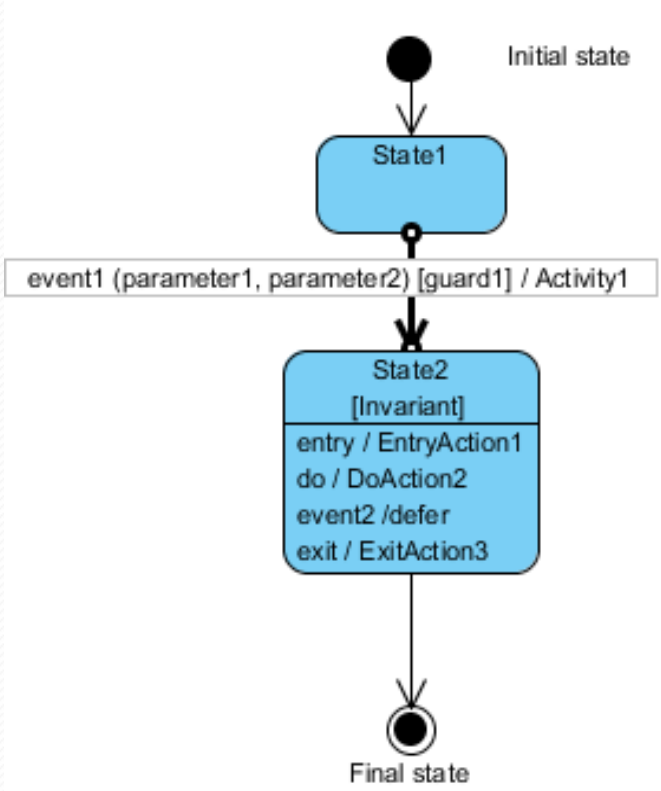
6th Lecture - Dynamic analysis of the computer system

- State chart diagram
- Interaction diagrams:
  - Sequence diagrams
  - Collaboration diagrams

State machine diagram

- It models the of a particular object (instance) **dynamic state**
- According to UML, a state is „a condition during an object's life when satisfies some criterion, performs some action, or waits for an event".
- It identifies **the events** that determine the transition of an object from one state to another state.
- Not all events are applicable in the context of all states. There are **conditions** that may cause the occurrence of a particular event.

State machine diagram

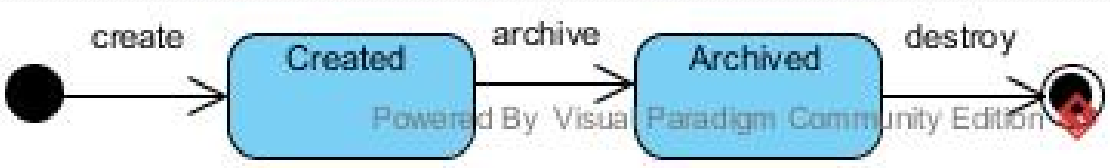


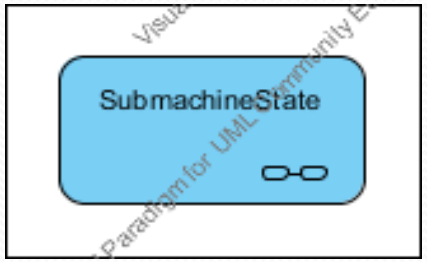
State:

- State **invariant**: Specifies conditions that are always true when this state is the current state.
- **On entry** action: An optional behavior that is executed whenever this state is entered
- **Do** activity: An optional behavior that is executed while being in the state
- **On exit** action: An optional behavior that is executed whenever this state is exited
- **A deferred trigger** is retained until the state machine reaches a state configuration where it is no longer deferred.

States

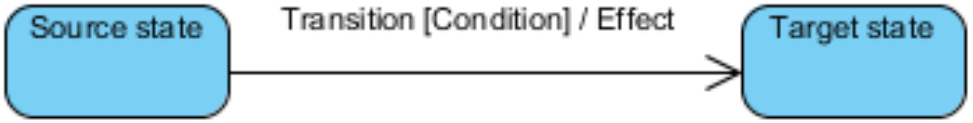
- **State**- represented by a rectangle with rounded corners.
- **Initial and final state** – they have same notations as in activity diagrams. Semnifică începutul şi sfârşitul vieţii unui obiect.
- **Sub-machine state** – It is a state that contains sub-states (embedded states).





Transitions

- **The object transits** from one state to another **when an event occurs** and **when certain conditions are met**.
- **Transition –** represented by an arrow from an source state to a target state.
- It may contain:
  - ***Trigger***: it is the cause of transitions, and it may be an event, a condition change or the time passing.
  - ***Condition***: A condition that must be true for the trigger to determine the transition.
  - ***Effect***: Action that will be invoked by object as a result of transition.

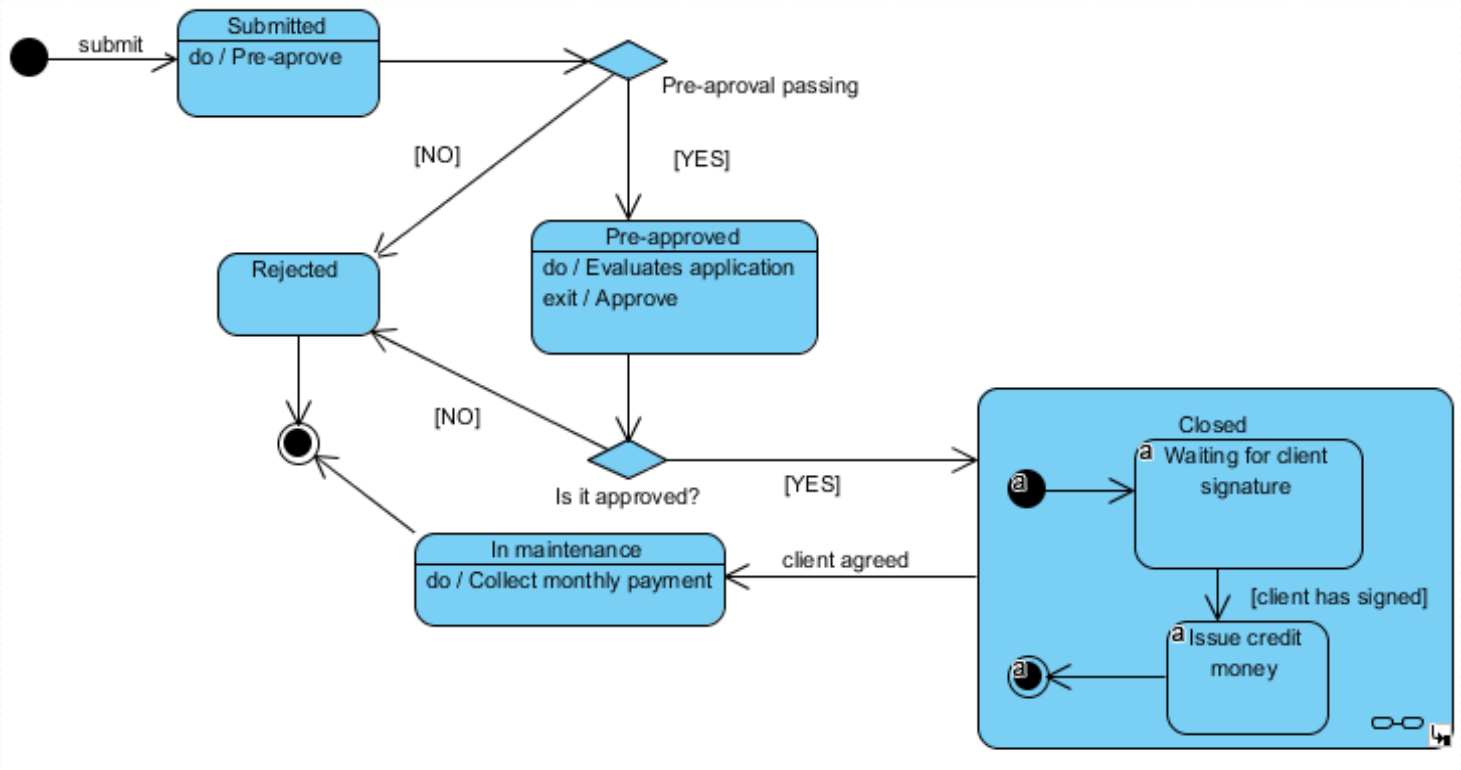


Actions

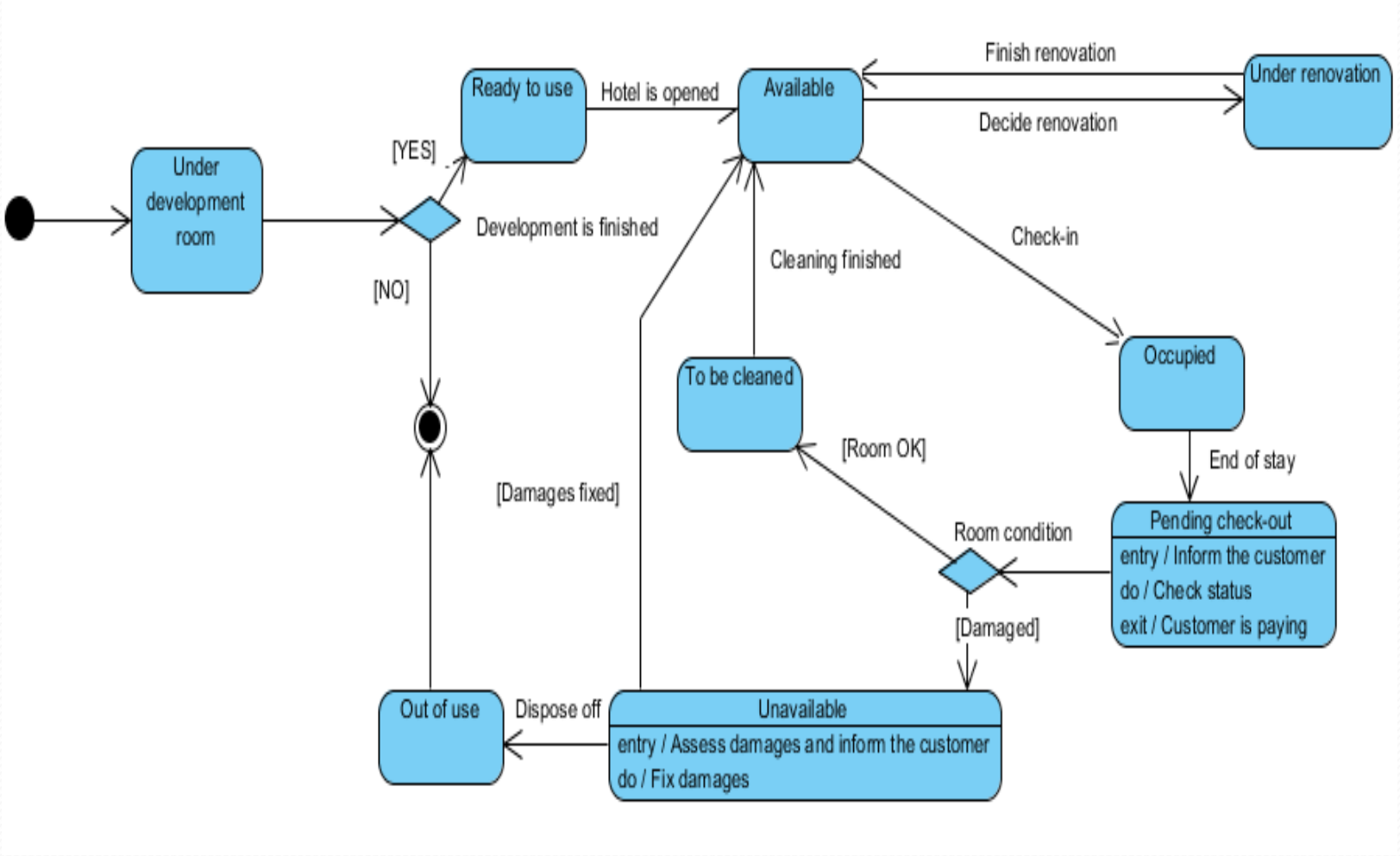
- Except for the initial and the final state, every state has a **name**, state **attributes**, **performed actions** and **activities**.
- Special actions are:
  - **Entry** – action taken when entering into a state.
  - **Exit** – action taken when leaving a state.
  - **Do** - action taken during a state; external events can interrupt Do action.

Decisions

- **Decision** (Choice) – it is a pseudo-state that make a conditional fork. It evaluates conditions for the triggers of output transitions in order to choose only one output transition.



State machine diagram – “Room” example



The role of interaction diagrams

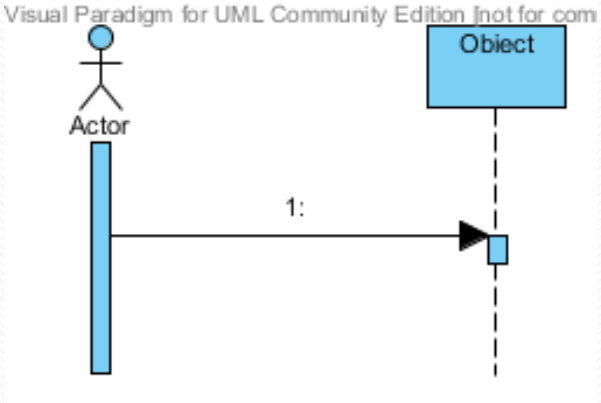
- Model the **dynamic aspects** of the system.
- They consist of a **set of objects** and **relationships** between them, including messages that objects send to each other
- There are two types of interaction diagrams: *sequence diagram* and *communication diagram* (called collaboration diagram in UML 1.4).
- The two diagrams are equivalent in terms of semantics and can transform from one to the other.

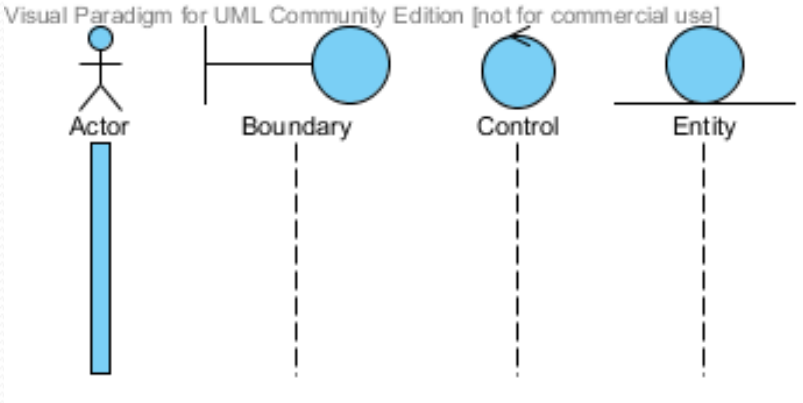
Sequence diagram

- It's an interaction diagram made up of *objects*, *messages* exchanged between them and the *temporal dimension* represented progressively vertically.
- It underlines the sequence of messages according to time
- **The objects** are placed in the upper part of the diagram, along the X-axis, from left to right
  - They are arranged in any order that allows the simplification of the diagram.
  - Typically, objects starting the interaction are placed to the left and objects that follow to the right
  - The existence of objects is illustrated by their *lifelines*

Sequence diagram - objects

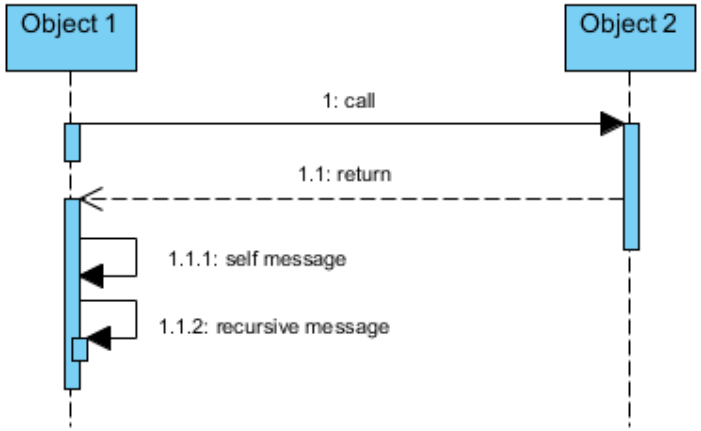
- ***Lifeline**:* vertical line that represents the existence of an object over a period of time. Most of the objects that appear in the diagram exist through the duration of the interaction, with the life line drawn from the top of the diagram to the bottom. Other objects can be **created** during the interaction.
- ***Activation** :* a tall, thin rectangle indicating the period of time **the object performs an action.** The upper end of the rectangle is aligned with the beginning of the action and the lower end to end of that action
- Objects can be represented using the following **stereotypes**: *actor*, *boundary*, *entity* and *control*.





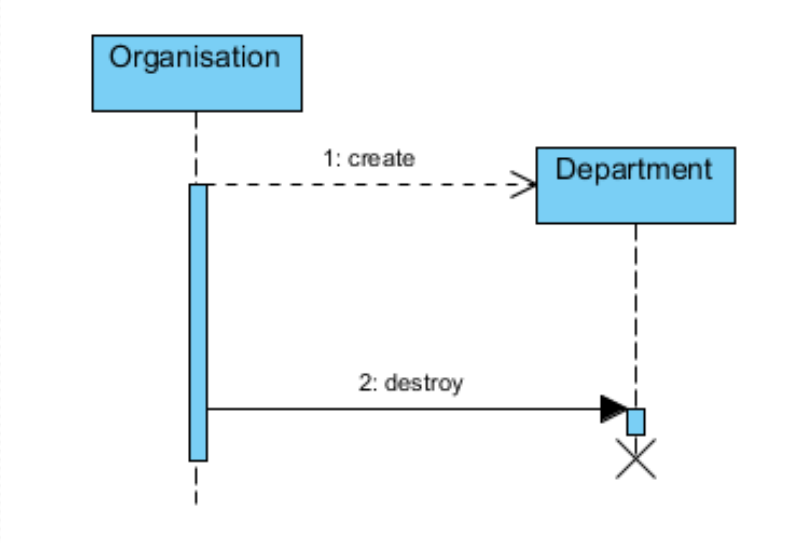
Sequence diagram - messages

- ***Messages*** are represented as arrows that start from the lifeline of an object and stop at the lifeline of another object. Messages can be of several types **and can include conditions**, the same as in the state diagrams.
- A **call message** represents a request of the object that sends the message to the object that receives the message. The request implies that the receiver will perform one of its operations.
- The sender waits for the receiver to execute the operation and to receive an answer from him. (*return*)
- An object can send messages to itself – self call. Such a message can mean a recursive call of an operation or method that calls another method of the same object.

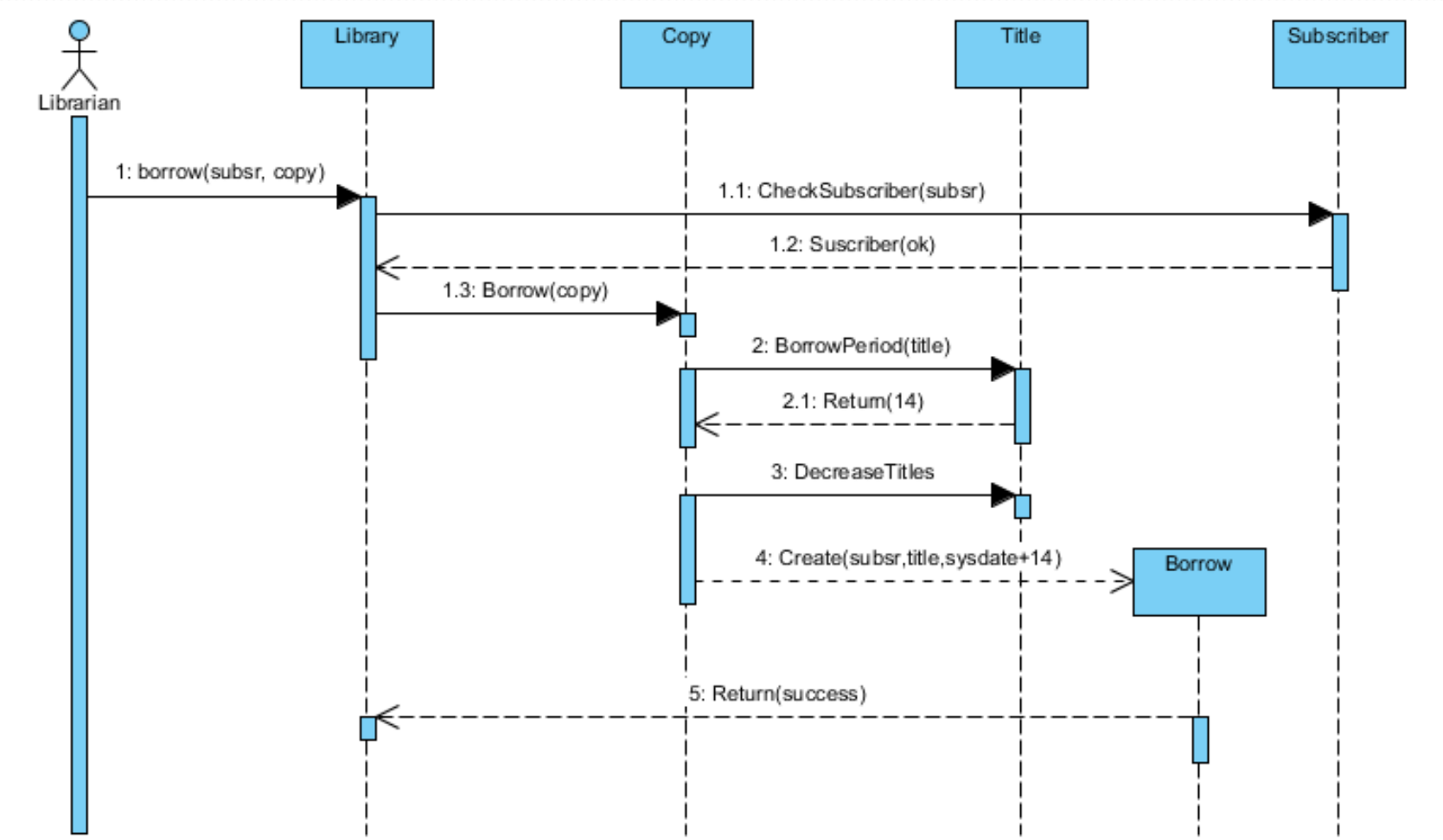


Sequence diagram - messages

- ***Create*** and ***destroy*** messages of an object begin and end the lifeline of an object. These are optional and are used when you want to explicitly specify these events.
- A *destroy* message can generate subsequent destruction of objects that it contains through composition. After destruction, an object can not be created again on the same lifeline.



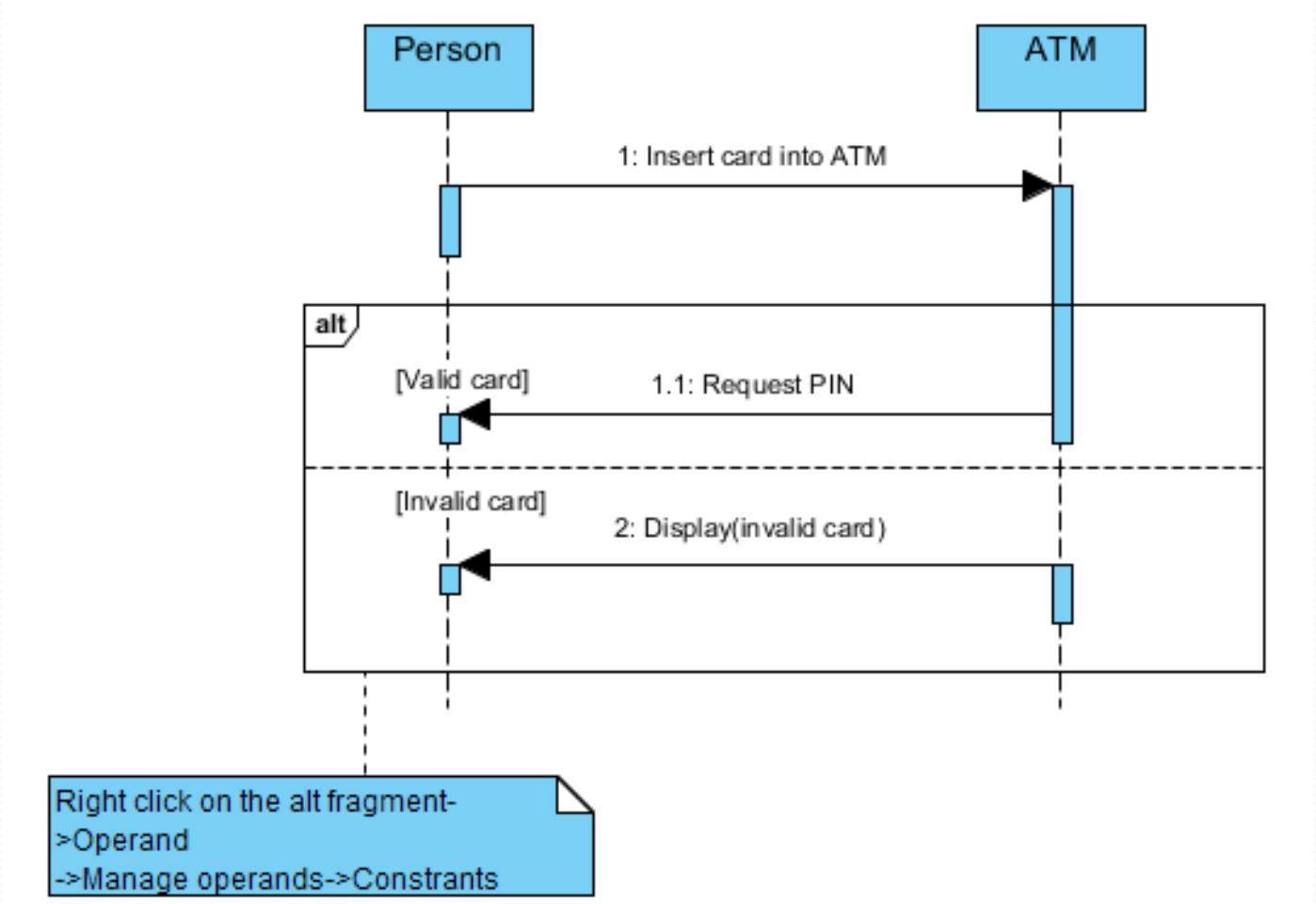
Sequence diagram - example



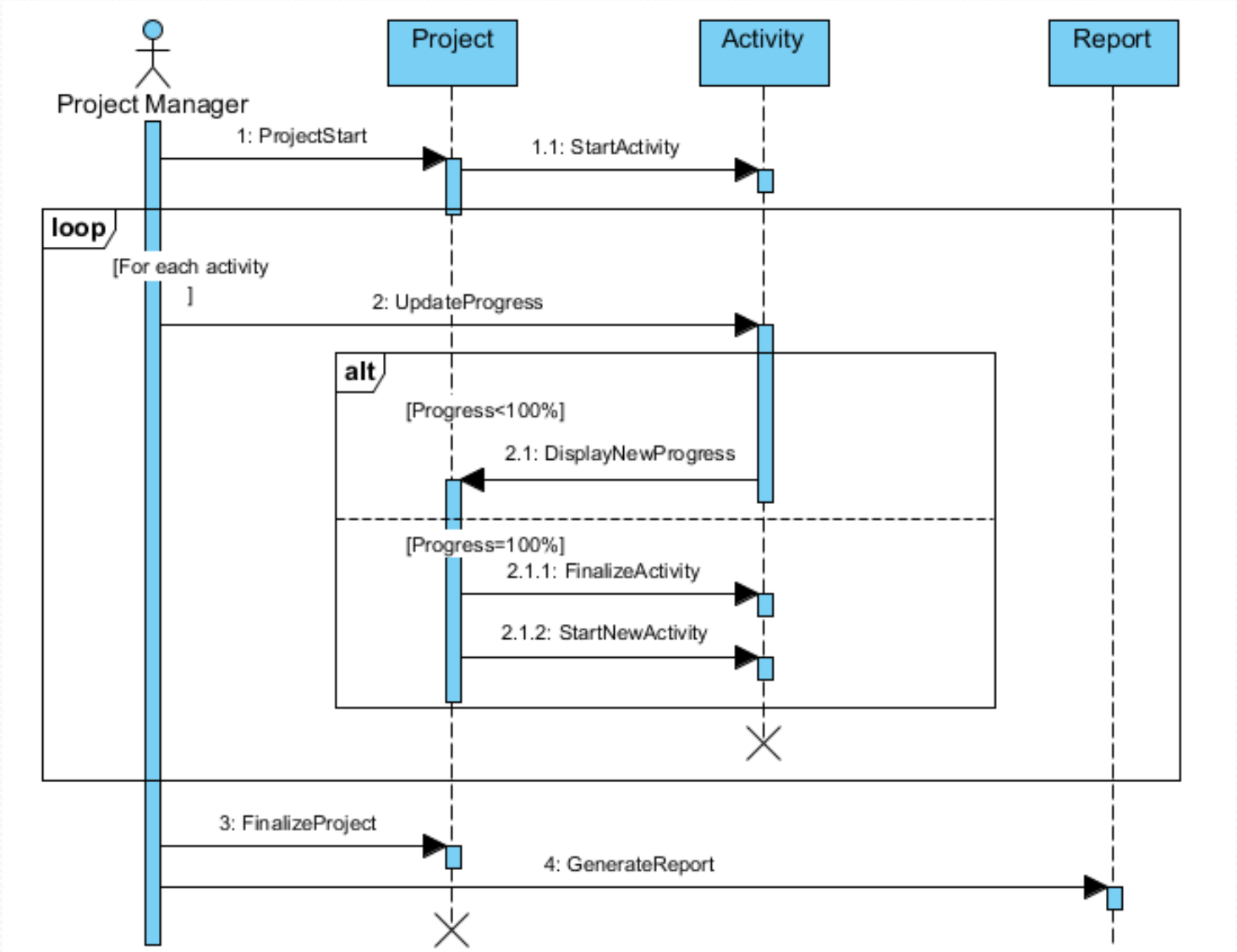
Combined fragments

- Sequence diagrams *are not used to represent complex procedural logic, but to model simple, sequential control flows.*
- However, *there are mechanisms that allow the addition of a certain level of procedural logic* in the diagrams through combined fragments.
- A **combined fragment** represents one or more processing sequences included in a frame and executed under certain circumstances.
- Most commonly used fragments are:
  - Alternatives (**Alt**) that model 'if..then..else' logic.
  - Repetitives (**Loop**) containing a series of interactions that are repeated several times.
  - Parallel fragments (**Par**) that model concurrent processing.

Combined fragments- example



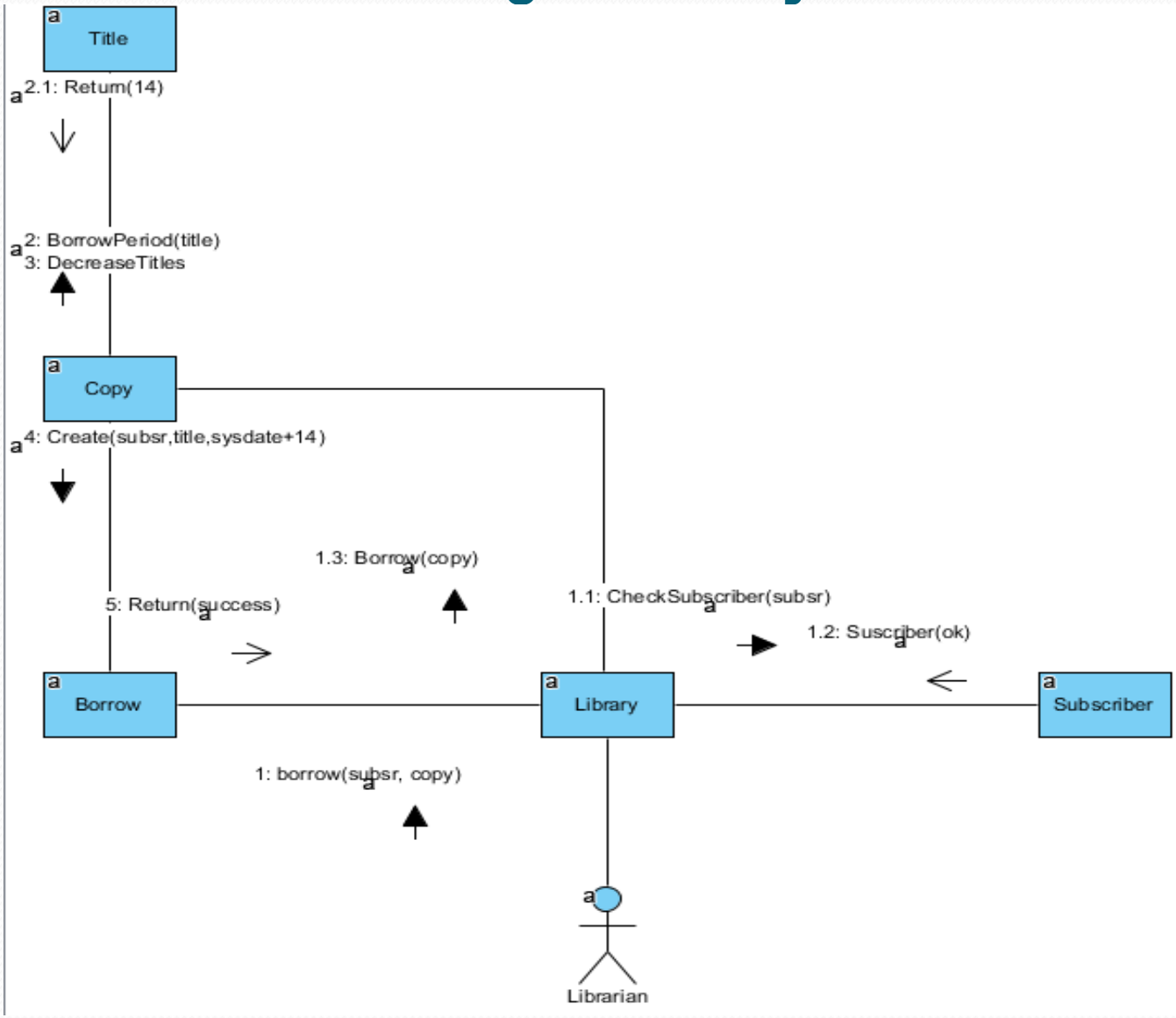
Combined fragments- example



Communication diagram

- **Communication diagram** (**collaboration diagram –** as it is named in UML 1.4) is an interaction diagram that emphasizes the structural organization of objects that send and receive messages
- Graphically, a collaboration diagram is a collection of vertices and edges
- It represents the same information as a sequence diagram, but emphasizes the *organization* of objects participating in the interaction.
- Objects are placed first, as **vertices of a graph**, next the connection between objects are drawn as **edges** of the graph and then the **messages** the objects receive or send are added to the connections
- To indicate the order, the message should be prefixed with a number starting from 1 and increasing

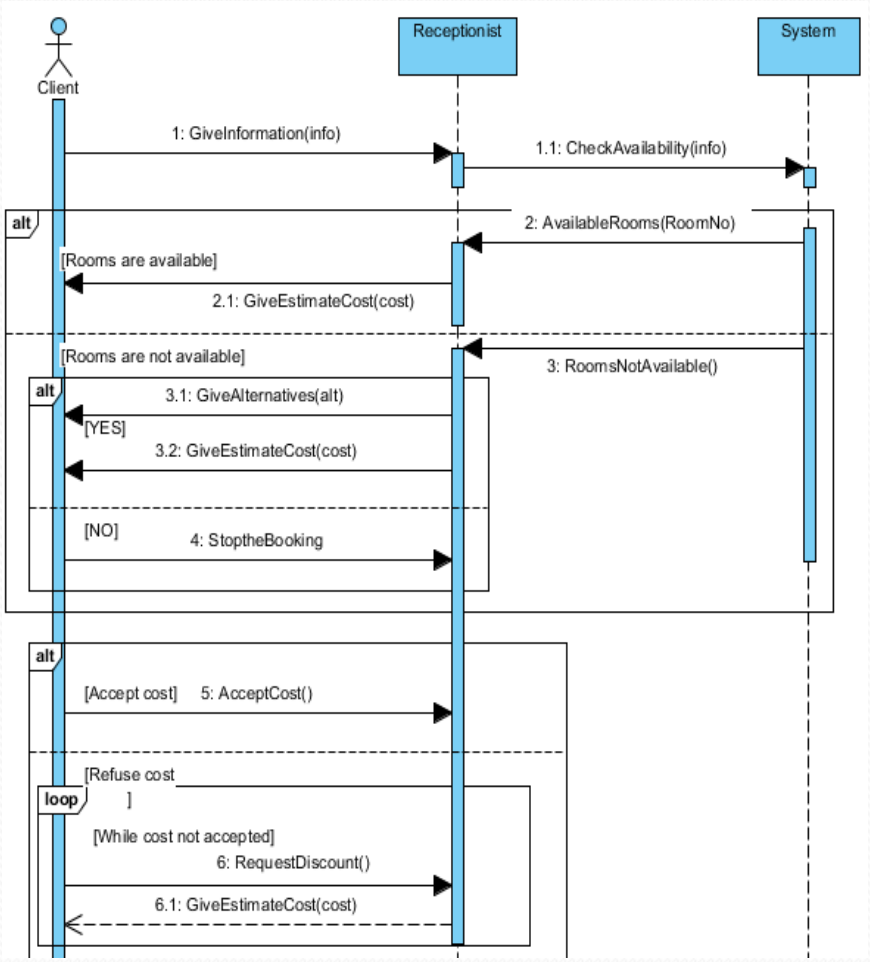
Communication diagram – objects and messages

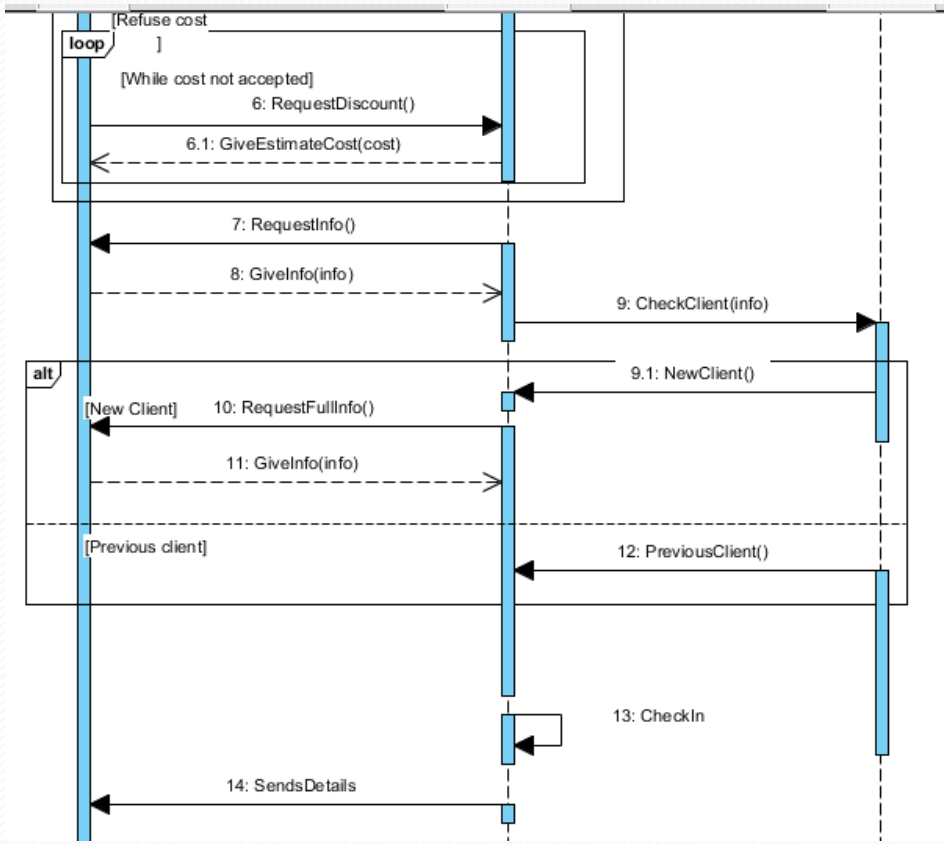


Interaction diagrams

- The two interaction diagrams are **equivalent** and **can be converted** one to another without loosing information.
- To convert a diagram to another in Visual Paradigm right-click on a diagram area and select the option *Synchronize to Communication / Sequence diagram.*
- Communication diagram shows how objects are connected, while the sequence diagram highlights returned messages and temporal order of interactions.

Sequence diagram - example





Communication diagram – hotel example