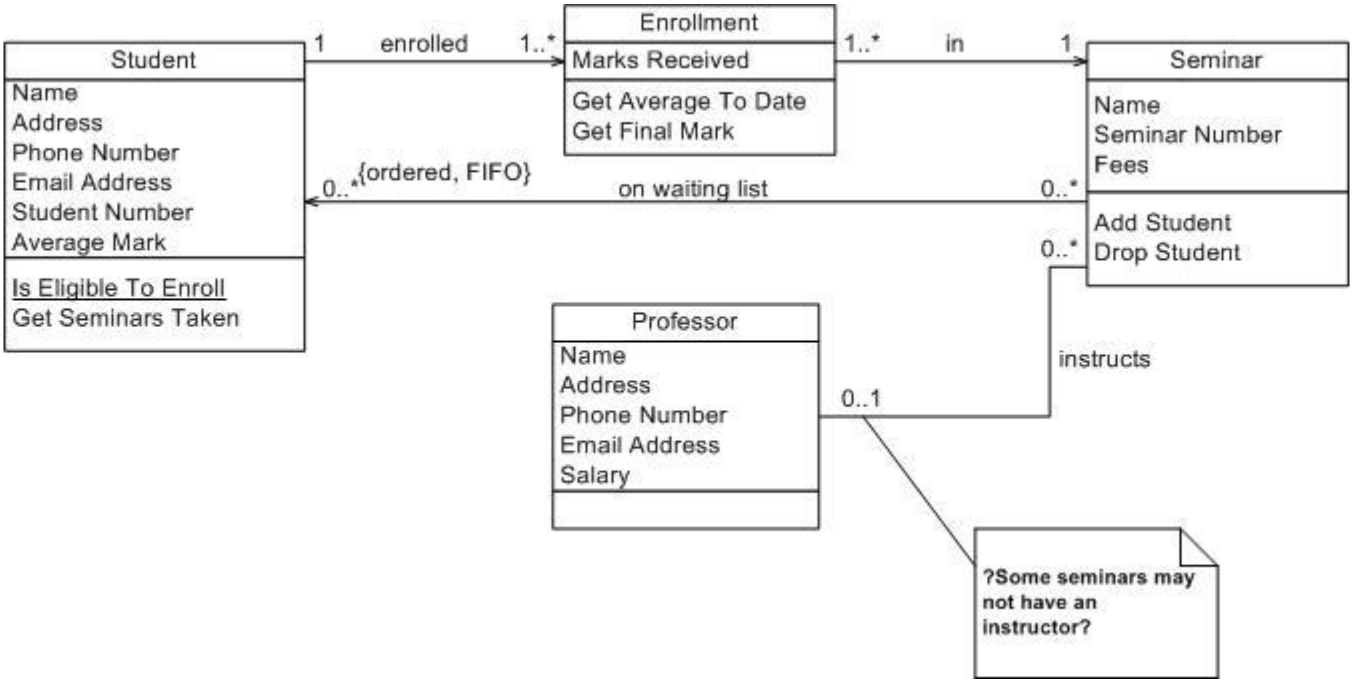
Laborator Diagrame UML - Teme

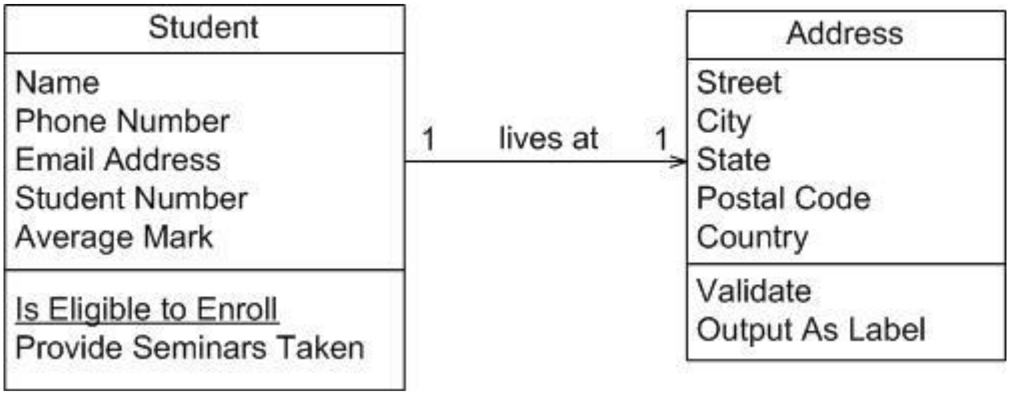
Pornind de la principalele elemente specifice diagramelor UML, descrieti pe scurt exemplele propuse. Cautati alte diagrame specifice fiecarui tip de diagrama. Analizati modul de generare considerand diferite limbaje si medii de programare.

1. **Exemple de utilizare ale diagramelor de clase:**

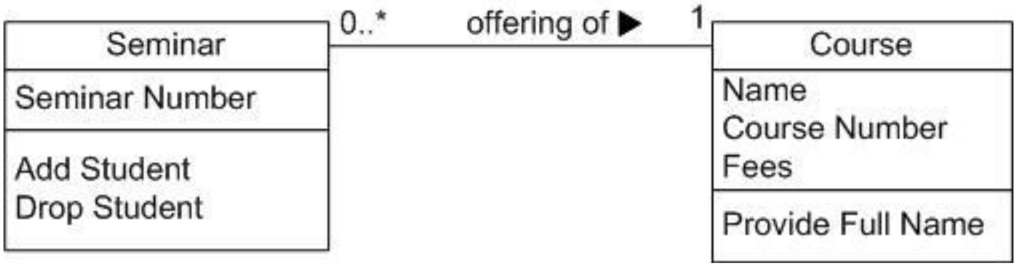
Conceptual, diagrama de clasa:



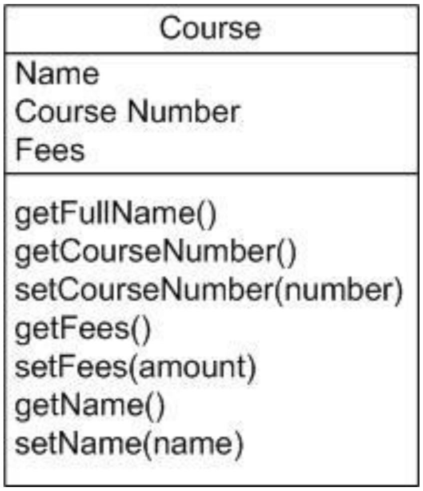
Clasele Student si Address (diagrama conceptuala de clasa)



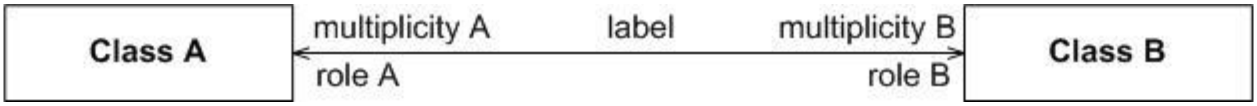
Clasa Seminar (diagrama conceptuala de clasa)



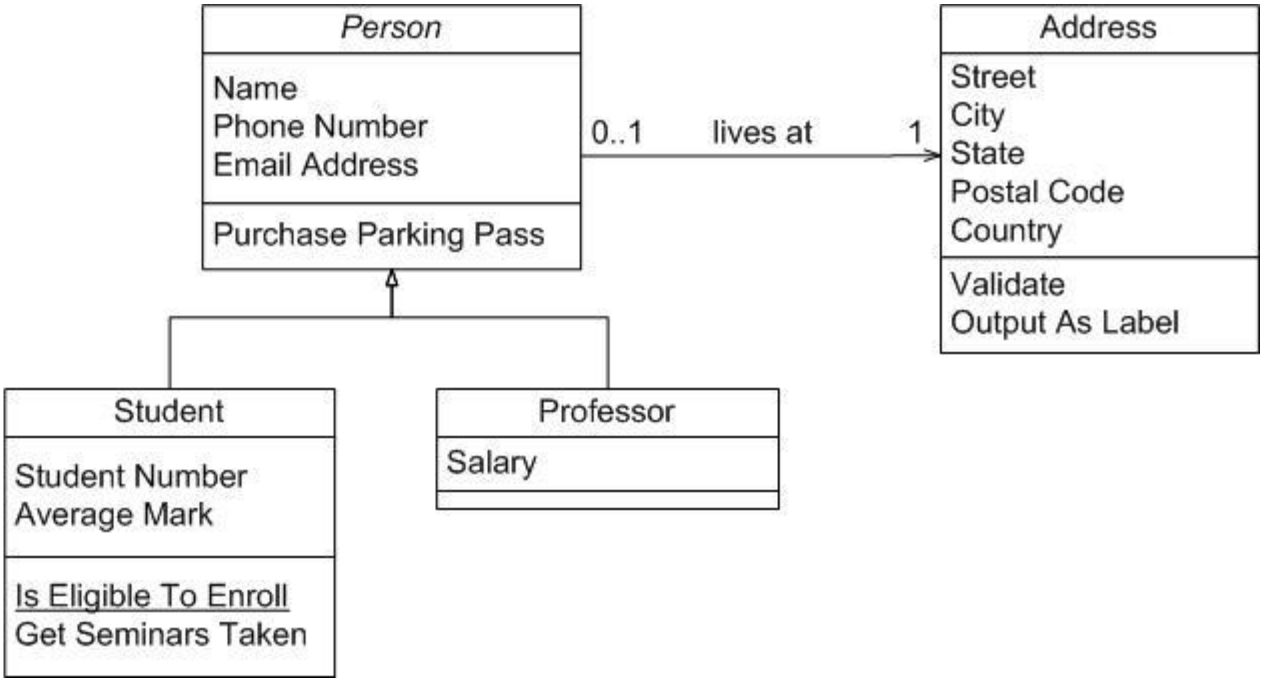
Clasa Course (cu metodele aferente)



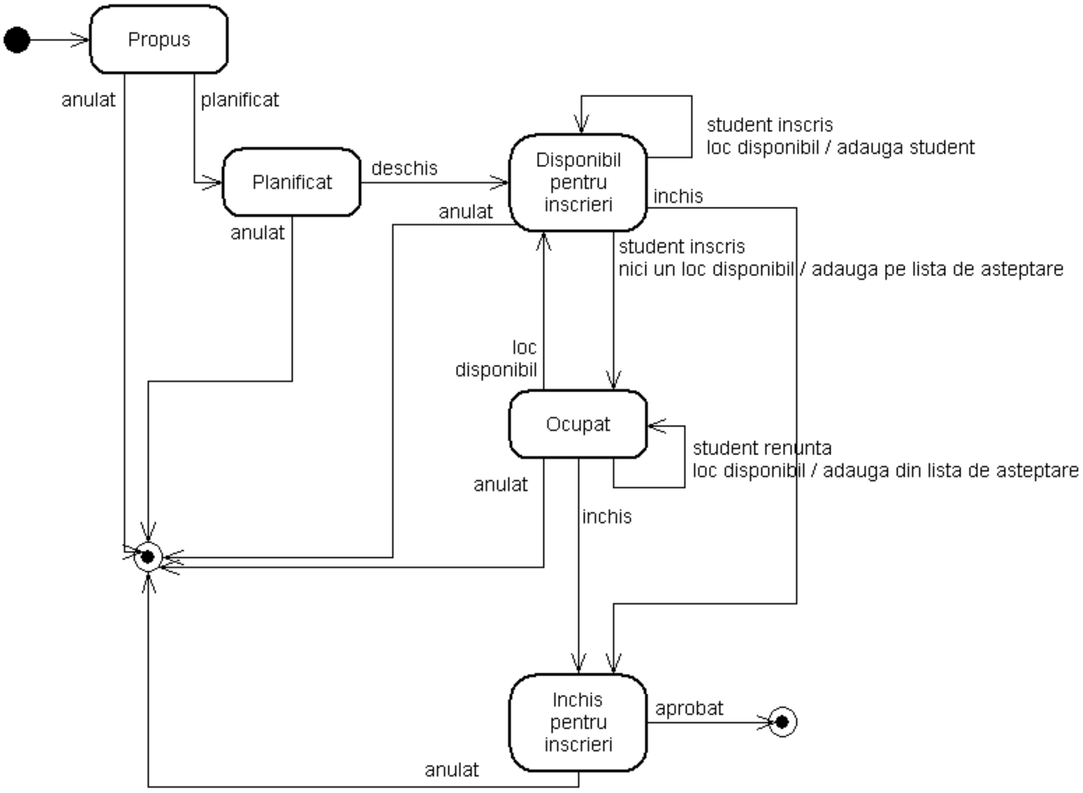
Pentru asociatii s-au folosit urmatoarele notatii:

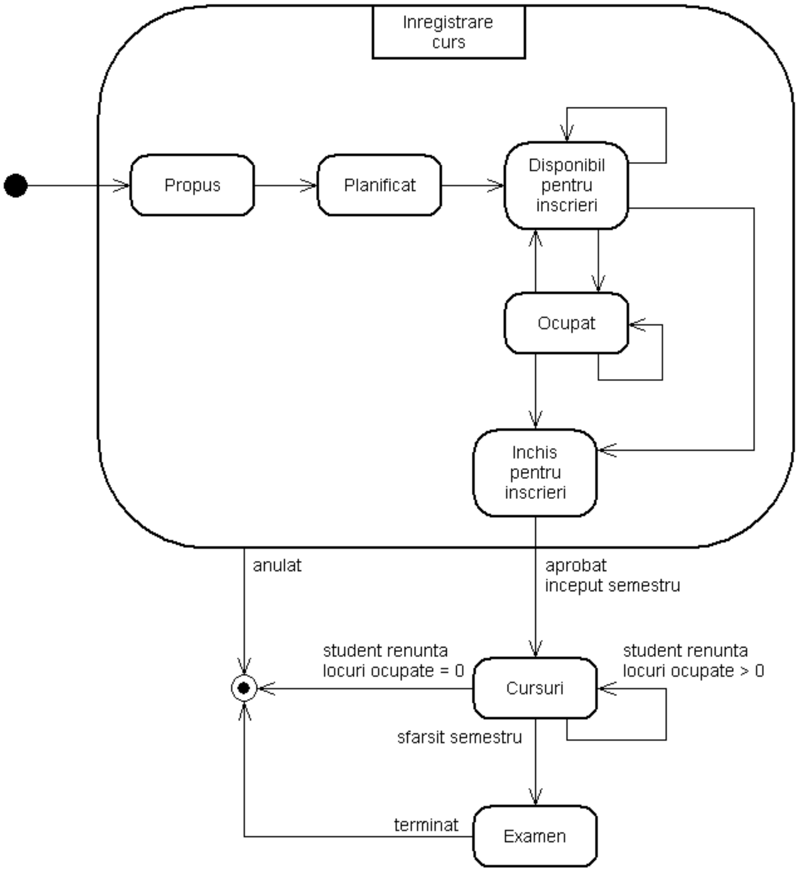


Ierarhia de mostenire intre clase:

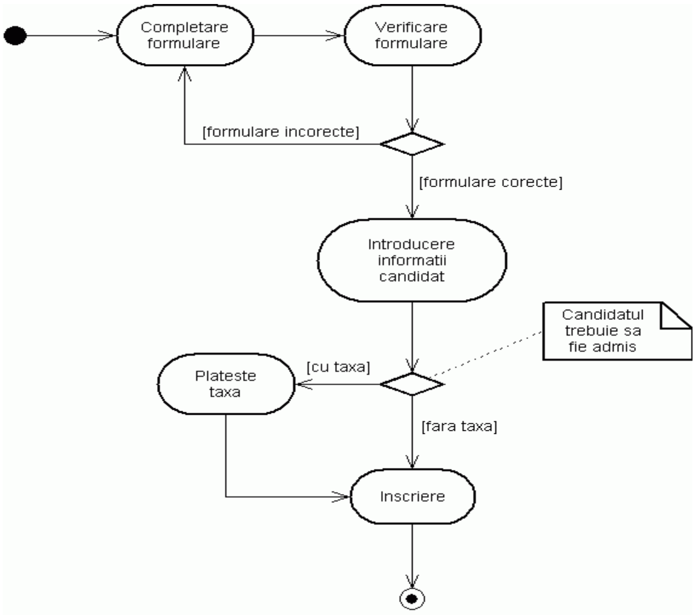


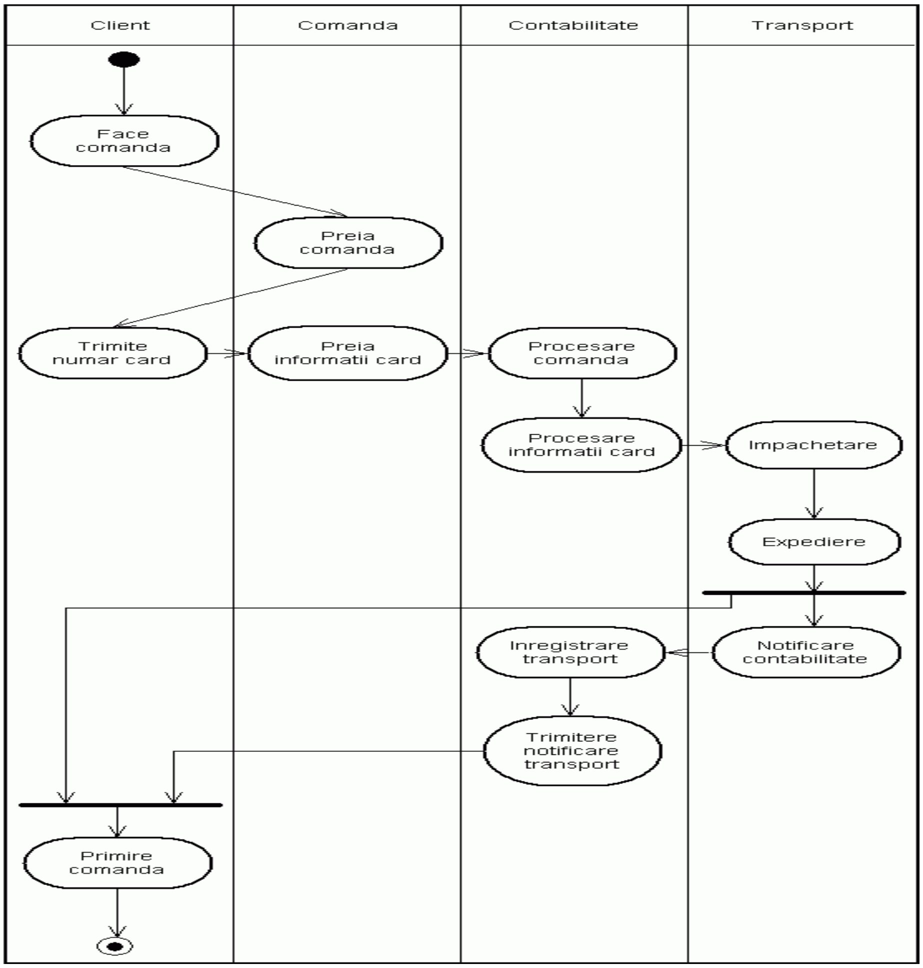
2. Exemple diagrame de stari:



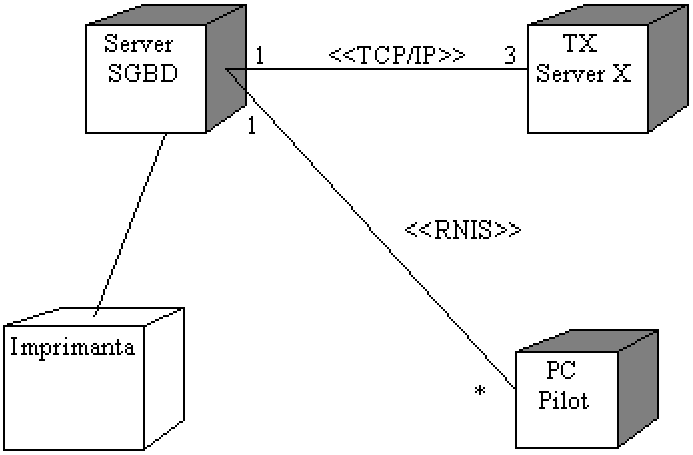


3. Exemple diagrame de activitati:



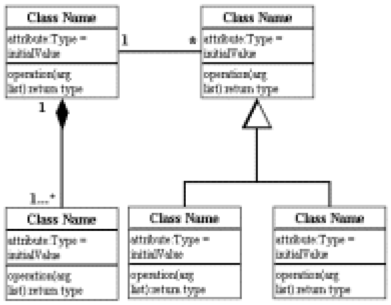


4. Exemplu diagrame de constructie:



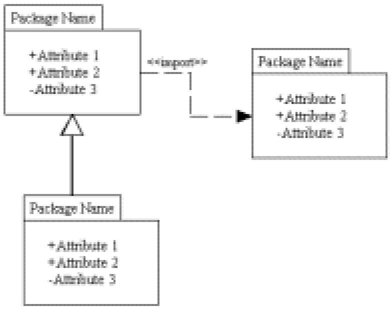
5. Concluzii

UML este un limbaj pentru specificarea, vizualizarea, construirea si documentarea elementelor sistemelor software. Este un standard de facto pentru modelarea software. UML permite modelarea cazurilor de utilizare si reprezentarea diagramelor de clase, de interactiune, de activitati, de stari, de pachete si de implementare. O sinteza a principalelor diagrame UML este prezentata in continuare.


Class Name
attribute:Type = initialValue
operation(arg list) return type
1
*
1
1...*

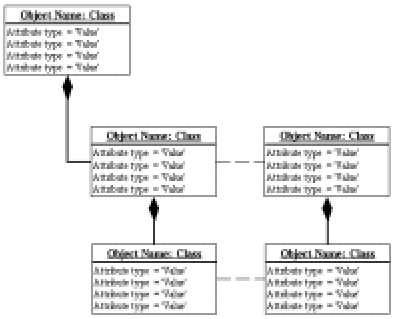
Class Diagrams

Class diagrams are the backbone of almost every object oriented method, including UML. They describe the static structure of a system.



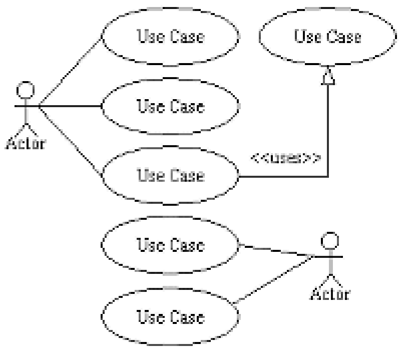
Package Diagrams

Package diagrams are a subset of class diagrams, but developers sometimes treat them as a separate technique. Package diagrams organize elements of a system into related groups to minimize dependencies between packages.



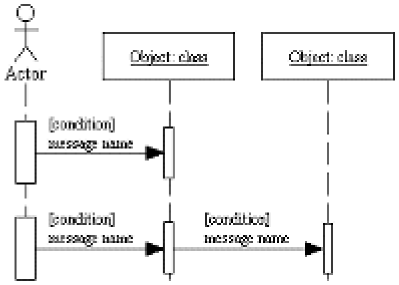
Object Diagrams

Object diagrams describe the static structure of a system at a particular time. They can be used to test class diagrams for accuracy.



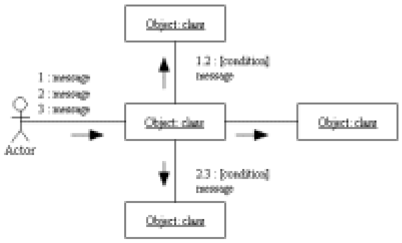
Use Case Diagrams

Use case diagrams model the functionality of system using actors and use cases.



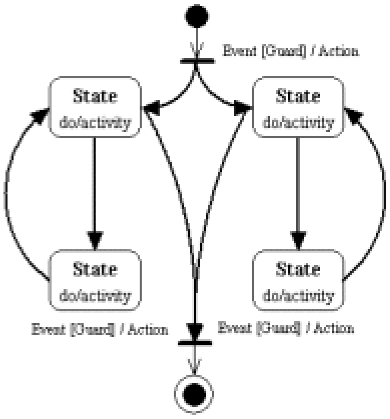
Sequence Diagrams

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.



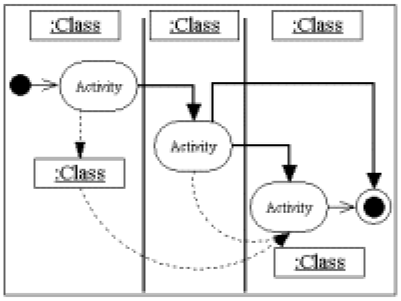
Collaboration Diagrams

Collaboration diagrams represent interactions between objects as a series of sequenced messages. Collaboration diagrams describe both the static structure and the dynamic behavior of a system.



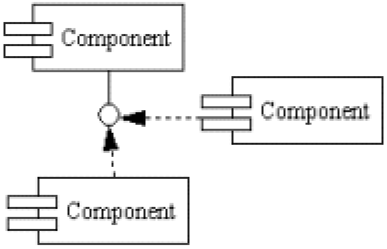
Statechart Diagrams

Statechart diagrams describe the dynamic behavior of a system in response to external stimuli. Statechart diagrams are especially useful in modeling reactive objects whose states are triggered by specific events.



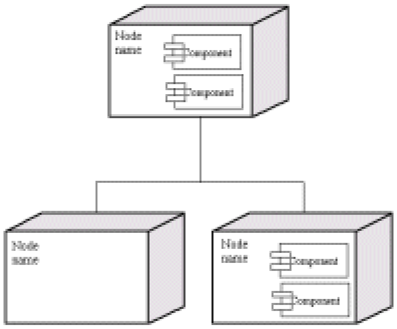
Activity Diagrams

Activity diagrams illustrate the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.



Component Diagrams

Component diagrams describe the organization of physical software components, including source code, run-time (binary) code, and executables.



Deployment Diagrams

Deployment diagrams depict the physical resources in a system, including nodes, components, and connections.

6. Anexa: Exemplu de caz pentru utilizarea diagramelor UML

0. Introduction

The aim of this tutorial is to show how to use UML in "real" software development environment.

1. Elevator Problem

A product is to be installed to control elevators in a building with m floors. The problem concerns the logic required to move elevators between floors according to the following constraints:

- Each elevator has a set of m buttons, one for each floor. These illuminate when pressed and cause the elevator to visit the corresponding floor. The illumination is canceled when the elevator visits the corresponding floor.
- Each floor, except the first floor and top floor has two buttons, one to request and up-elevator and one to request a down-elevator. These buttons illuminate when pressed. The illumination is canceled when an elevator visits the floor and then moves in the desired direction.
- When an elevator has no requests, it remains at its current floor with its doors closed.

2. Unified Modeling Language

UML is a modeling language that only specifies semantics and notation but no process is currently defined. Thus, we decided to do the analysis as follows;

- Use Case Diagram
- Class Diagram
- Sequence Diagram
- Collabration Diagram
- State Diagram

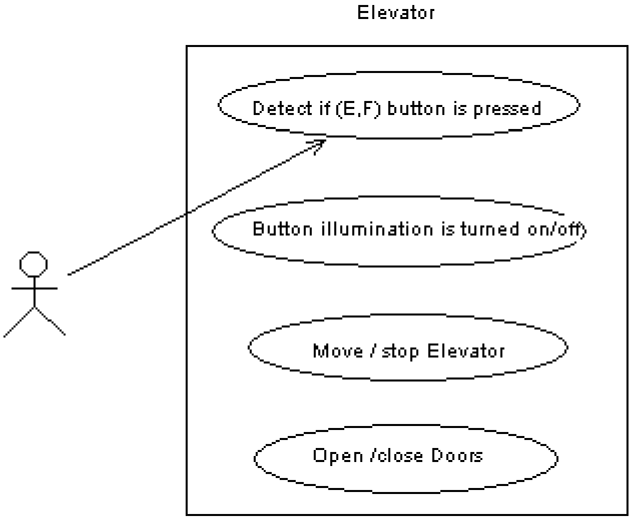
3. Analysis

3.1. Use case diagram

Use case description:

- A generalized description of how a system will be used.
- Provides an overview of the intended functionality of the system.
- Understandable by laymen as well as professionals.

Use Case Diagram:


Elevator
Detect if (E,F) button is pressed
Button illumination is turned on/off)
Move / stop Elevator
Open /close Doors

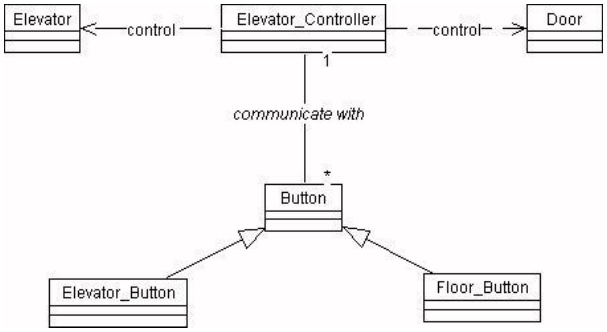
Elevator basic scenario that can be extracted from Use Case Diagram:

- Passenger pressed floor button
- Elevator system detects floor button pressed
- Elevator moves to the floor
- Elevator doors open
- Passenger gets in and presses elevator button
- Elevator doors closes
- Elevator moves to required floor
- Elevator doors open
- Passenger gets out
- Elevator doors closes

3.2. Class Diagram

Class diagrams show the static structure of the object, their internal structure, and their relationships.

Class diagram:



3.3. State diagram

A state diagram shows the sequences of states an object goes through during it's life cycle in response to stimuli, together with its responses and actions.

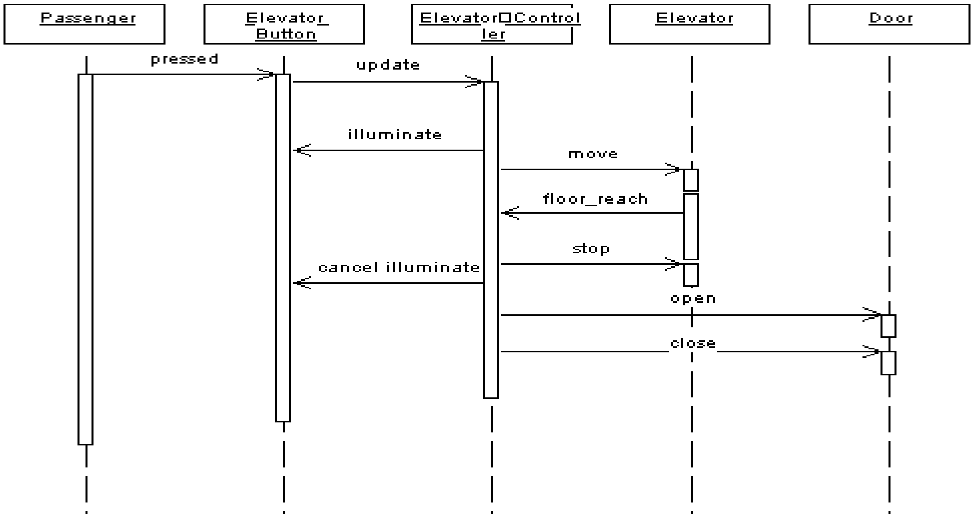
4. Design

The design phase should produce the detailed class diagrams, collaboration diagrams, sequence diagrams, state diagrams, and activity diagram. However, the elevator problem is too simple for an activity diagram. Thus, we are not using an activity diagram for the elevator problem.

4.1. Sequence Diagram

A sequence diagram and collaboration diagram conveys similar information but expressed in different ways. A Sequence diagram shows the explicit sequence of messages suitable for modeling a real-time system, whereas a collobration diagram shows the relationships between objects.

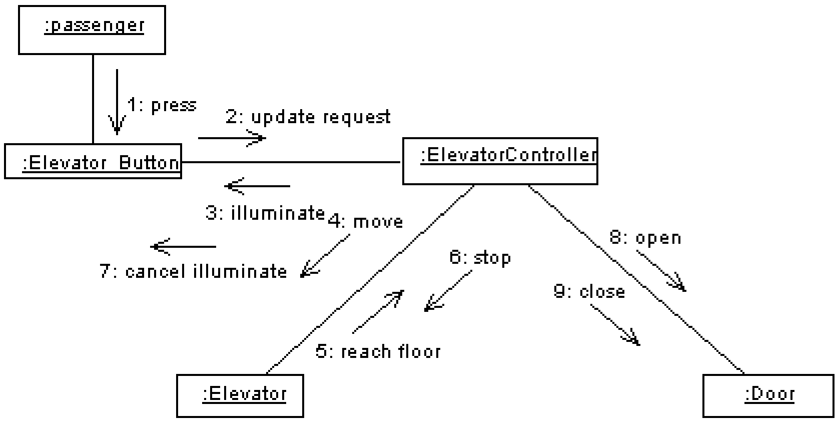
Sequence Diagrams:



4.2. Collaboration diagram

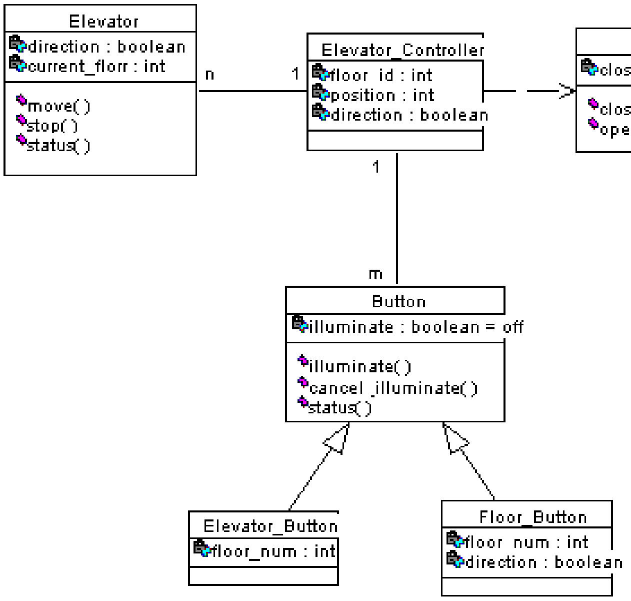
- Describes the set of interactions between classes or types
- Shows the relationships among objects

Collabration diagrams:



5. Detail Design

5.1. Detail Class Diagram



5.2. Detail Operation Description

Module Name	Elevator_Control::Elevator_control_loop
Module Type	Method

Input Argument	None
Output Argument	None
Error Message	None
File Access	None
File Change	None
Method Invoke	button::illuminate, button::cancel_illumination, door::open, door::close, elevator::move, elevator::stop
Narative	

5.3. Pseudo-Code

```
void elevator_control (void)
{
  while a button has been pressed
      if button not on
      {
          button::illuminate;
          update request list;
      }
      else if elevator is moving up
      {
            if there is no request to stop at floor f
                Elevator::move one floor up;
            else

}
```