MUE107C
Project Work 1 for DSAI
UML

Cecilia Zanni-Merk

Scientific Coordinator of the DSAI master at UFAZ

Resources in the Web

- Tutorials
  - https://www.tutorialspoint.com/uml/index.htm
  - https://sparxsystems.com/resources/tutorials/uml/part1.html

- UML editors
  - https://staruml.io/
  - https://plantuml.com/en/
  - https://www.bouml.fr/
  - http://www.pacestar.com/edge/
  - https://www.eclipse.org/papyrus/

Agenda

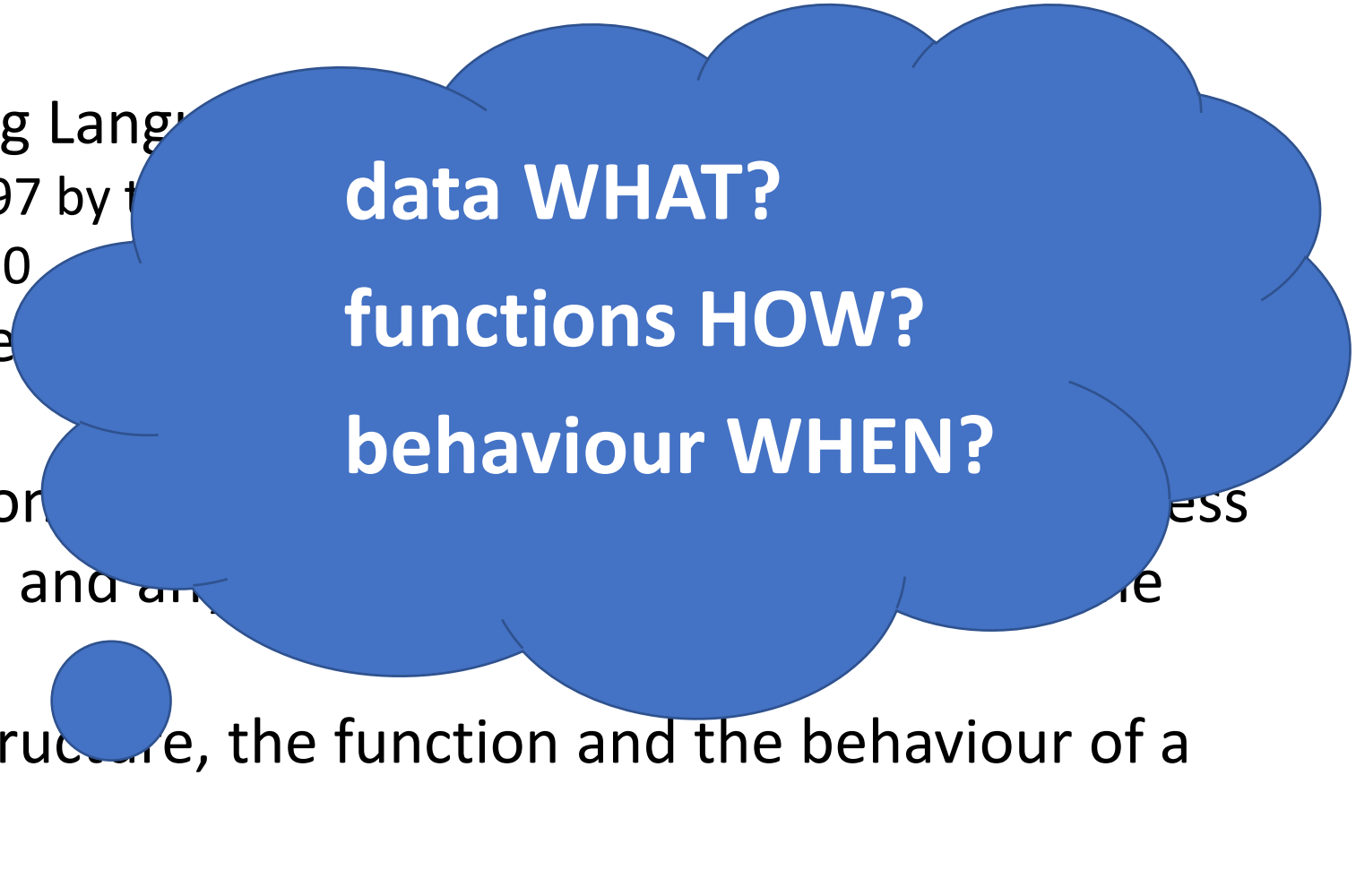
- UML
- Simplified SE approach with UML
- UML by the example
- Exercices

UML

- UML = Unified Modeling Language
  - Initially proposed in 1997 by the Object Management Group (OMG)
  - Latest version is UML 2.0
- It’s a graphical language for modeling objects, in a set of different diagrams
- UML diagrams are not only made for developers but also for business users, common people, and anybody interested to understand the system.
- UML can express the structure, the function and the behaviour of a program

UML

- UML = Unified Modeling Lang
  - Initially proposed in 1997 by t
  - Latest version is UML 2.0
- It's a graphical language
  diagrams
- UML diagrams are not or
  users, common people, and a
  system.
- UML can express the struc e, the function and the behaviour of a program

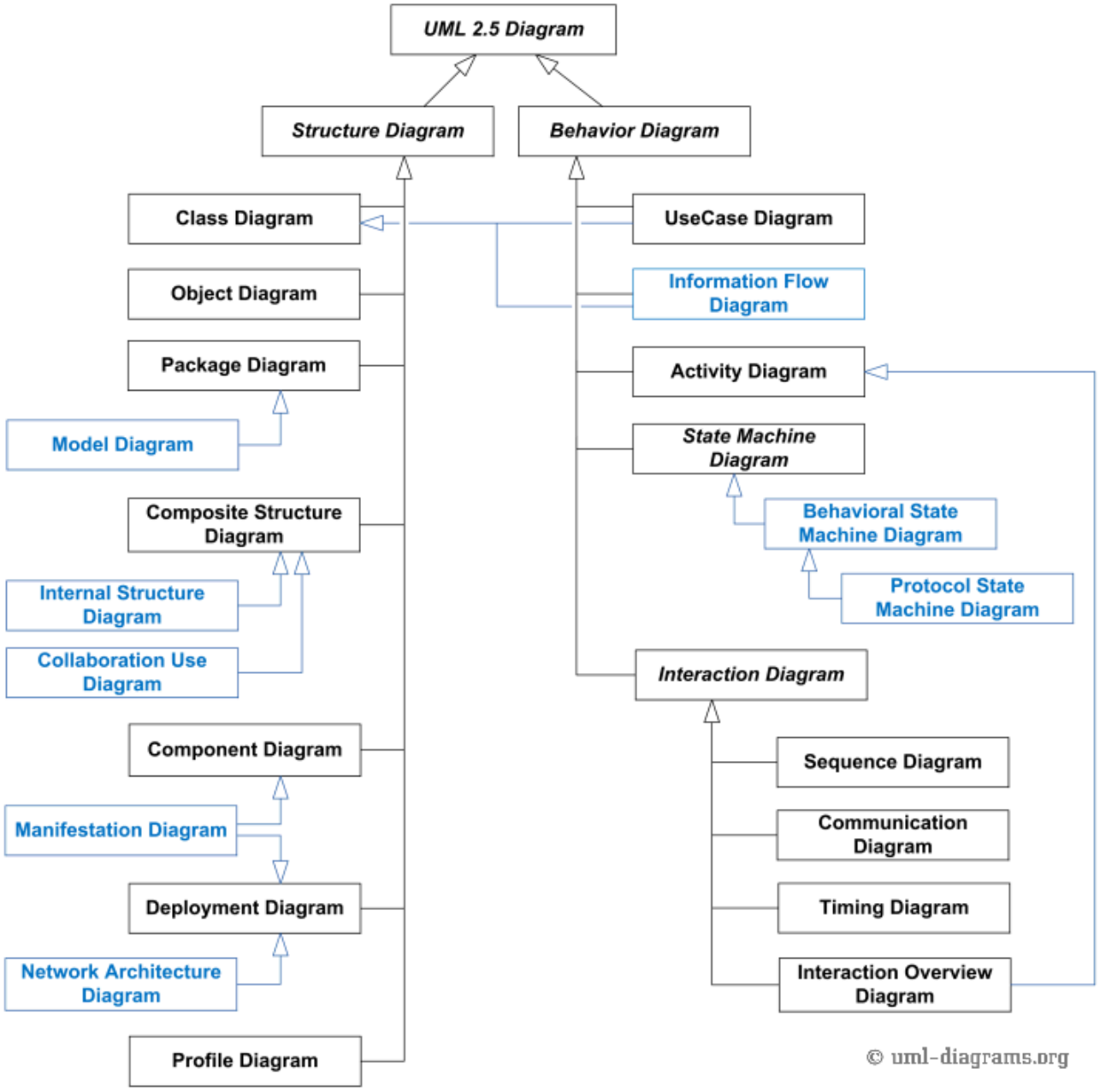


Official definition

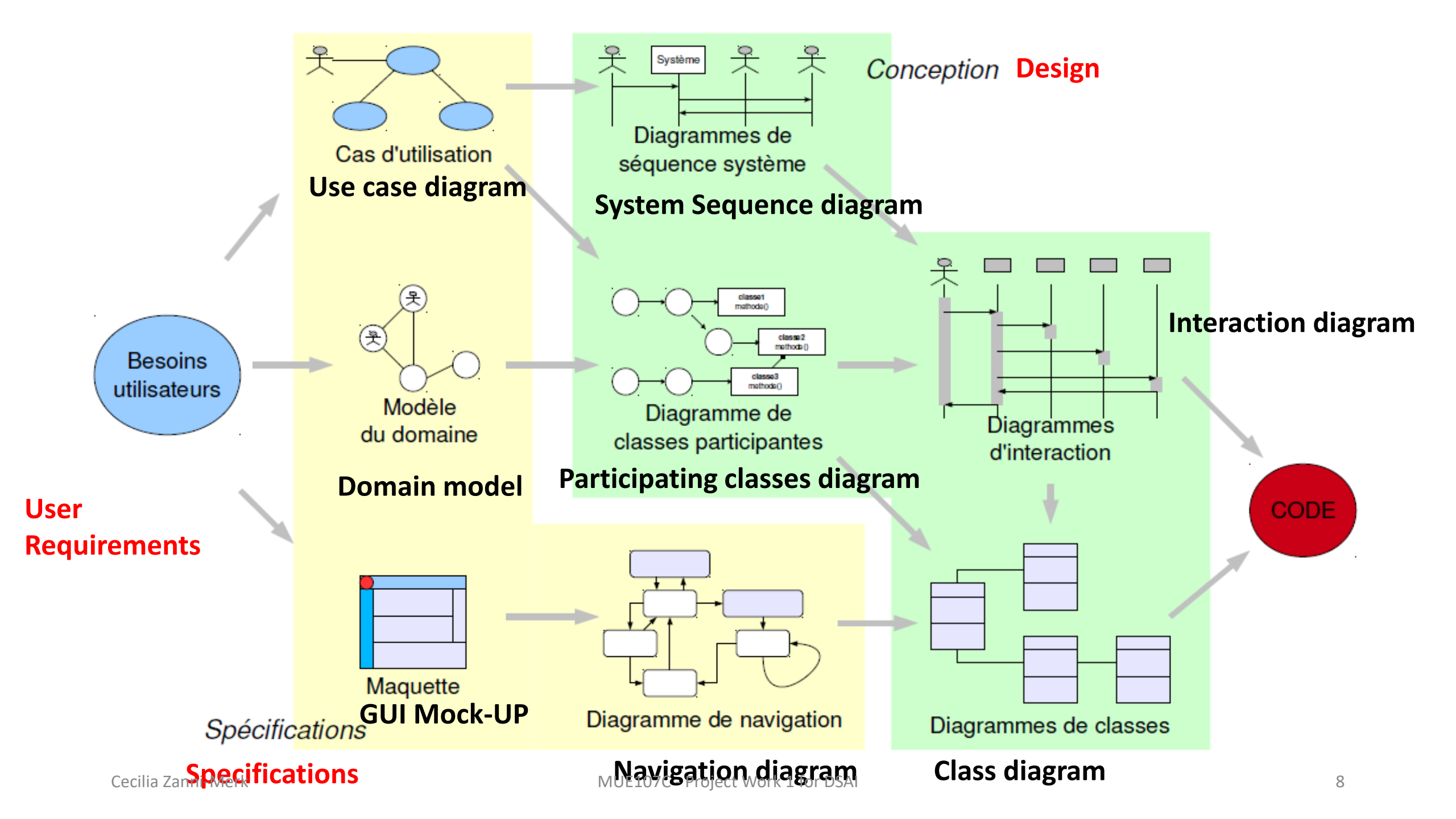
- The Object Management Group (OMG) specification states:

"The Unified Modeling Language (UML) is a ***graphical language*** for ***visualizing, specifying, constructing, and documenting*** the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components."

Summary UML 2.0



Simplified SE approach with UML


Conception
Design
Cas d'utilisation
Use case diagram
Système
Diagrammes de
séquence système
System Sequence diagram
Besoins
utilisateurs
Modèle
du domaine
Domain model
classe1
methode()
classe2
methode()
classe3
methode()
Diagramme de
classes participantes
Participating classes diagram
Diagrammes
d'interaction
Interaction diagram
CODE
User
Requirements
Maquette
GUI Mock-UP
Spécifications
Specifications
Diagramme de navigation
Navigation diagram
Diagrammes de classes
Class diagram

Specifications : identification of user requirements

Requirements are modeled by a use case diagram.

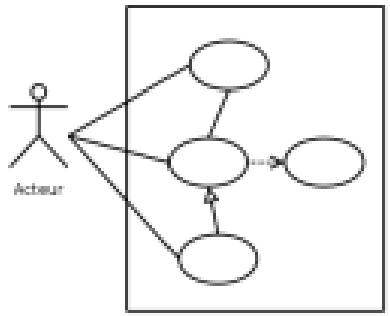
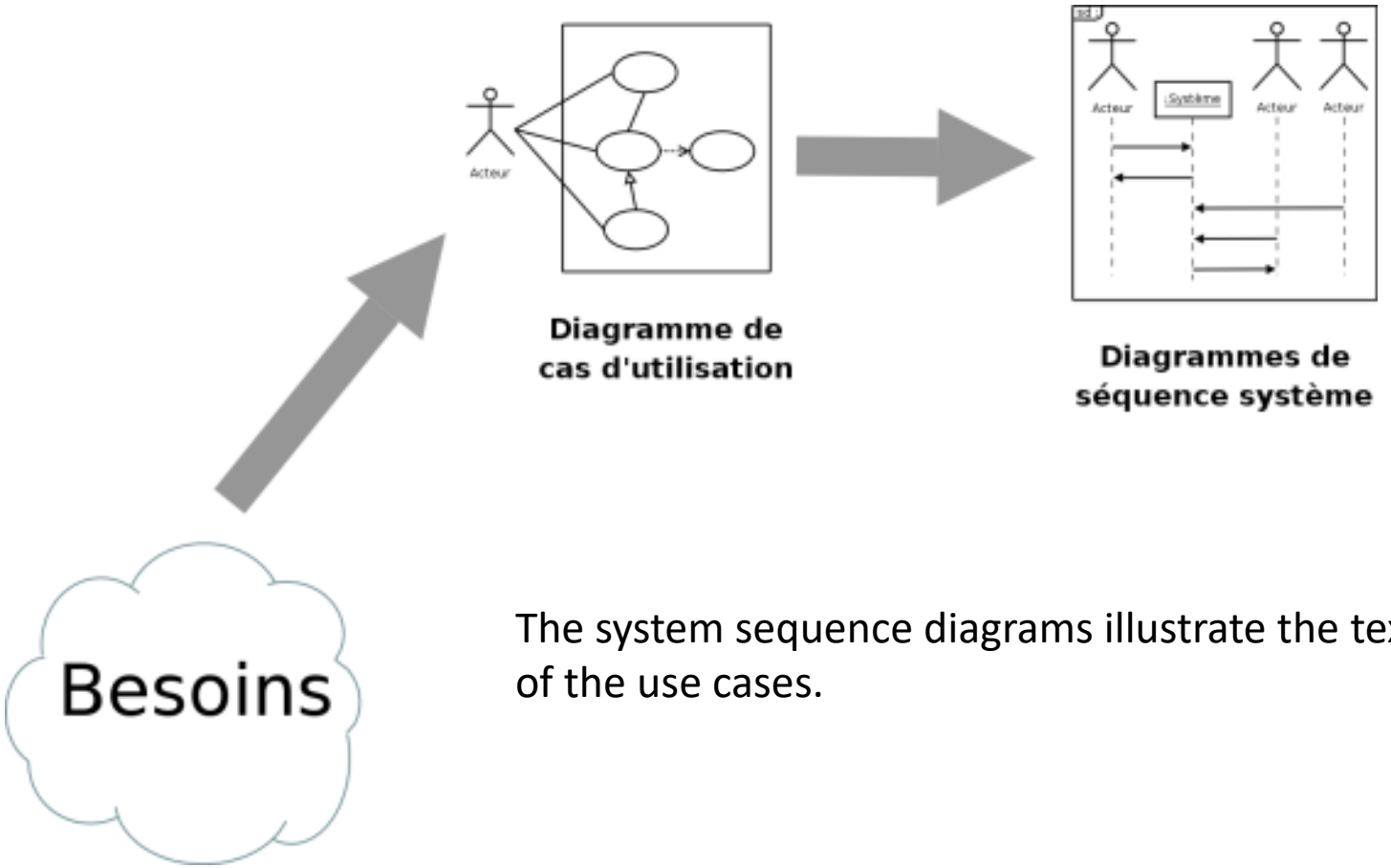


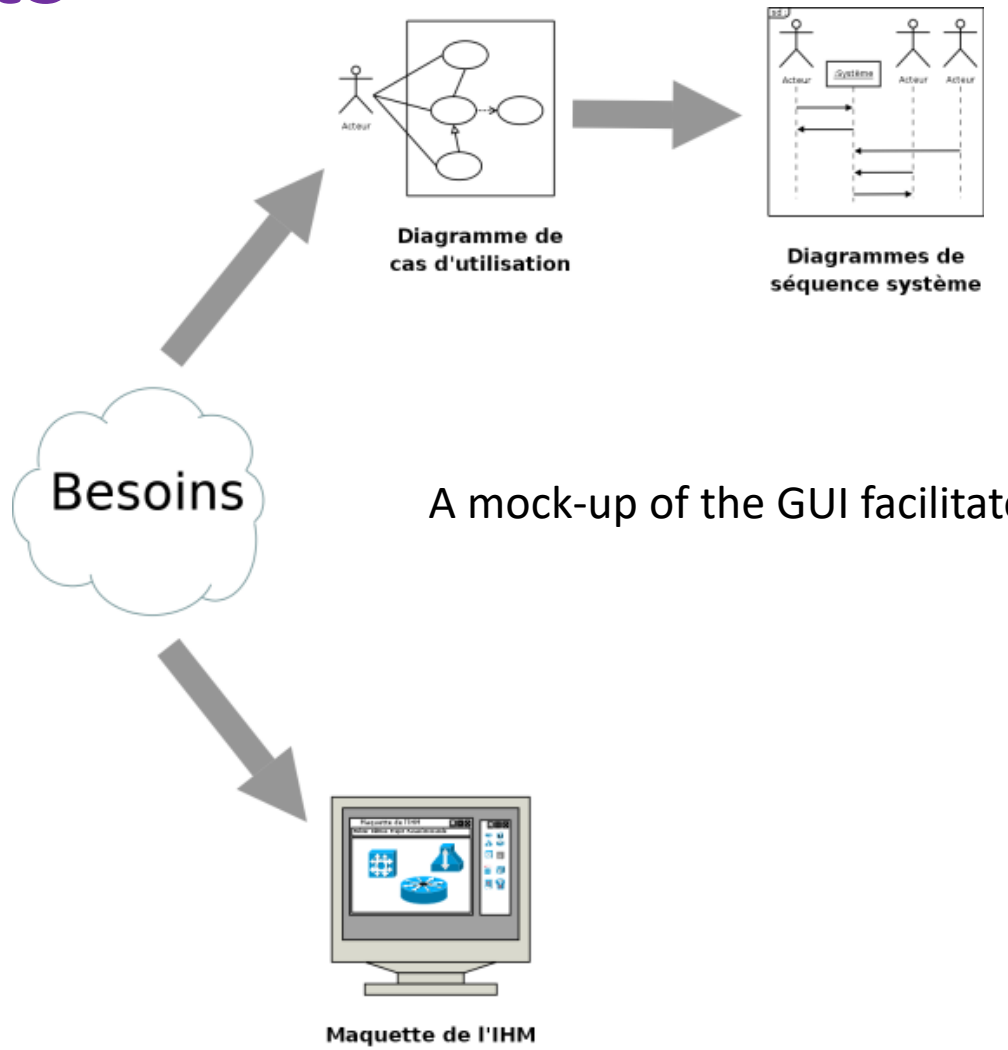
Diagramme de cas d'utilisation

Specifications : identification of user requirements



The system sequence diagrams illustrate the textual description of the use cases.

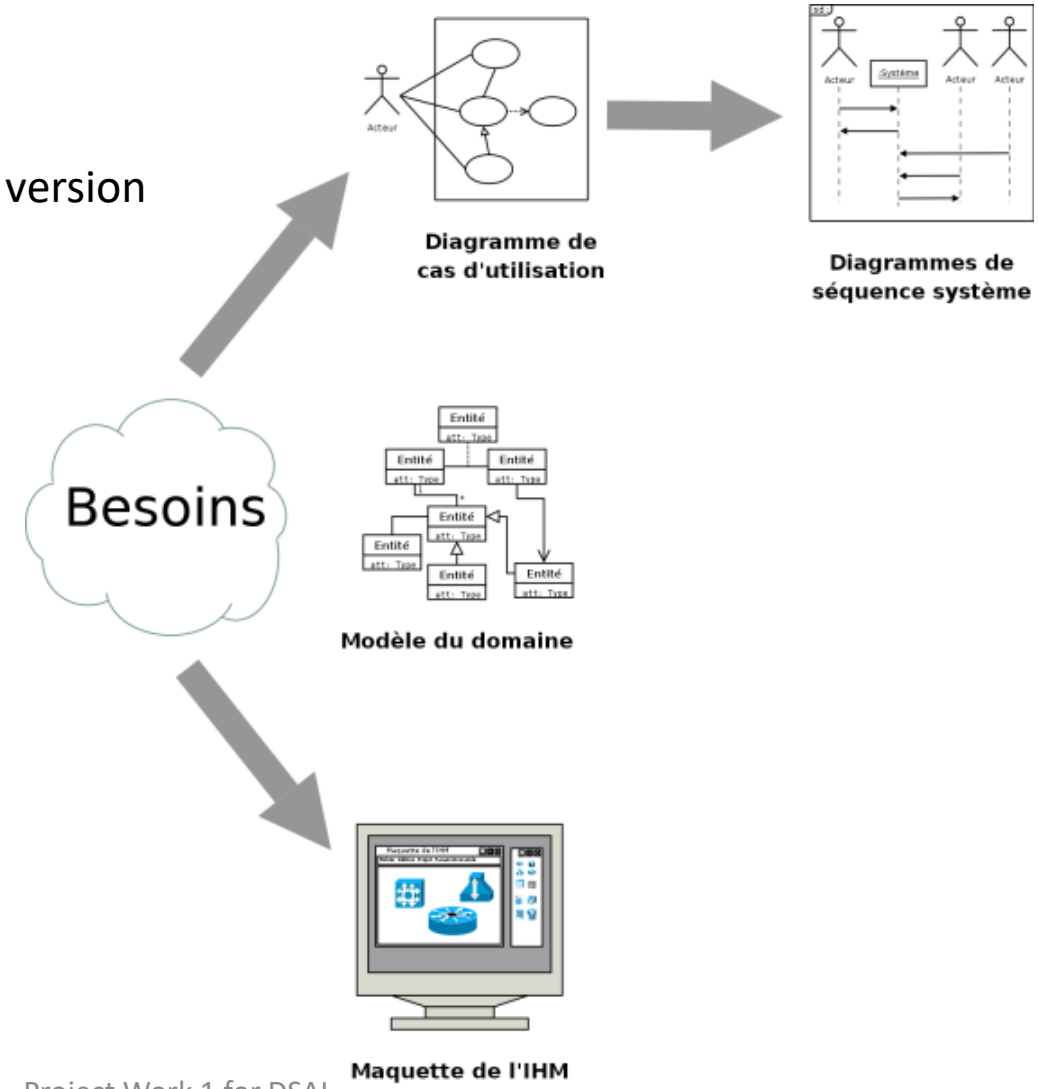
Specifications : identification of user requirements



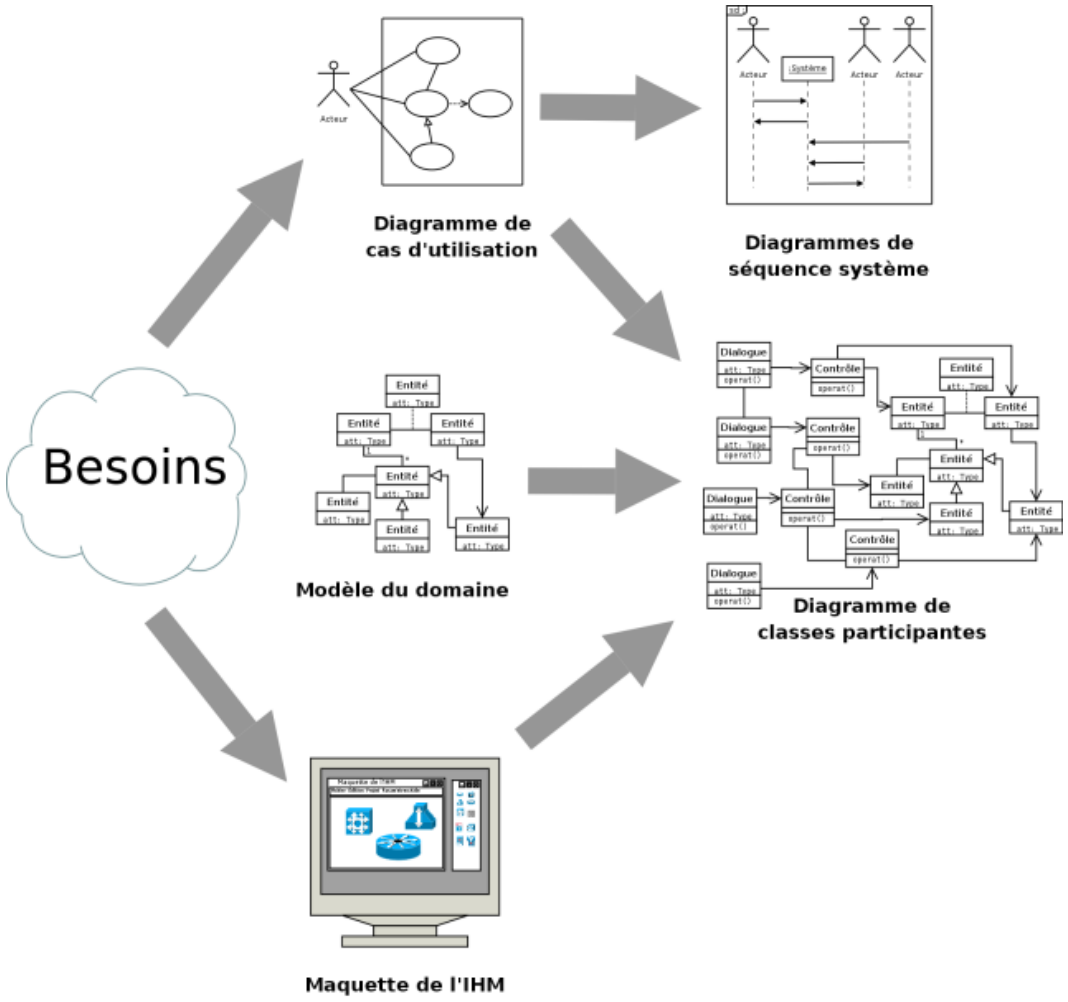
A mock-up of the GUI facilitates discussions with future users.

Specifications

The domain analysis phase is used to develop the first version of the class diagram.

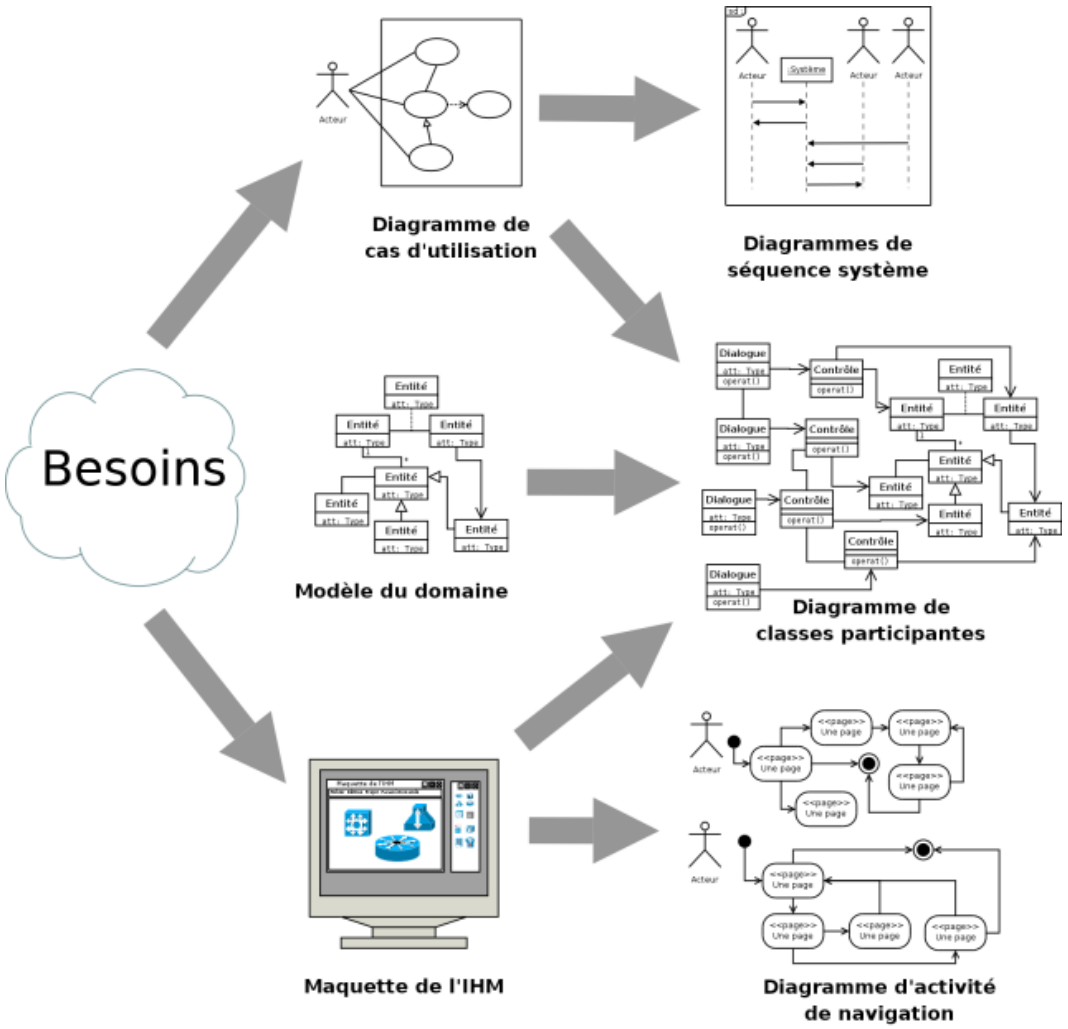


Specifications



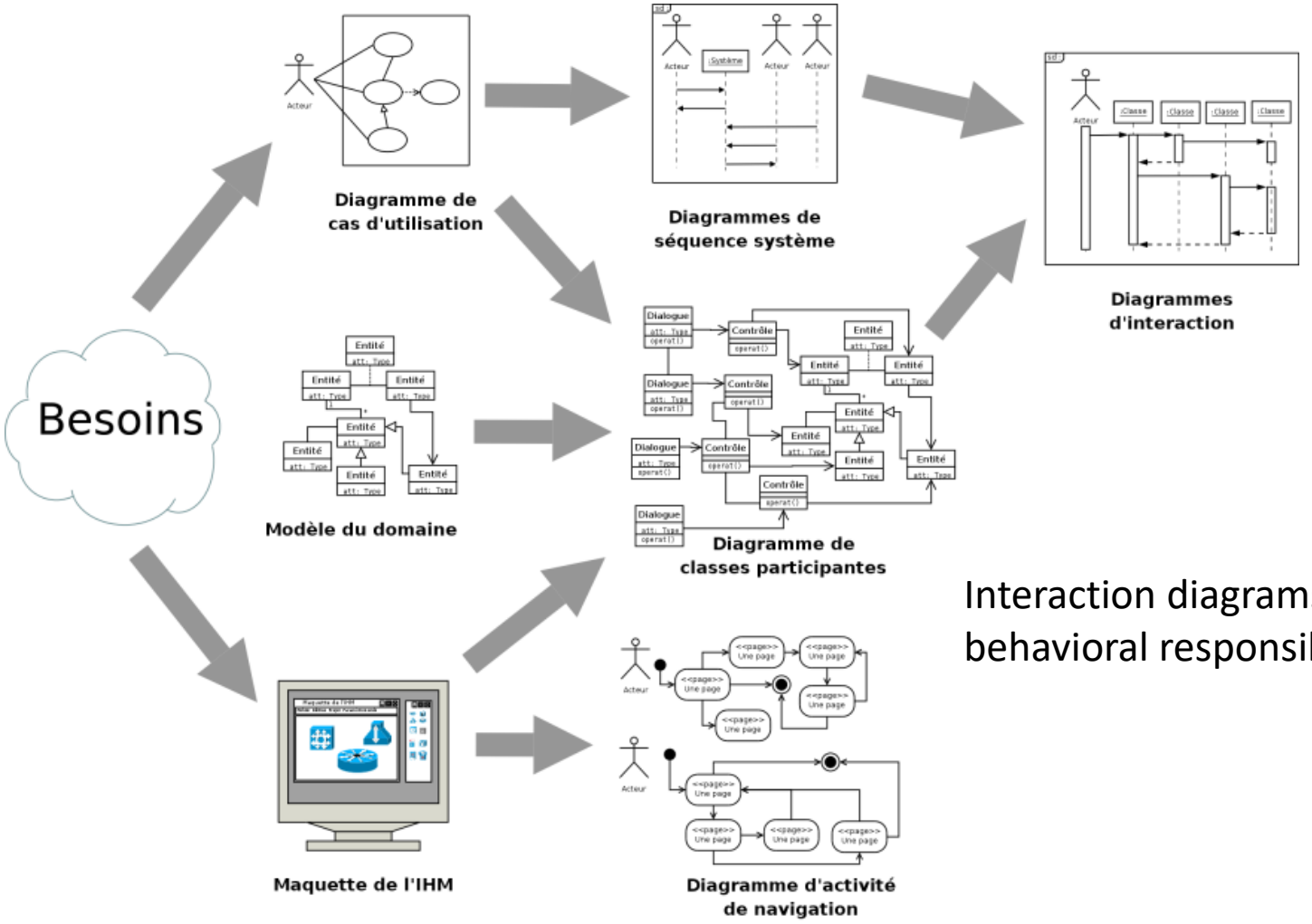
The participating class diagram connects the use cases, the domain model and the mock-ups.

Specifications



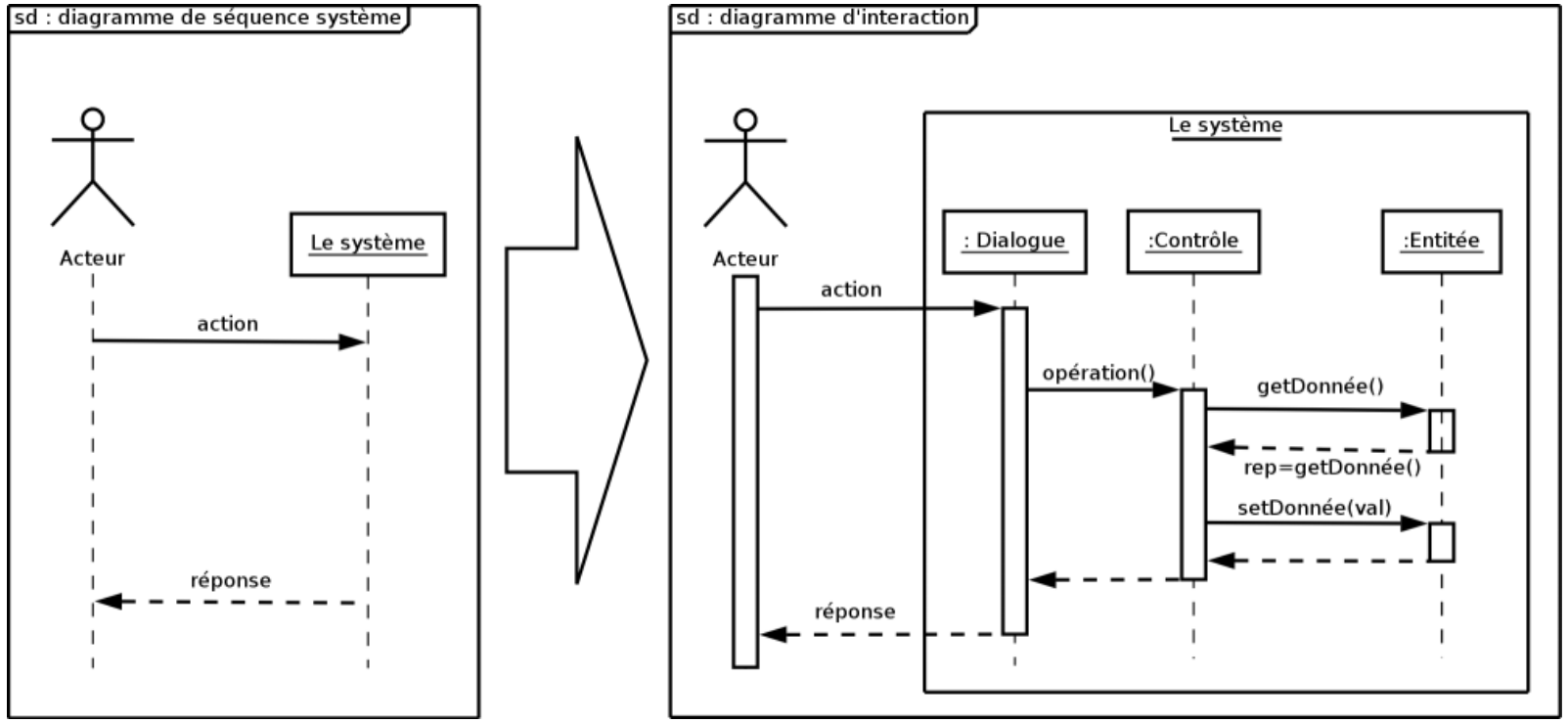
Navigation activity diagrams graphically represent the navigation activity in the GUI

Design phase



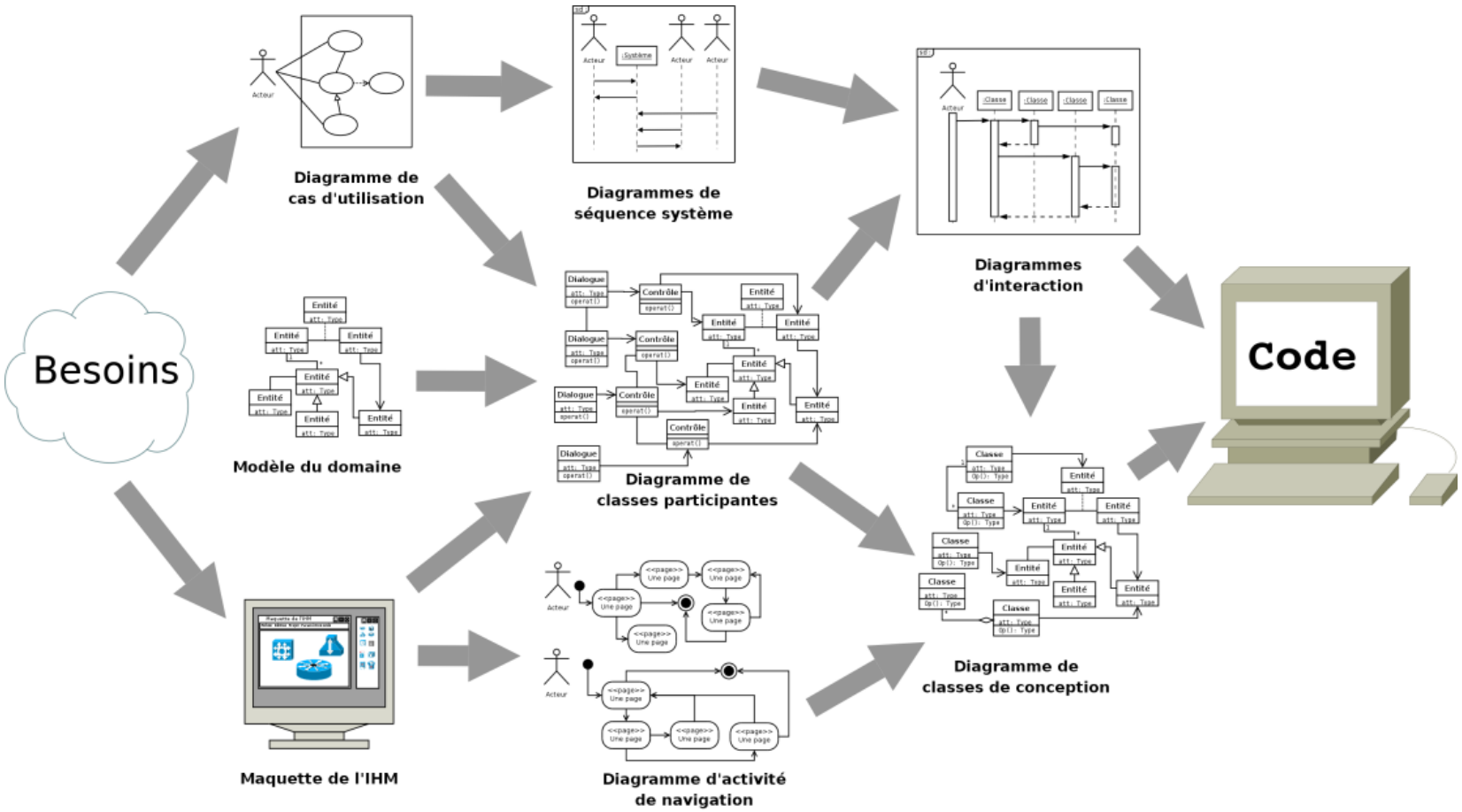
Interaction diagrams make it possible to precisely assign behavioral responsibilities to analysis classes.

Design phase



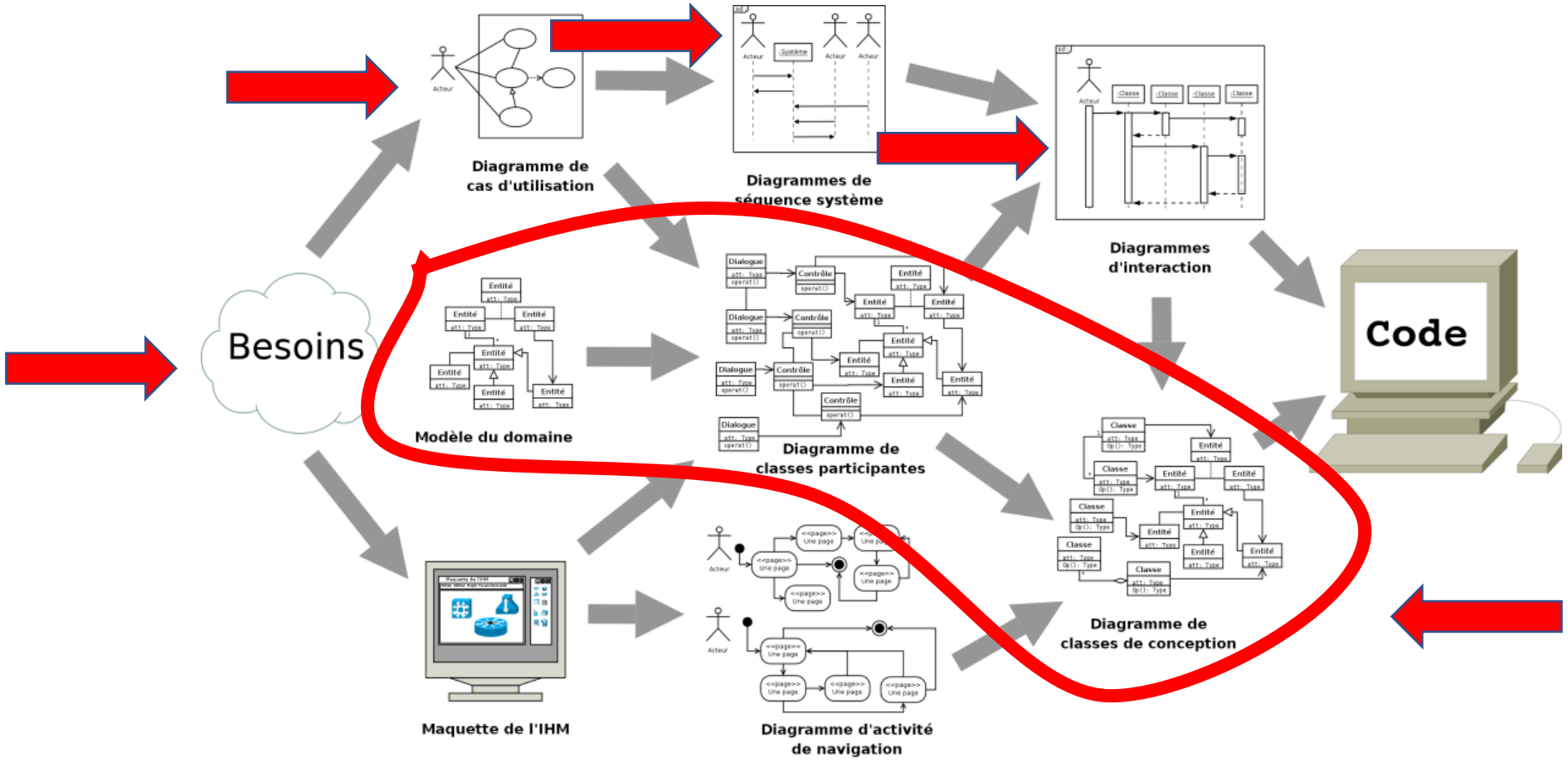
The set of system sequence diagrams, seen as a black box, is replaced by a set of collaborative objects.

Design phase



The design class diagram will be used for the implementation

We will concentrate on …



UML by the example

User requirements

Develop software for boat management in a marina

Specification: Goals of the software

- In a marina, there are sailing boats and outboard boats. The boats have a home port and move from port to port.
- ***We want to keep track of the boats' movements and be able to contact their owner***

Specification: Analysis of the needs

- This is the list of features they want to see integrated into the software.
- In general, with the help of the "use cases" of the UML
- Use case diagrams are UML diagrams used to give an overview of the functional behavior of a software system.

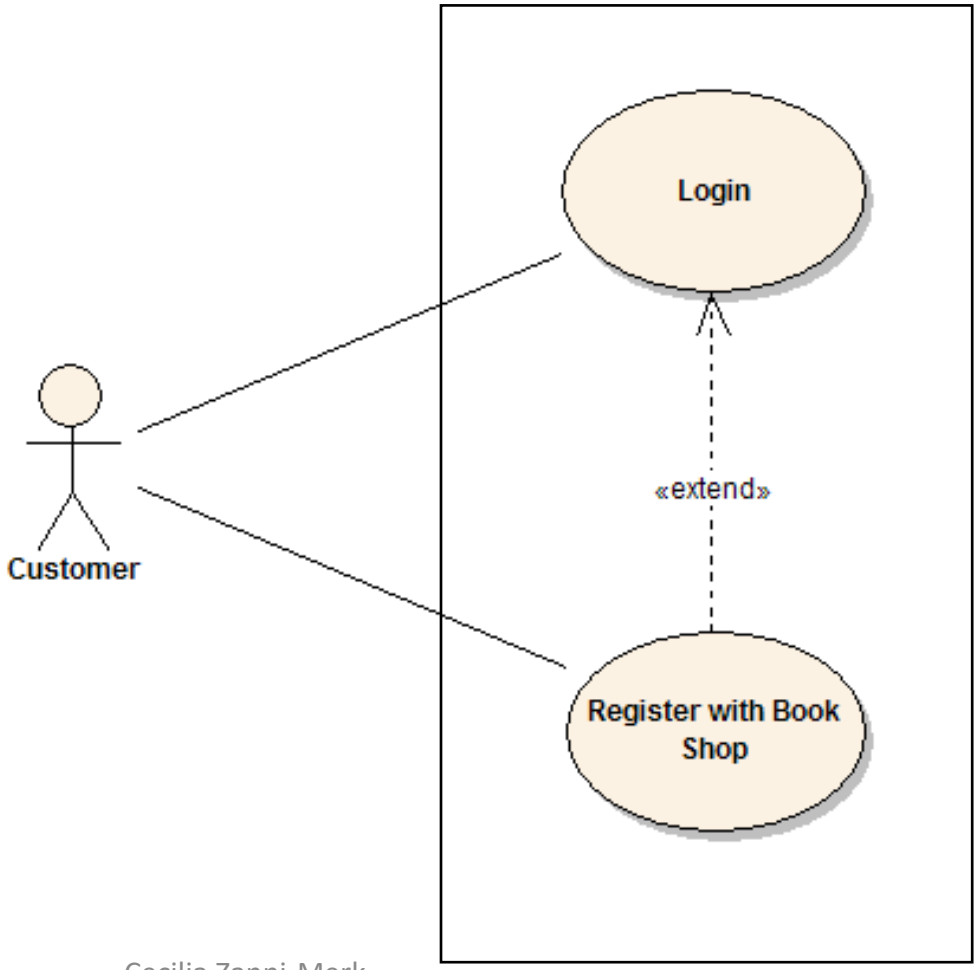
Specification: Analysis of needs

1. Add / delete / modify a boat
2. Find the owner of a boat
3. Calculate the annual tax of a boat in its home port. The tax depends on the length of the boat, and its power if the boat is motorized
4. List the boats for a given home port
5. List the ports visited by a boat
6. Make the history of a boat's movements
7. List the ports sorted by number of attached boats
8. Send a letter to each owner of a boat in a port specifying the amount of the tax to pay

Use Case diagrams

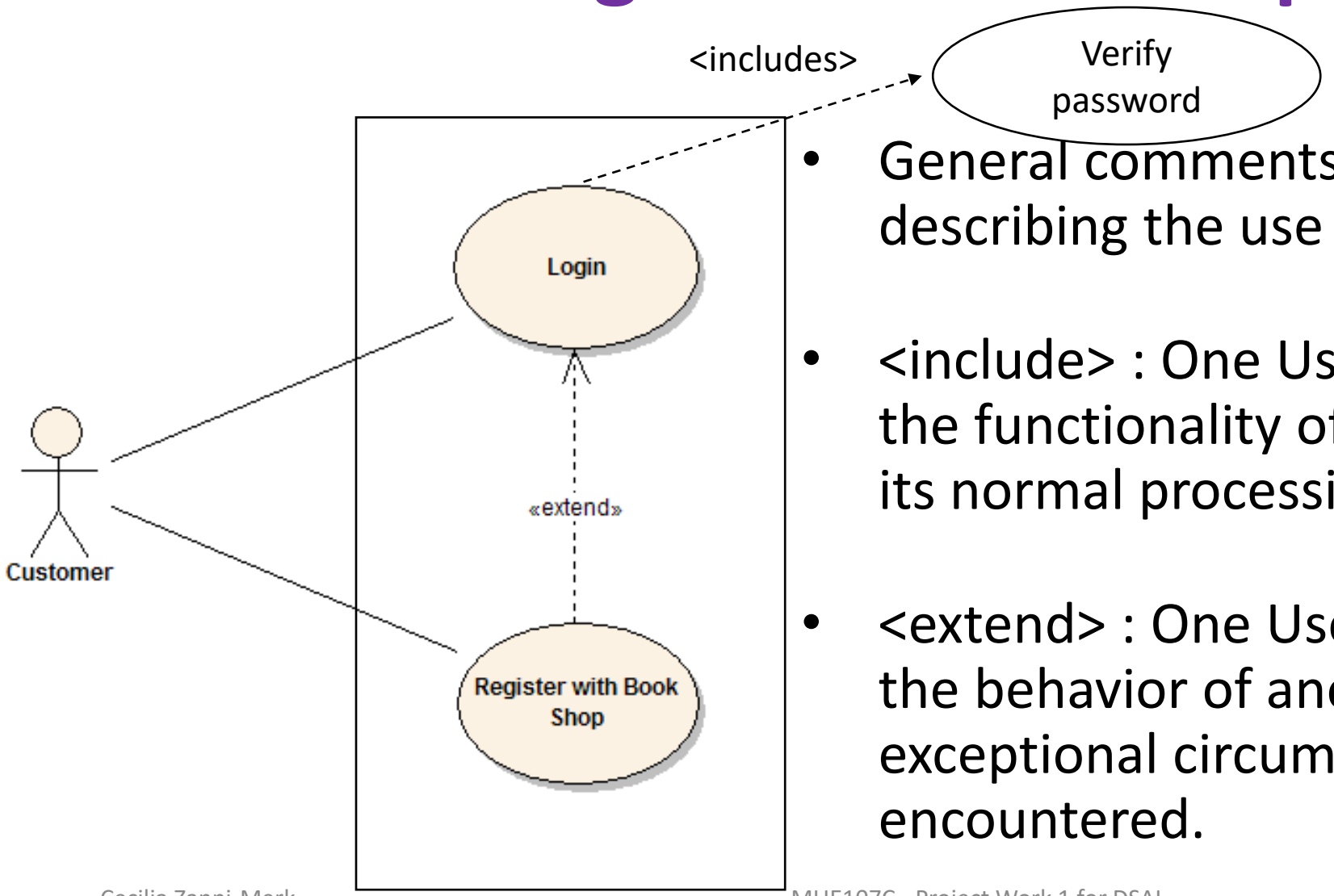
- They are useful for presentations to the management or stakeholders of a project
- A use case represents a discrete unit of interaction between a user (human or machine) and a system. It is a significant unit of work.
- In a use case diagram, users are called actors, they interact with use cases

Use Case diagrams … an example



- General comments and notes describing the use case are needed
- <include> : One Use Case could include the functionality of another as part of its normal processing.
- <extend> : One Use Case can extend the behavior of another, typically when exceptional circumstances are encountered.

Use Case diagrams … an example

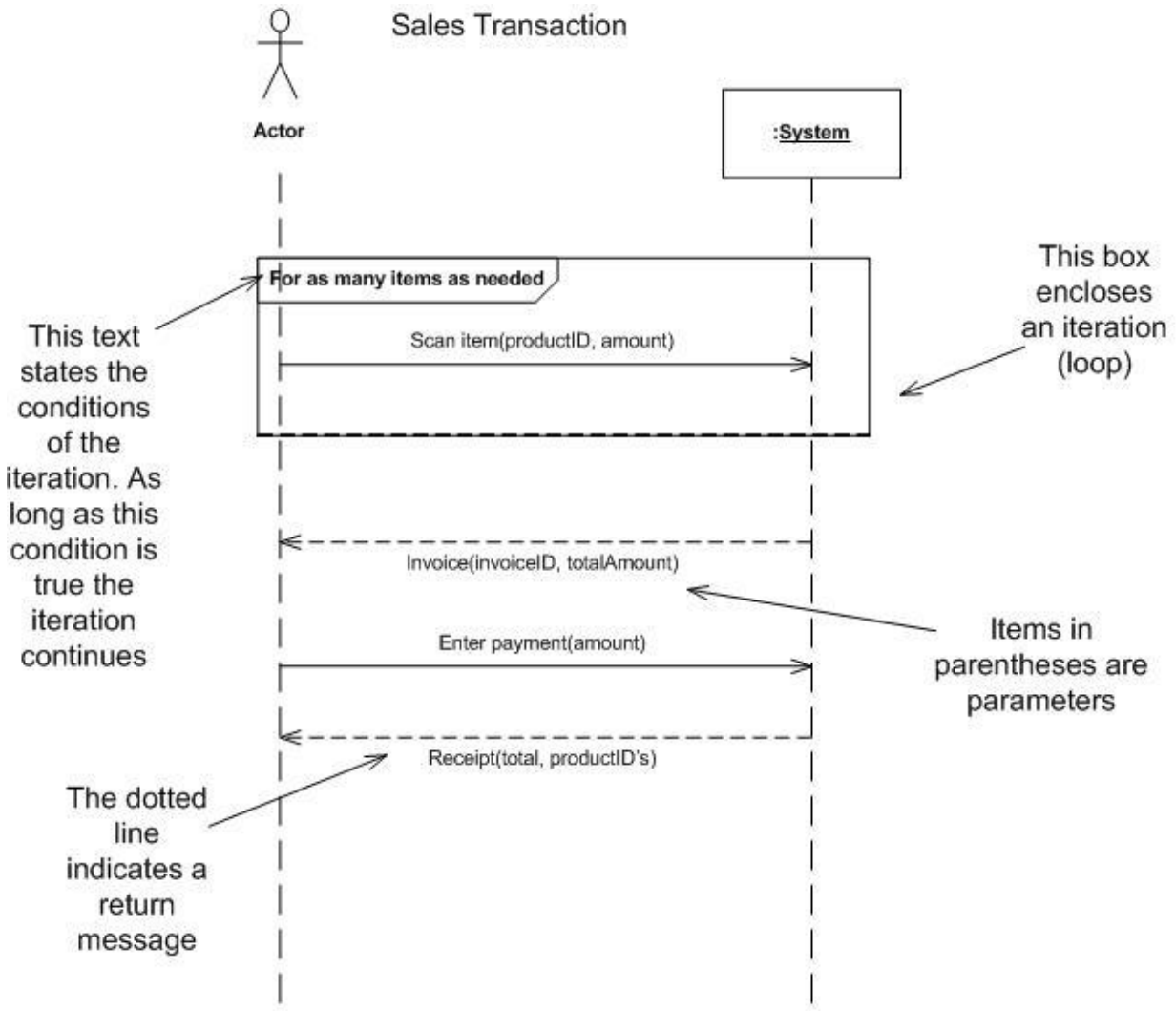


- General comments and notes describing the use case are needed
- <include> : One Use Case could include the functionality of another as part of its normal processing.
- <extend> : One Use Case can extend the behavior of another, typically when exceptional circumstances are encountered.

From Specification to Design

- The scenarios in the textual description of the use cases can be seen as instances of use cases and are illustrated by system sequence diagrams (one by use case)
- At a minimum, the nominal scenario of each use case should be represented by a sequence diagram that accounts for the interaction between the actor(s) and the system.
- The system is considered here as a whole and is represented by a lifeline. Each actor is also associated with a lifeline.

Sequence System Diagrams … an example



Up to you …

- Draw the system sequence diagrams for the use cases in slide 25

Design

Several points of view

- The user's point of view (already addressed during the needs analysis)
- The logical point of view: static view of the internal components of the system : ***UML class diagrams***
- The implementation point of view: decomposition into modules to be coded, tested and modified independently : ***Components or packages UML diagrams***
- The process view: dynamic vision of the system, how components interact over time : ***Interaction UML diagrams***

Design

Using this simplified approach, we will only use UML class diagrams and interaction diagrams for design

UML Class diagrams

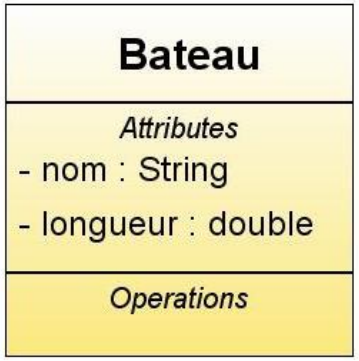
- Used to describe the static structure of the domain
  - **Concepts** (or classes or entities) are represented by boxes.
    A concept represents a set of concrete or abstract objects with their own identity. Ex: Person, Car
  - **Instance or Class Occurrence**; A particular entity.
    Ex person Jean Dupont, François; Picasso with registration number 222 ABE 67
  - **Attribute** or Characteristic or Property
    Atomic information relating to an entity. E.g.: the name of a person, the registration of a car
    Each entity can only have one value for a given attribute.
  - **Attribute Occurrence**
    The value of an attribute for a particular instance. Ex "222 ABE 67" for the attribute "registration" of the authority "my car"

UML Class diagrams

WARNING

- A class defines a set of objects with common characteristics (attributes).
- Attributes are therefore defined at the class level
- An instance has a value for each of the attributes defined at the class level

UML Class diagrams



class

instances

UML Class Diagrams

Association or relationship

- Semantic link between several entities.
- Ex: owner (between Person and Car)

Relationship Occurrence

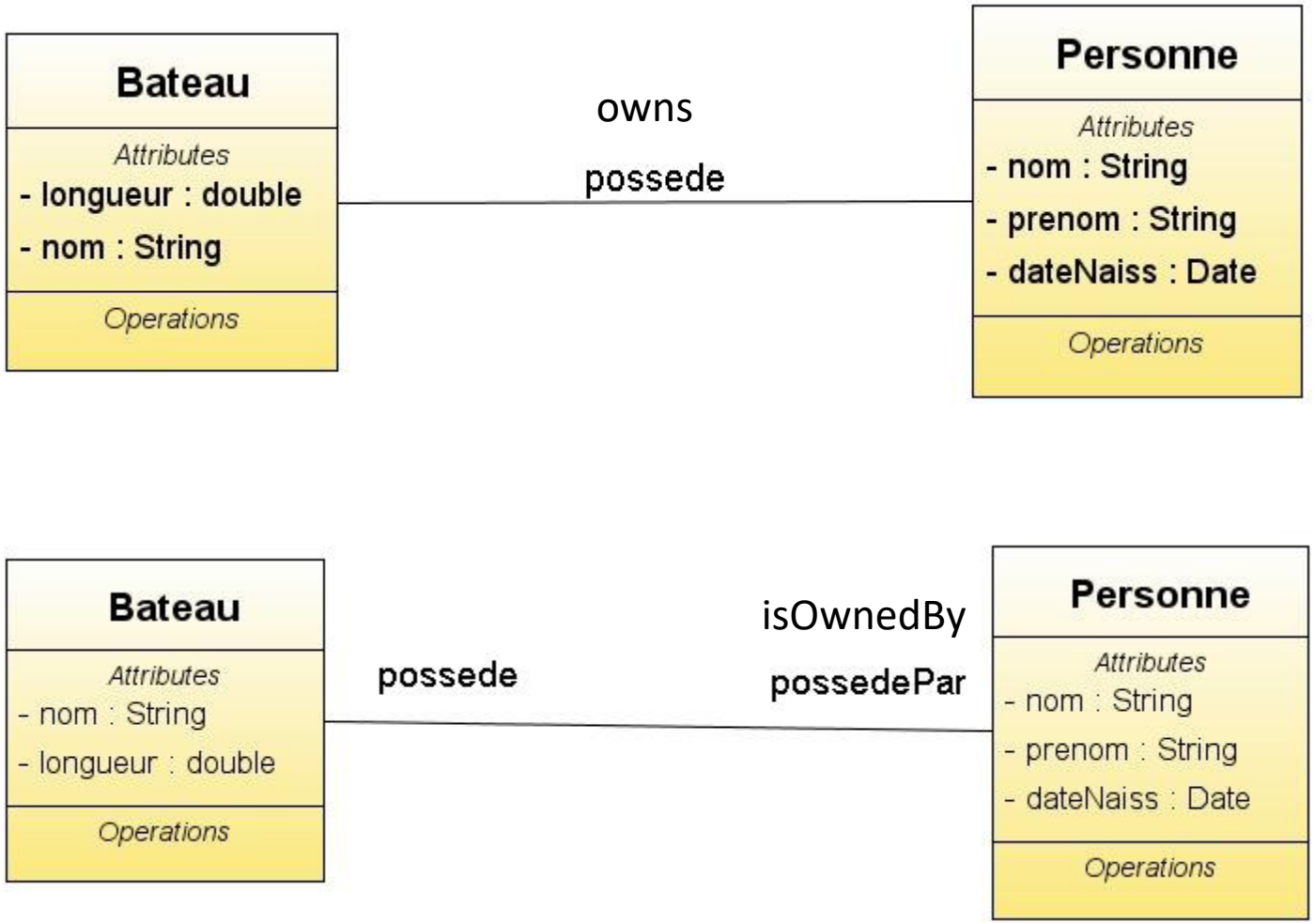
- A tuple of instances.

UML Class diagrams

The associations are represented by simple arrows.

- Can have a name and also a multiplicity.
- Bidirectional associations are represented by a single line, and unidirectional associations by a single arrow

UML Class diagrams



UML Class diagrams

Let R be a relationship between entities E1 and E2

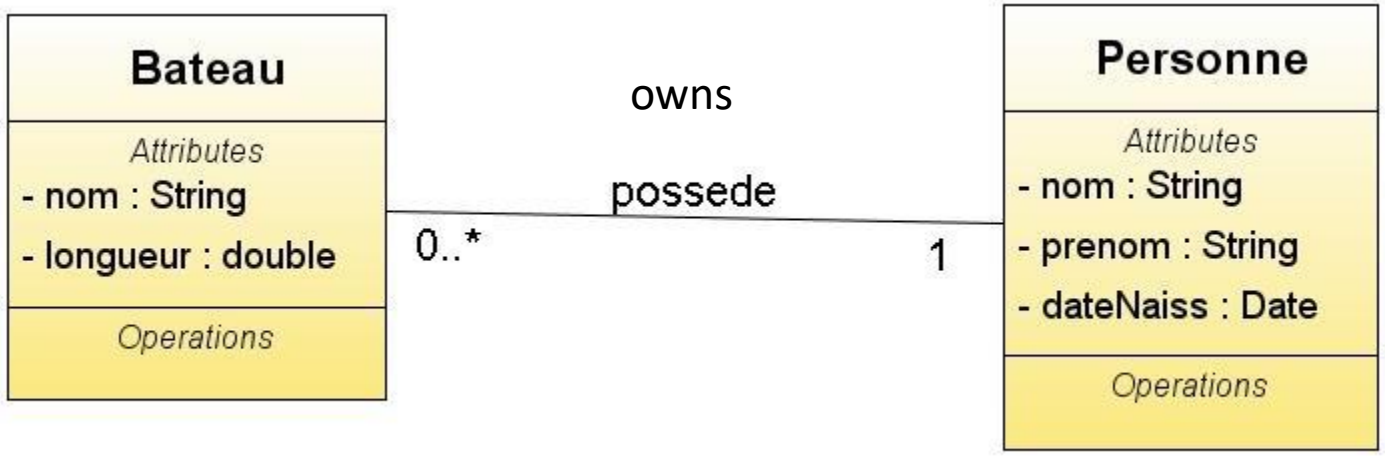
- For e1 an instance of E1, how many (minimum/maximum) entities e2i of E2 can be related to e1
- Similarly, for e2 an instance of E2, how many (minimum/maximum) entities e1i of E1 can be related to e2

owns

UML Class diagrams

Let R be a relationship between entities E1 and E2

- For e1 an instance of E1, how many (minimum/maximum) entities e2i of E2 can be related to e1
- Similarly, for e2 an instance of E2, how many (minimum/maximum) entities e1i of E1 can be related to e2



UML Class diagrams

An entity E1 generalizes an entity E2 if any instance of E2 is also an instance of E1. (E2 is included in E1). We can also say:

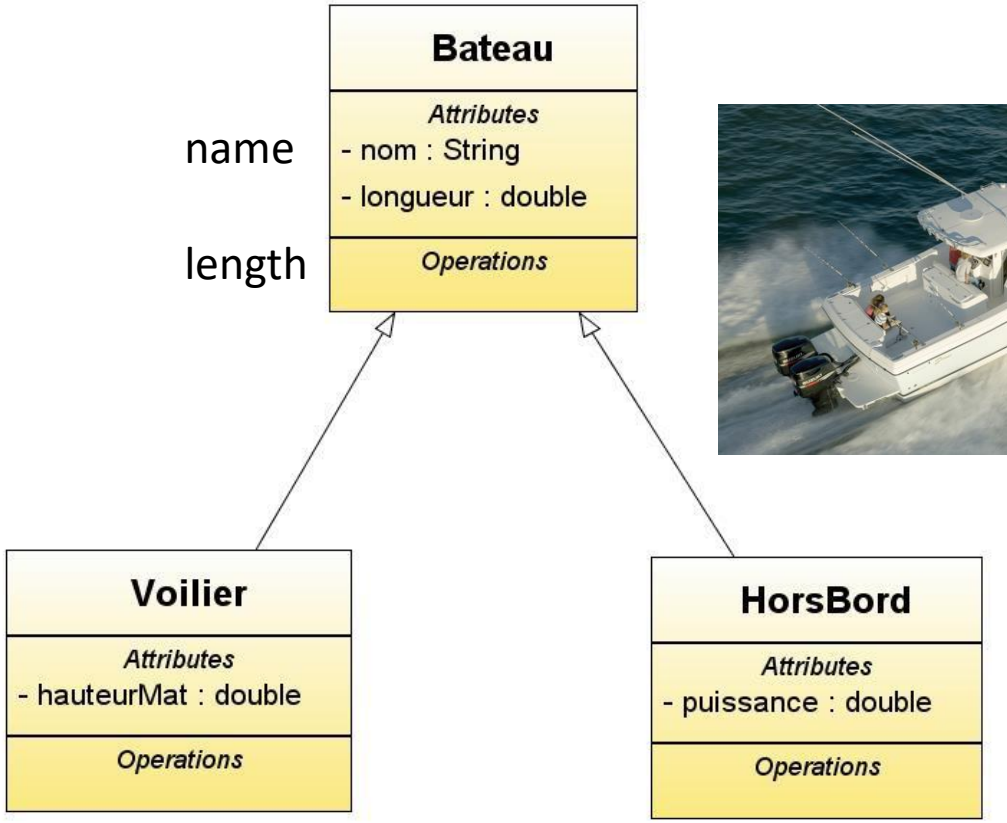
- E1 subsumes E2
- E2 specializes E1
- E2 inherits from E1

Ex: Woman specializes Person

Represented by a small hollow triangle. The arrow points to the superconcept

UML Class diagrams

Inheritance: If E1 specializes E2 any instance e of E1 is a fortiori instance of E2, so has all the attributes of E2, and can intervene in all the relationship defined for E2.



UML Class diagrams

A white "diamond" represents an **aggregation**, which is more specific than associations

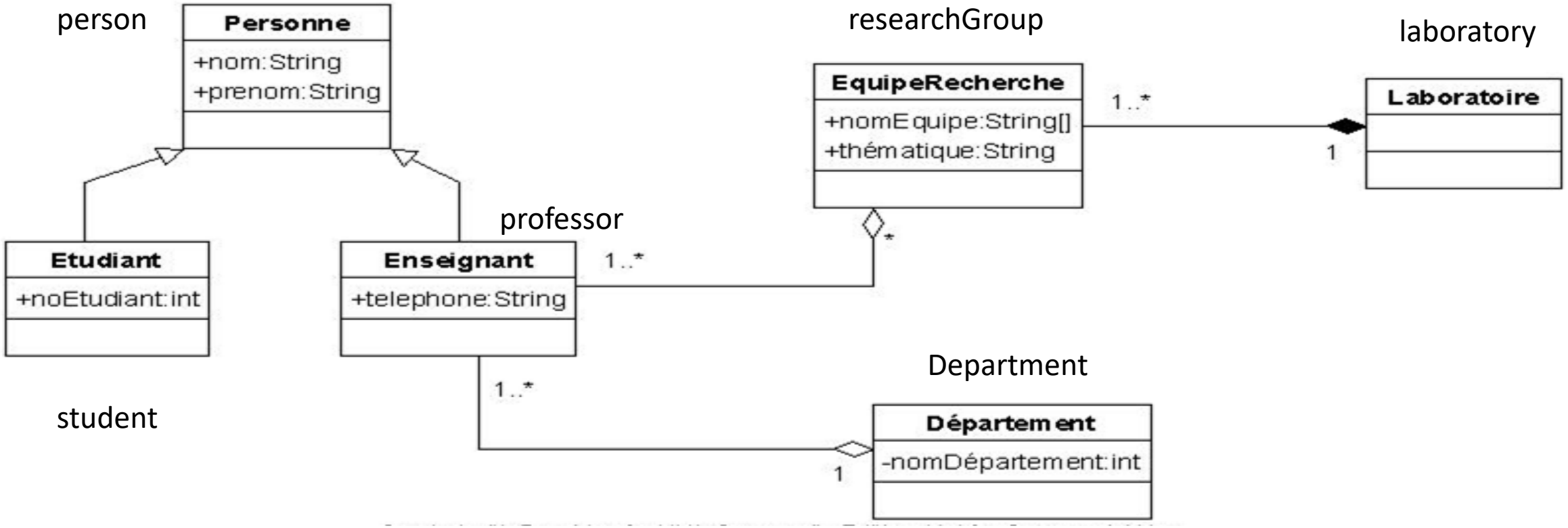
- It is an association that represents a "part-whole" type of relationship
- They can have names and cardinality
- Aggregation occurs when a concept "contains" other concepts. If the container is destroyed, the contents remain.

UML Class diagrams

A solid "diamond" represents a **composition**, a stronger association

- More specific than an aggregation
- In this case, if the container is destroyed, the contents are also destroyed.

Aggregations vs Compositions: an example



Let's return to the example

Goal of the development

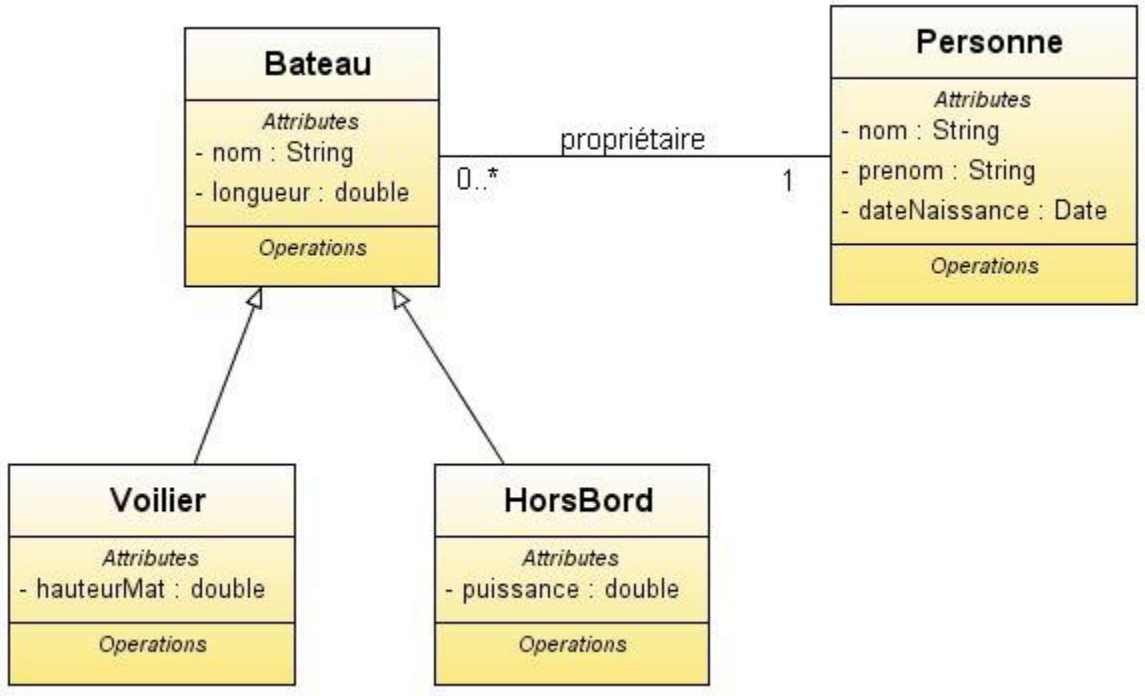
- In a marina, there are sailing boats and outboard boats. The boats have a home port and move from port to port
- We want to keep track of the boats' movements and be able to contact their owner

Let's return to the example

Features to be developed

1. Add / delete / modify a boat
2. Find the owner of a boat
3. Calculate the annual tax of a boat in its home port. The tax depends on the length of the boat, and its power if the boat is motorized
4. List the boats for a given home port
5. List the ports visited by a boat
6. Make the history of a boat's movements
7. List the ports sorted by number of attached boats
8. Send a letter to each owner of a boat in a port specifying the amount of the tax

A first version of the class diagram



Ports and home ports

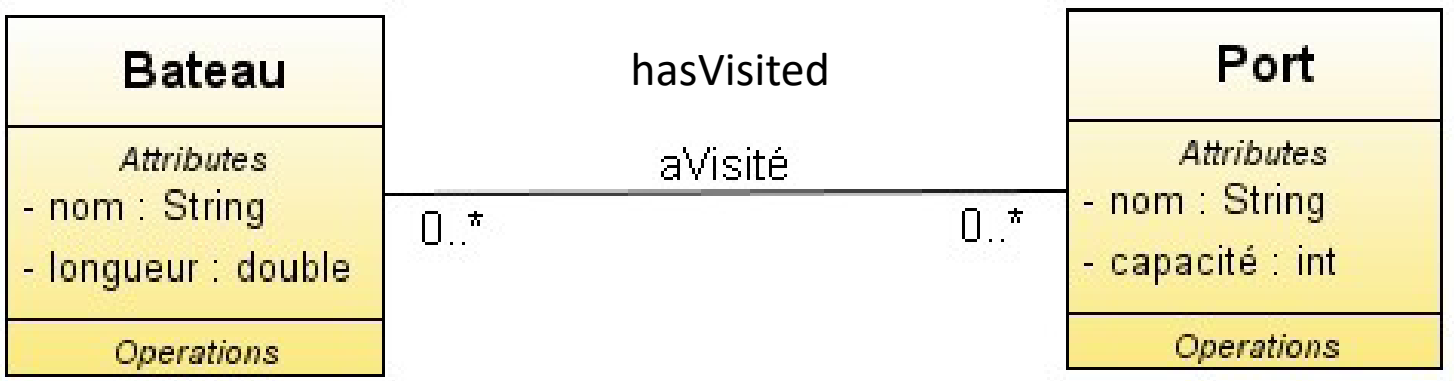
We also need to define the **Port** class, and a **portDAttache (homePort)** association that connects boats and ports.

A boat has at most one home port, but a port is the home port of several boats.

The aVisité (hasVisited) relationship

Point 5 of the needs analysis indicates a new association between Port and Boat.

A ship may visit more than one port, and a port may have been visited by more than one ship.

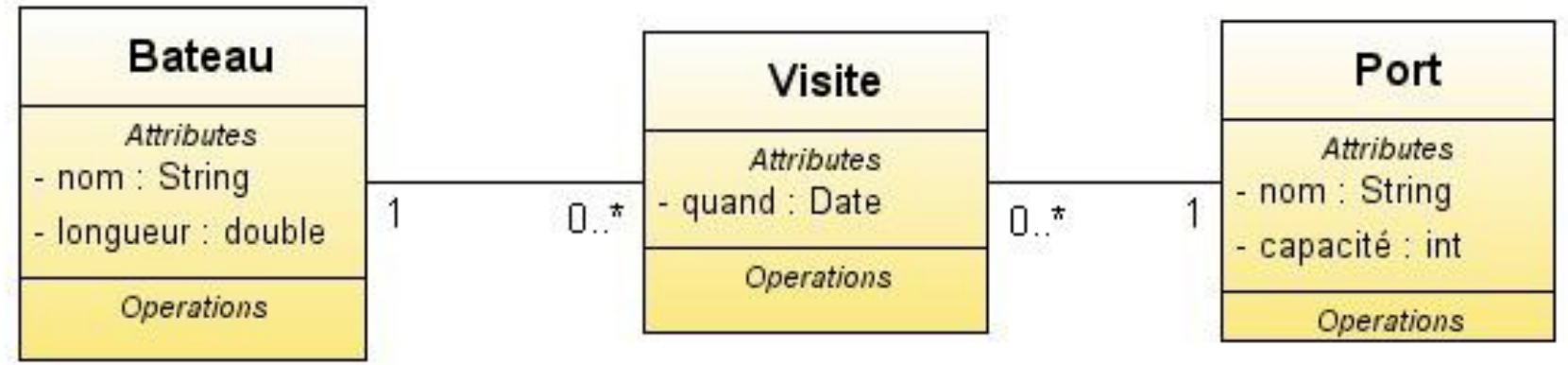


History of visits

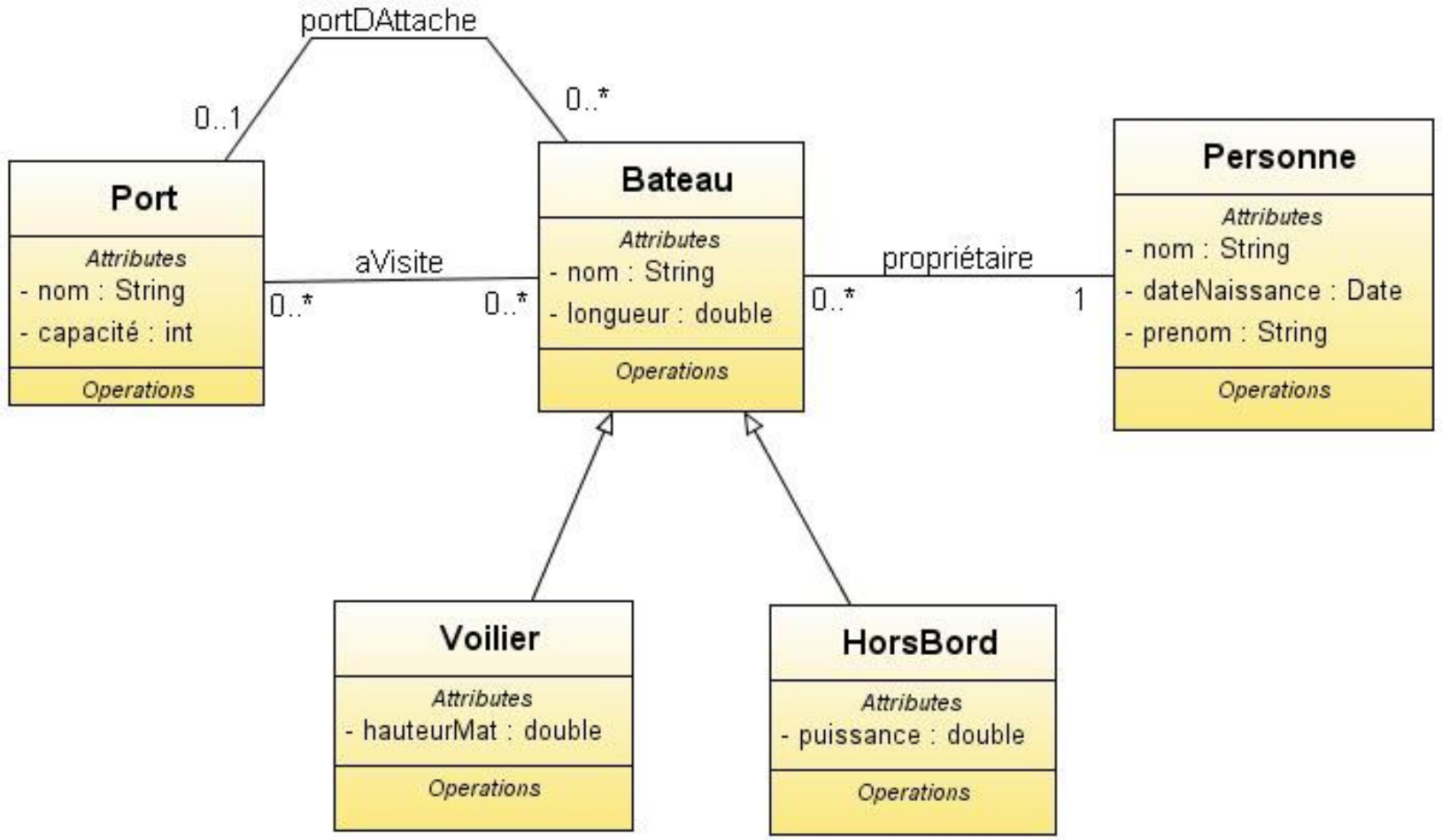
- To be able to make the history of a boat's movements, we need to know when it visited the different ports.
- This information is not a boat attribute (remember that an attribute can only have one value, or a boat can visit several ports, with several visit dates), and is not a port attribute (for the same reason).
- This information should therefore be associated with the relationship itself.

History of visits

- We create a normal class that corresponds to the relationship. This transformation id sone by **reifying** of the relationship.



The whole diagram



What is lacking

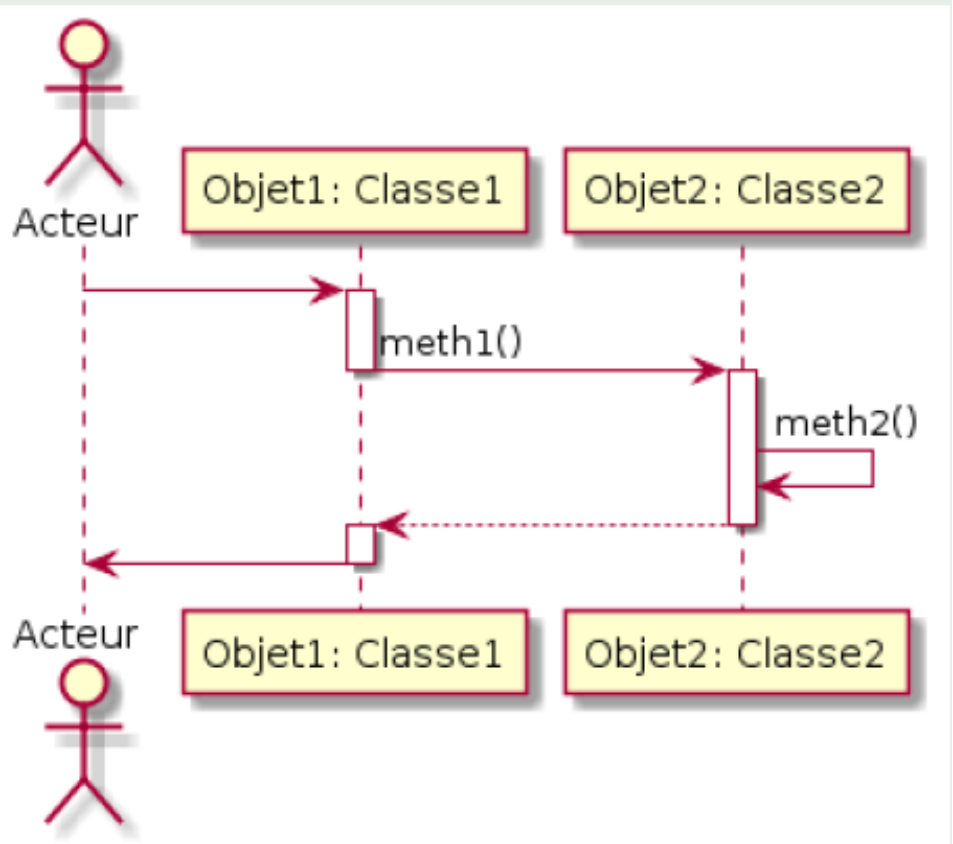
The verification of the rest of the needs

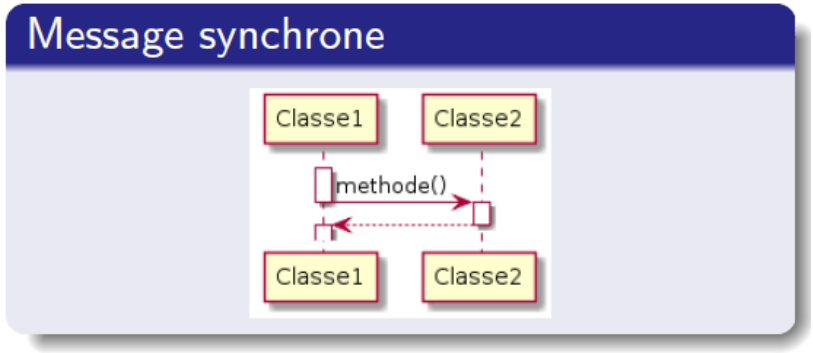
- List the ports sorted by number of boats attached to this port
- Sending an invoice

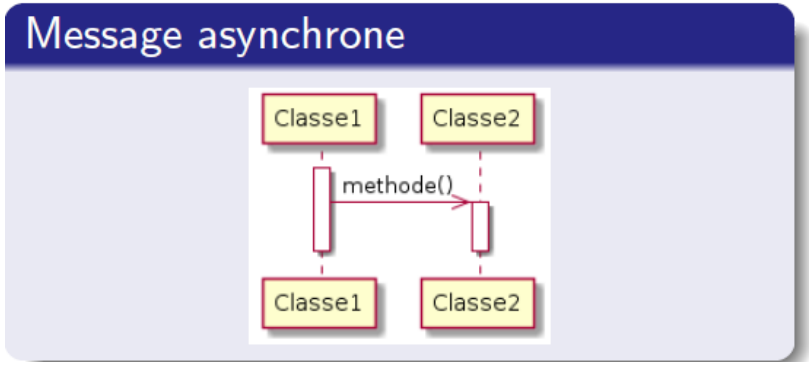
Interaction diagrams

- It is a more detailed view of the system sequence diagrams. Now, the system, is described in more detail, and not as a “black box”
- This diagram represents the interactions between users, screens, objects and entities within the system by chronology of method calls
- A sequence diagram illustrates a use case
- Time is represented by a vertical axis

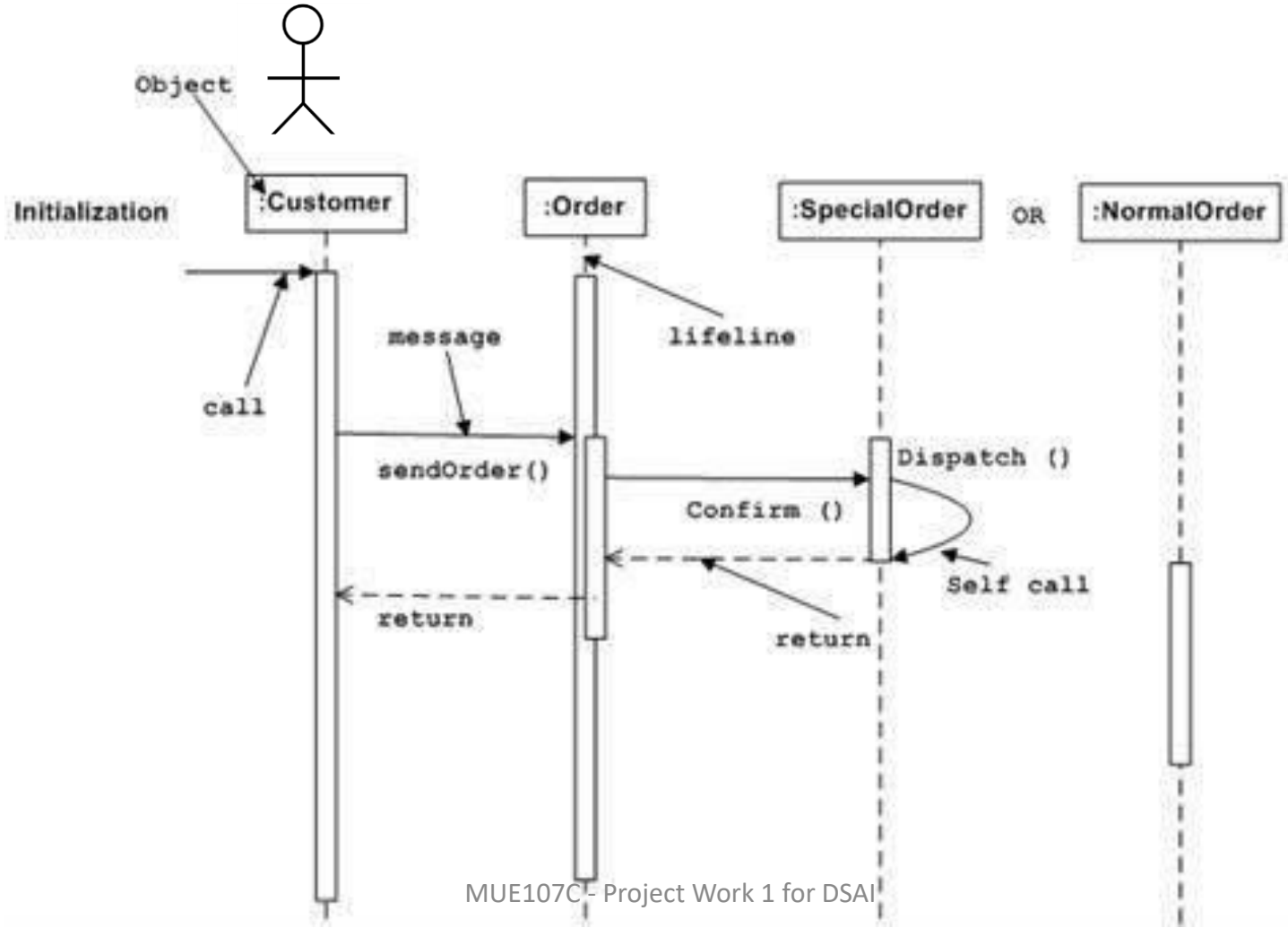
Interaction diagrams … notation





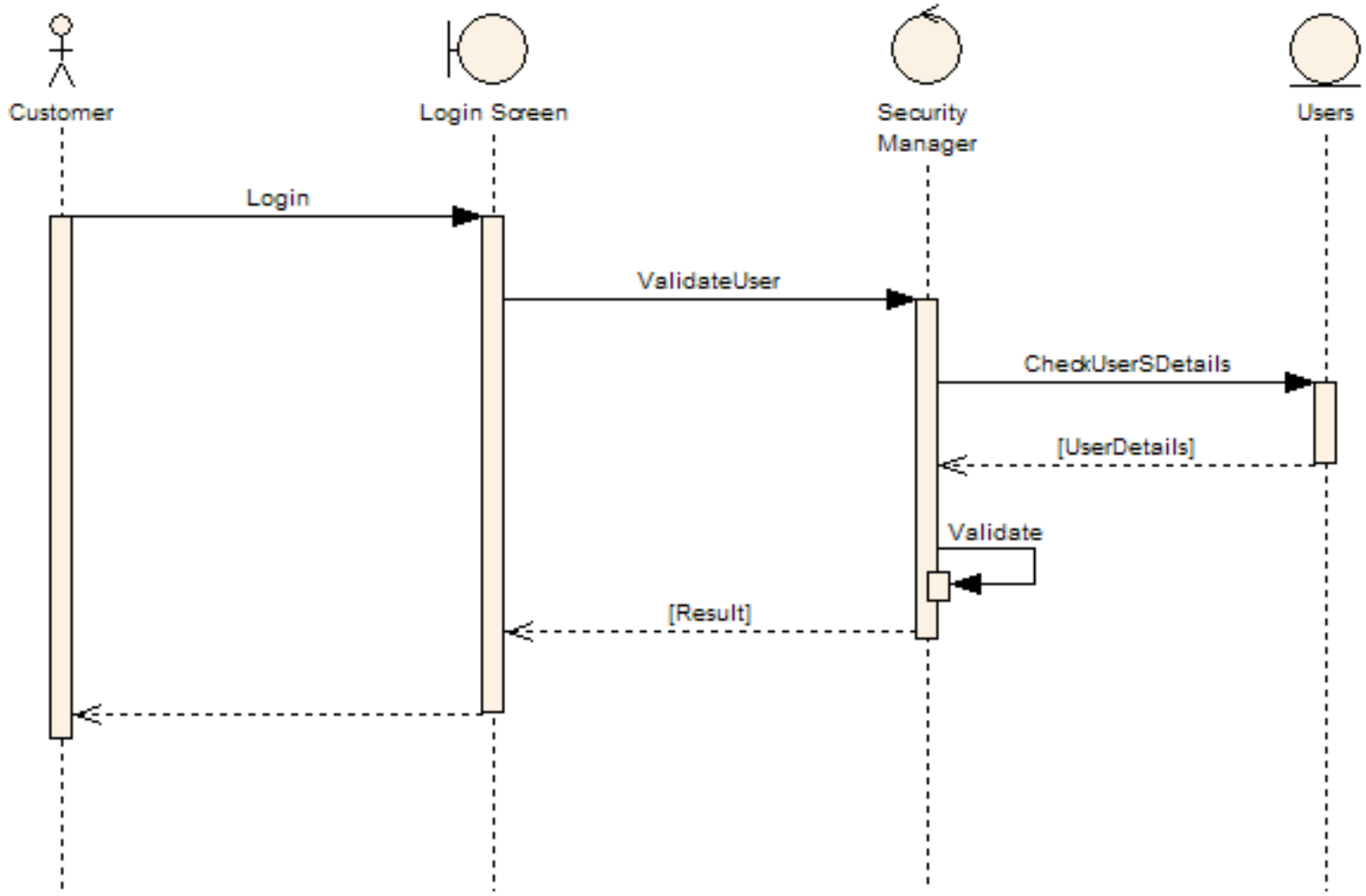


Interaction diagrams … an example



Interaction diagrams … another example

Often, the objects are represented using special stereotyped icons, such as, a user interface icon, a controller icon, an entity icon



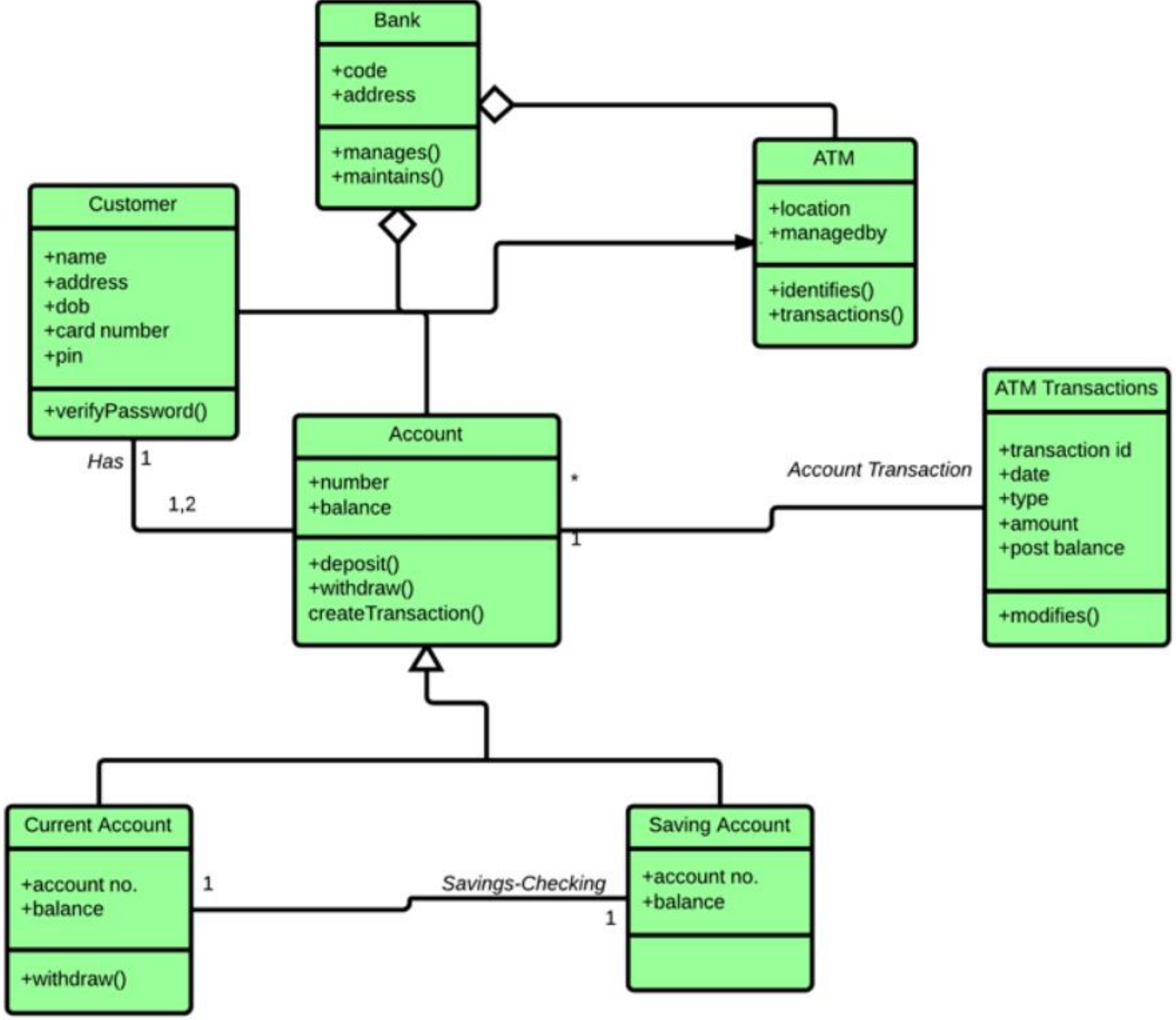
Exercices

Exercices

1. Interpret the following class diagram to give a natural language specification

Exercices

1. Interpret the specification



Exercices

2. Complete the Boats and Ports diagram to include the two features that are missing
   a) List the ports sorted by number of boats attached to this port
   b) Sending an invoice

Exercices

3. For the following description do:
    a. The specification (exhaustive list of features, use case diagram, system sequence diagramss)
    b. The design (class diagram, interaction diagram)

Exercices

A hotel consists of a number of rooms. A hotel manager manages the rental of the rooms. Each room is rented at a given price.

Access to the bathrooms is included in the price of a room rental. Some rooms have a bathroom, but not all. Guests in rooms without bathrooms can use a bathroom available in the same floor. These can be used by multiple hosts.

Hotel rooms that are neither bedrooms nor bathrooms (reception hall, kitchen...) are not part of the study (out of scope).

People can rent one or more rooms in the hotel to reside there. In other words: the hotel accommodates a certain number of people, its hosts (these are people who rent at least one room in the hotel…)