III - Object Oriented Analysis and Development - UML

Equipa de SIG

Table of Contents

What is Object Oriented Approach?

- It is supported on a way of analyzing the world that reproduces how human being perceives and expresses the reality that surrounds him
- The world is classified and subdivided into different objects, based on the differences and similarities existing in the characteristics and behaviors.
- The OO approach allows you to support:
  - complex data types,
  - richer data models
  - reuse

UML *(Unified Modeling Language)* (1/2)

- It is a language to specify, visualize, construct and document artifacts of diverse systems (software, business or others);
- Allows the developer to specify:
  - » the structure of the systems (static elements),
  - » the behavior of the systems (dynamic elements);
- The UML resulted from the fusion of 3 OO methods for analysis and design:
  - » Object Modeling Technique (OMT) (by Rumbaugh)
  - » Method of Booch (de Booch)
  - » Object Oriented Software Engineering (OOSE) (by Jacobson).

UML *(Unified Modeling Language)* (2/2)

- The specification is performed using diagrams;
- Each diagram presents a specific perspective on the system;
- More information at http: //www.omg.com

Core items (1/3)

Type of Diagrams

- **Functional Vision Diagrams**
- **Use Case Diagrams - allow the system to be viewed from the perspective of its users**
- **Activity Diagrams - represent the various activities carried out by the actors**

Core items (2/3)

Type of Diagrams

- Structure or Static Vision Diagrams
  - Class Diagrams - static structure of the information system
  - Component Diagrams - dependencies between software components
  - Deployment diagrams - configuration of physical components (software and hardware).

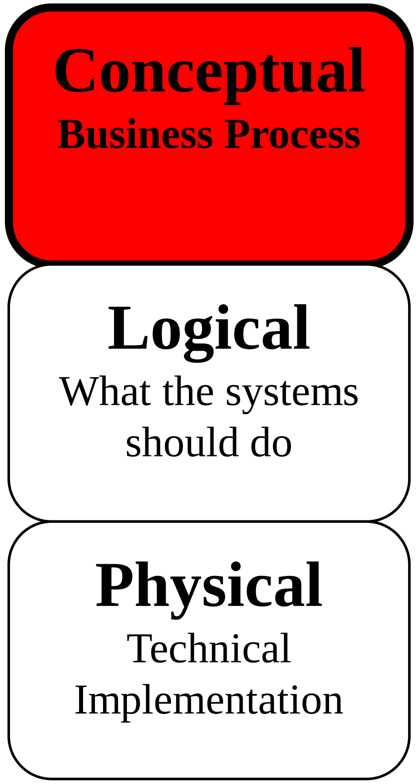
Core items (3/3)

Type of Diagrams

- Dynamic Vision Diagrams
  - **Interaction Diagrams**
    - Sequence diagrams - messages between objects over time
    - Communication Diagrams - Non-temporal view of communication between objects
  - **State diagrams** (or state machines) - representation of the life cycle of the objects of the system, through the states they are passing through

Levels of Detail (1/3)

- UML diagrams can be created with different levels of detail, depending on their purpose:



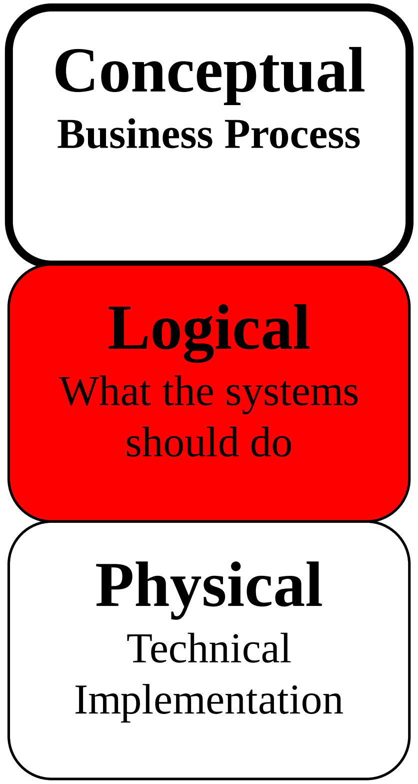
» **Little detail (or high level - conceptual):**

» they only describe the operation of the business, both at the structure level and at the (macro) level.

» They are adequate to discuss with users or to represent a first blueprint of the system.

Levels of Detail (2/3)

- UML diagrams can be created with different levels of detail, depending on their purpose:



» **Medium detail (Logical):**

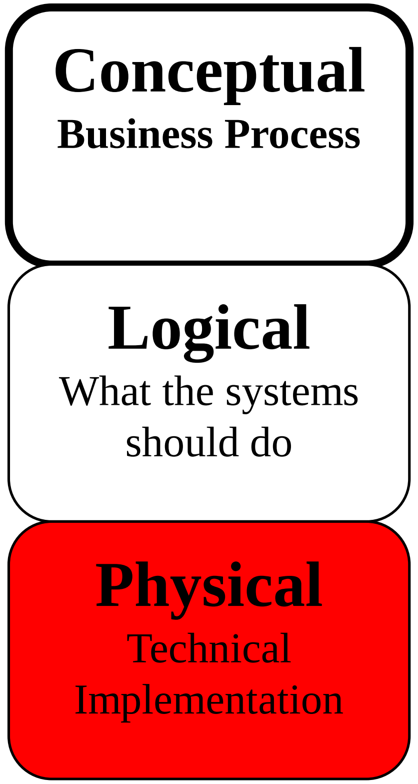
» give more detail than the previous level, focusing more on what the system should do.

» They allow us to establish the bridge between what the customer knows and what developers should know about the system initially.

» They support the phases of analysis, design and testing.

Levels of Detail (2/3)

- UML diagrams can be created with different levels of detail, depending on their purpose:



» **Great detail (or low level - Physical):**

» describe in great detail the system to be built presenting aspects connected with the technology that will be used.

» They are usually prepared by and for developers to support and summarize their coding work or to support communication between technical teams.

Use Case Diagrams

- They are frequently used in conjunction with Use Case descriptions and scenarios
- They allow capturing the functional requirements, according to an approach focused primarily on the use of the system
- They give a global and high-level vision of the IS
- They refer to the large modules of the system by external agents to the system, called actors

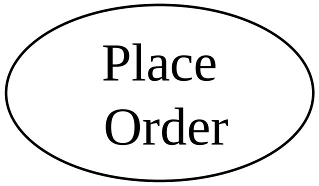
Use Case Diagrams (UCD)

- Before creating a project we can used UCD to model the business
  - » UCD allow all participants to obtain a uniform knowledge of the concepts involved (eg customer, worker, business activities)
- In the analysis phase, UCD must be designed to capture system requirements and present what the system must do.
- Use cases and actors can help identifying classes that the system needs.
- In the testing phase, use case diagrams can be used to identify suitable tests for the system.

Use Case Diagrams' elements (1/3)

- Use Case:
  - » describes a sequence of actions that provide something of measurable value to an actor
  - » is a sequence of actions that one or more actors perform in a system in order to obtain a particular result
  - » is drawn as a horizontal ellipse (or oval).

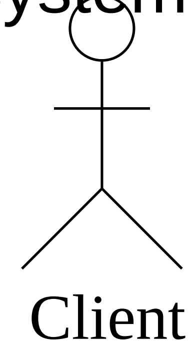
Sintaxe



must be written
in the active voice, with a verb in the infinitive

Use Case Diagrams' elements (2/3)

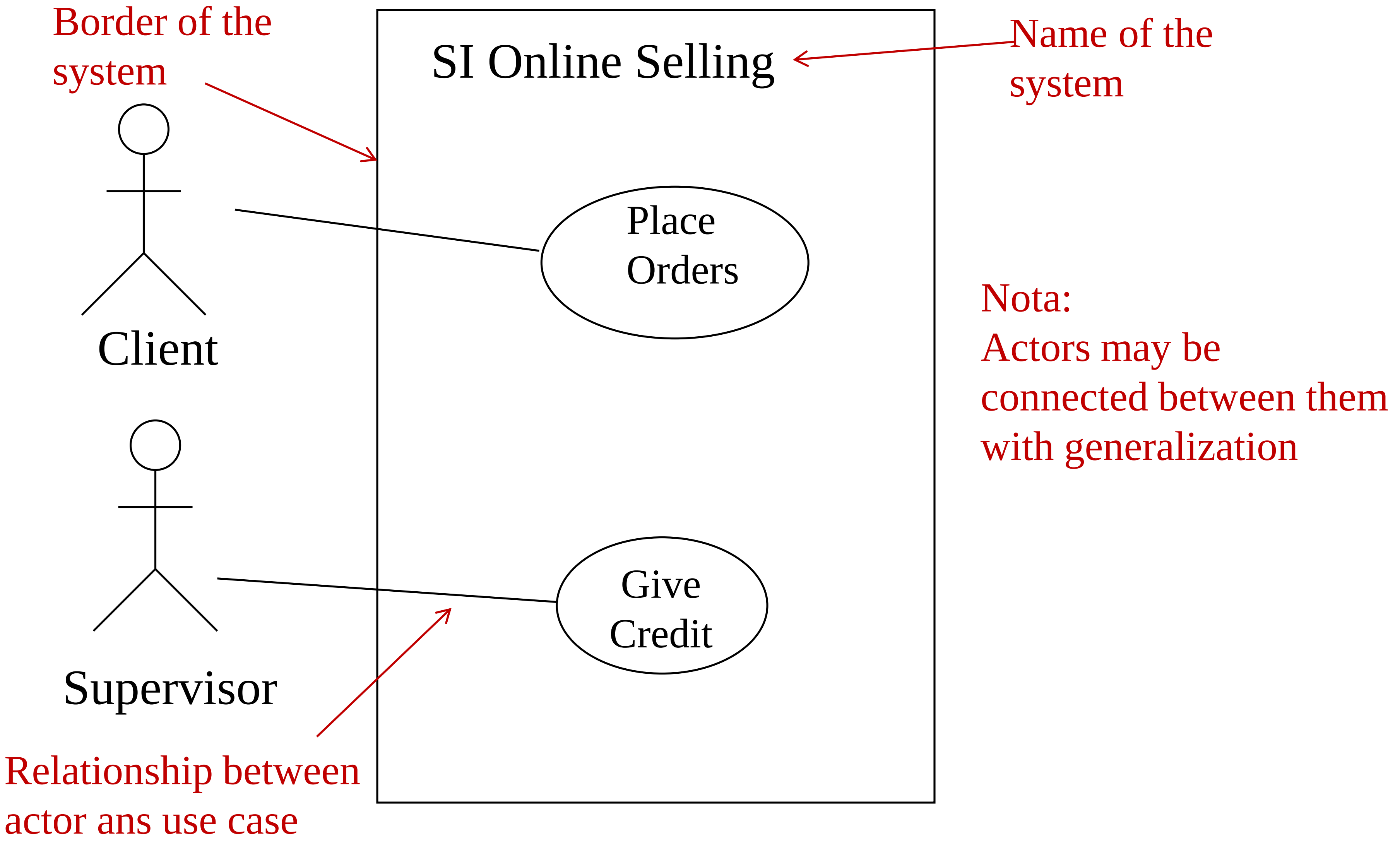
- Actor:
  - » An actor is a person, organization, or external system that plays a role in one or more interactions with your system.
  - » The actor does not have to be a human user, and can represent another information system, hardware
  - » although the actors are not part of the system to implement, their modeling is fundamental to establish the interfaces of the system with the outside world (its border)
  - » Actors are drawn as stick figures



Use Case Diagrams' elements(3/3)

- Actor (cont.):
  - » an actor is one who interacts directly with the system
  - » At the conceptual level, our purpose is modelling the business,
    - – which means that the system is the business;
  - » At the logical level, modeling is performed in order to describe a computer system,
    - – which means that the system is the computer application and the actors the users of this application.

Use Case Diagram Exemple



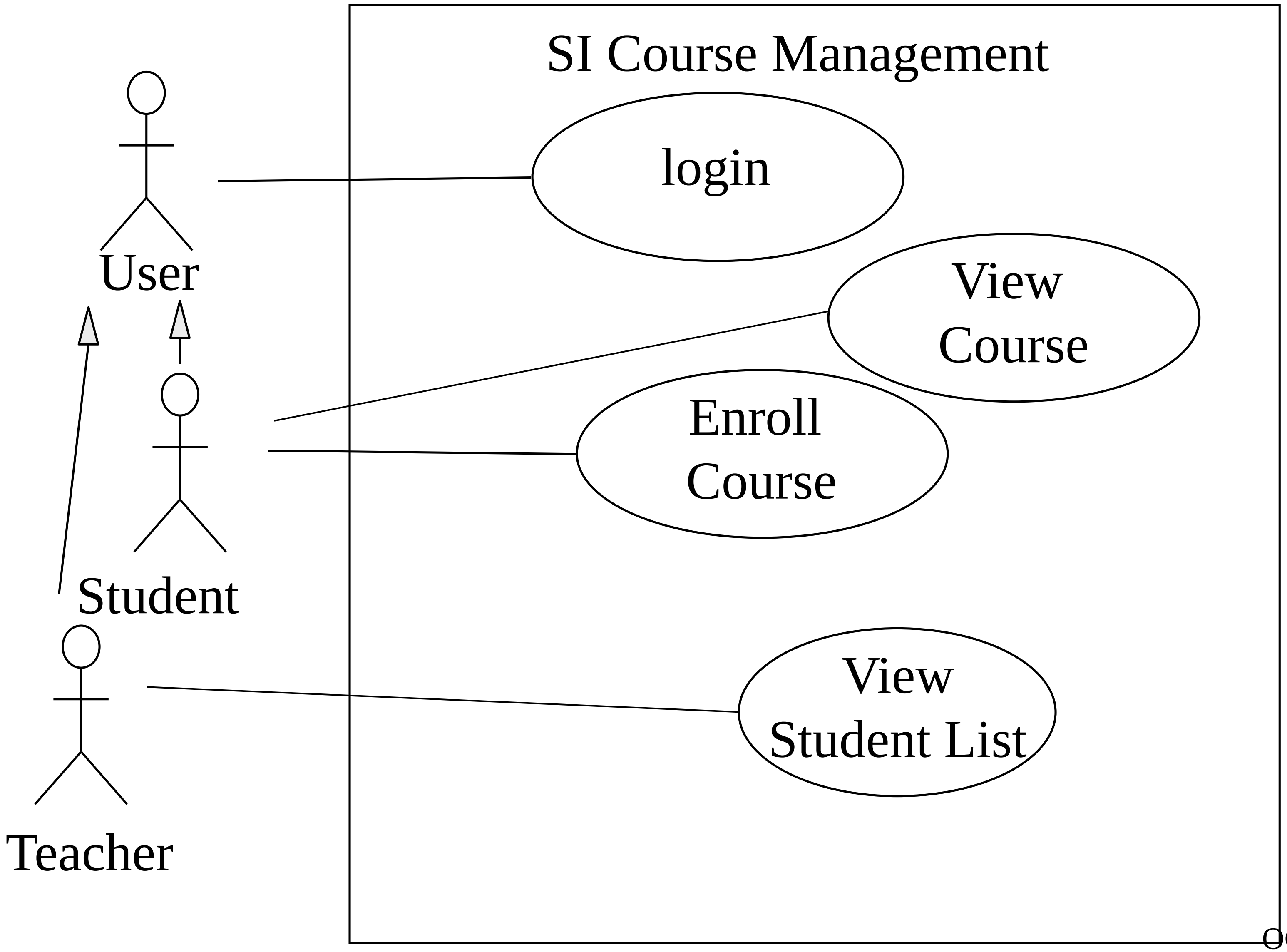
Nota:
Actors may be connected between them with generalization

Course Management example (part 1)

- It is intended to develop an IS that allows to manage a course of a university according to the following specification:
- "Students should enroll in the course available for their program, and may consult their list in advance for better planning.
- Teachers from the corresponding course can view the enrolled students' list at any time, and may select the students by shift, by program or by type of student (student / student worker)
- Users must log in with their username and password to perform any functionality on the system. "

Course Management

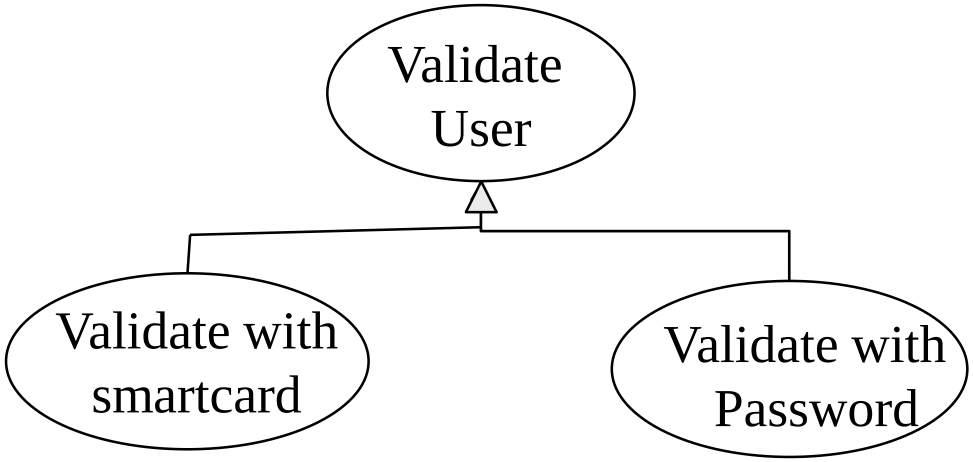
Use Case Diagram



Relationships between use cases

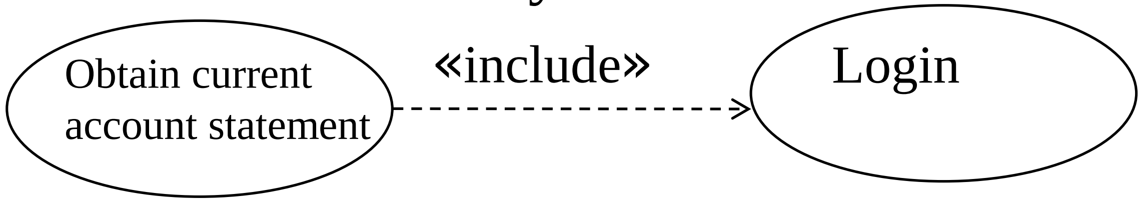
1. Inheritance between Use Cases

- It allows defining cases based on the existing ones, and the child use case inherits the properties and semantics defined for its parent
- This connection should be avoided by alternately adding and extending links

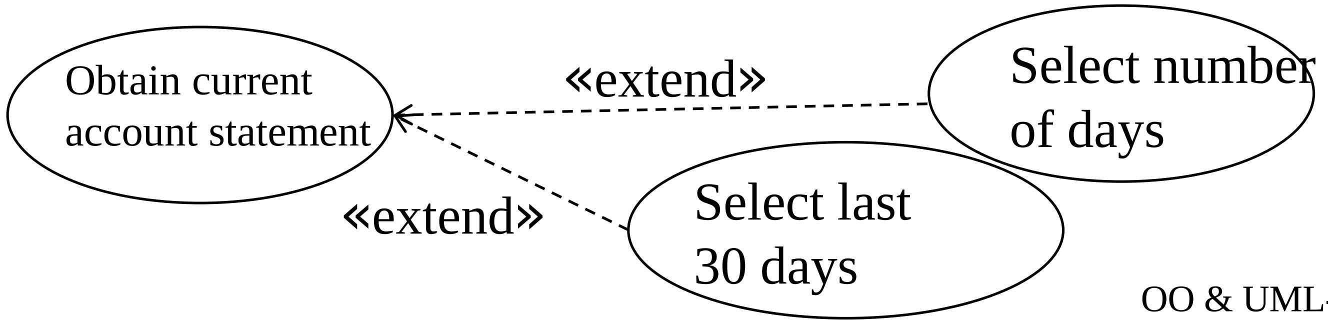


Relationships between use cases

- Include
  - » An include dependency, is a relationship denoting the inclusion of the behavior described by another use case.
  - » The best way to think of an include dependency is that it is the invocation of a use case by another one.



- Extend
  - » A use case B extends a use case C when B applies optionally, under some condition (usually specified in the scenario)
  - » The target case can be extended with the behavior of another case (s)
  - » Allows to represent the part of a case that a user sees as optional, or as having several alternatives

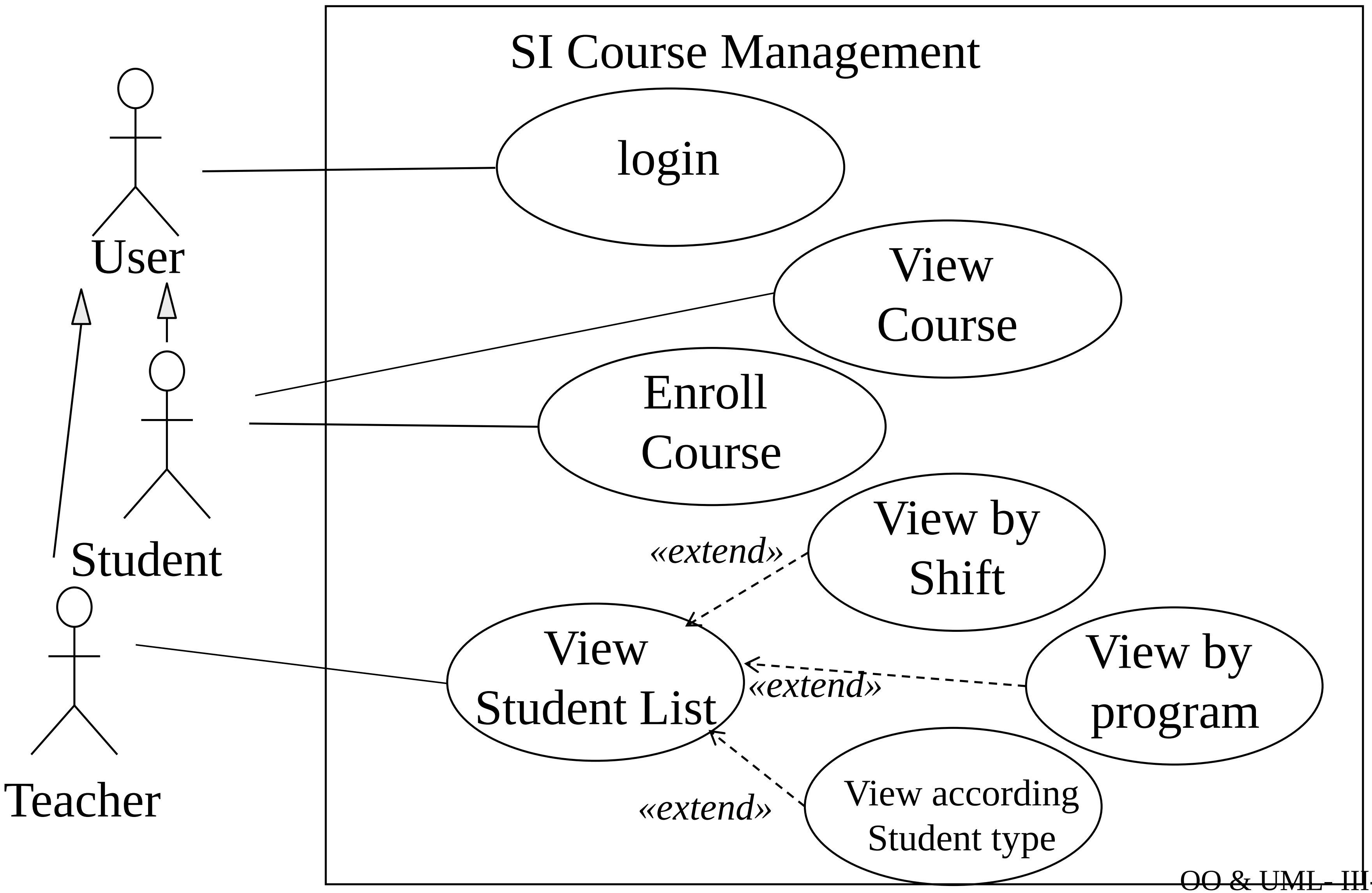


Course Management example (part 1)

- It is intended to develop an IS that allows to manage a course of a university according to the following specification:
- "Students should enroll in the course available for their program, and may consult their list in advance for better planning.
- Teachers from the corresponding course can view the enrolled students' list at any time, and may select the students by shift, by program or by type of student (student / student worker)
- Users must log in with their username and password to perform any functionality on the system. "

Course Management

Use Case Diagram



Scenarios

What are scenarios?

- Scenarios are descriptions of how the system is used in practice, or should be used in practice.
- Scenarios are important for the early stages of development, and for communicating with stakeholders
- Scenarios allow us to answer questions such as "what actions should the system accomplish"; "what if?"

Scenario Description

Description of scenarios should contain:

- The system state at the beginning of the scenario
- The normal flow of events in the scenario
- What may fail and how we may deal with failure
- Other concurrent activities
- The system state at the end of the scenario

Example of Scenario description

Place order in the Internet(Main scenario)	
Preconditions	The cliente is valid user of the system
Description	1. The use case starts when the customer selects the option to Order 2. Simultaneously with your order, the system shows the product catalog 3. The customer adds products to the order by entering the product code 4. Automatically, the system shows the name, description and price of the product 5. Each time a product is added, the total value of the order is recalculated 6. The customer confirms his order through the option Confirm 7. The system asks for the credit card details 8. The system confirms the payment data and assigns an identification number to the order
Postcondition:	The order will be delivered to the customer's address

Example of Scenario description

Place order in the Internet(Secondary scenario)	
Preconditions	The cliente is valid user of the system
Description	1. The use case starts when the customer selects the option to Order 2. Simultaneously with your order, the system shows the product catalog 3. The customer adds products to the order by entering the product code a. If a code is invalid the system warns the client with a message 4. Automatically, the system shows the name, description and price of the product 5. Each time a product is added, the total value of the order is calculated 6. The customer confirms his order through the option Confirm 7. The system asks for the credit card details 8. The system confirms the payment data and assigns an identification number to the order a. If the card is invalid, the system informs the customer via a message, then returns to step 7
Alternative path	At any time the customer can cancel his order by pressing the Cancel button
Postcondition	The order will be delivered to the customer's address

exercise Course Management

- List of Use Case Scenarios - Enroll in Discipline:
  - » Desired course has no previous course
  - » Desired course has previous course and the student had approval to all, so the enrollment is made with success
  - » Desired course has previous courses and the student did not have approval at all those courses, so the enrollment is not carried out and the system show a warning message.

Exercise Course Management

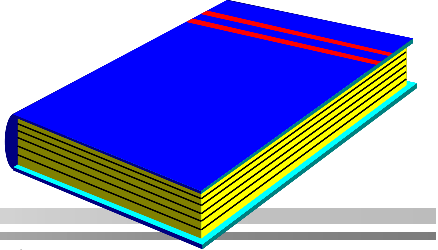
- Consider the following excerpt from the scenario description of the " Course Enrollment" use case:

Course Enrollment	
Precondition:	The student is a valid user in the system
Description	1. The use case starts when the student selects the "Enroll in Course" 2. The system asks for the school year of the intended course 3. The student enters the academic year of the desired course 4. Then the system shows the list of courses that the student has available for the selected year 5. The student selects the name of the course 6. Automatically, the system will check if the chosen course has some course of precedence and if so, verifies that the student has passed the course (s). If there is some previous course that the student does not have approval, the system shows a message informing the student of the situation, otherwise the message should inform that the registration was successful.

Exercise

- Draw a use case diagram and a related scenario for the following situation:
- A user can borrow a book from a library;
  - » extend it with borrowing a journal
- a user can give back a book to the library
  - » including the use case when the user is identifie

Bibliography



1 - Alhir, S., S., ***UML in a Nutshell***, O'Reilly & Associates, Inc., USA., 1998.

2 - Oestereich, Bernd, ***Developing Software With UML***, Addison Wesley, second edition, 2002.

3 - Silva, Alberto, Videira, Carlos ***UML Metodologias e Ferramentas CASE***, Centro Atlântico, Publishing Ltd, 2004.

5 - Rumbaugh, J., Blaha, M., Premerlani. W., Eddy, F., Lorensen, W.; ***Object-Orientes Modeling and Design***, Prentice-Hall Inc., USA., 1991.

6 - Stevens, P., Pley, R.; ***Using UML software Engineering with Objects and Componentes***, Addison Wesley.

7 - Rumbaugh, J., Jacobson, I.,Booch, G.; ***The UML reference manual***, Addison Wesley, 2004.