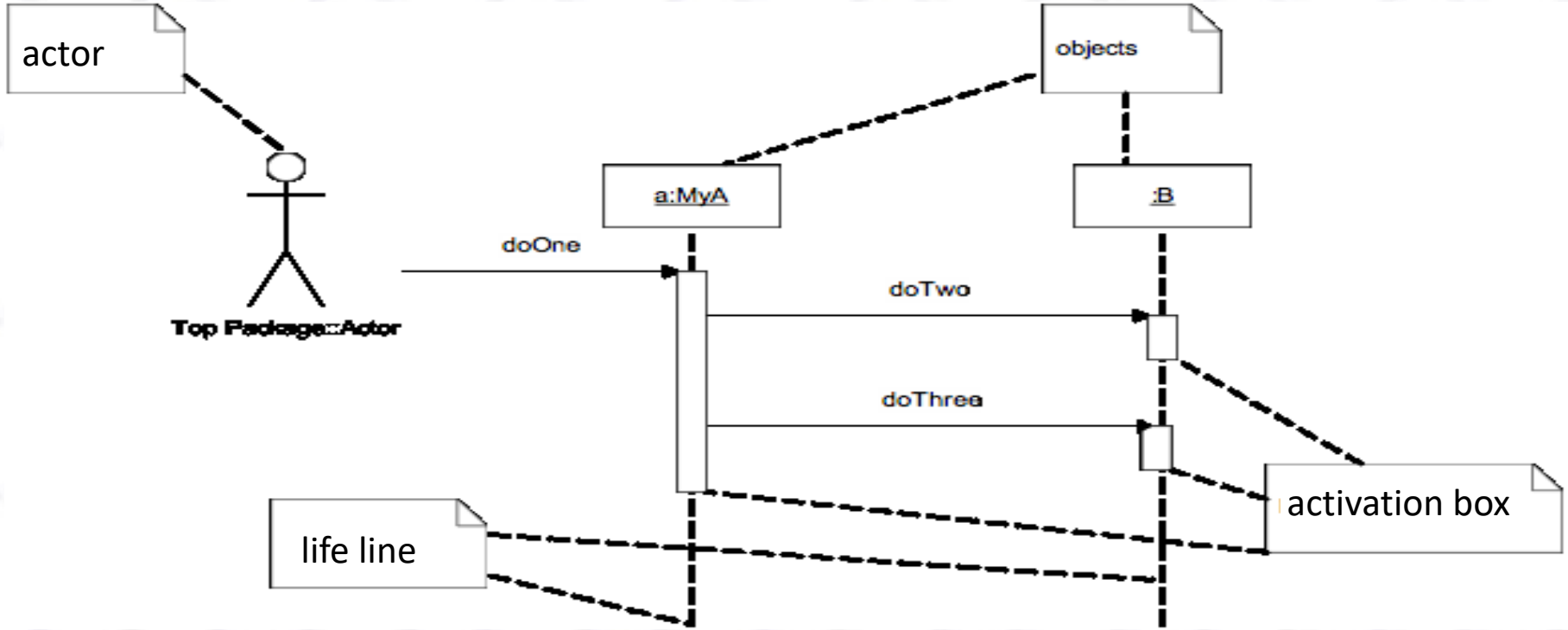
- The essential elements of an interaction diagrams
  - Objects
  - Actors
  - Messages
- Actions between objects and actors are
  - message sendings
  - object creations and destructions
- **Two types of interaction diagrams**
  - **Sequence diagrams**
    - The temporal sequence of interactions
  - **Communication diagrams**
    - An instance of class diagram

Sequence diagram

- A **sequence diagram** describes the **temporal sequence** of exchanges of messages between objects and the actor to perform a certain task
  - The **actor** who initiates interactions is usually found on the far left
  - The **objects** are placed horizontally on the diagram
  - The vertical dimension represents time
  - Each object or actor is associated with a **life line** representing the time where the object or actor is
  - An **activation box** represents the object activation period

Sequence diagram

- Notation



Sequence diagram

- Messages
  - Message is the medium of communication between objects
  - The general form of message

[guard]message(parameters)

  - **guard**: a condition must be satisfied in order to send the message
  - **message**: the identifier of the sent message
  - **paramaters**: a list of parameter values

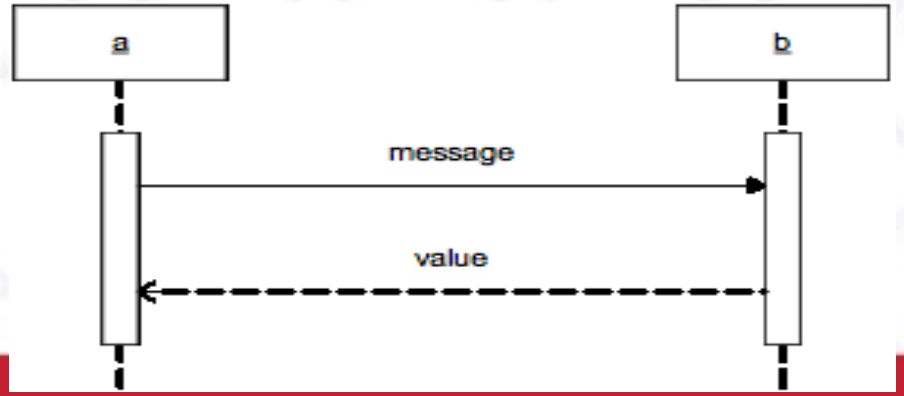
  - Note: guard and parameters can be omitted

Sequence diagram

- The return values
  - Sending a message to an object cause **the execution of a method** of this object
    - This method can optionally return a value
  - The return values may be omitted or be explicitly described

[guard]value := message(parameters)

    - either as the following form
    - or by a return message that represents graphically

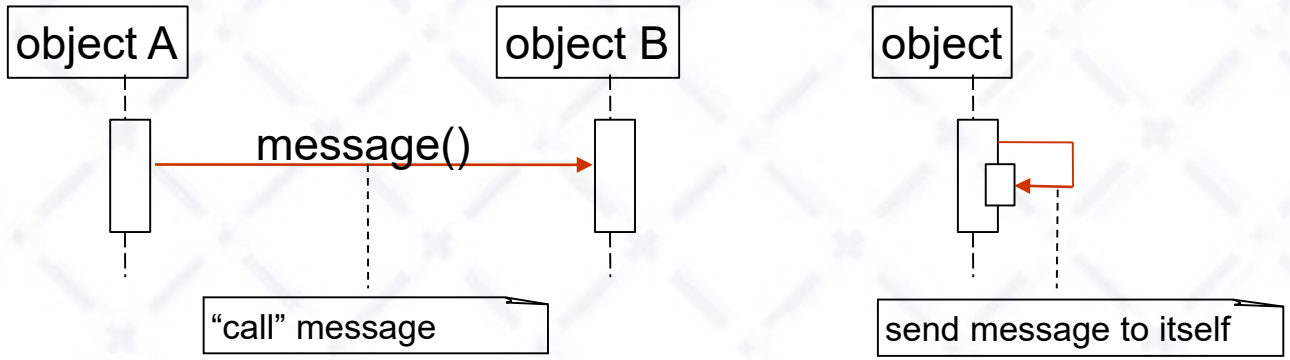


Sequence diagram

- Types of message
  - “call” message
  - “return” message
  - “send” message
  - “create” message
  - “destroy” message

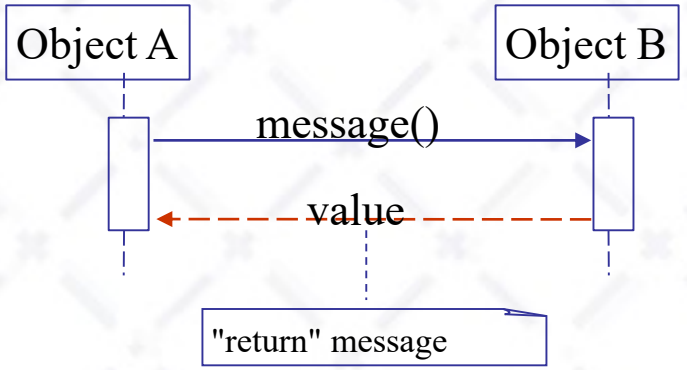
Sequence diagram

- "call" message
  - A "call" message invokes an operation/method of the object
  - A "call" message is a **synchronous message**: the object that sends the message must wait for the termination of the execution of the message before doing other tasks
  - An object can send message to itself
  - Notation



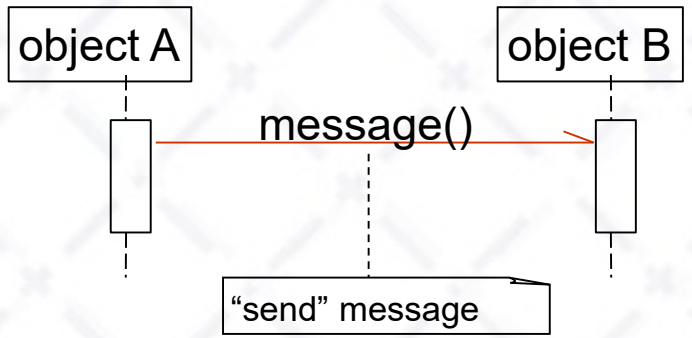
Sequence diagram

- The “return” message returns a value for the calling object
- Notation



Sequence diagram

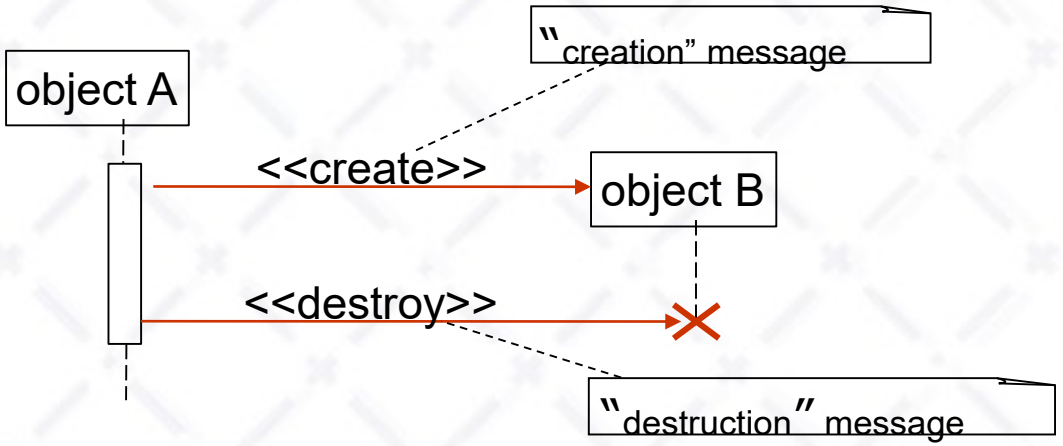
- "send" message
  - A "send" message sends a signal to an object
  - A "send" message is an **asynchronous message**: once the object sends the message, it expects nothing and continues to do other tasks
  - Notation



  - Asynchronous message is often used in multi-threaded environment
    - For example, *Thread.start(), Runnable.run()* in Java

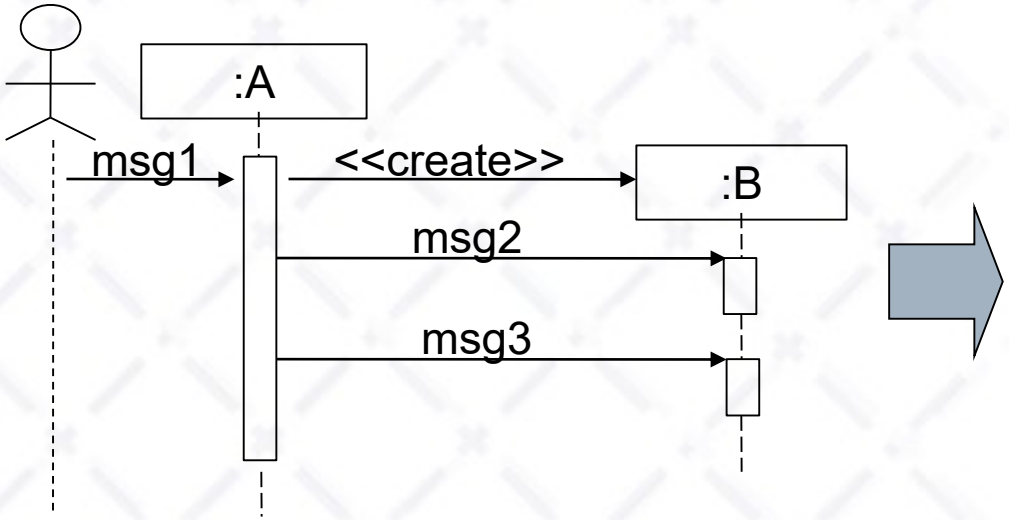
Sequence diagram

- "creation" message
  - invokes the creation method of object (constructor)
- "destruction" message
  - invokes the destruction message of message (destructor)
- Notation



Sequence diagram

- Example
  - The sequence diagram and the corresponding code

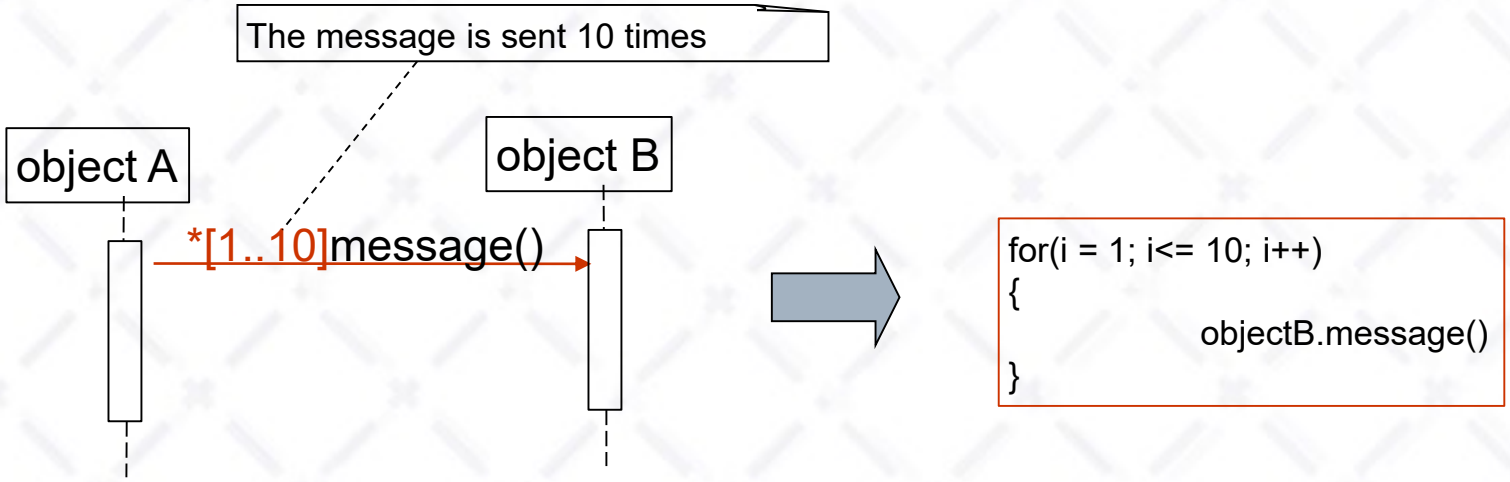



```
public class A
{
  private B objB;
  public void msg1()
  {
     objB = new B();
     objB.msg2();
     objB.msg3();
  }
}

public class B
{
  …
   public void msg2() { … }
   public void msg3() { … }
}
```

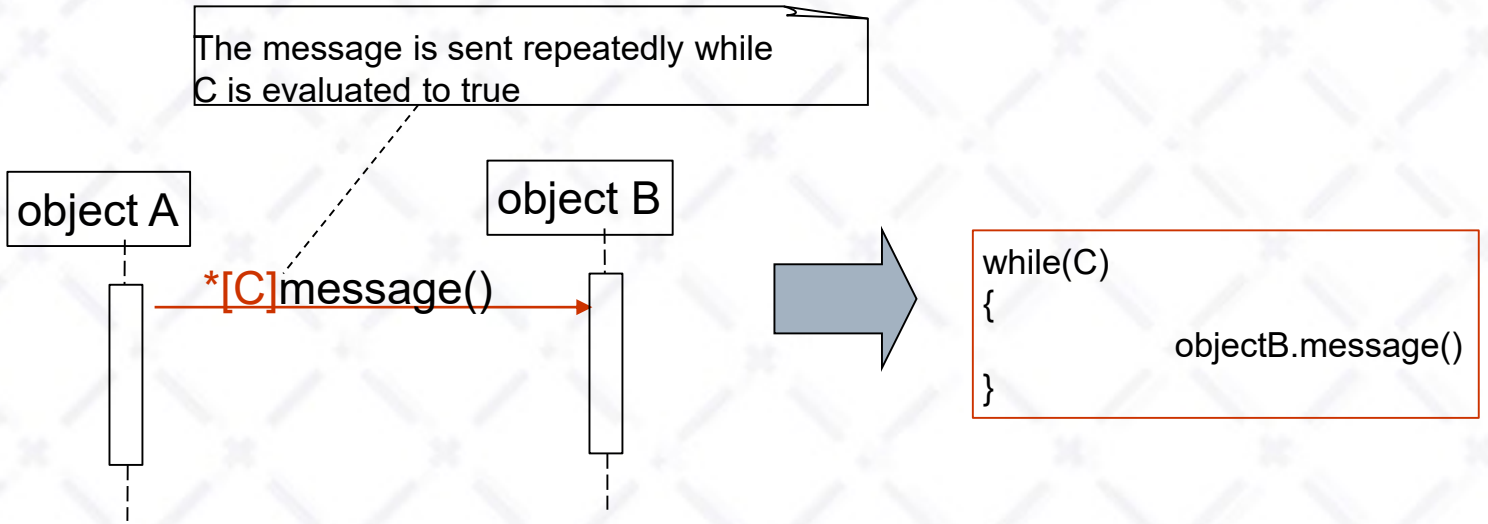
Sequence diagram

- A message can be **sent iteratively**
- Example



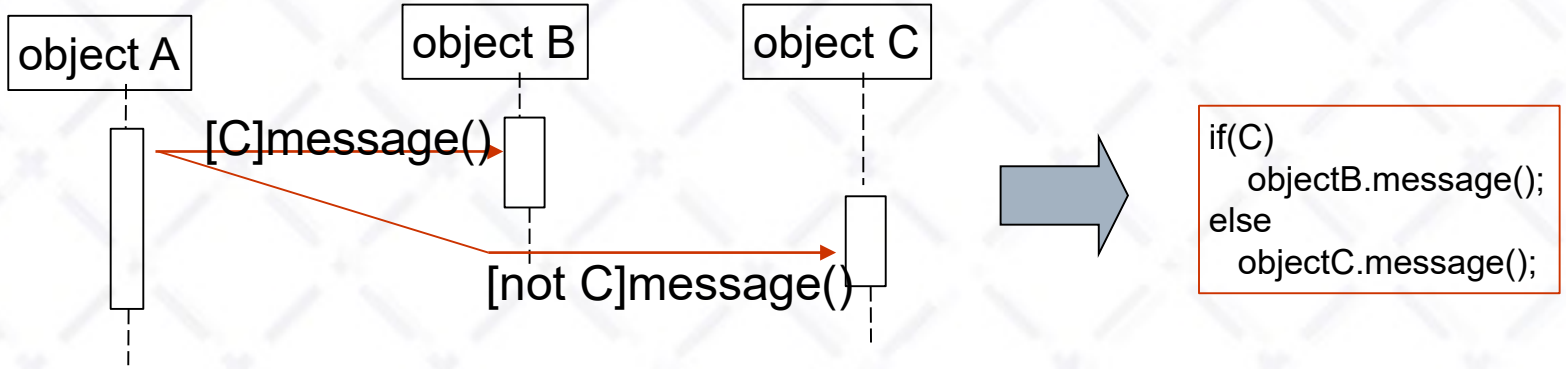
Sequence diagram

- A message can be sent iteratively based on a condition
- Example



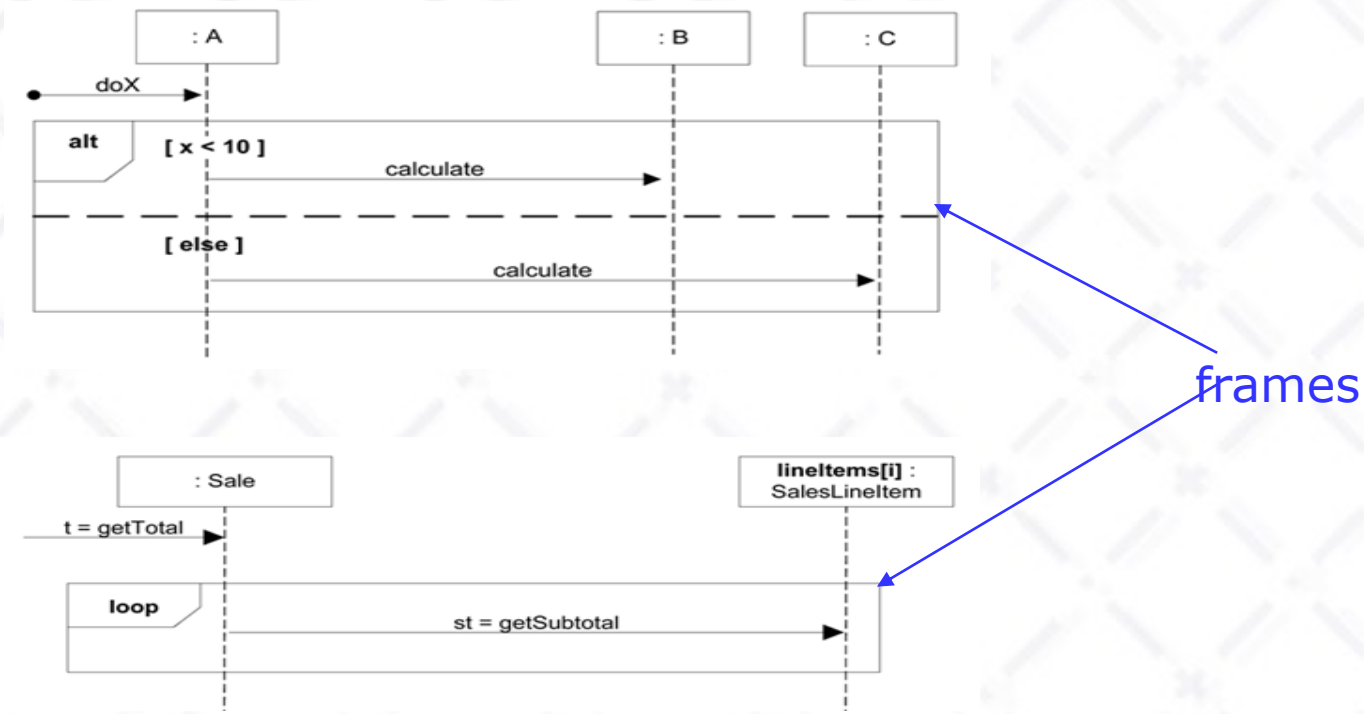
Sequence diagram

- The sending of a message can depend on a **decision**
- Example



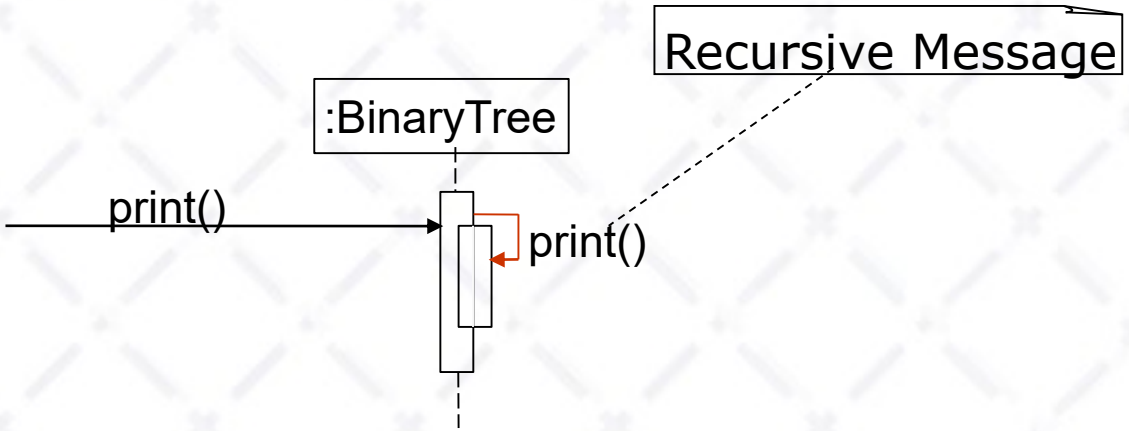
Sequence diagram

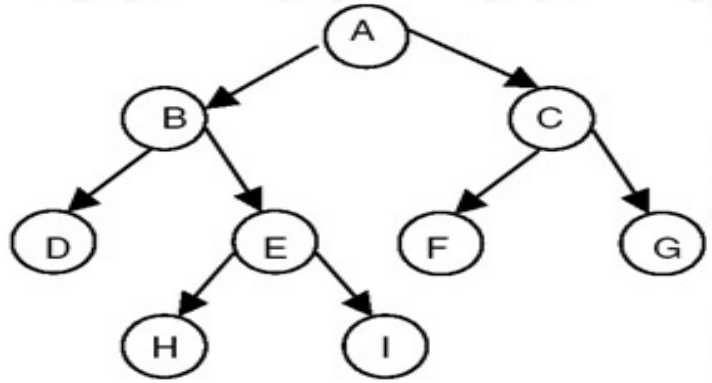
- Note: UML 2.x notations allow the use of frames to represent the conditions or iterations



Sequence diagram

- A message can be called **recursively**
- Notation

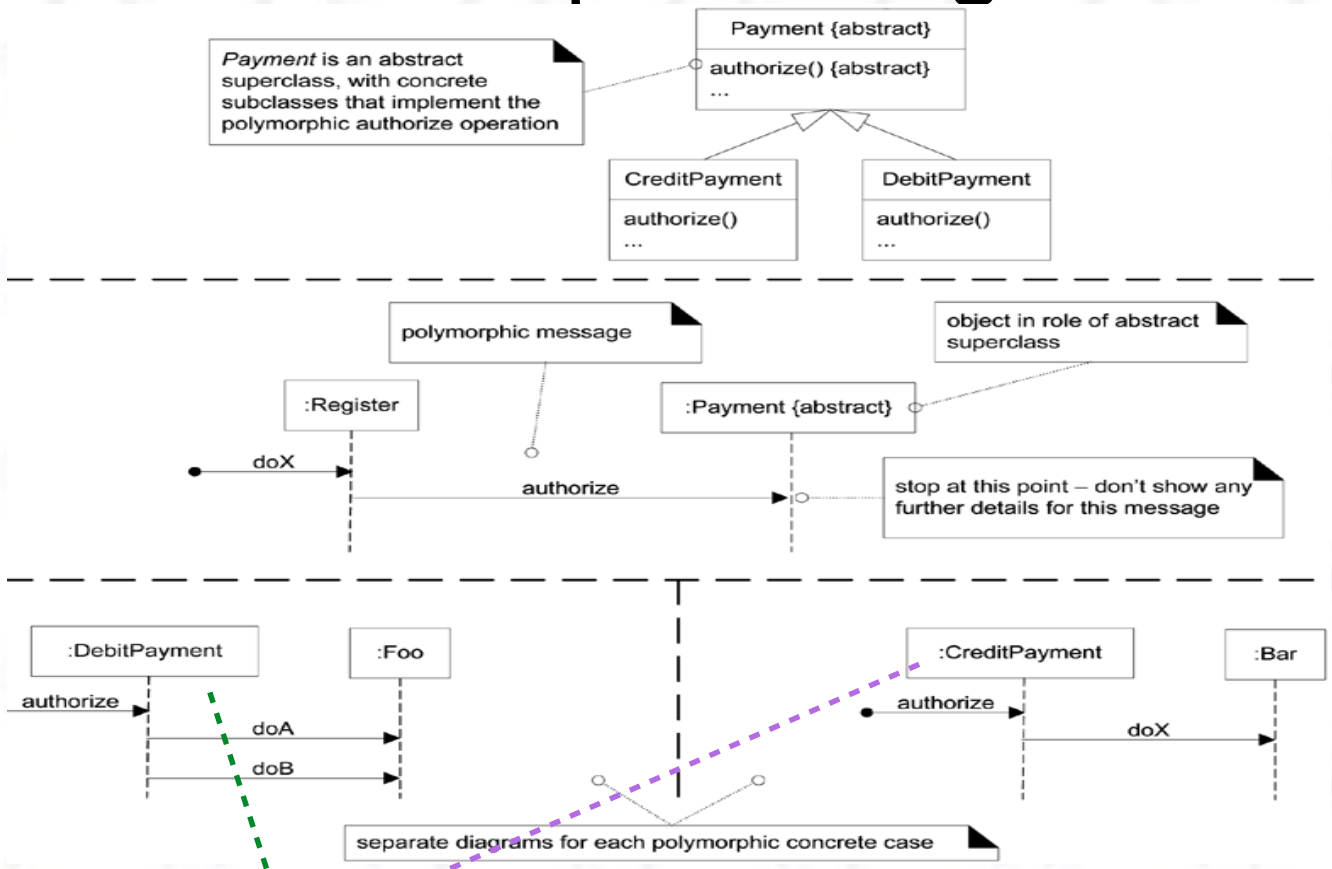


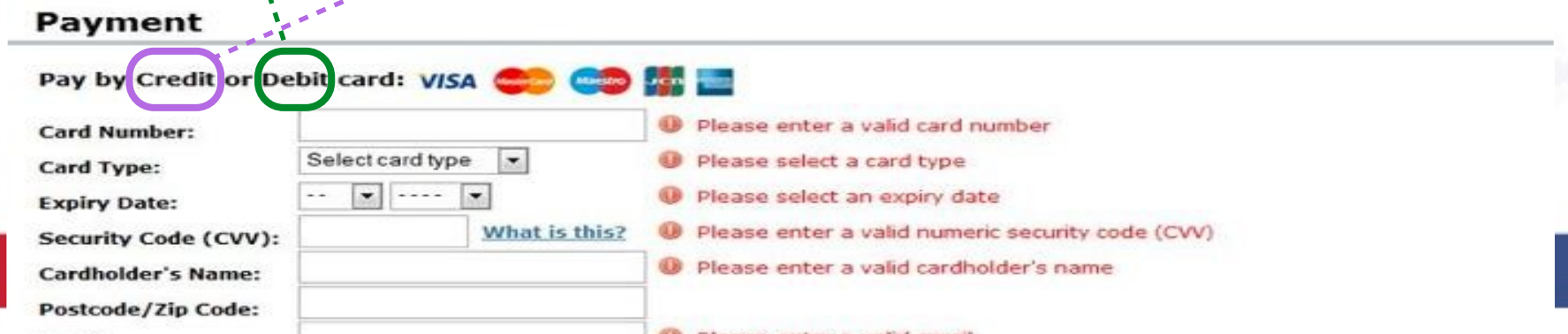


Inorder : DBHEIAFCG
Preorder : ABDEHICFG
Postorder : DHIEBFGCA

Sequence diagram

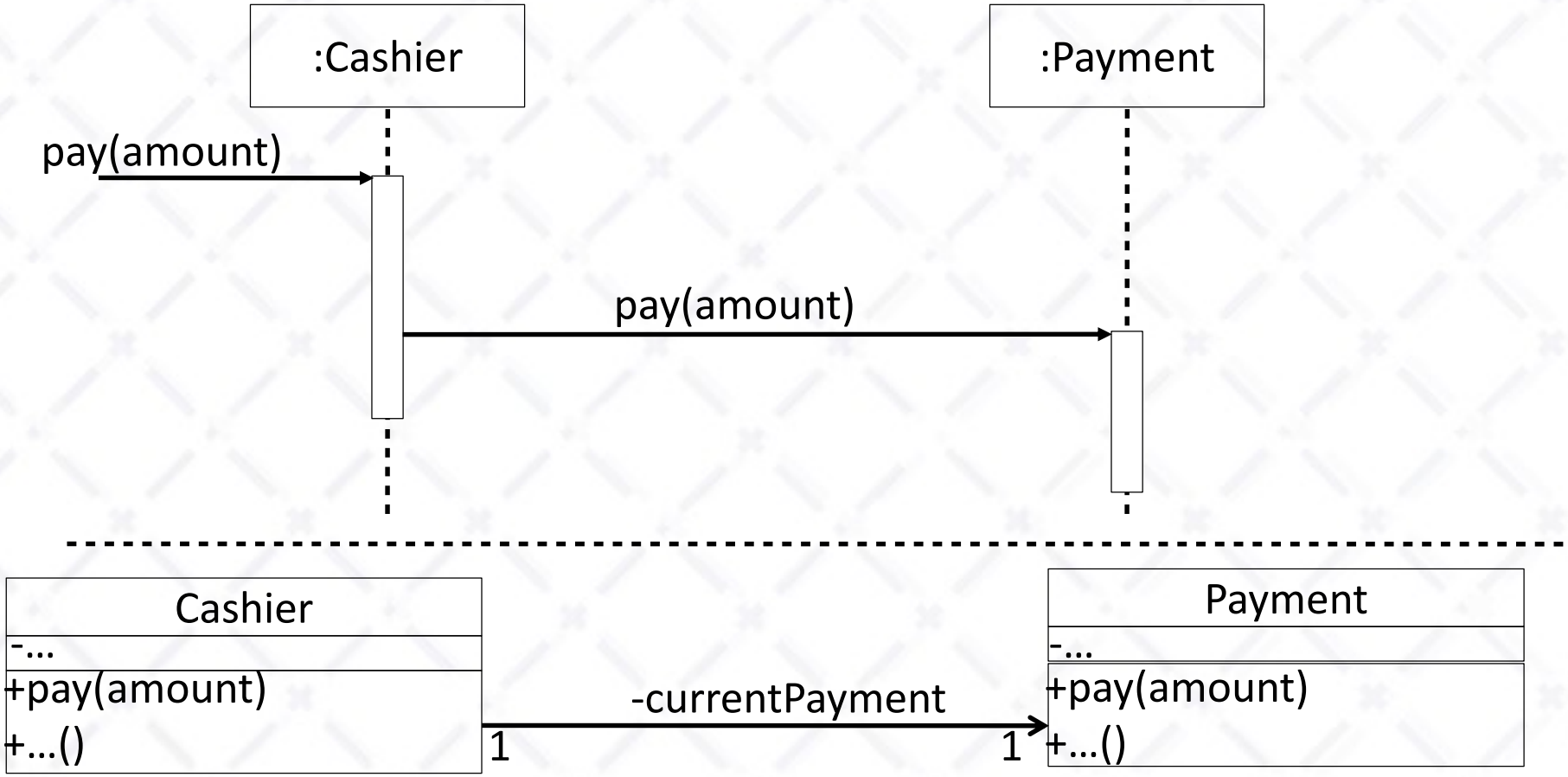
modeling a polymorphic message





Sequence diagram

- Relationship between class diagram and sequence diagram



Sequence diagram from use-case

- Boudary class
- Entity class
- Control class

Sequence diagram from use-case

Basic Course

The Customer specifies an author on the Search Page and then presses the Search button.

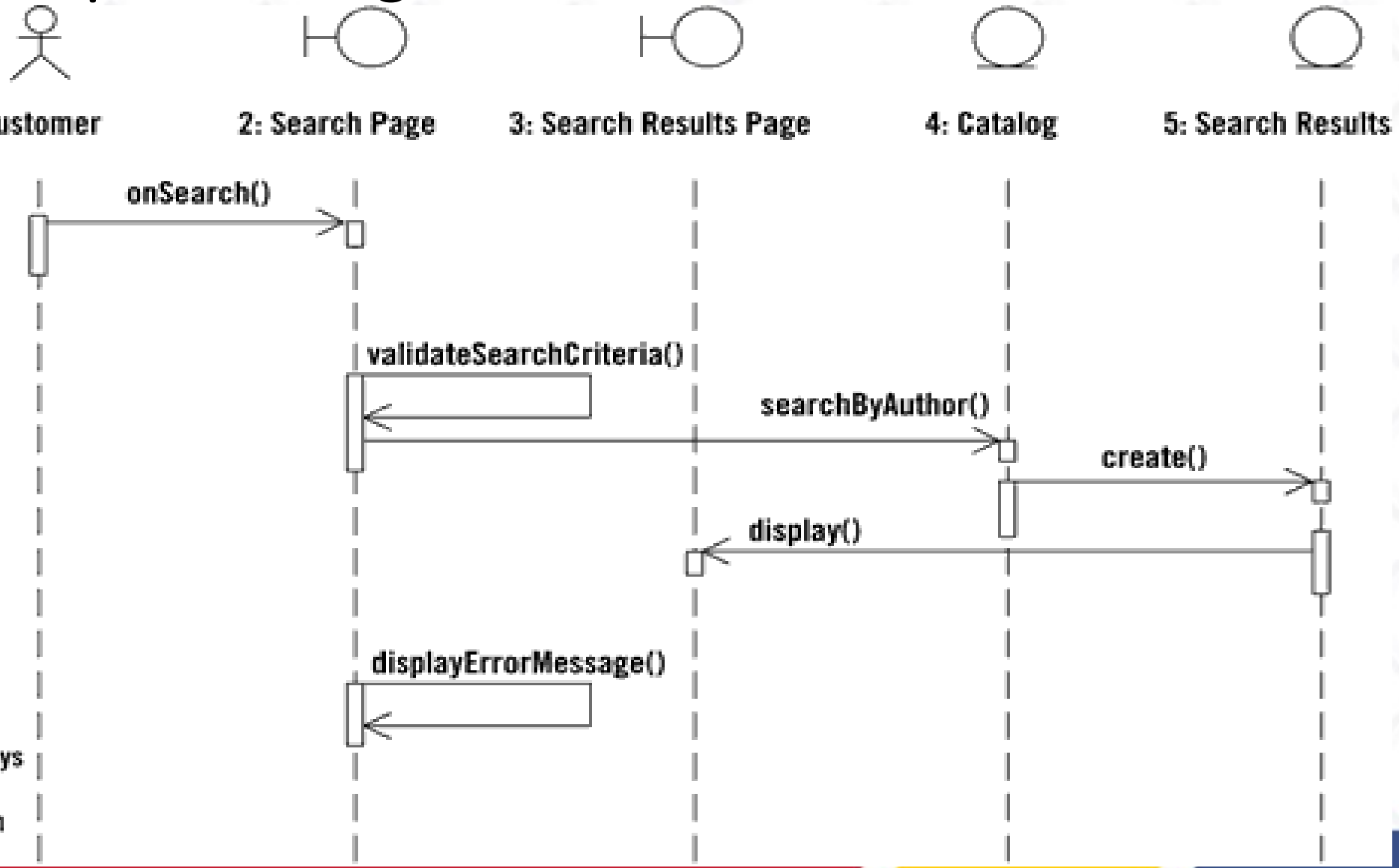
The system validates the Customer's search criteria.

The system searches the Catalog for books associated with the specified author.

When the search is complete, the system displays the search results on the Search Results Page.

Alternate Course

If the Customer did not enter the name of an author before pressing the Search button, the system displays an error message to that effect and prompts the Customer to re-enter an author name.

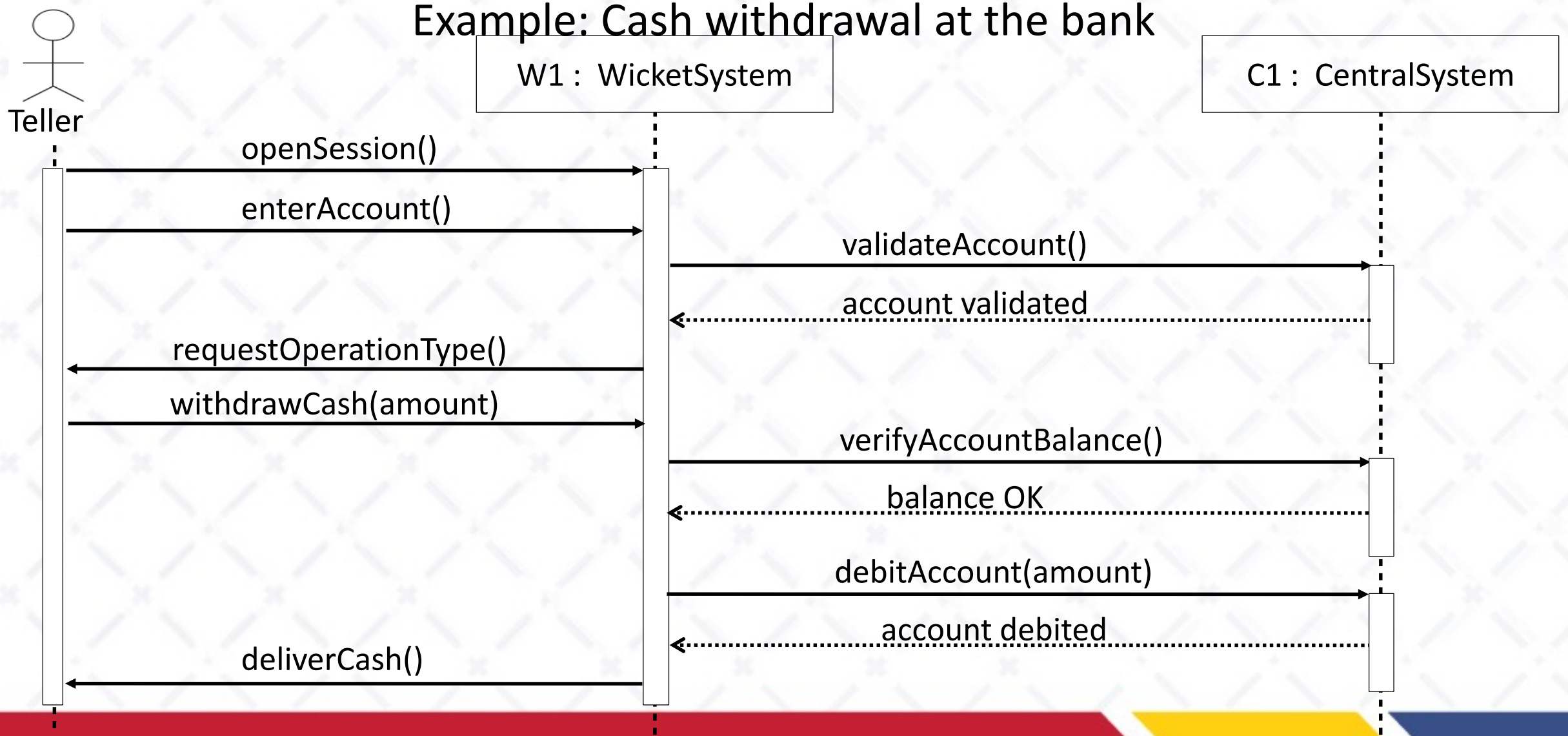


Sequence diagram

- Example: Cash withdrawal at the bank

Sequence diagram

Example: Cash withdrawal at the bank

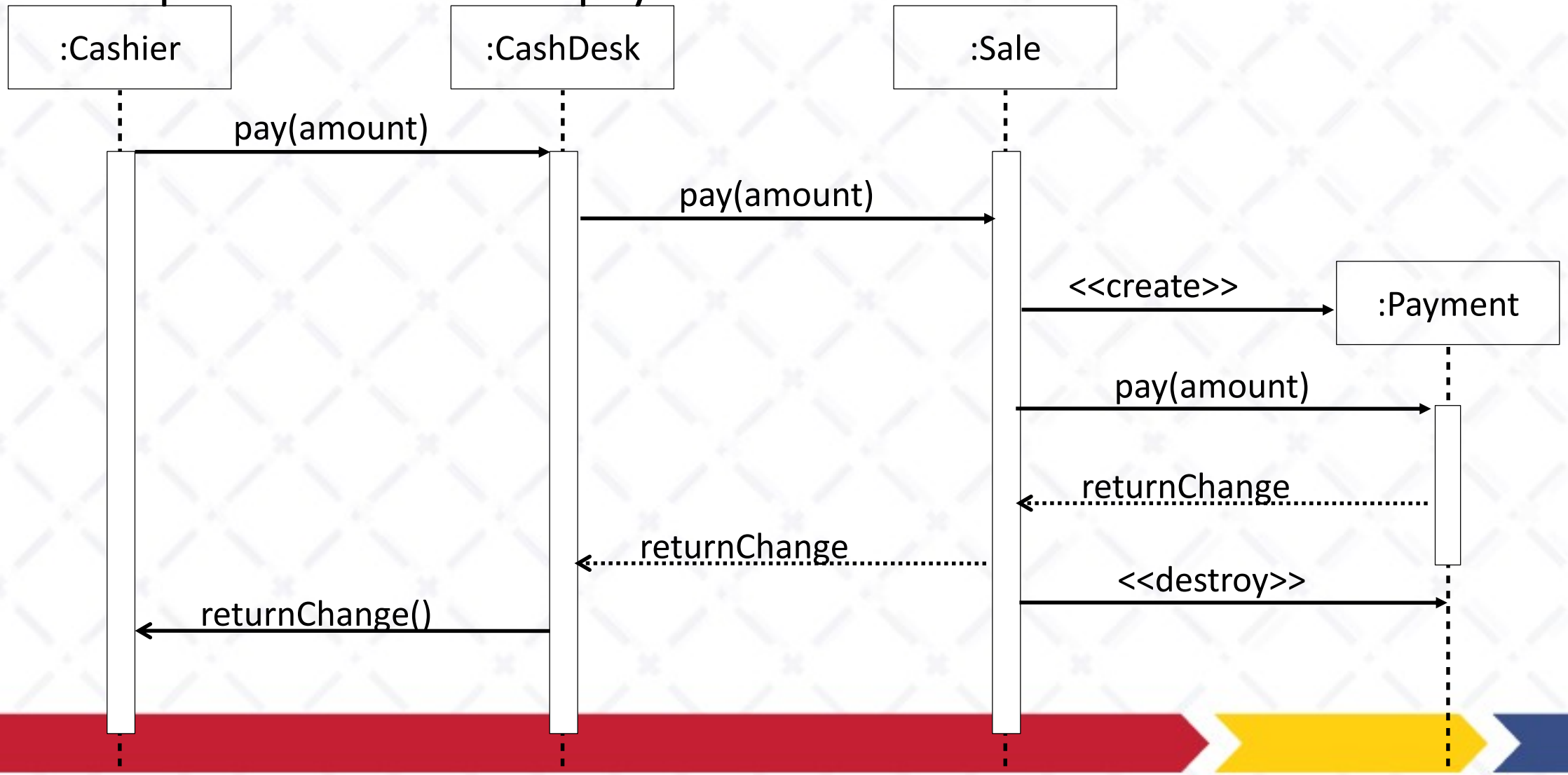


Sequence diagram

- Example: Use-case “cash payment”

Sequence diagram

- Example: Use-case “cash payment”



Why not just code it?

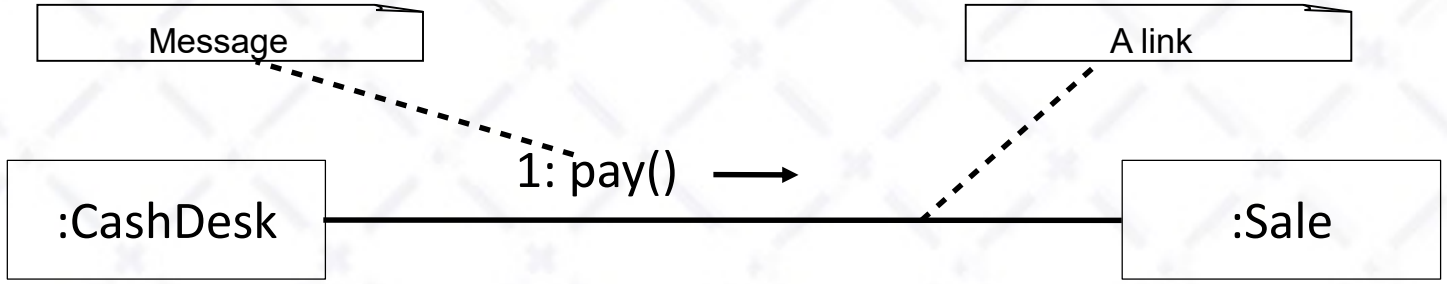
- Sequence diagrams can be somewhat close to the code level. So why not just code that algorithm rather than drawing it as a sequence diagram?
  - a good sequence diagram is still a bit above the level of the real code (not EVERY line of code is drawn on the diagram)
  - sequence diagrams are language-agnostic (can be implemented in many different languages)
  - non-coders can do sequence diagrams
  - easier to do sequence diagrams as a team
  - can see many objects/classes at a time on same page (visual bandwidth)

Communication diagram

- A Communication diagram describes the interaction between objects
  - A Communication diagram is a graph whose
    - nodes represent object
    - edges represent the communication between objects
  - The temporal ordering of messages is represented by a **numbering** of messages
  - Communication diagram is an extension of class diagram

Communication diagram

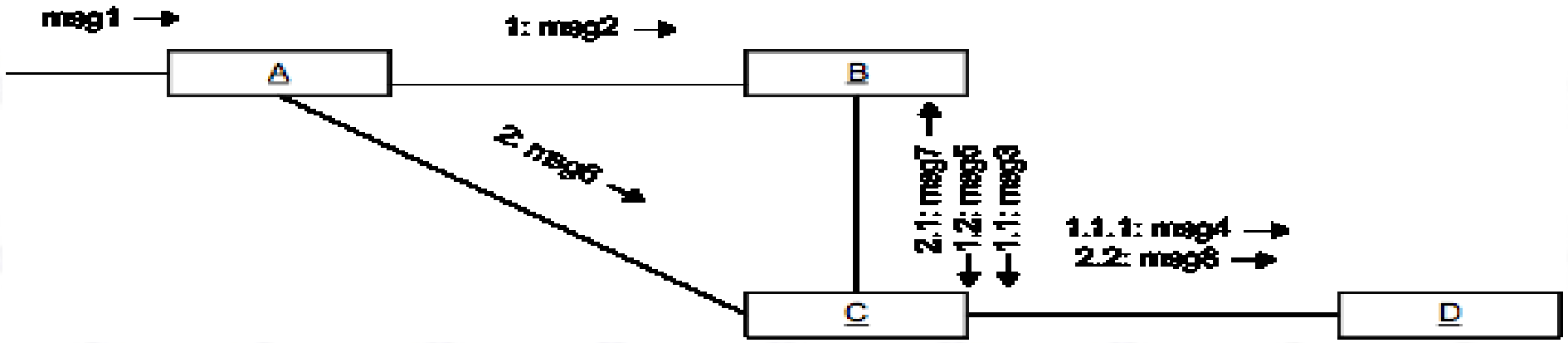
- Links
  - A link shows the sending of a message from an object to another object
  - Formally, a link is an instance of an association
- Messages
  - Each message between objects is presented by an expression of message and an arrow showing the direction of the message



Communication diagram

- "creation" message and "destruction" message

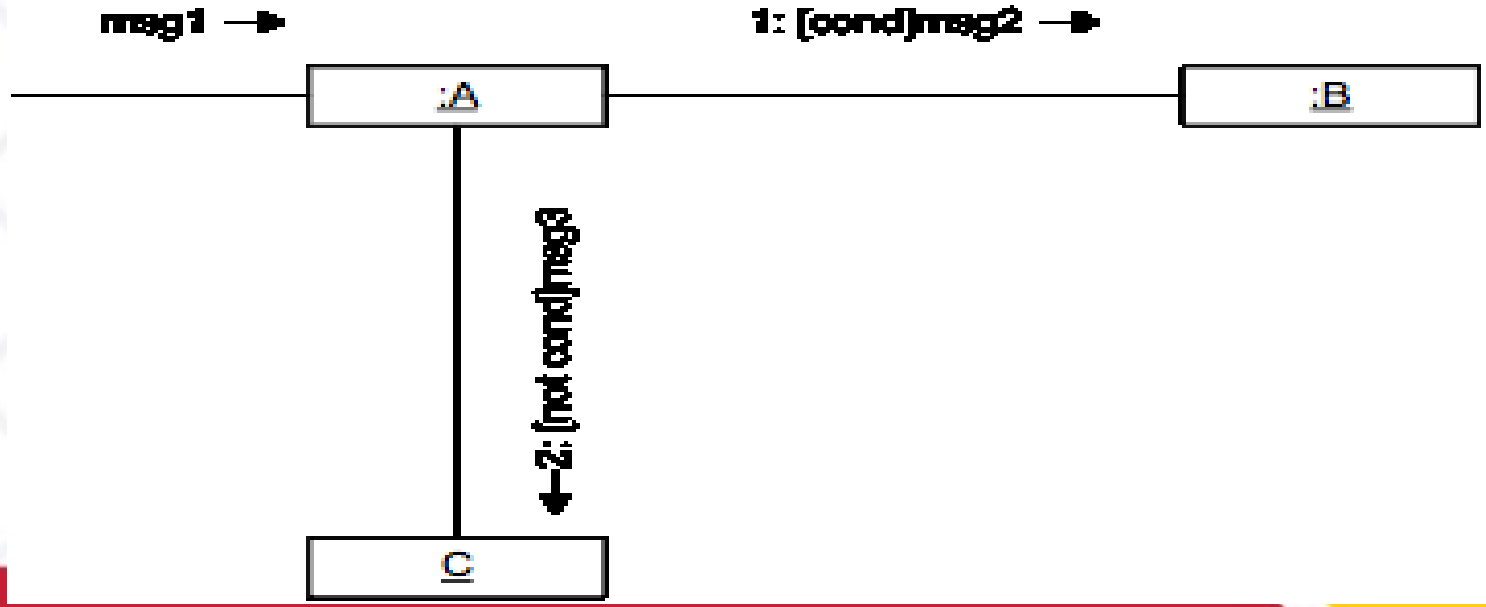
- Message numbering



Communication diagram

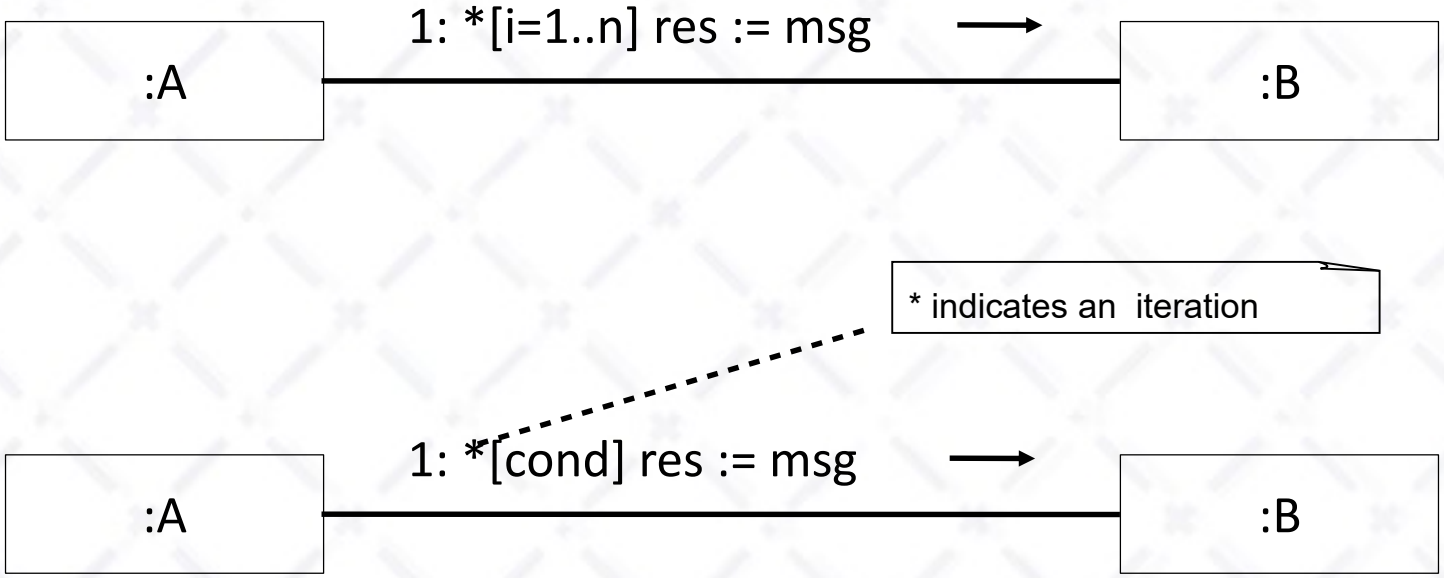
- Conditional m

- modeling a decision



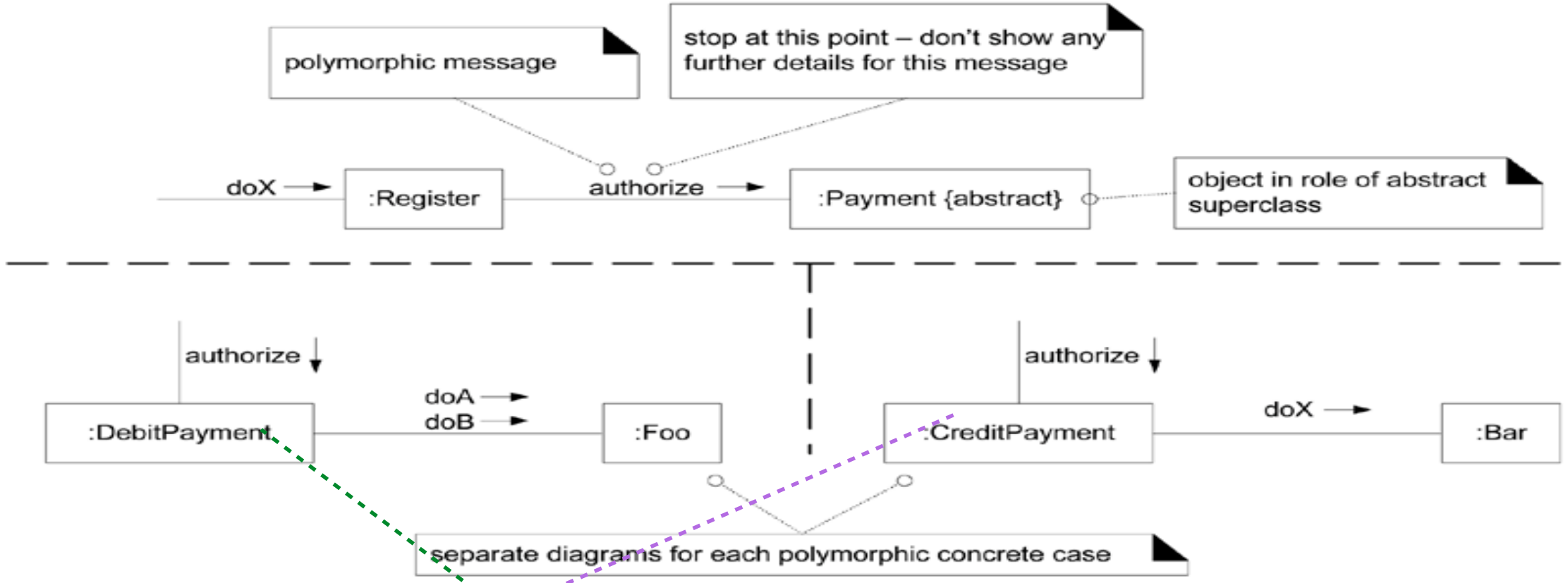
Communication diagram

- modeling an iteration



Communication diagram

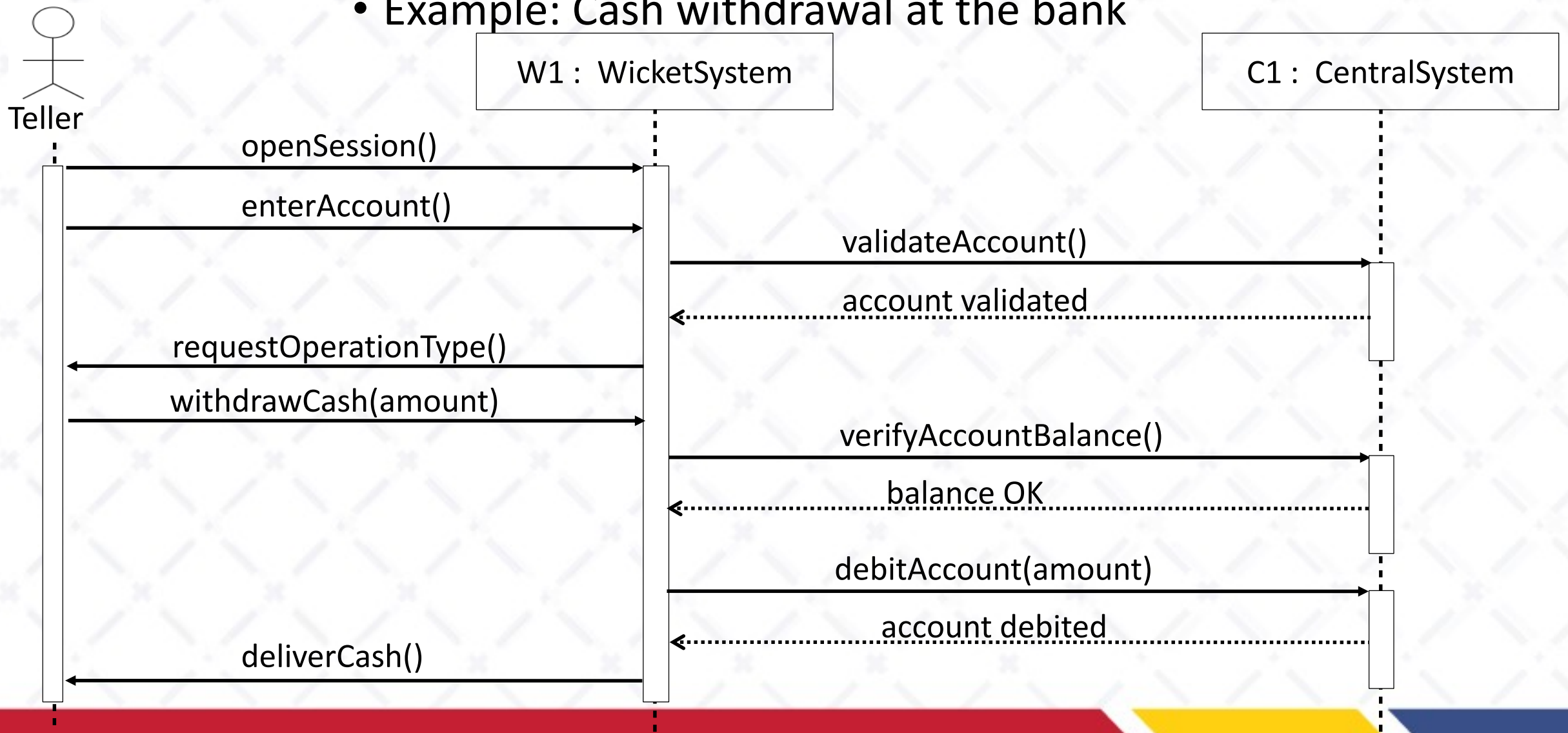
- modeling a polymorphic message



Cash withdrawal at the bank

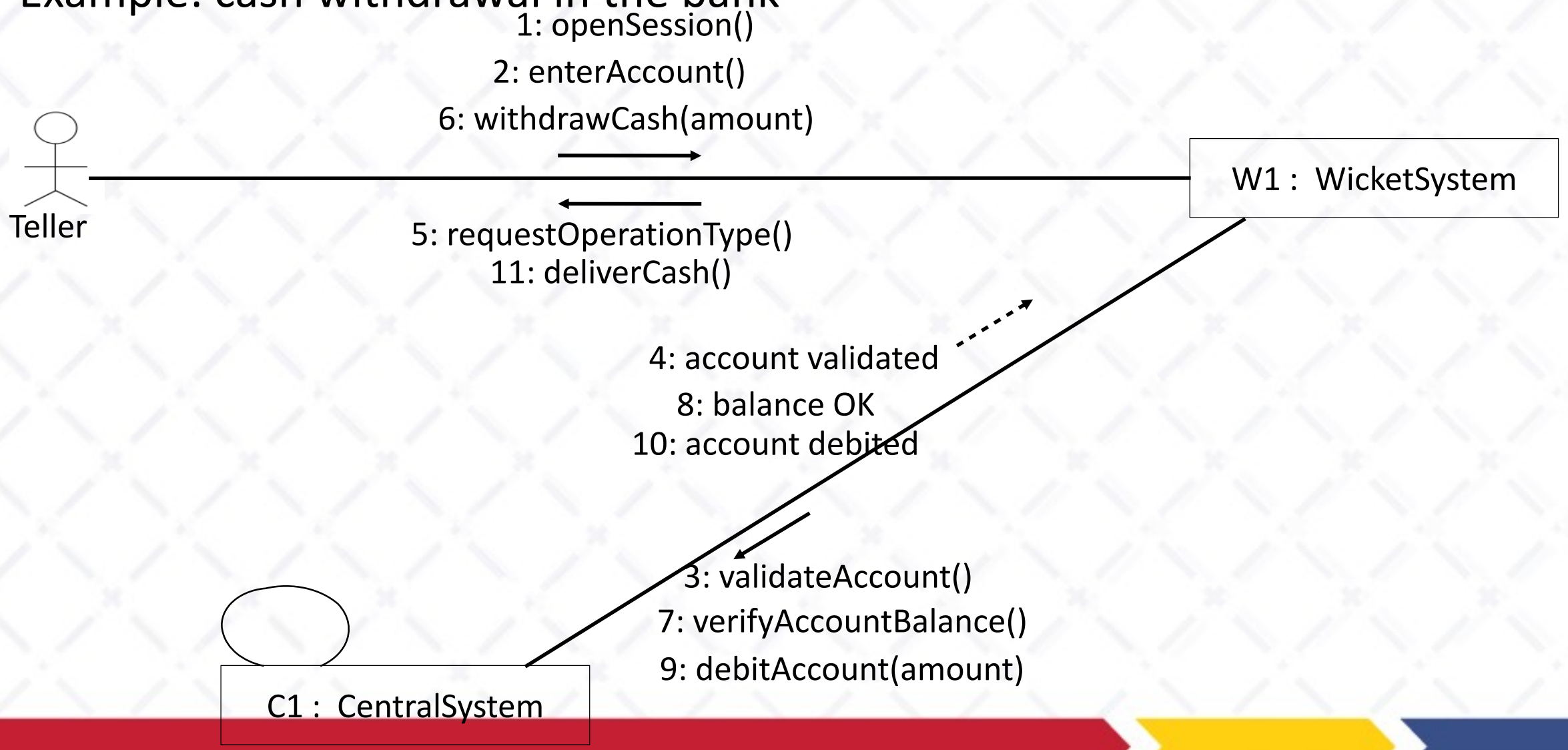
Sequence diagram

- Example: Cash withdrawal at the bank



Communication diagram

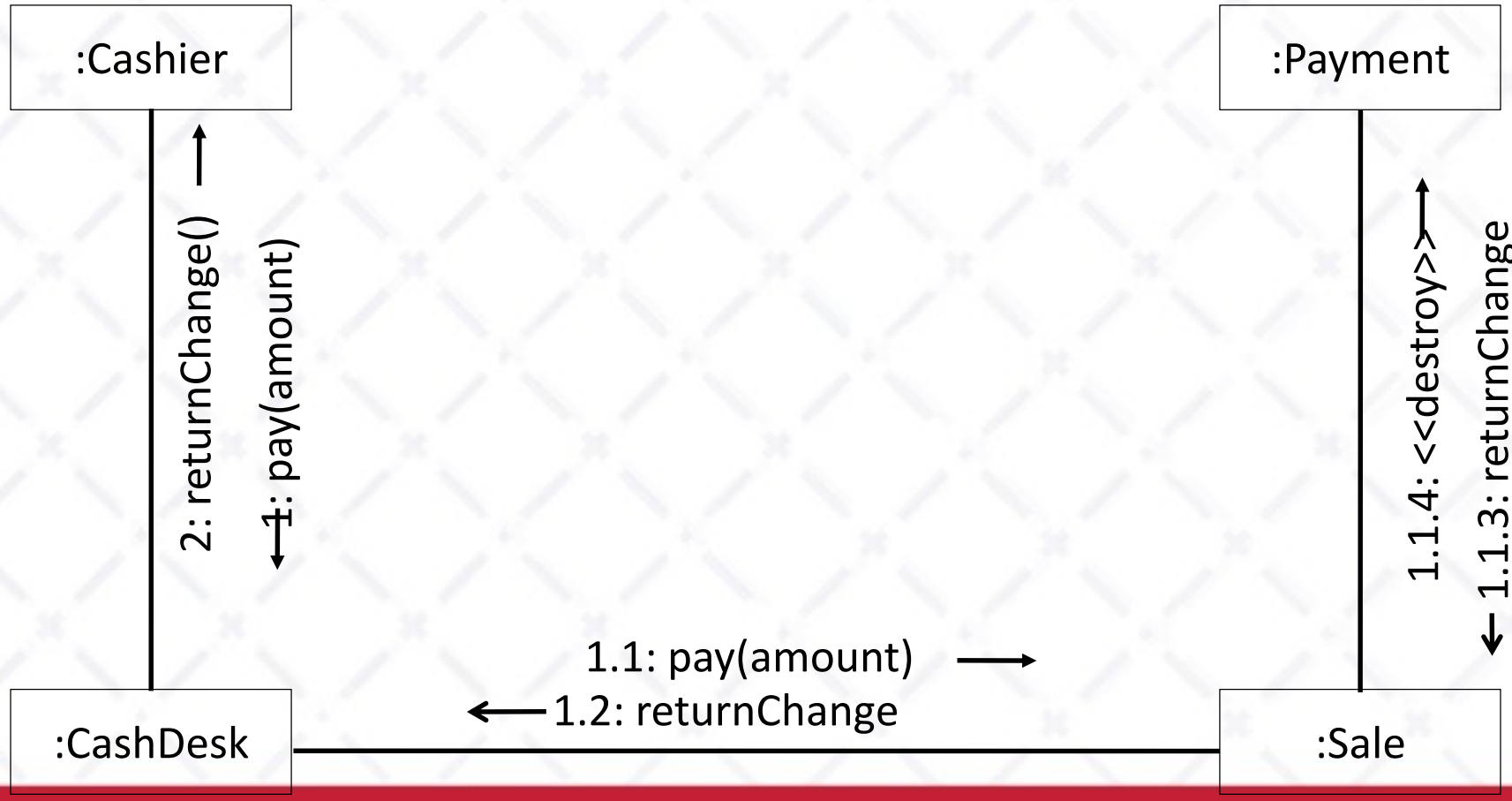
- Example: cash withdrawal in the bank



Use-case “cash payment”

Sequence diagram

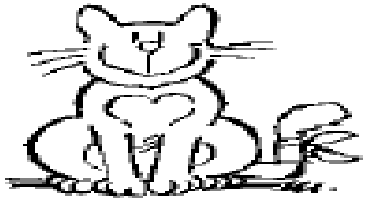
Communication diagram



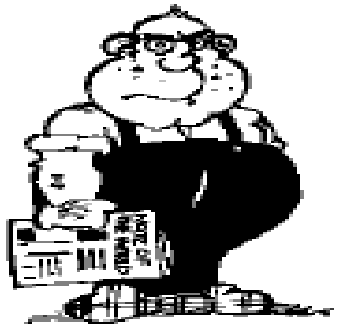
Sequence diagram v.s. Communication diagram

- Both sequence diagram and Communication diagram are alternate representations of an interaction
- Sequence diagram
  - is a graphical view of a scenario
  - shows object interaction in a time-based sequence of what happens first, what happens next
  - establishes the roles of objects and help provide essential information to determine class responsibilities and interfaces
  - is normally associated with a use-case
- Communication diagram
  - shows how object associate with each other (objects, links and messages)
  - provides the structural relationships between objects

Fun example



:Cat

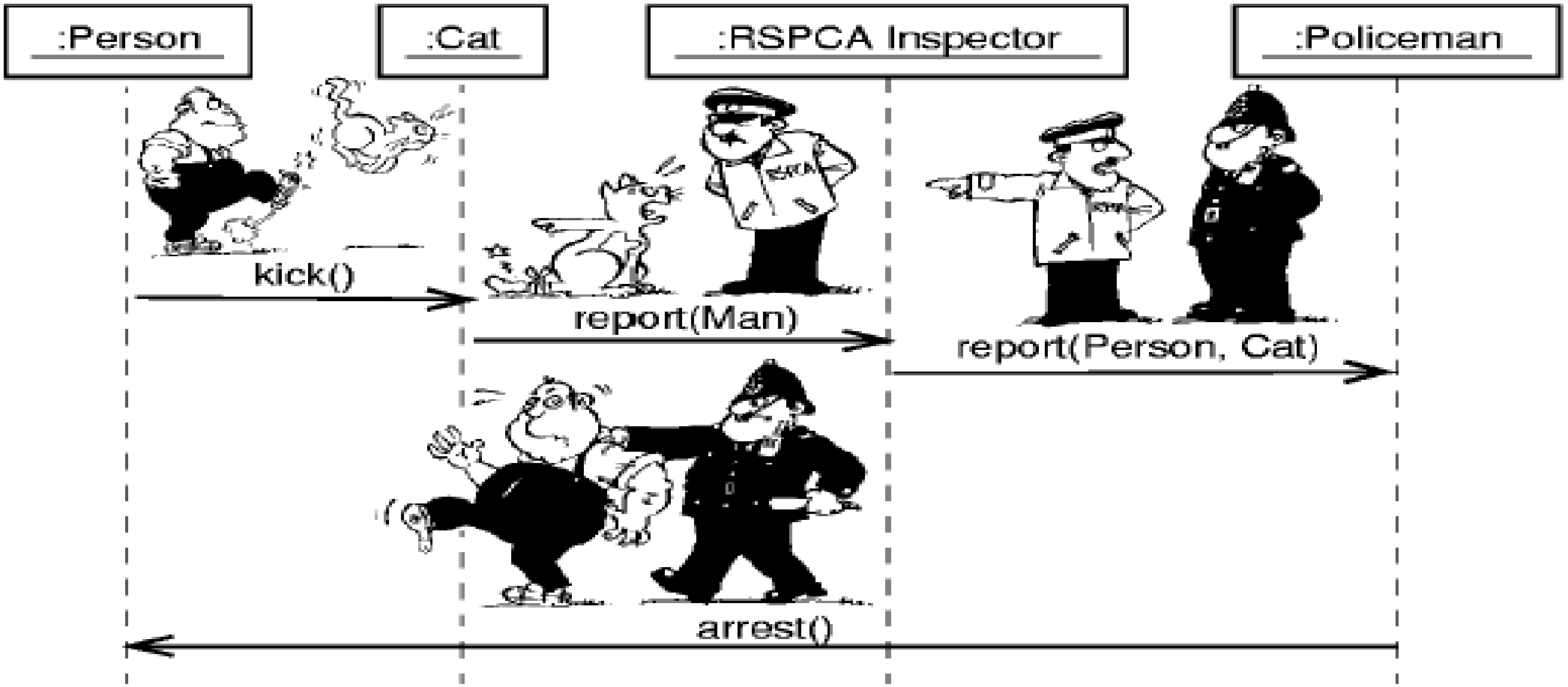


:Person

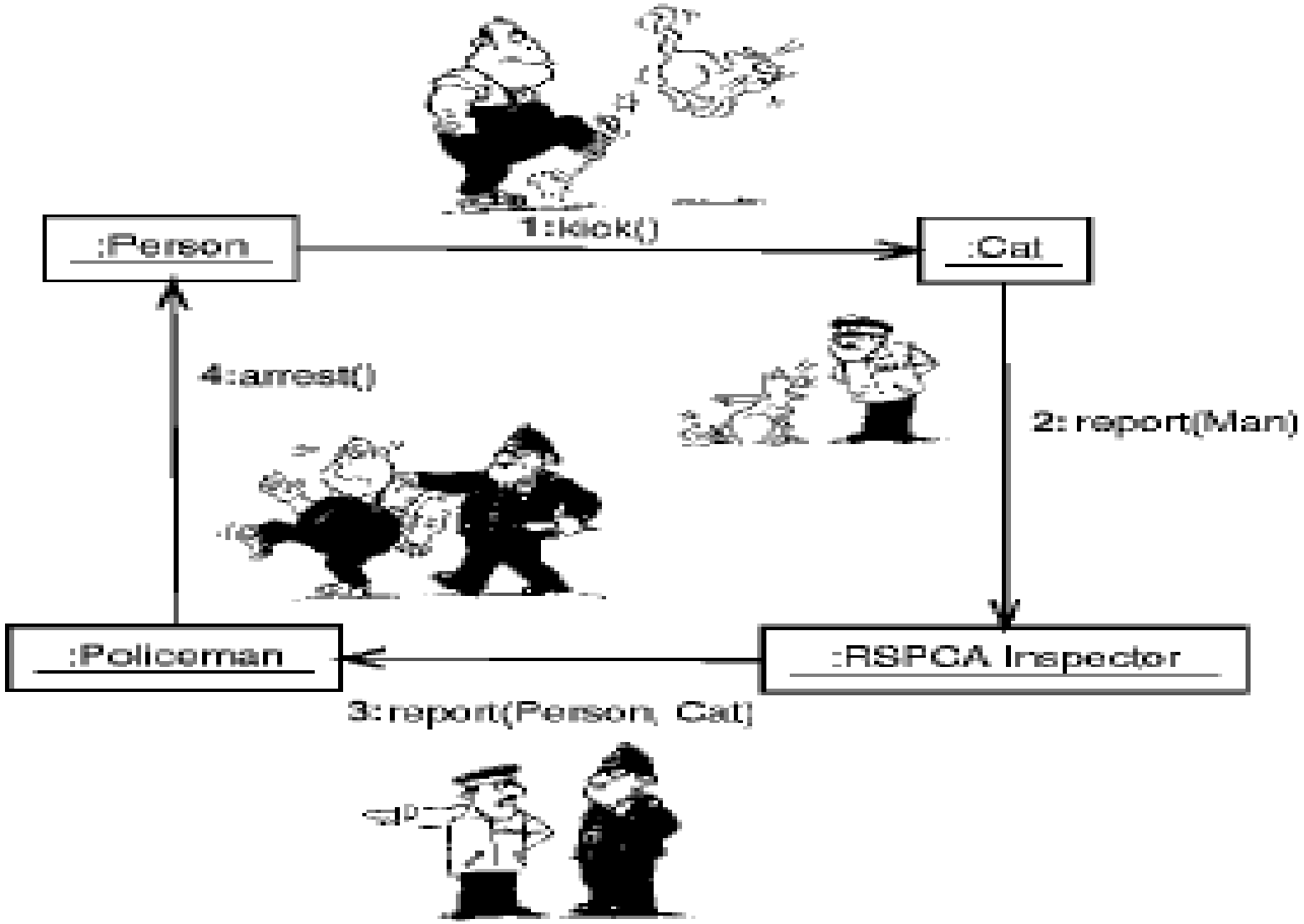
:Policeman

:RSPCA Inspector

Fun example: Sequence diagram



Fun example: Communication diagram



Project (1)

- Divide groups of 4-5 students
- Each group chooses a problem
- Building Activity diagrams, State diagrams: Choose one of the following tools:
  - Microsoft Visio
  - StarUML: http://staruml.io/
  - Argo UML: https://argouml.jaleco.com/
  - Lucidchart: https://www.lucidchart.com/pages/examples/uml_diagram_tool

Project (2)

- Divide groups of 4-5 students
- Each group chooses a problem
- Building Interaction diagrams: Choose one of the following tools:
  - Microsoft Visio
  - StarUML: http://staruml.io/
  - Argo UML: https://argouml.jaleco.com/
  - Lucidchart: https://www.lucidchart.com/pages/examples/uml_diagram_tool