Abdelhafid Boussouf University Center Mila

Faculty of Math and Computer Science

Department of Computer Science

# Software engineering

# Chapter 5

## UML Diagrams: Dynamic View

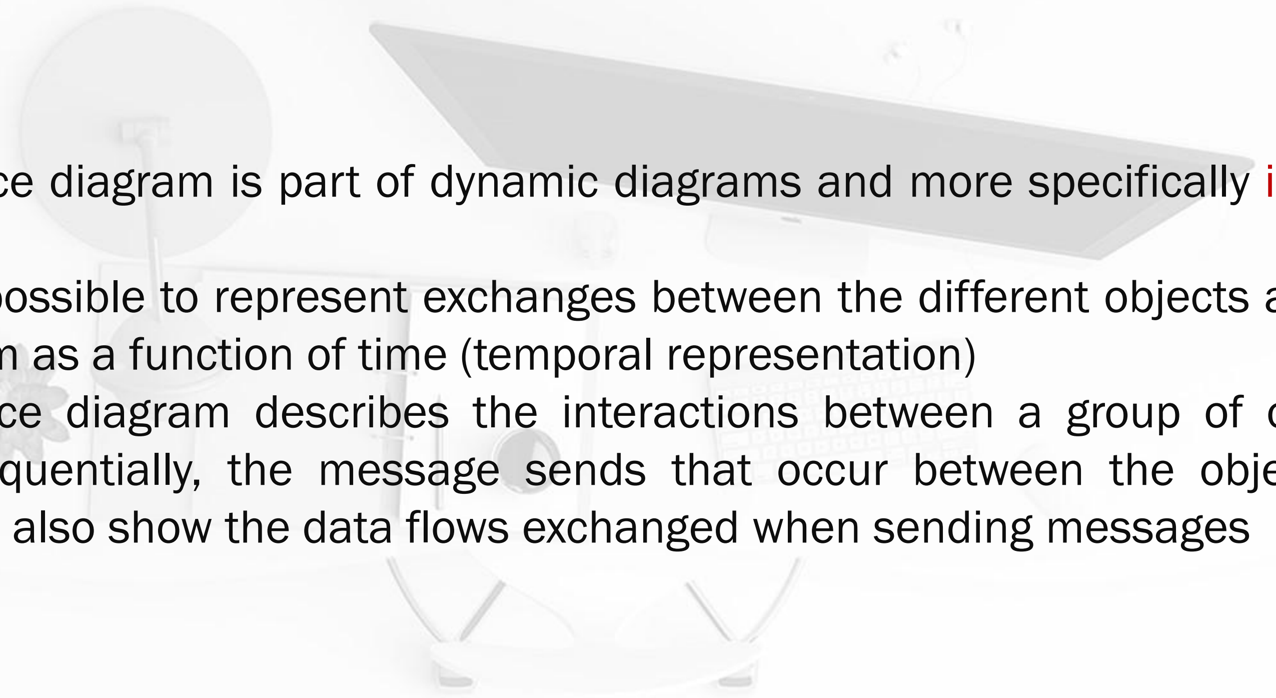
Mrs. S.HEDJAZ

2025/2026

# V. UML Diagrams: Dynamic View

- 01 Interaction diagram (sequence)
- 02 Interaction diagram (communication)
- 03 Activity Diagram
- 04 Transition state diagram

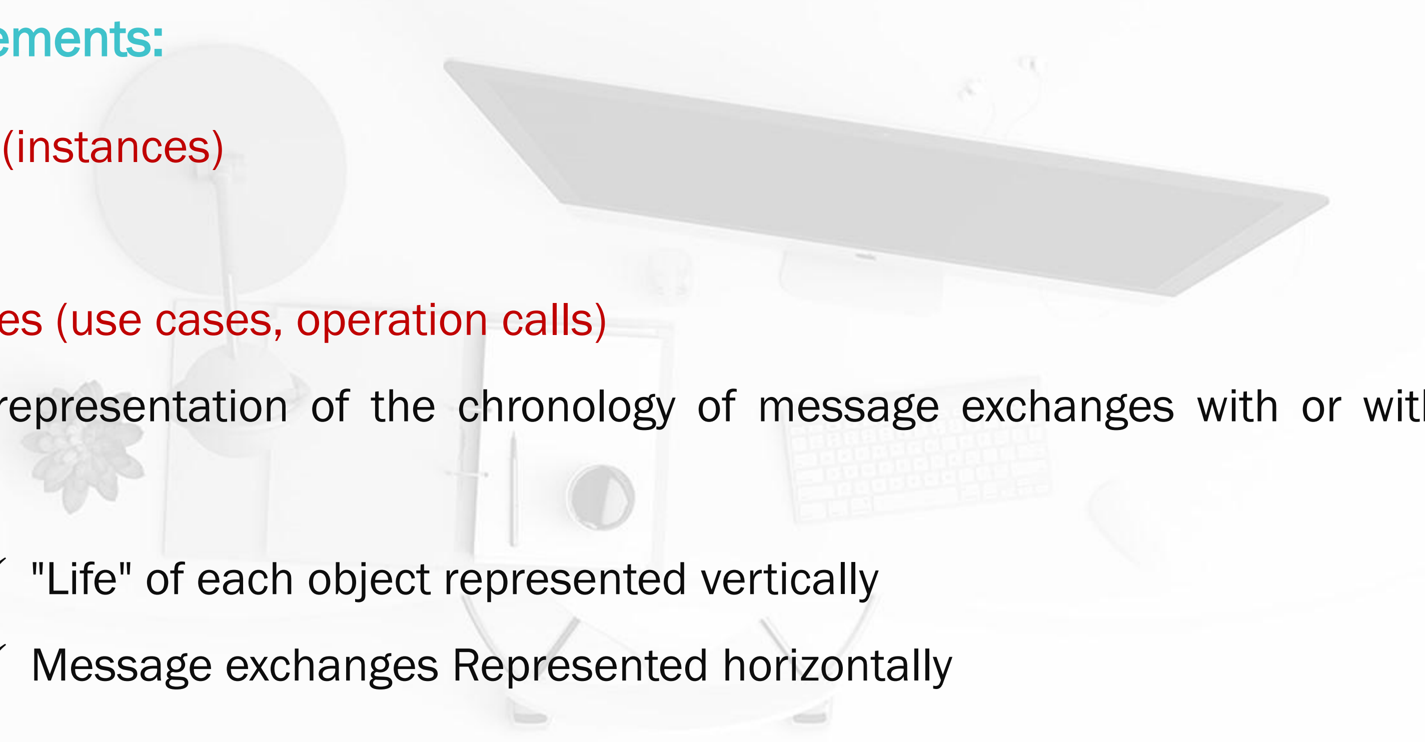
## 01 Interaction Diagram (Sequence)



### 1- Objective:

- The sequence diagram is part of dynamic diagrams and more specifically interaction diagrams
- It makes it possible to represent exchanges between the different objects and actors of the system as a function of time (temporal representation)
- The sequence diagram describes the interactions between a group of objects by showing, sequentially, the message sends that occur between the objects. The diagram can also show the data flows exchanged when sending messages

## 01 Interaction Diagram (Sequence)



### 2- Basic elements:

- Objects (instances)
- Actors
- Messages (use cases, operation calls)

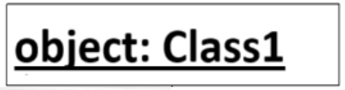
Graphical representation of the chronology of message exchanges with or within the system

- ✓ "Life" of each object represented vertically
- ✓ Message exchanges Represented horizontally

## 01 Interaction Diagram (Sequence)

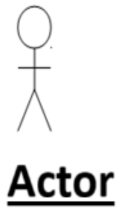
### 2- a- Object:

- In a sequence diagram, the object has the same representation as in the object diagram. That is to say, a rectangle in which the name of the object appears



### 2- b- Actor:

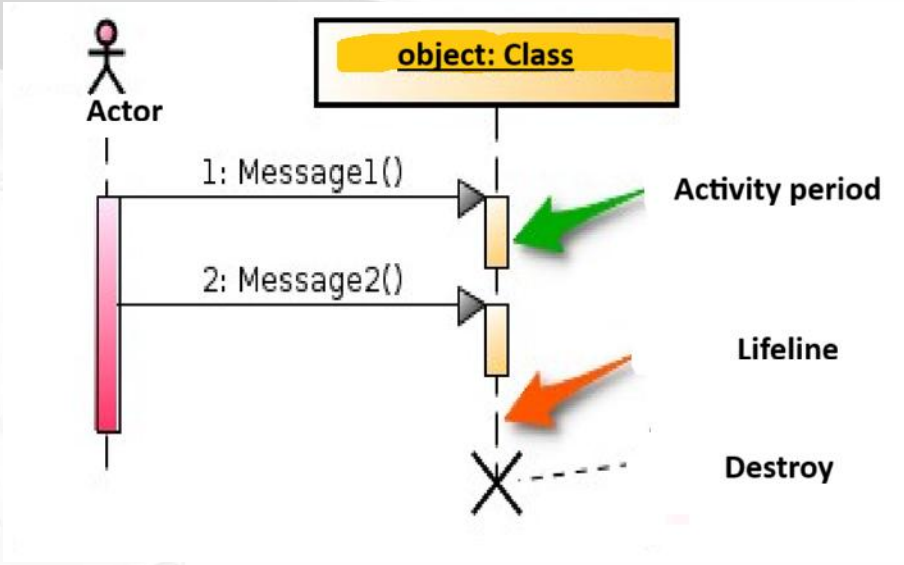
- The sequence diagrams representing the exchanges between the objects but also the exchanges with the actors, we will also find the representation of the stick man (which can be considered as an object).



## 01 Interaction Diagram (Sequence)

### 2- c- Lifeline:

- Each object is associated with a lifeline (in dotted lines vertical to the object) which can be considered as a temporal axis (time flows from the top to the bottom)
- The lifeline indicates the periods that the object is active (typically, the times when the object executes one of these methods).
- When the object is destroyed, the lifeline ends with a cross.



## 01 Interaction Diagram (Sequence)

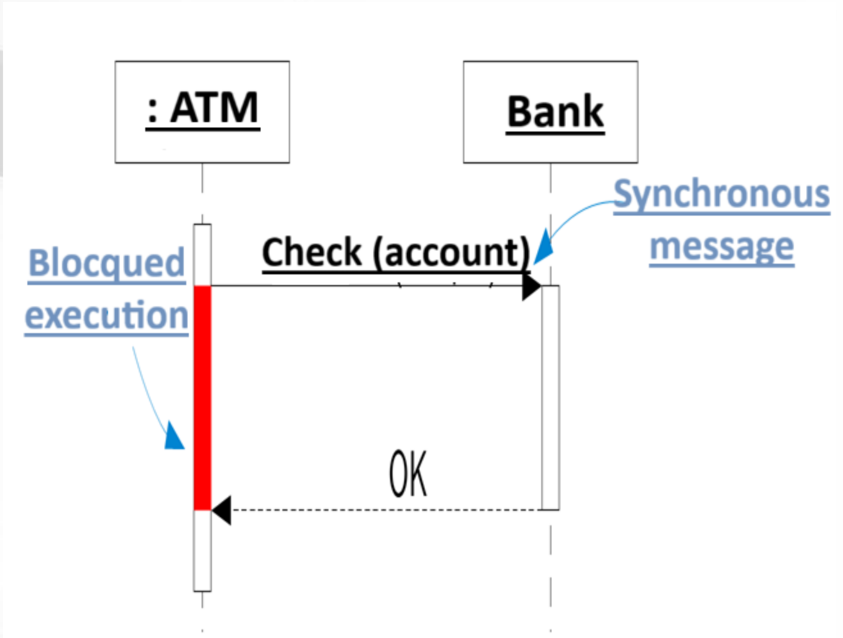
### 2- d- The messages:

- A message defines a particular communication between lifelines. Thus, a message is a communication from one object to another object.
- The reception of a message is considered by the receiving object as an event that must be processed (or not).
- Several types of messages exist, the most common are:
  - ✓ Invoking an operation: synchronous message (calling a method of the target object).
  - ✓ Sending a signal: asynchronous message (typically used for event management).
  - ✓ The creation or destruction of a class instance during the main cycle.

01 Interaction Diagram (Sequence)

## 2- d- 1- Synchronous messages:

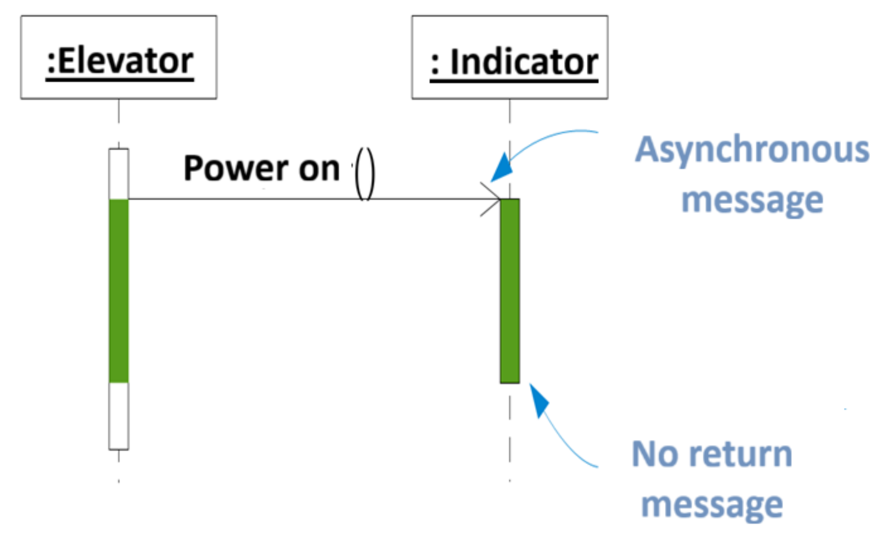
- Receiving a synchronous message must cause the recipient to launch one of its methods (which often has the same name as the message).
- The sender of the message remains blocked for the entire execution of the method, so they wait for the method to complete before they can start a new message. This is the most frequently used message.
- Represented by an arrow with a solid triangle at its end



01 Interaction Diagram (Sequence)

## 2- d- 2- Asynchronous messages:

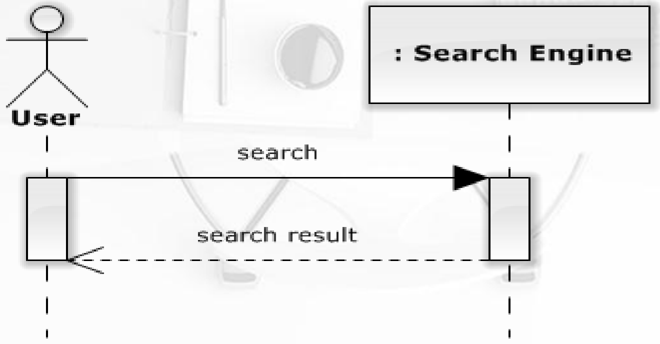
- In the case of an asynchronous message, the sender does not wait for the recipient to complete the activation of the invoked method.
- An asynchronous message represented by a simple arrow



01 Interaction Diagram (Sequence)

## 2- d- 3- Feedback messages:

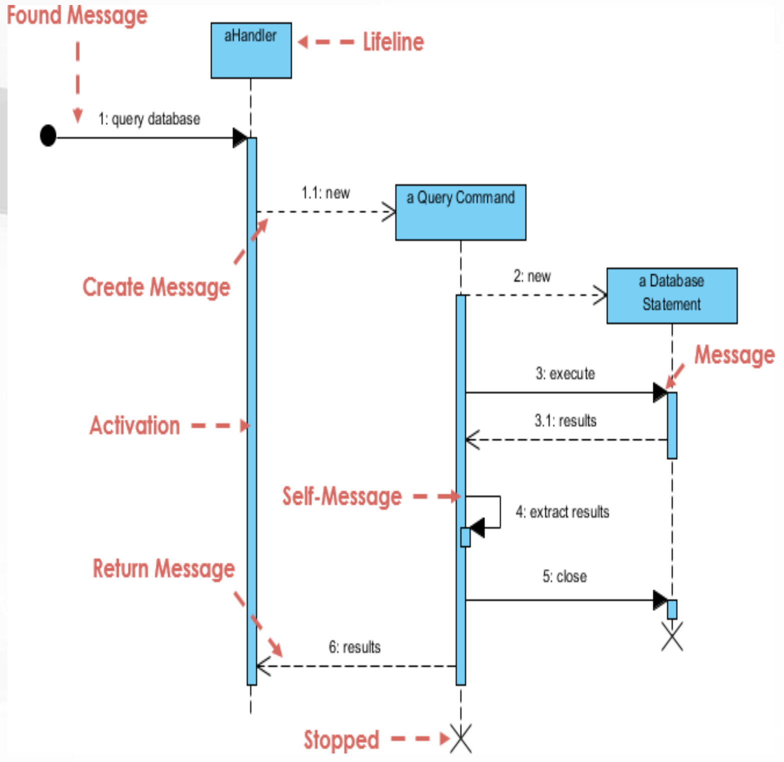
- If a method that has been activated (by a message) needs to return values at the end of its activation, this is done by a return message.
- The return message is therefore not a method call (so it does not cause an object to be activated)
- The return message has often named of the item being returned.
- The return message represented by a simple dotted arrow



01 Interaction Diagram (Sequence)

## 2- d- 4- Creation and Destruction Messages:

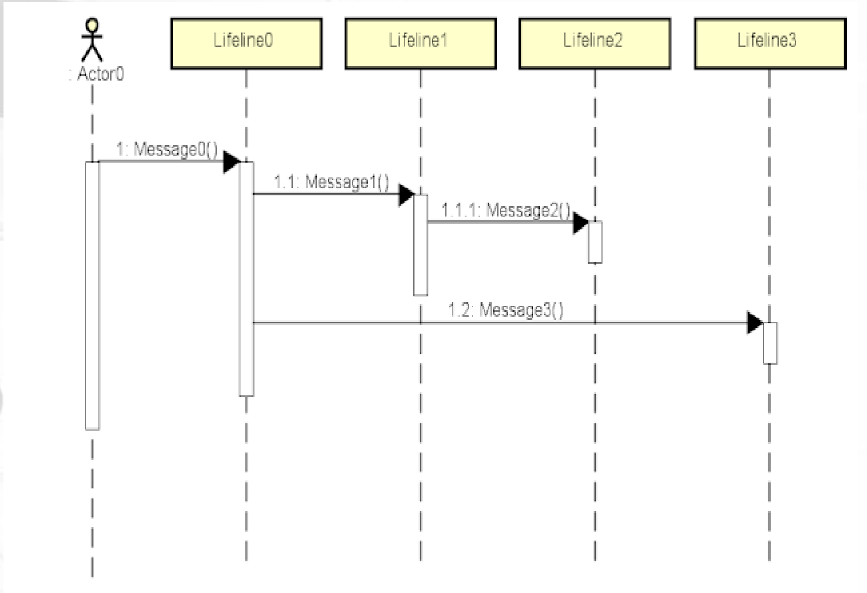
- The creation of an object is materialized by a specific message, a call from a constructor, usually accompanied by the stereotype "create" which points to the beginning (the vertex) of the lifeline of the created object
- The destruction of an object is represented by a cross at the end of its lifeline. Often the object is destroyed following the receipt of a message, but this is not mandatory. In this case, he carries the stereotype "destroy".



01 Interaction Diagram (Sequence)

## 2- e- Synchronous and asynchronous message formalism:

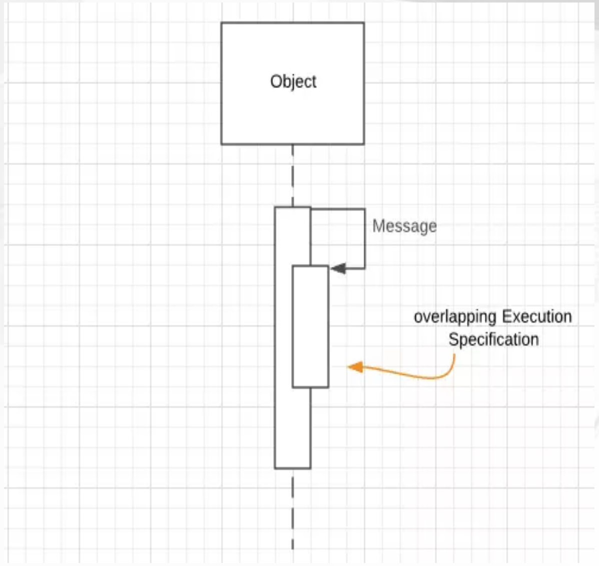
- Its name (which is the name of the method called or the signal sent). We can optionally add:
  - ✓ A numbering in front of the message name (separated from the message name by 2 dots " : ")
  - ✓ Parameters passed to the method or signal (in parentheses after the message name)



## 01 Interaction Diagram (Sequence)

### 2- f- Recursive messages:

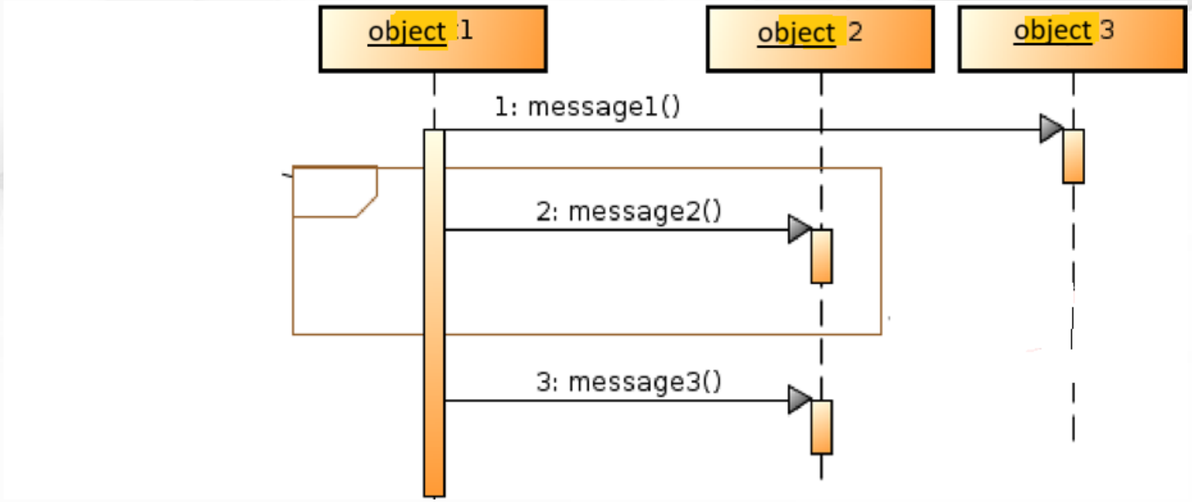
- An object can send a message to itself (using another method of the same object).
- This is also represented by a doubling of the activation band.



01 Interaction Diagram (Sequence)

## 3- Interaction Frame:

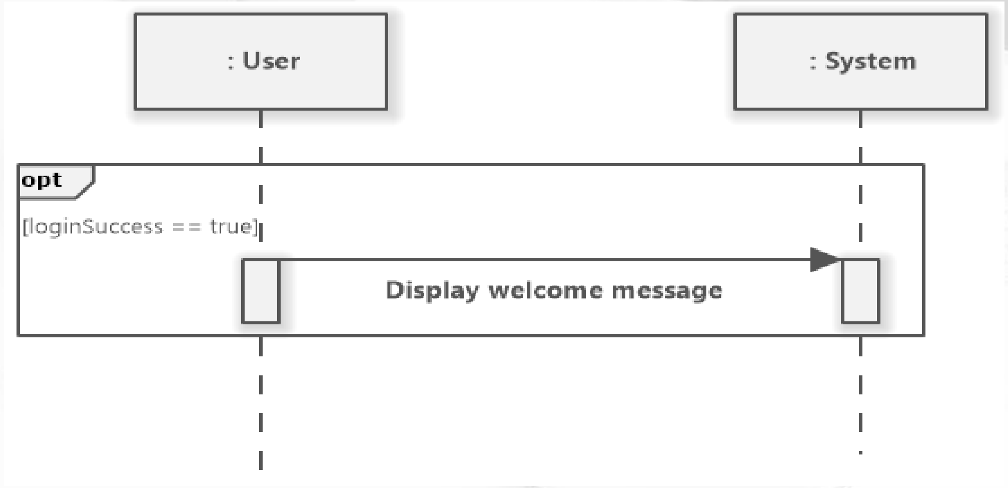
- An interaction frame is a part of the sequence diagram associated with a label.
- It contains an operator that determines how it is executed. The main modalities are skip logic and loop.



## 01 Interaction Diagram (Sequence)

### 3-a- Frame of interaction with "opt":

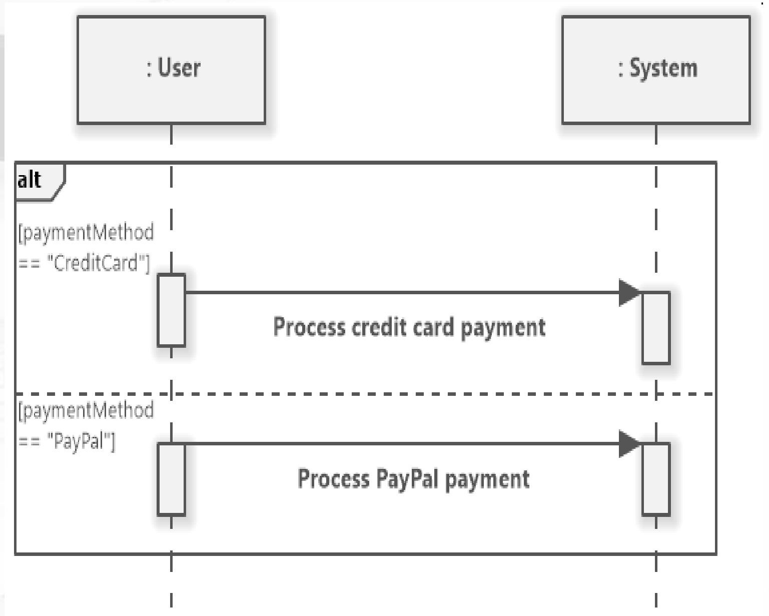
- The option operator (opt) has an operand and an associated guard condition. The subfragment executes if the guard condition is true and does not execute otherwise.



# 01 Interaction Diagram (Sequence)

## 3-b- Frame of interaction with « alt »:

- The alternative operator (alt) is a conditional operator with multiple operands separated by dotted lines. This is the equivalent of a multiple-choice execution.
- Each operand has a guard condition. Only the subfragment with the true condition is executed.
- The else condition is executed only if no other condition is valid.

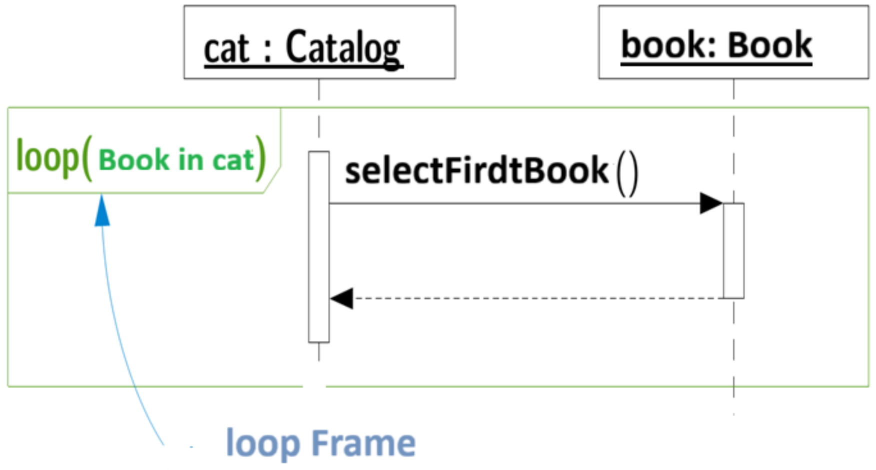

: User
: System
alt
[paymentMethod == "CreditCard"]
Process credit card payment
[paymentMethod == "PayPal"]
Process PayPal payment

## 01 Interaction Diagram (Sequence)

### 3-c- Frame of interaction with "loop":

- The loop is performed by the loop operator followed by the min, max parameters, and a test condition

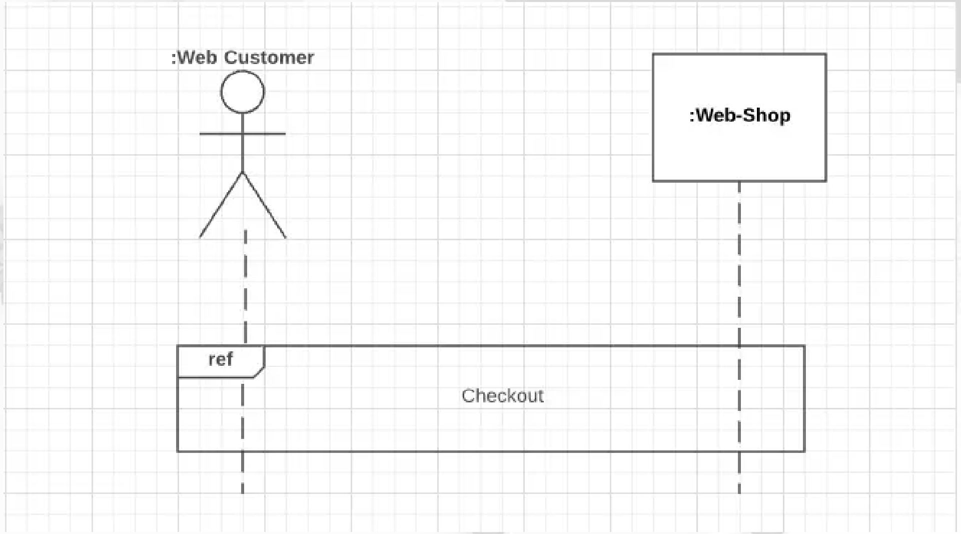
- The contents of the frame are executed min. times, then as long as the test condition is true, and as long as the maximum number of runs in the loop does not exceed max.
- Each parameter is optional



# 01 Interaction Diagram (Sequence)

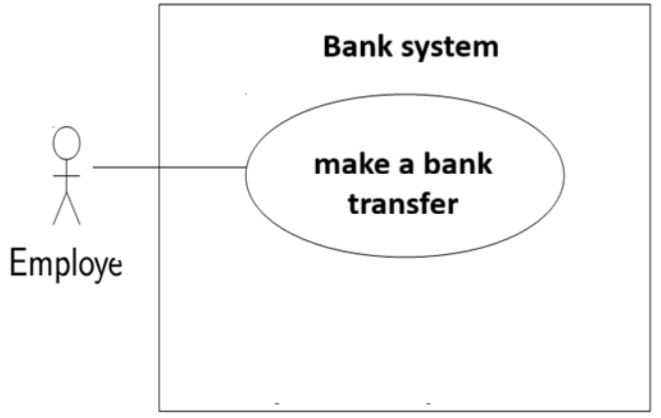
## 3-d- Frame of interaction with "ref":

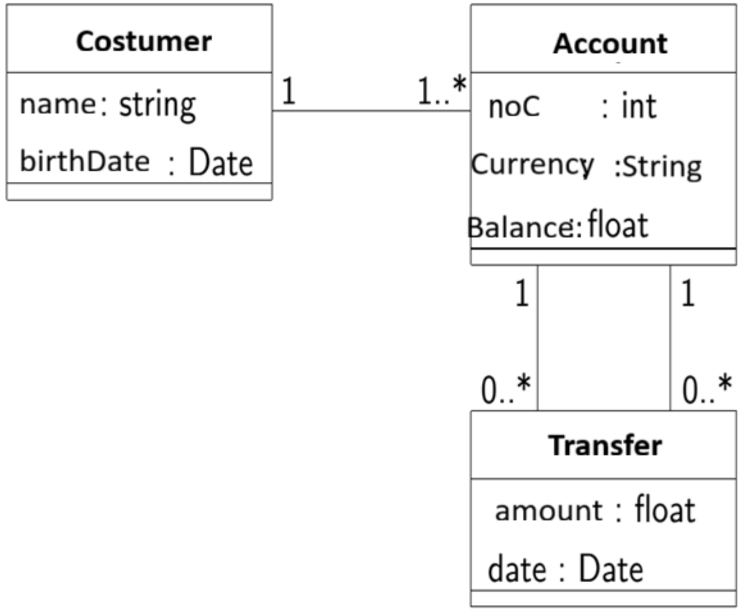
- The operator (ref) allows you to use another sequence diagram.



01 Interaction Diagram (Sequence)

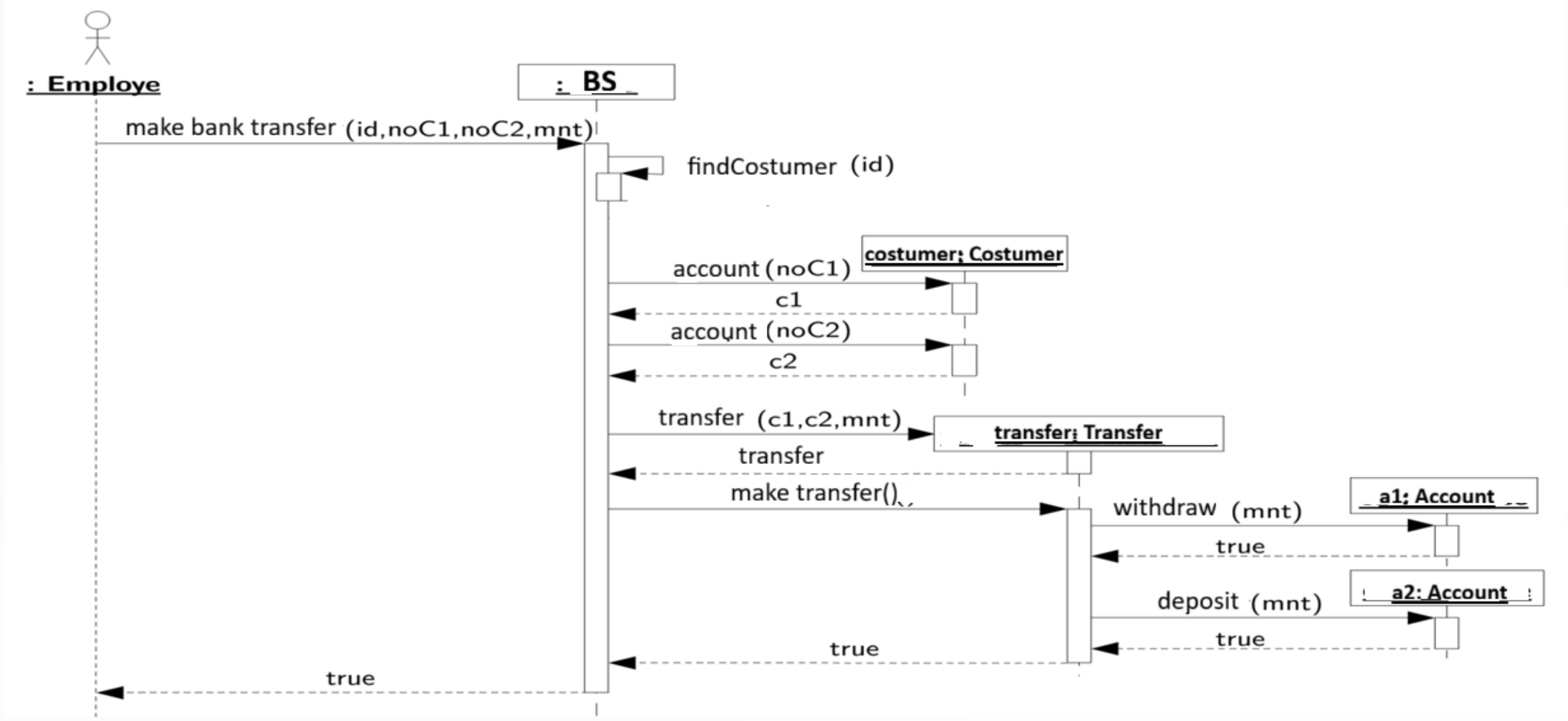
## 4- Example: Use Case Diagram and Class Diagram





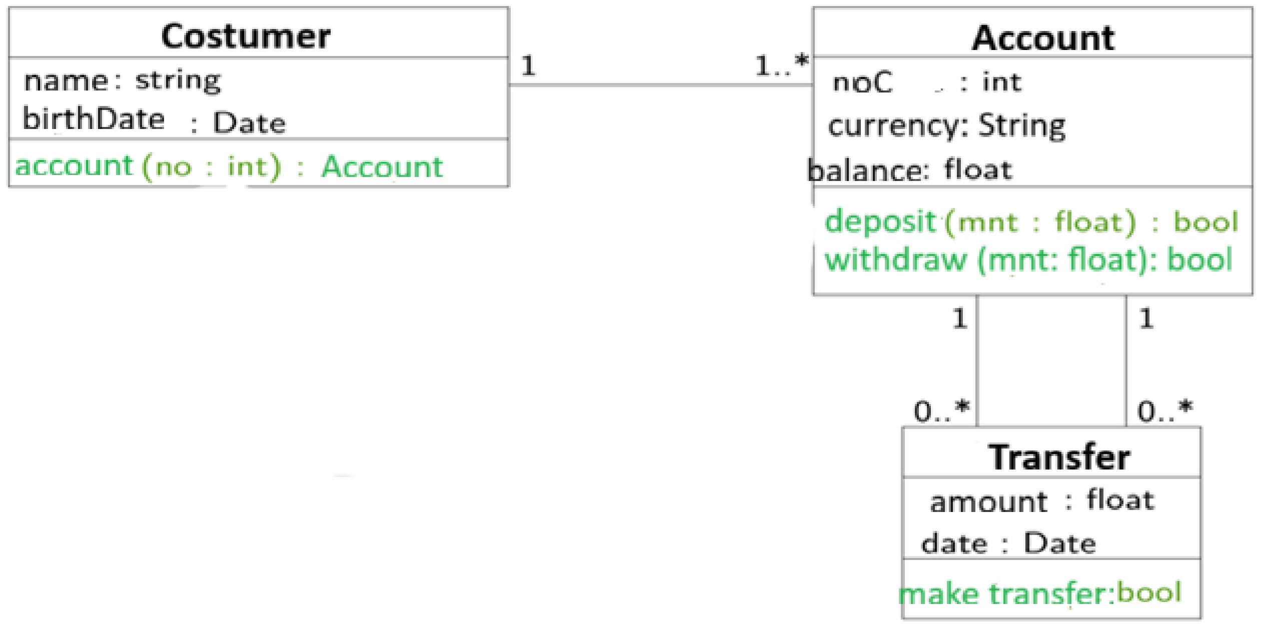
# 01 Interaction Diagram (Sequence)

## 4- Example: Corresponding sequence diagram



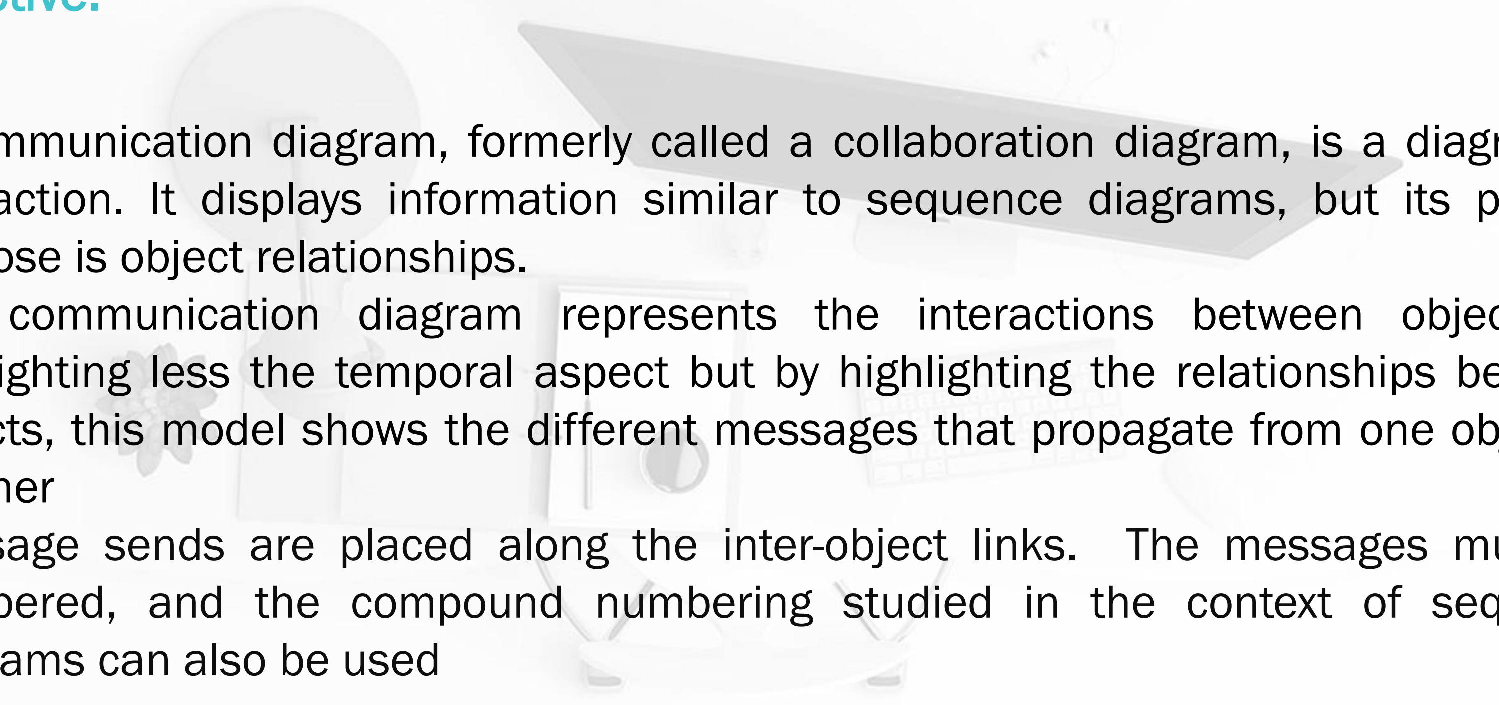
01 Interaction Diagram (Sequence)

## 4- Example: Final Class Diagram



## 02 Interaction diagram (Communication)

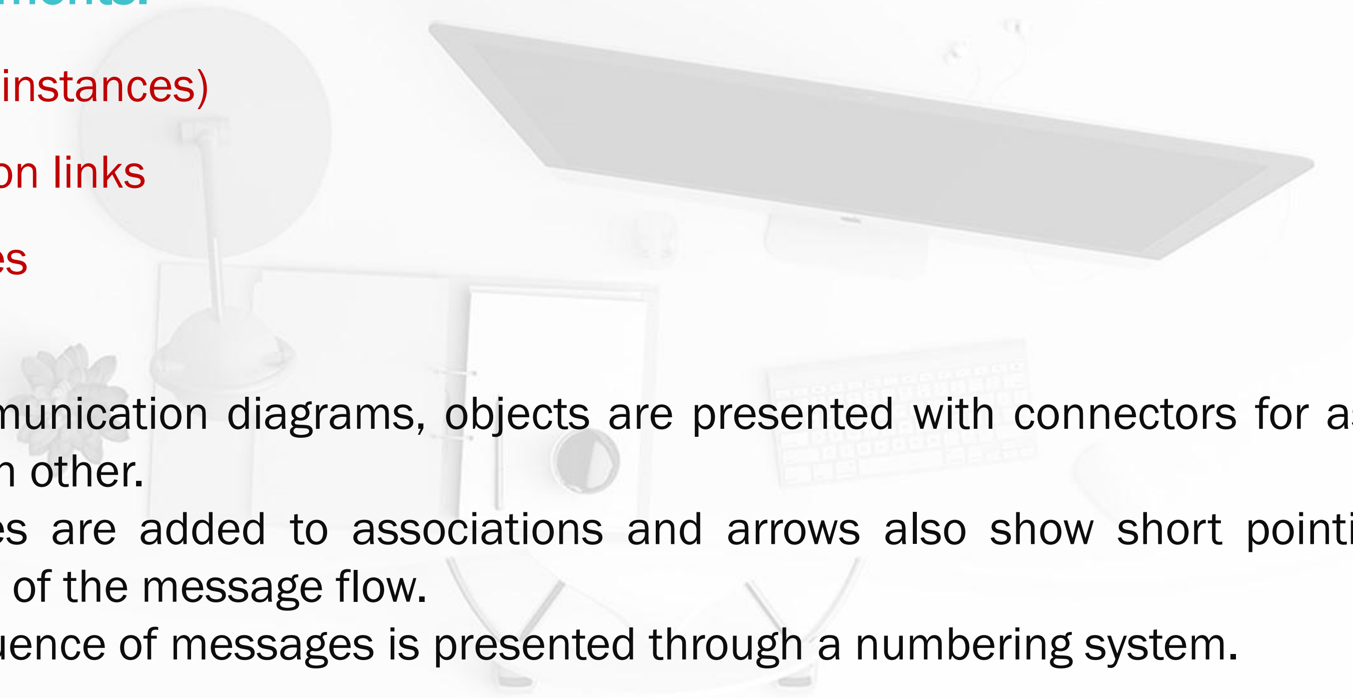
### 1- Objective:



- A communication diagram, formerly called a collaboration diagram, is a diagram of interaction. It displays information similar to sequence diagrams, but its primary purpose is object relationships.
- The communication diagram represents the interactions between objects by highlighting less the temporal aspect but by highlighting the relationships between objects, this model shows the different messages that propagate from one object to another
- Message sends are placed along the inter-object links.  The messages must be numbered, and the compound numbering studied in the context of sequence diagrams can also be used

# 02 Interaction diagram (Communication)

## 2- Basic elements:



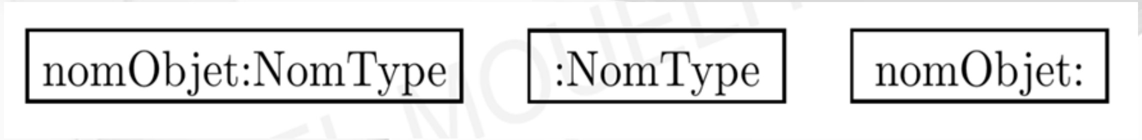
- Objects (instances)
- Interaction links
- Messages

- On communication diagrams, objects are presented with connectors for associating with each other.
- Messages are added to associations and arrows also show short pointing in the direction of the message flow.
- The sequence of messages is presented through a numbering system.

## 02 Interaction diagram (Communication)

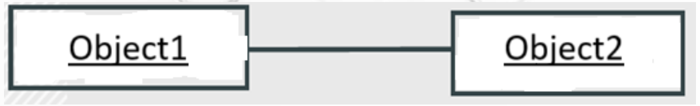
### 2- a- Objects:

- In a communication diagram, the object has the same representation as in the object diagram. That is to say, a rectangle in which the name of the object appears



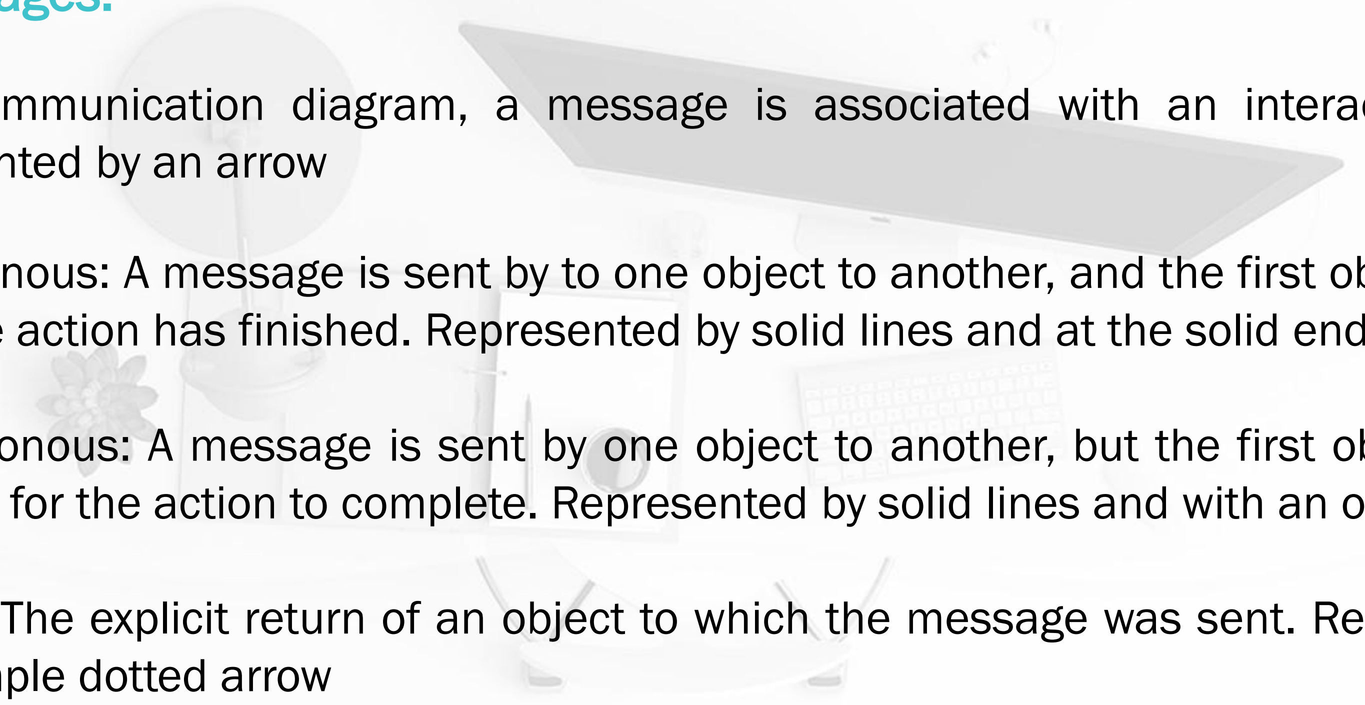
### 2- b- Links:

- Specifies a communication path between two objects, over which messages pass



## 02 Interaction diagram (Communication)

### 2- c- Messages:

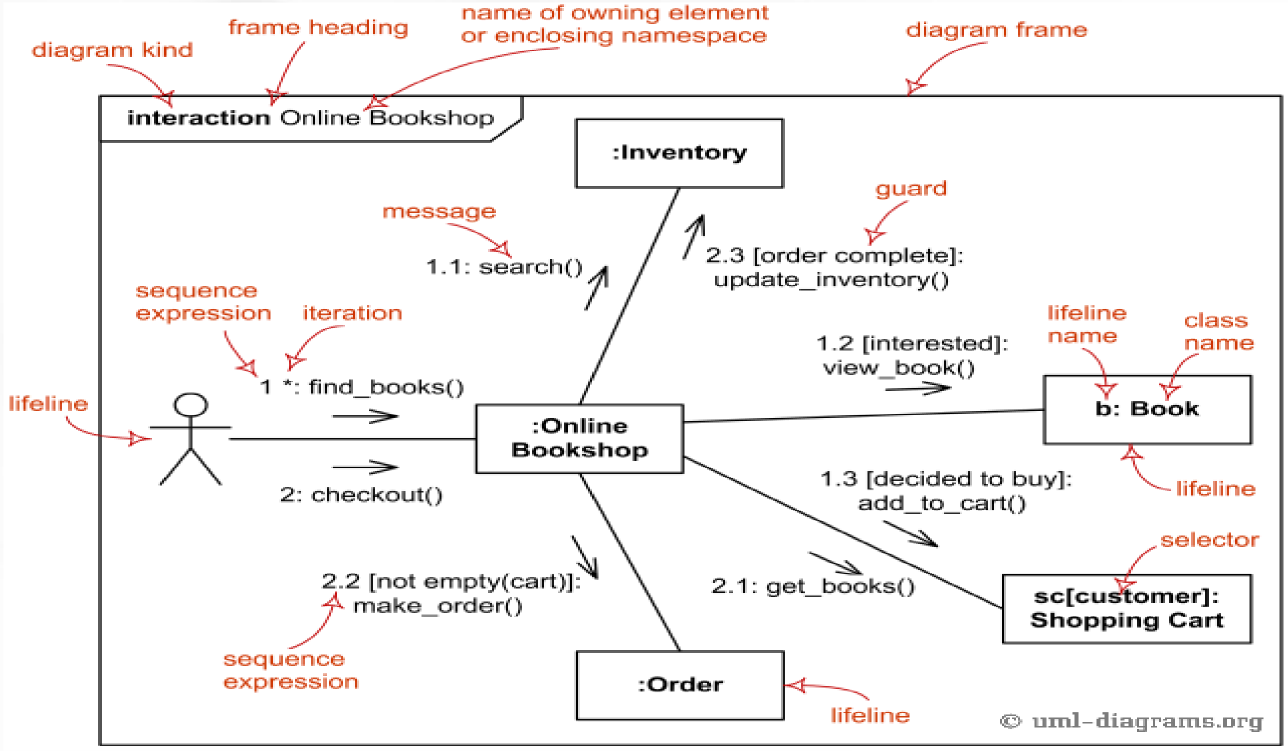


- In a communication diagram, a message is associated with an interaction link, represented by an arrow
- Synchronous: A message is sent by to one object to another, and the first object waits until the action has finished. Represented by solid lines and at the solid end
- Asynchronous: A message is sent by one object to another, but the first object does not wait for the action to complete. Represented by solid lines and with an open end
- Return: The explicit return of an object to which the message was sent. Represented by a simple dotted arrow

02

# Diagramme d'interaction (Communication)

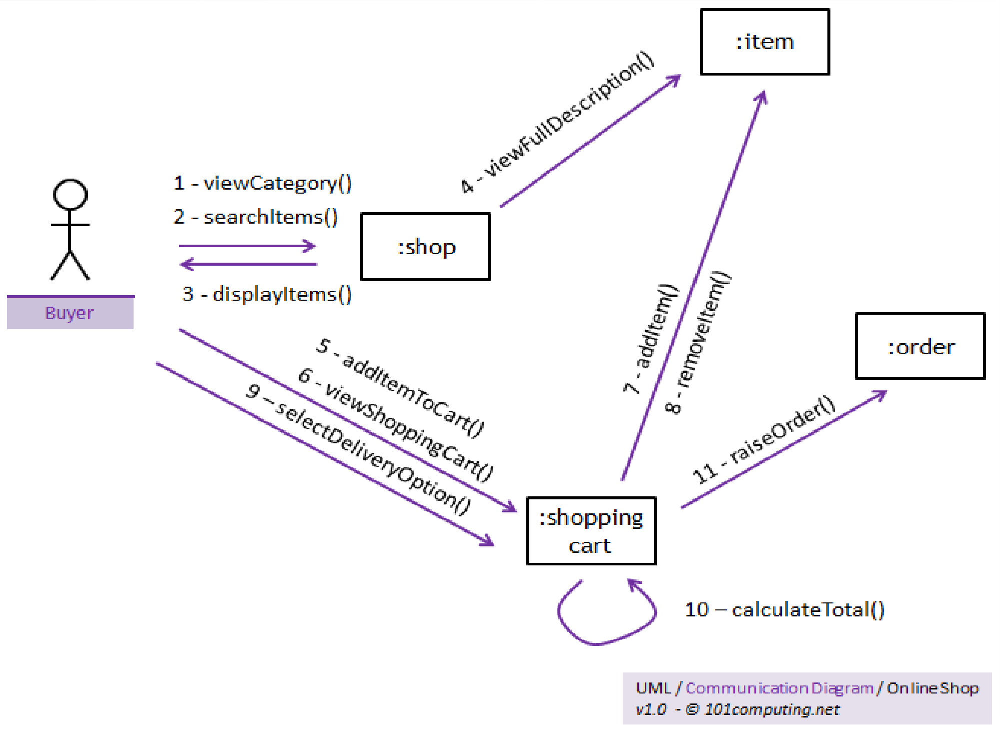
## 2- c- Messages: example:



02 Interaction diagram (Communication)

## 2- c- Messages:

- The messages can be accompanied by several pieces of information:
  - ✓ Order: Numbering
  - ✓ crossing condition: adding a logical expression [condition = true]
  - ✓ Loop: Loop Constraint * [i=0..9]
  - ✓ Return Value



## 03 Activity Diagram

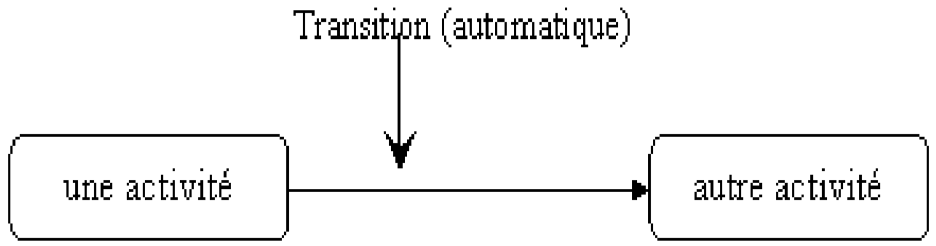
### 1- Objective:

- The activity diagram is used to model the dynamic aspects of a system. It is a question of representing the operations of a process and their consequences on objects (software or hardware). Modeling can be used to describe how a use case or method works.
- Activity diagrams allow for an emphasis on treatments. We can attach a flowchart to any modeling element in order to visualize, specify, construct, or document the behavior of that element.
- In the design phase, activity diagrams are particularly suitable for a greater description of use cases. More precisely, they illustrate and consolidate the textual description.

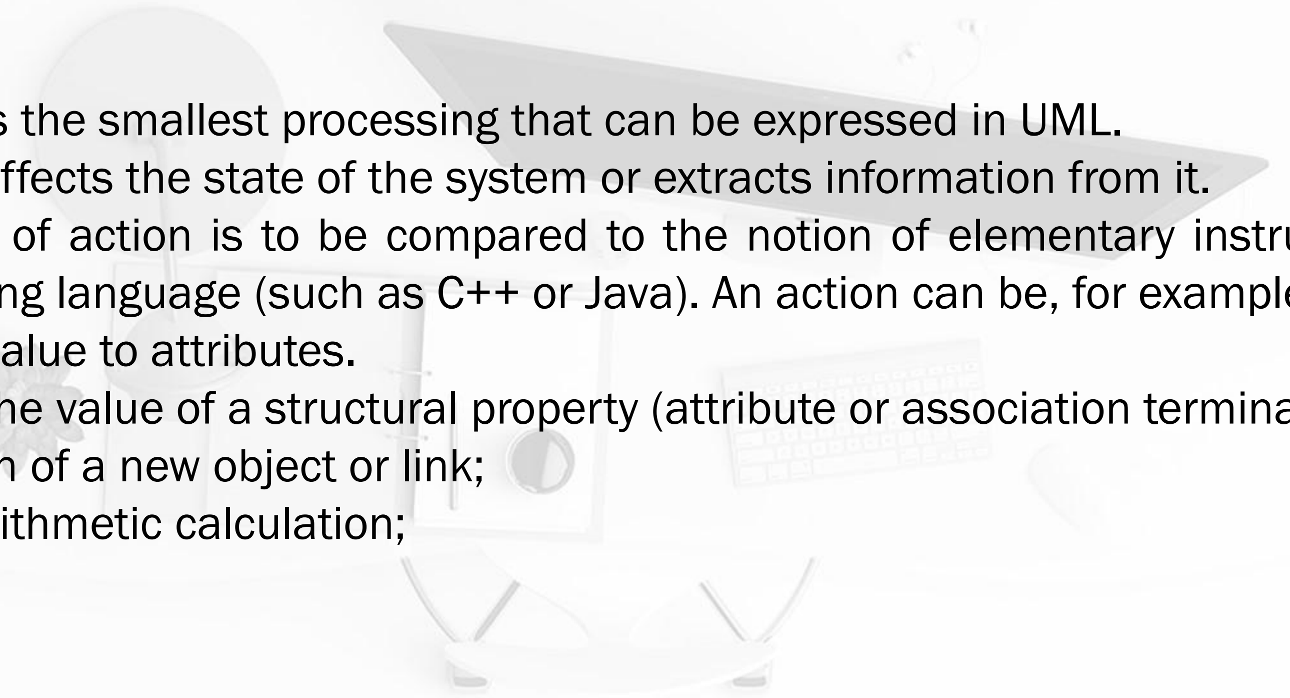
# 03 Activity Diagram

## 2- Basic elements:

- Activities: (one activity = one execution step, activity-state). An activity represents the execution of a mechanism, a sequence of steps. The transition from one activity to another is materialized by a transition.
- Transitions: which are automatic between activities, it is also unnecessary to specify the events. Transitions are triggered by the end of one activity and cause another to begin immediately.
- In theory, all dynamic mechanisms could be described by an activity diagram, but only complex or interesting mechanisms are worth representing.



## 03 Activity Diagram

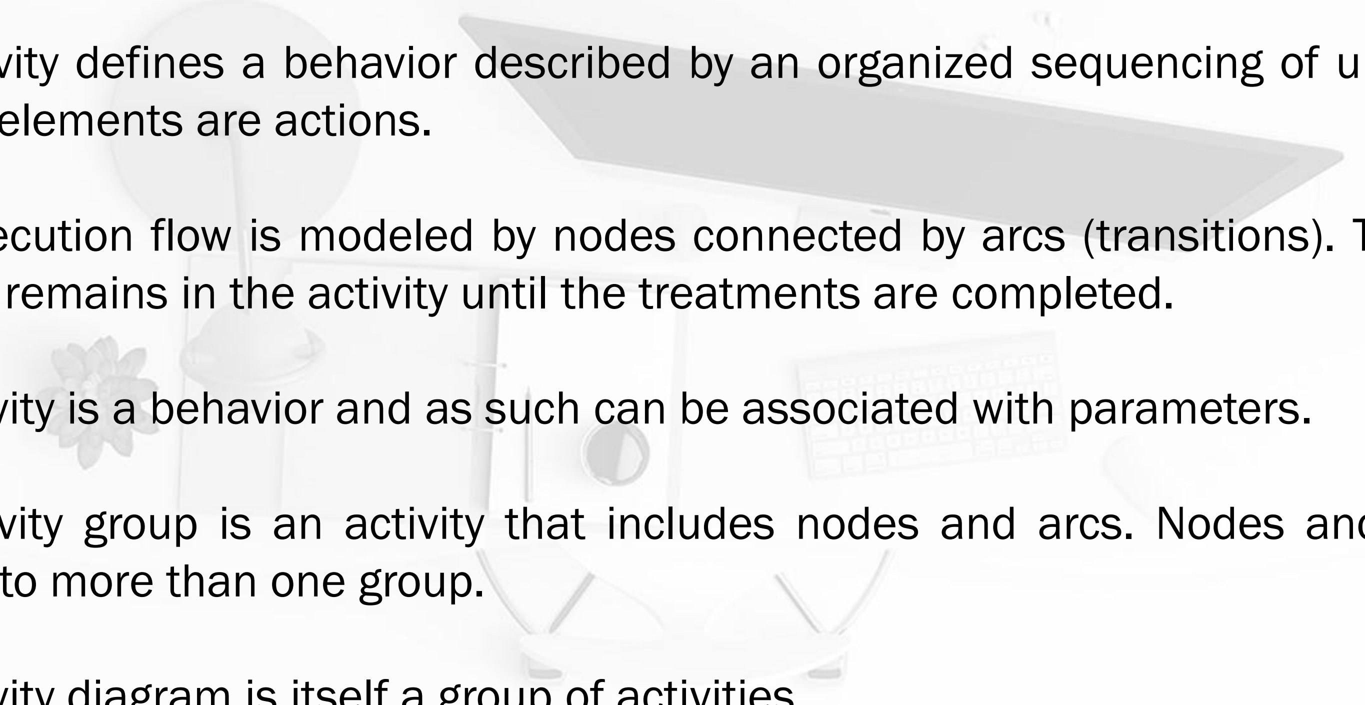


### 2- a- Action:

- An action is the smallest processing that can be expressed in UML.
- An action affects the state of the system or extracts information from it.
- The notion of action is to be compared to the notion of elementary instruction of a programming language (such as C++ or Java). An action can be, for example:
- Assigning value to attributes.
- Access to the value of a structural property (attribute or association termination).
- the creation of a new object or link;
- a simple arithmetic calculation;
- ...

## 03 Activity Diagram

### 2- b- Activity and activity group:

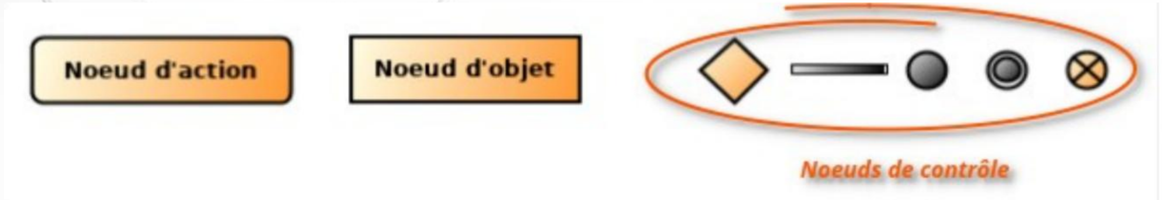


- An activity defines a behavior described by an organized sequencing of units whose simple elements are actions.
- The execution flow is modeled by nodes connected by arcs (transitions). The control stream remains in the activity until the treatments are completed.
- An activity is a behavior and as such can be associated with parameters.
- An activity group is an activity that includes nodes and arcs. Nodes and arcs can belong to more than one group.
- An activity diagram is itself a group of activities.

03 Activity Diagram

## 2- c- Activity nodes:

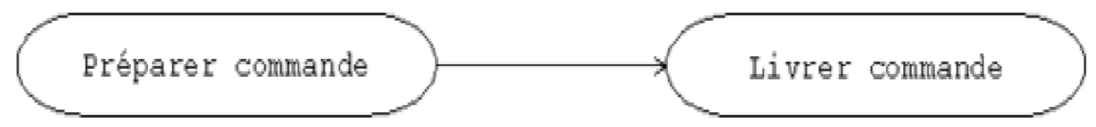
- An activity node is a type of abstract element used to represent the steps along the flow of an activity. There are three families of activity nodes:
  - ✓ Execution nodes or action
  - ✓ Object nodes
  - ✓ and control nodes
- Graphical representation of activity nodes: from left to right:
  - ✓ The node representing an action
  - ✓ An object node,
  - ✓ a decision or merge node,
  - ✓ a bifurcation or union node,
  - ✓ an initial node
  - ✓ A final node
  - ✓ a final node of flow.



# Activity Diagram

## 2- d- Transition:

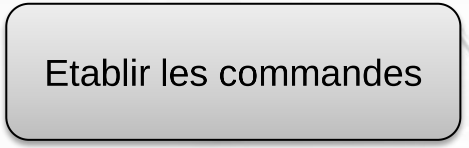
- The transition from one activity to another is materialized by a transition. Graphically, the transitions are represented by arrows in solid lines that connect the activities to each other.
- They are triggered as soon as the source activity is completed and automatically and immediately cause the next activity to be triggered (the target activity) to start.



## 03 Activity Diagram

### 2- c- 1. Actions node:

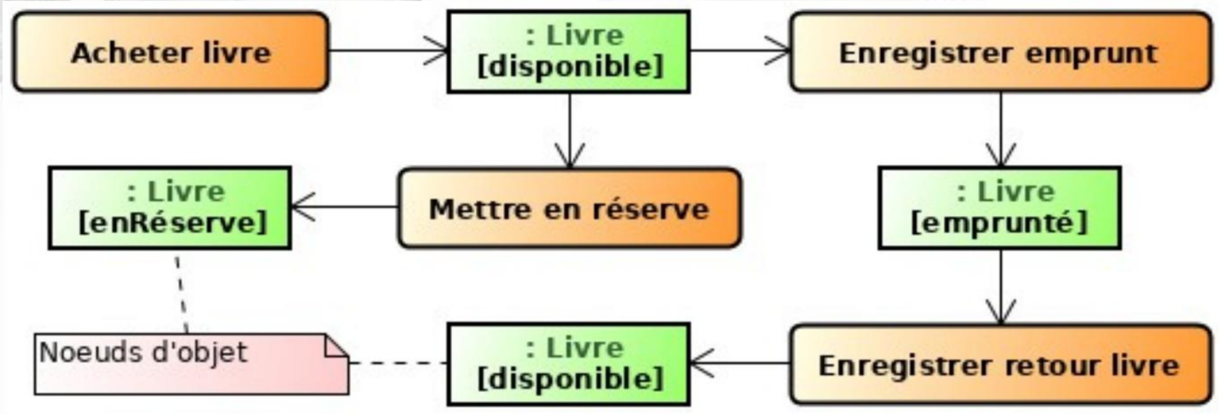
- An action node is an executable activity node that is the fundamental unit of executable functionality in an activity.
- The execution of an action represents a transformation or calculation of some kind in the modeled system.
- An action node must have at least one incoming arc.
- Graphically, an action node is represented by a rectangle with rounded corners that contains its textual description.



03 Activity Diagram

## 2- c- 2. Object node:

- Object nodes are used to define a flow of objects (data flow) in a flowchart. Each node represents the existence of an object generated by an action in an activity and used by other actions.
- Graphically, such an object node is represented by a rectangle in which the type of the object is mentioned. Arcs then connect this object node to source and target activities



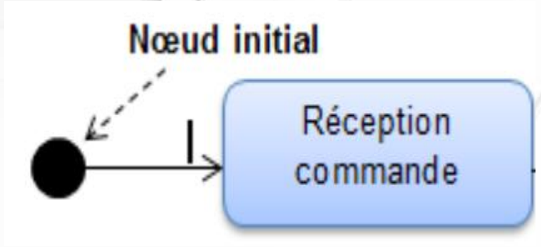
## 03 Activity Diagram

### 2- c- 3. Control node:

- A control node is an abstract activity node used to coordinate flows between nodes in an activity. There are several types of control nodes:

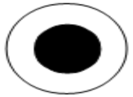
### 1. Initial node:

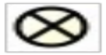
- An initial node is a control node from which the flow begins when the enveloping activity is invoked. An activity can have multiple initial nodes. An initial node has an outgoing arc and no incoming arc.
- Graphically, an initial node is represented by a small solid circle.



03 **Activity Diagram**

## 2. Final Node:

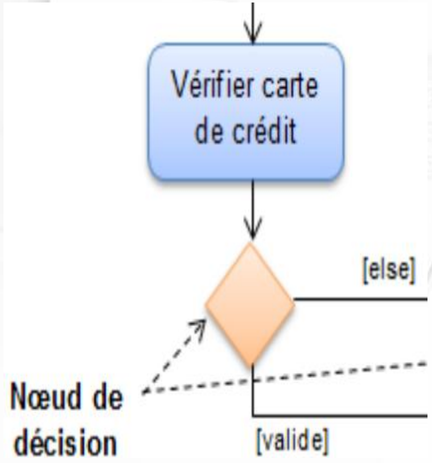
- A final node is a control node with one or more incoming arcs and no outgoing arcs. There are two types of final nodes:
- End of activity node: When one of the edges of an end of activity node is activated, the execution of the wrapping activity completes and any active nodes or flows within the wrapping activity are abandoned. Graphically, an end-of-activity node is represented by an empty circle containing a small full circle. 
- End of Flow Node: When a runstream reaches an end of flow node, the flow in question is complete, but that end of flow has no impact on the other active streams of the wrapping activity.
- Graphically, an end-of-flow node is represented by an empty circle crossed out b y an X



03 Activity Diagram

## 3. Decision node:

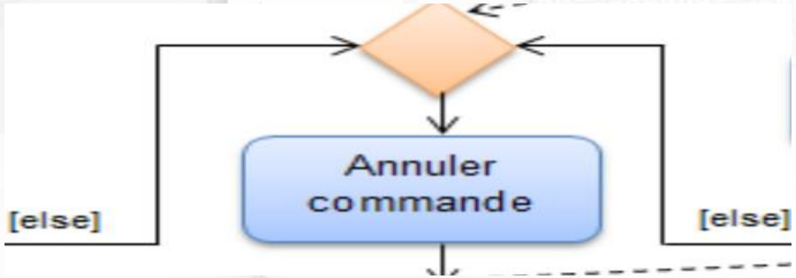
- A decision node is a control node that allows you to choose between several outgoing flows. It has an inbound arc and several outgoing arcs. These are generally accompanied by garde conditions to condition the choice.
- Graphically, we represent a decision node by a diamond.



03 Activity Diagram

## 4. Merge node:

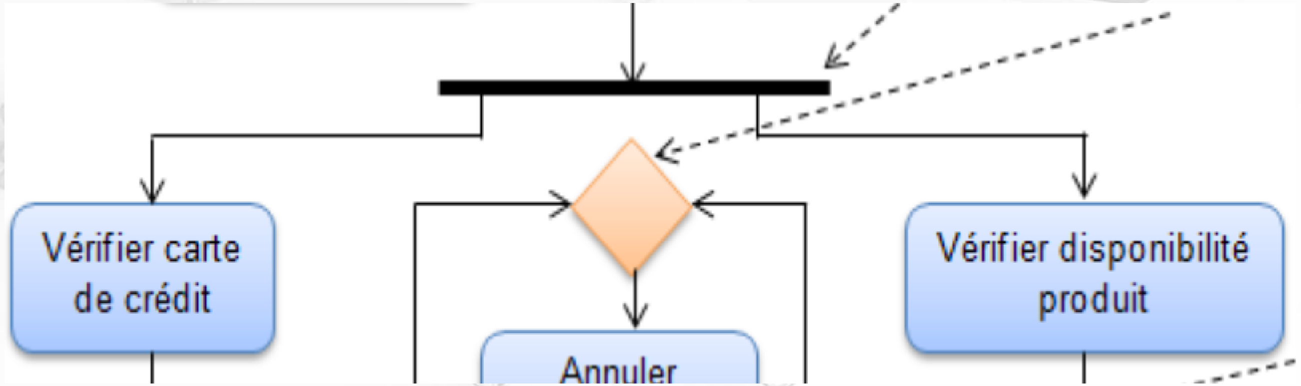
- A merge node is a control node that brings together multiple incoming alternate streams into a single outgoing stream. It is not used to synchronize concurrent streams (this is the role of the union node) but to accept one stream among several.
- Graphically, we represent a merge node, as a decision node, by a diamond.



03 **Activity Diagram**

## 5. Bifurcation node:

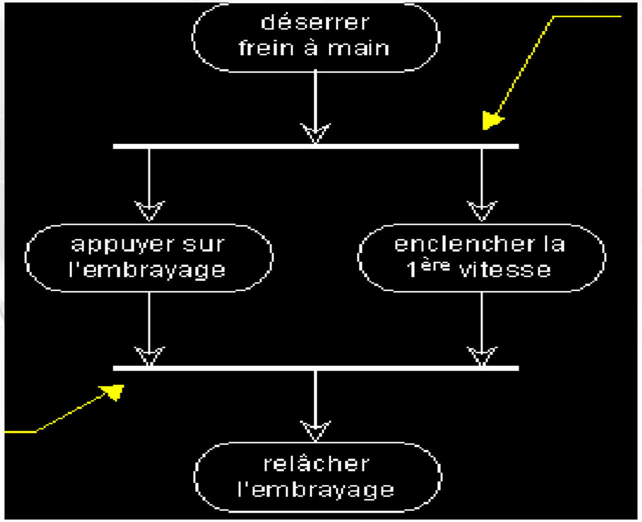
- A bifurcation node, also known as a disconnect node, is a control node that separates a stream into multiple concurrent streams. Such a node therefore has one inbound arc and several outbound arcs.
- Graphically, we represent a bifurcation node with a solid line



03 Activity Diagram

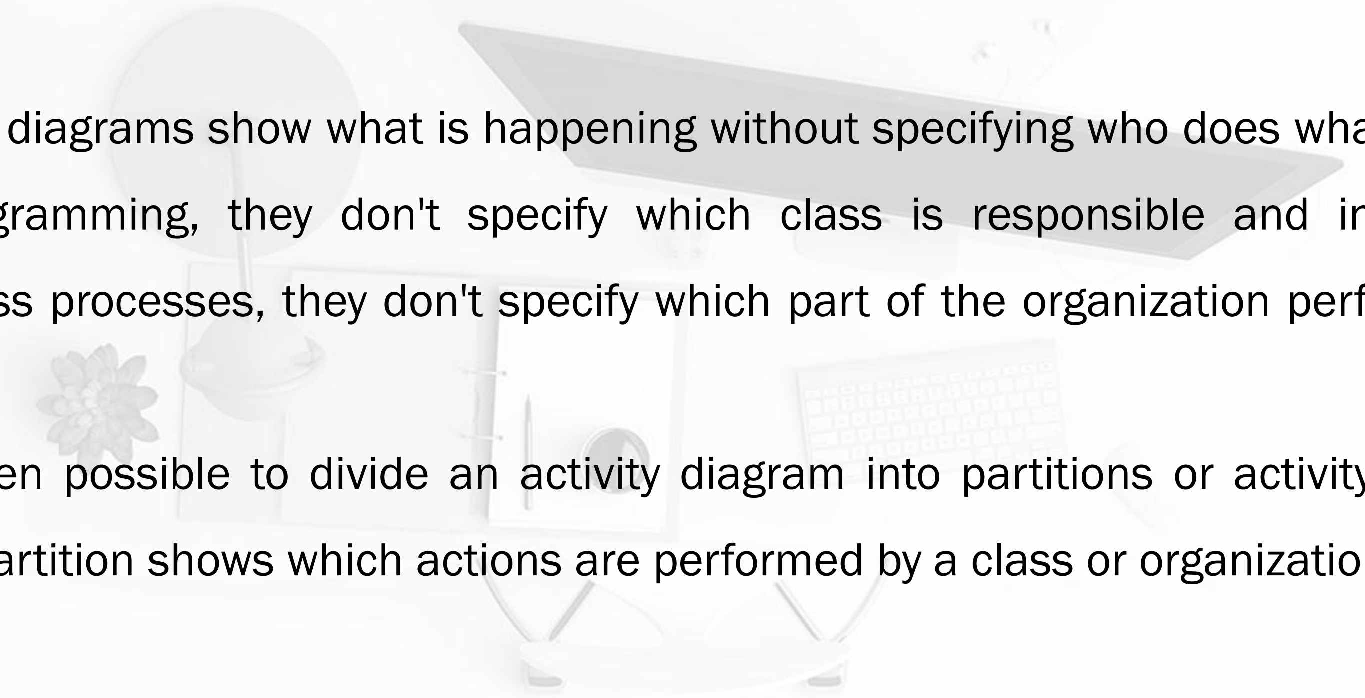
## 6. Union node:

- A union node, also known as a join node, is a control node that synchronizes multiple streams. Such a node therefore has several incoming arcs and a single outgoing arc.
- Graphically, we represent a union node, like a bifurcation node, by a solid line.



## 03 Activity Diagram

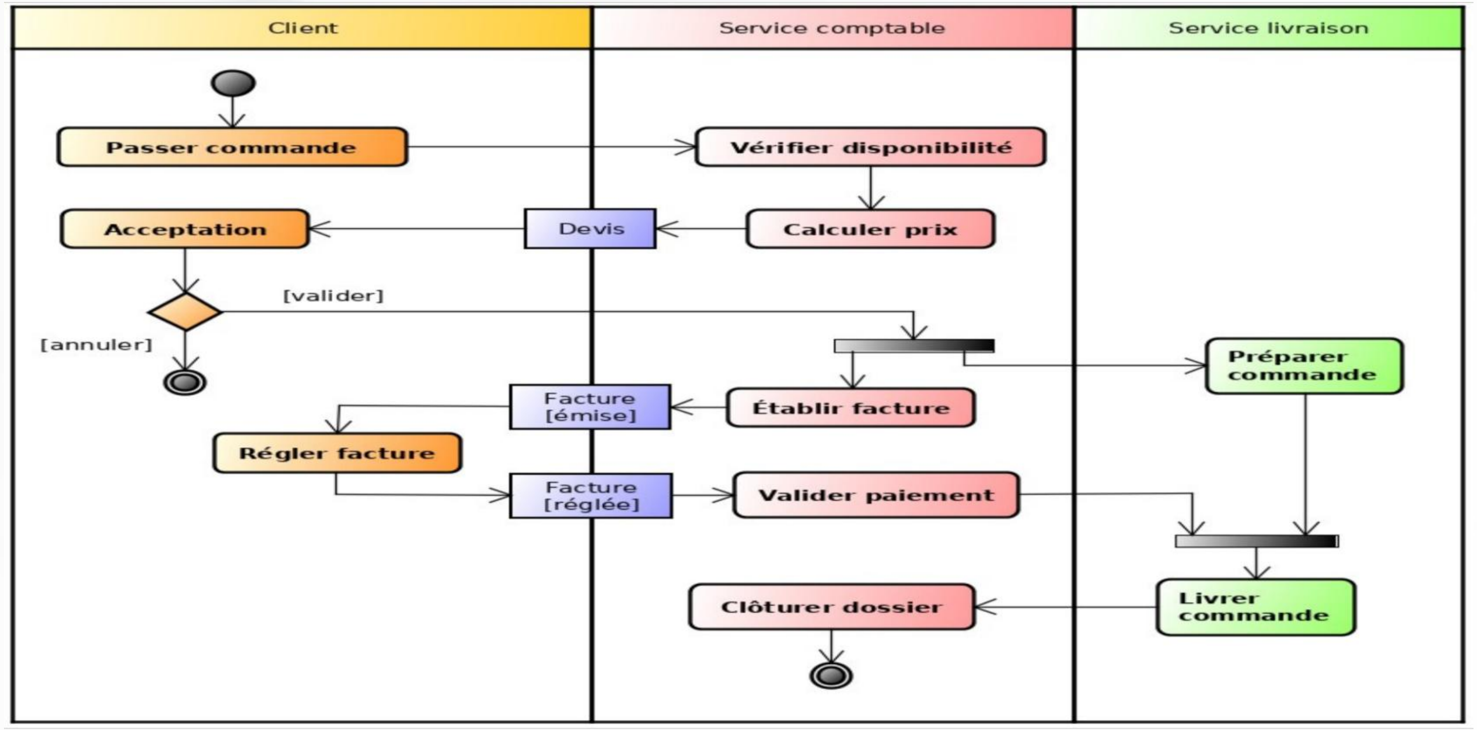
### 7. Partitions or activity corridors:



- Activity diagrams show what is happening without specifying who does what (in terms of programming, they don't specify which class is responsible and in terms of business processes, they don't specify which part of the organization performs each action).
- It is then possible to divide an activity diagram into partitions or activity corridors. Each partition shows which actions are performed by a class or organizational unit.

03 Activity Diagram

## 7. Partitions or activity corridors: example



## 04 Transition State Diagram

### 1 Definition:

- Describes the behavior of objects in a class using a state automaton associated with the class

- It is desirable to construct a state-transition diagram for each workbook (which is us ually a class) with significant dynamical behavior. A state-transition diagram can onl y be associated with one class

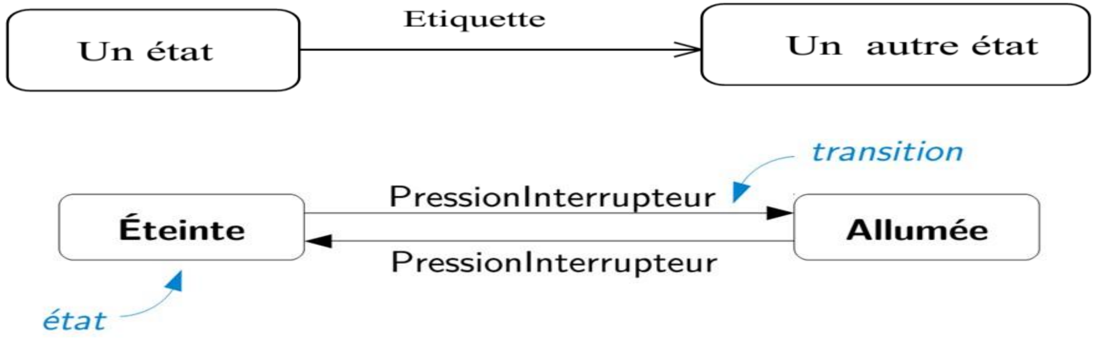
- The behavior is modeled by a graph:
  - Nodes = possible states of objects
  - Arcs = state-to-state transitions

## 03 Transition State Diagram

### 2- Graphic notation:

- State-transition diagrams visualize finite-state automata, from the point of view of st ates and transitions.

- States are represented by rectangles with rounded corners, while transitions are rep resented by oriented arcs linking states together.



## 03 Transition State Diagram

### 3- State:

- An object can go through a series of states during its lifetime
- A state represents a period in the life of an object during which it awaits an event or performs an activity
- The configuration of the overall state of the object is the set of (elementary) states t hat are active at a given time
- The name of the state can be specified in the rectangle and must be unique in the di agram of transitions, or in the wraparound state

## 03 Transition State Diagram

### 3-1. State Types:

#### 1. Initial state:

- The initial state is a pseudostate that indicates the starting state, by default, when t he Diagram of transition-states, or the wrapping state, is invoked
- When an object is created, it enters the initial state

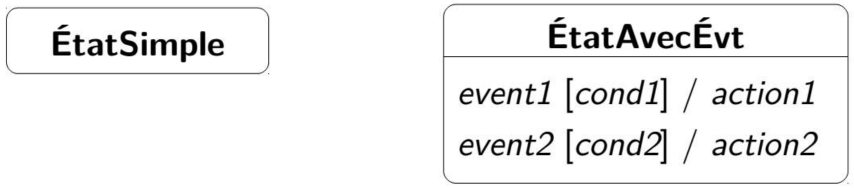
#### 2. End state:

- The final state is a pseudostate that indicates that the state-transition diagram, or w raparound state, is complete

## 03 Transition state diagram

### 3. Intermediate state:

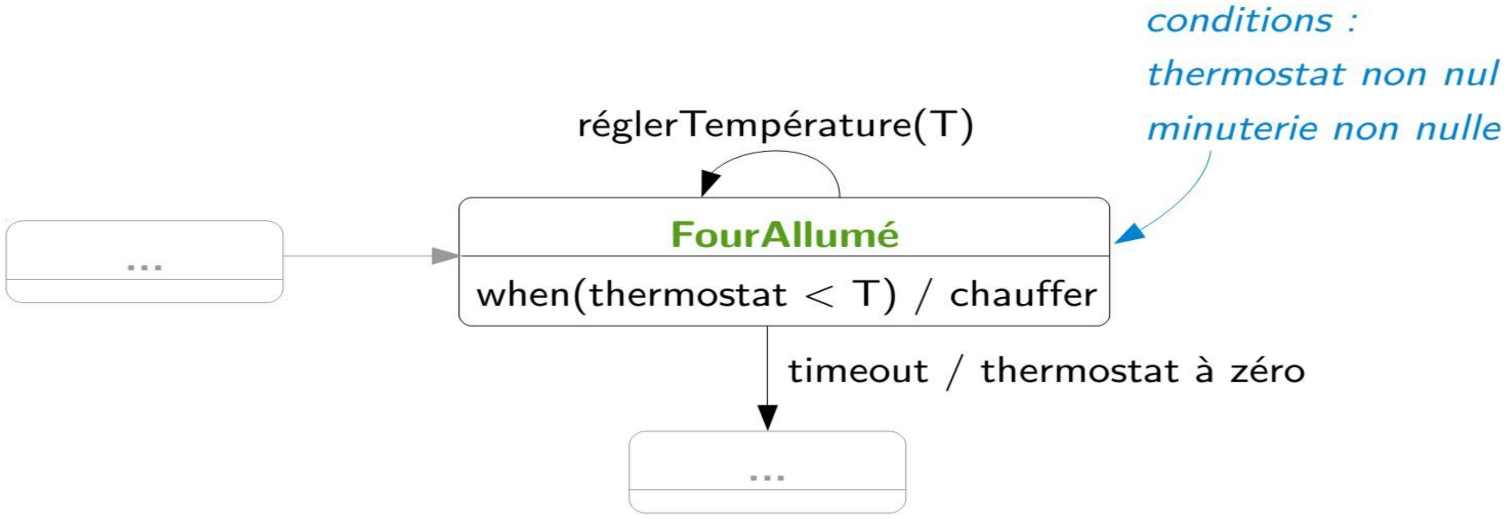
- Reports can also contain actions; They are executed when entering or exiting the state or when an event occurs while the object is in the state in question. Like classes, state scan be divided into buckets that contain:
  - The "no" of the state;
  - The list of internal actions or activities performed while the item is in this state.



## 03 Transition State Diagram

### 3-2. Condition characteristics:

#### 3.2.1. Conditions:



## 03 Transition State Diagram

### 3-2. Condition characteristics:

#### 3.2.2. Events:

- An instantaneous event coming from outside the system and occurring at a given mo ment Types of events:
  - Signal : Receiving an asynchronous message
  - Call of a (synchronous) operation: linked to use cases, class diagram operation, etc...
  - Satisfying a Boolean condition: when(cond), evaluated continuously until true
  - Time

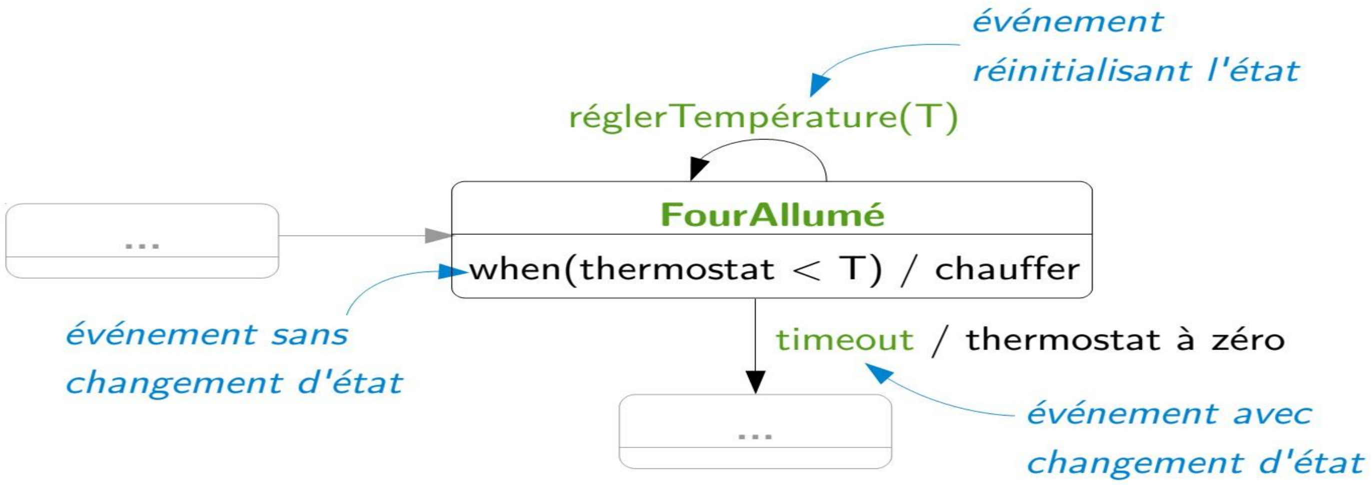
    Relative date : when(date = date)

    Absolute date: after(duration)

## 03 Transition State Diagram

### 3-2. Condition characteristics:

#### 3.2.2. Events:


événement réinitialisant l'état
réglerTempérature(T)
FourAllumé
when(thermostat < T) / chauffer
...
événement sans changement d'état
timeout / thermostat à zéro
...
événement avec changement d'état

# 03 Transition State Diagram

## 3-2. Condition characteristics:

### 3.2.3. Action:

- Action : System reaction to an event

- Characteristics : atomic, instantaneous, uninterruptible

- Examples of actions:
  - affectation
  - Sending a signal
  - Calling an operation
  - Creating or Destroying an Object

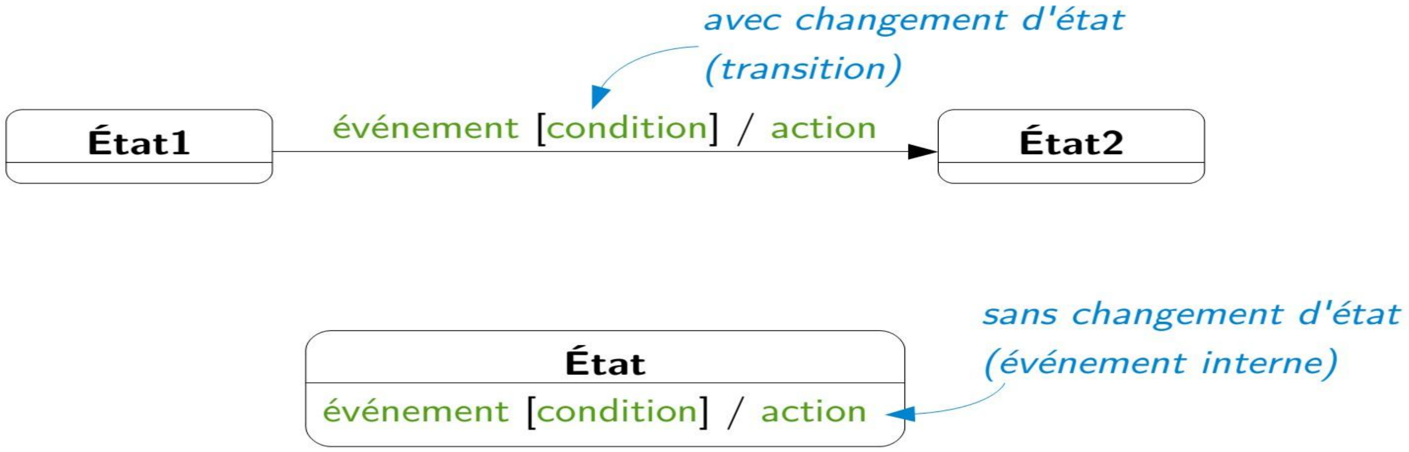
# 03 Transition State Diagram

## 3-2. Condition characteristics:

### 3.2.3. Action:

événement [condition] / action

**When the event occurs, if the condition is true, then the action is performed**



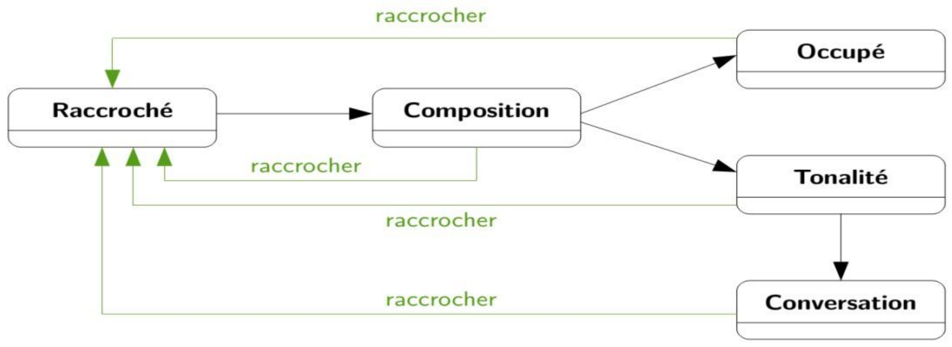
# 03 Transition State Diagram

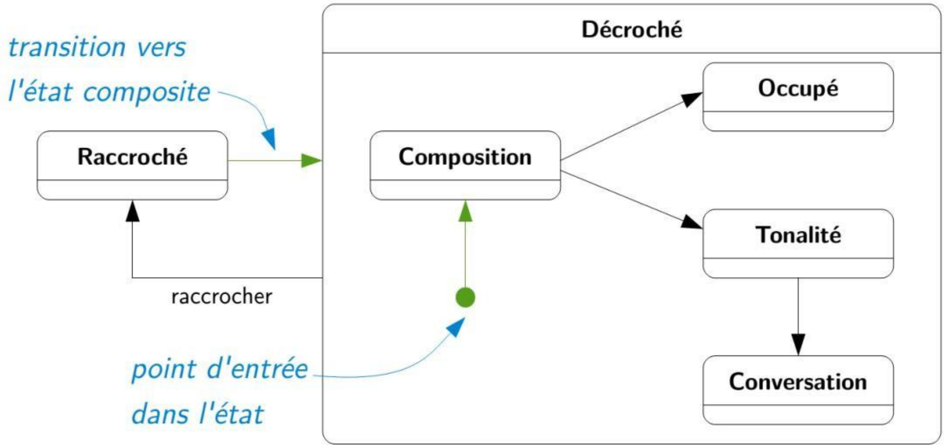
## 3.3. Composite state:

A state that includes a set of states Objecti

ves :

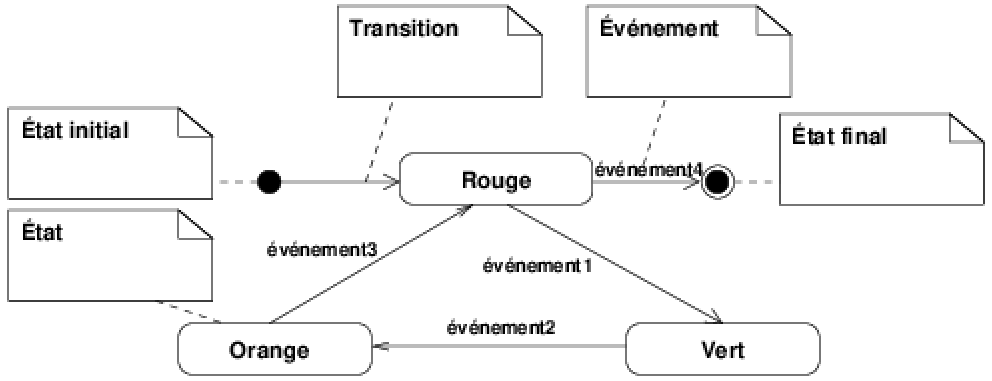
- Prioritize states
- Structure complex behaviors
- Factoring actions





## 03 Transition State Diagram

### 4. Exemple:



## Bibliographies

- **Uml 2 pratique de la modélisation**, Benoît Charroux, Yann Thierry-Mieg, Aomar Osmani https://fr.slideshare.net/nassimamine3994/uml-2-pratique-de-la-modlisation
- **Uml 2 par la pratique, Pascal roques**
- **Les cahiers du programmeur, Pascal roques**
- **Uml en action, Pascal roques**
- **.....**