Exercise 1

Introduction
Vision of the IS
Repetition of the UML

2021/2022

Martin Radvanský

Teacher

RNDr. Ing. Martin Radvanský, M.Sc., Ph.D.

Email:

martin.radvansky@vsb.cz

consultation:

EA 439, Wednesday 9:00 - 11:00, MS TEAMS

Web:

homel.vsb.cz/~rad0028

The course of the exercise

General tasks for the exercise:

Attendance is compulsory (max. 2 × excused absence)

Late submission of artifact -1p, every 2 days -1p

- Discussion
- Presentation
- Notes on IS development and use of technology.
- Consultation of the semester project.
- Programming assignments given in lecture.

Submission of artifacts:

Submit via school email only

- File name and subject in the form of: **VIS_personal_number_Group_Ax (VIS_RAD0028_C01_A1)**
- Send in Pdf, Zip format (remove exe and dll from zip file)

Semester project

IS design and implementation

- Minimized scope.
- More complex architecture.
- Limited use of technology.
- Integration task.
- Minimal (complete) documentation.
- 2 different UIs and 2 different storage.
- Assignment - in the form of a project vision to be ready for approval up to **29.9.2021** (next exercises).

Artifacts I.

- [2/1] Vision (document describing the system from the customer's point of view).
- [7/4] Functional specification (use case model - description of individual cases, use case diagram, activity diagrams).
- [3/2] Technical specification (first domain model, documentation for technological decisions, selected technologies and processes).
- [2/1] Sketch (wireframe, prototype) of the user interface.

Artifacts II.

- [7/4] Domain model design (classes, relationships, interactions - static class diagram, sequential diagram, patterns used).
- [3/2] Description of the system architecture (layout and interconnection of the logical and physical layers, diagram of components).
- [18/9] Consistent functional part of the selected information system with high emphasis on the architecture and design (layering, design in individual layers, patterns). It is assumed implementation of two simple user interfaces on different platforms, at least three non-trivial use cases in each of them and the use of two data storage methods (SQL database, XML etc.).

Vision of semestral project

Vision of Information system

- **Project initiation phase**
  - **Understanding what is to be created. Identify all external elements of the system (actors) and their main requirements for the emerging system (key use cases)**
  - Determining at least one possible solution. At least one potential architecture is selected to enable the system to be built. The risk elements of this architecture shall be implemented and verified.
  - Identification of potential risks. At the end of the initiation phase, there must be a rough idea of the problems that may arise in the development of the system.
  - Clarification of the cost, schedule, resources required and economic value of the project.
  - Determination of the software process and tools to be used.

Vision - document describing the system from the customer's point of view.

- Range approx. ½-1 page of text.
- Answers questions
  - **WHAT, WHO, WHEN, HOW, WHERE, WHY**
  - **Clear, simple wording**
- Discussion of the visions presented.

Scooter rental IS

What?

Who?

When?

How?

Why?

We design an information system for comprehensive management of electric scooter rental. Four categories of users will work with this system. A customer who wants to rent a vehicle accesses the system remotely through a web interface. The customer can have a customer account created in the system, which allows him to reserve a specific scooter, pay cashless and receive a bonus on repeat rentals. The IS is also operated by a staff member at the branch who makes bookings/orders of scooters and works on a desktop computer in a windows application. She enters the status of the vehicle into the IS when it is handed over to the customer and when it is taken back. May make adjustments to user orders, cancel them and check for payment. The system is operated by a supervisor who can obtain information on the economics of the operation and set discounted prices for customers or issue invoices for damage to the vehicle. The last type of IS user is the technical department worker, who enters information about detected faults into the system and blocks the possibility of renting a scooter.

Scooter rental IS

The IS monitors information on customer bookings, customer payments, technical condition of scooters, allows for price advantages for regular clients and generates documents for the economic department. The system is essential for the fulfilment of the business plan.

The IS is available 24 hours a day via the Internet for customer needs and including data is located on the local company server.

UML

- UML is a universal language for visual modelling of systems. It is primarily associated with the modeling of object-oriented software systems.
- The individual models are presented graphically by means of diagrams. UML offers 13 types of diagrams. This allows different views of the system to be recorded.
- When creating diagrams, the use of CASE (computer-aided software engineering) tools is expected. Currently, there are a number of them. https://www.visual-paradigm.com/editions/

UML diagrams

- Use case diagram
- Object diagram (domain objects)
- Class diagram (classes)
- State diagram (technological processes)
- Activity diagram (dynamics)
- Sequence diagram (timing)
- Collaboration diagram (cooperation)
- Component diagram (components)
- Deployment diagram (deployment/location)

Use case diagram

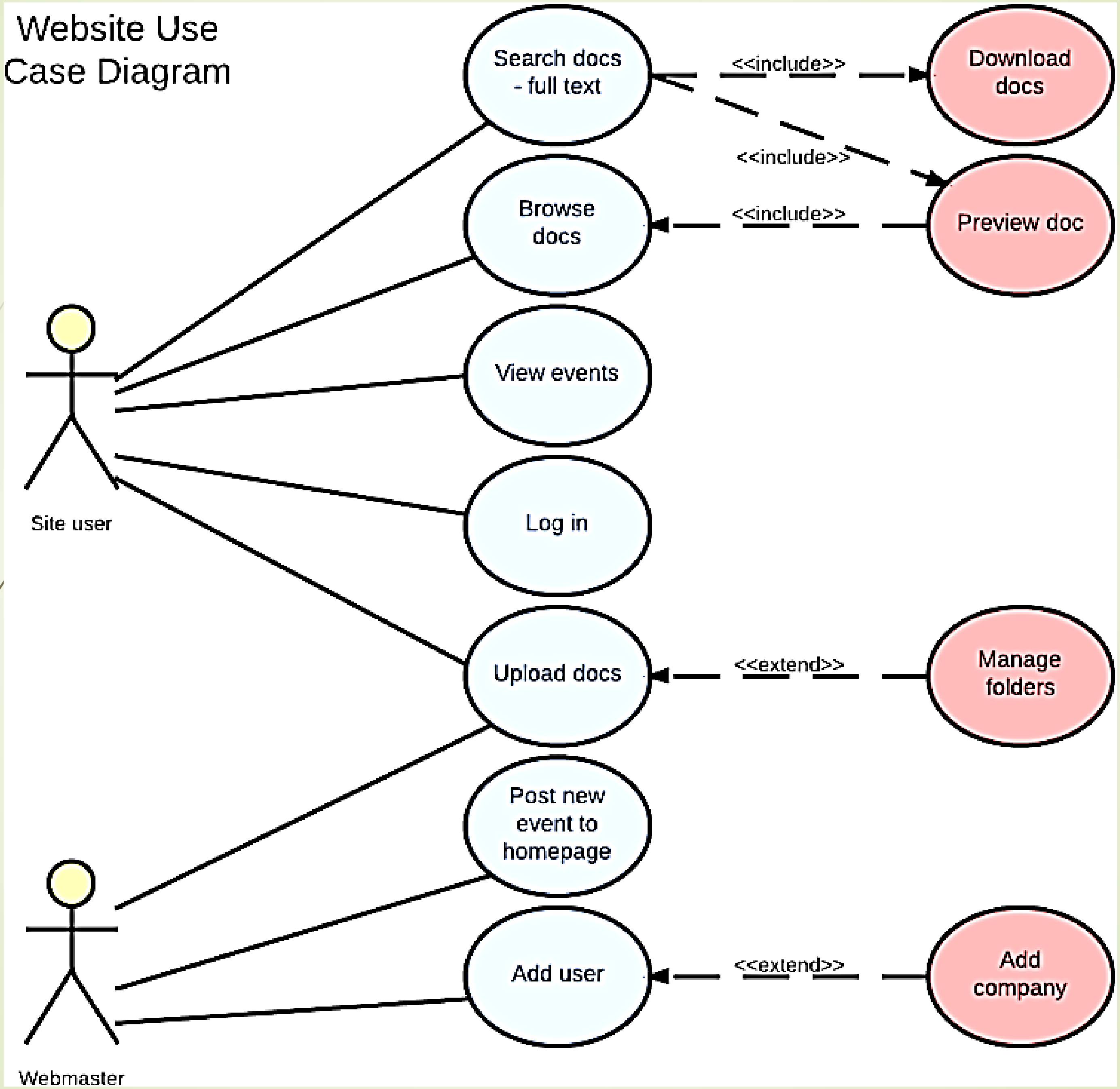
UML supports software processes that are driven by user requirements for the system. It provides the following concepts to record them.

- **Actor** specifies the role in which a particular entity (person or other system) interacts with the system being modelled. It is an element that invokes some functionalities of the modelled system and controls their course, but is not part of it.
- **A use case** defines one possible use of the system by an actor. It describes the sequence of interactions between the actor and the system leading to the achievement of the goal that the user has pursued by invoking the use case.

Use Case Diagram - Bindings

The diagram does not include information on how each use case proceeds. This must be described in a separate text document. Relationships between use cases refer to situations where some activities are repeated in multiple use cases.

- **Include** says that each time one use case is executed, the scenario associated with another use case is executed simultaneously.
- **Extend** allows you to extend the behavior of a basic use case by invoking another use case. This is how we handle cases that **extend the base use case** behavior. The binding is drawn in reverse, i.e. the arrow points from the extending behaviour to the extended use case.


Website Use Case Diagram
Search docs - full text
Browse docs
View events
Log in
Upload docs
Post new event to homepage
Add user
Download docs
Preview doc
Manage folders
Add company
<<include>>
<<include>>
<<include>>
<<extend>>
<<extend>>
Site user
Webmaster

Use case scenarios

Scenarios are an essential part of use case diagrams, especially for complex activities or when the name of the use case does not clearly indicate its functionality.

Scenarios or use-case description:

- System status before the start of the use case.
- The sequence of events after a use case is triggered by an actor in the baseline scenario.
- The sequence of events in the alternative scenario/scenarios.
- System state after the end of the use case.

Use case scenario - an example

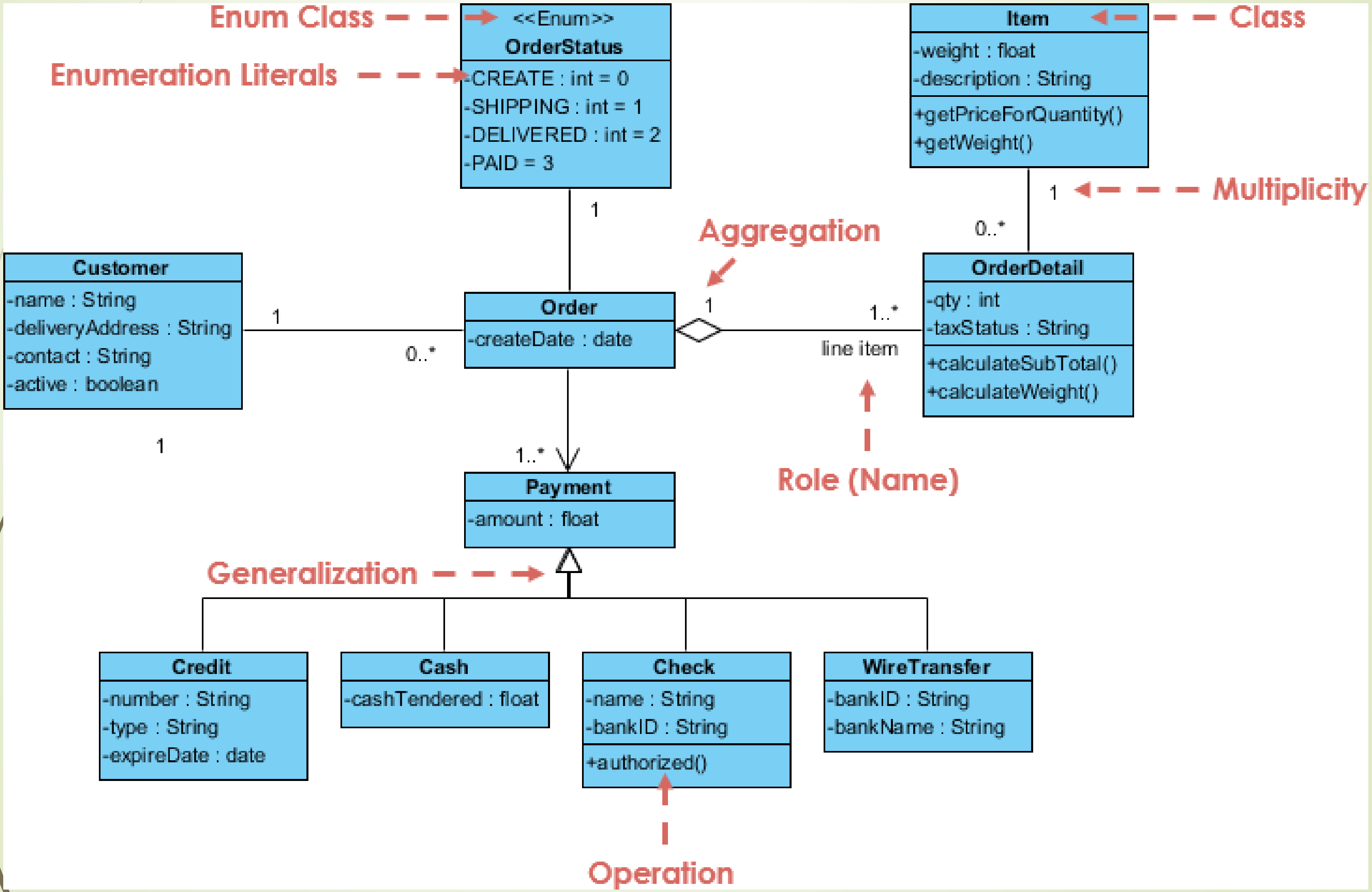
Use case:	UC01: Search doc – full text
Goal:	Finding and viewing a document on the web.
Actor:	Site user
Anothers actors:	System
Pre-conditions:	Fulltext index over web site
Post-conditions:	

Main run:

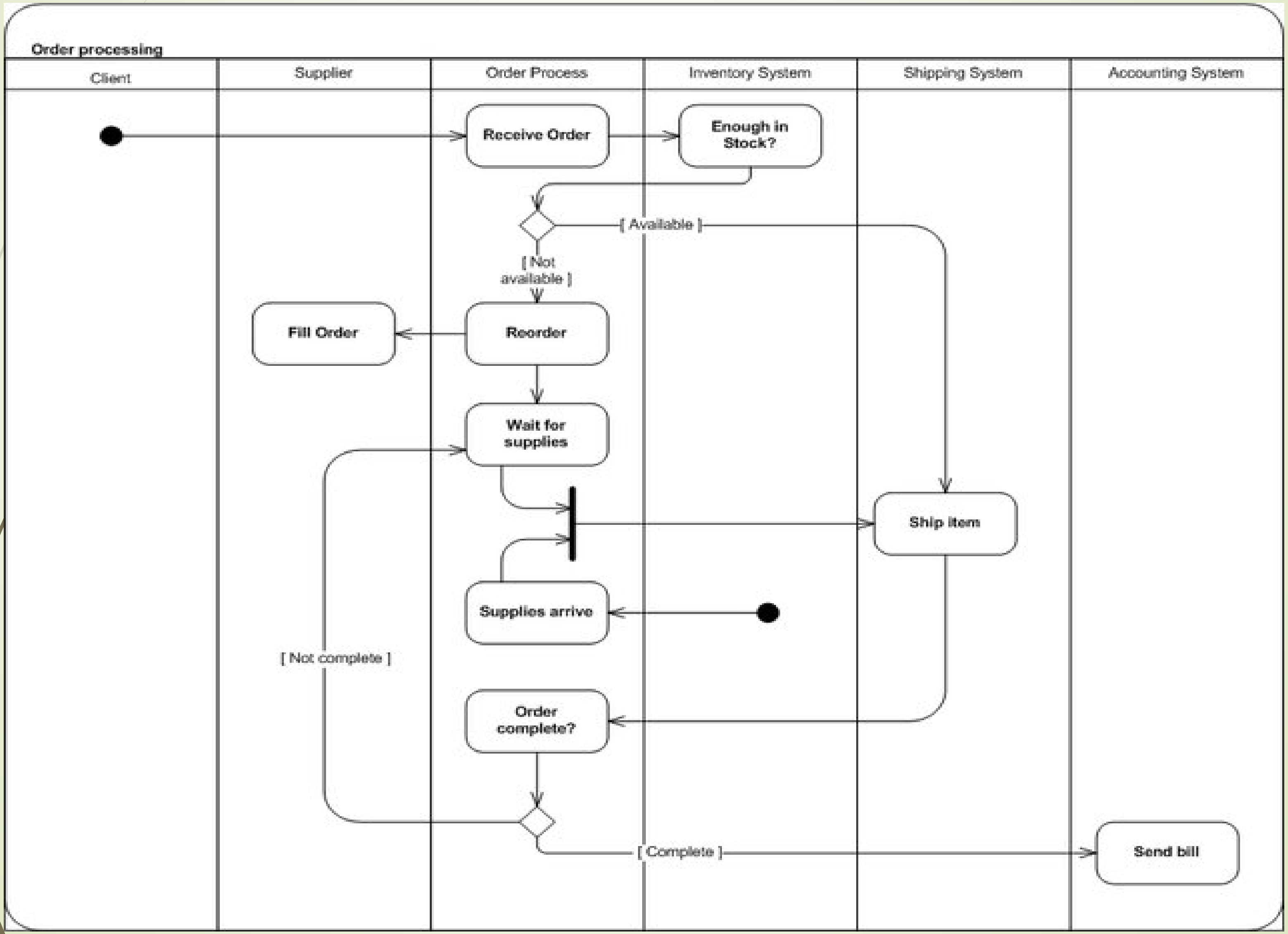
1. **Site user -** enters the search text in the search field.
2. **System -** performs a full-text search.
3. **System -** displays a list of found files with the Download option and displays a Preview.

Alternative run: ???

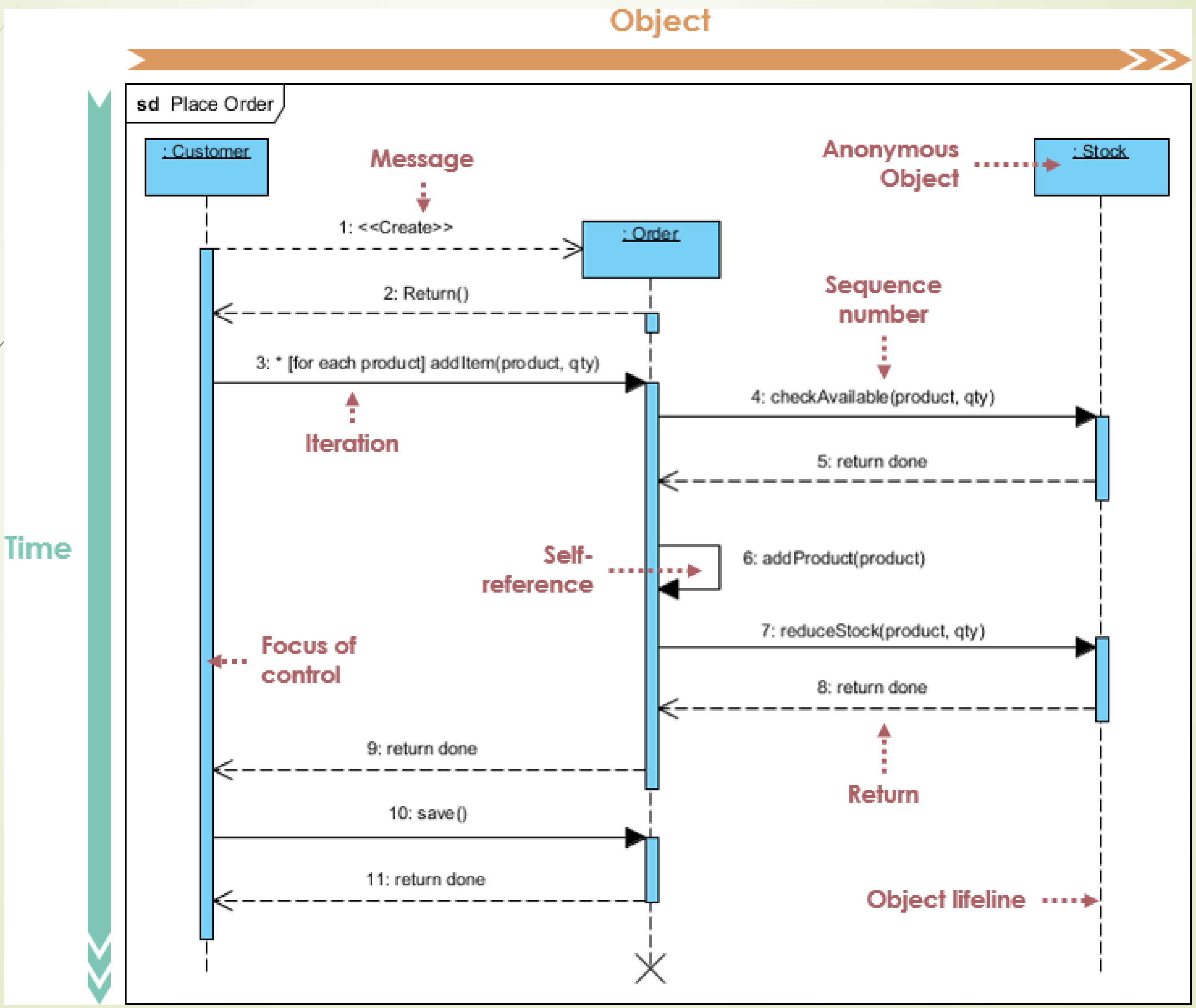
Class diagram – Ordering system



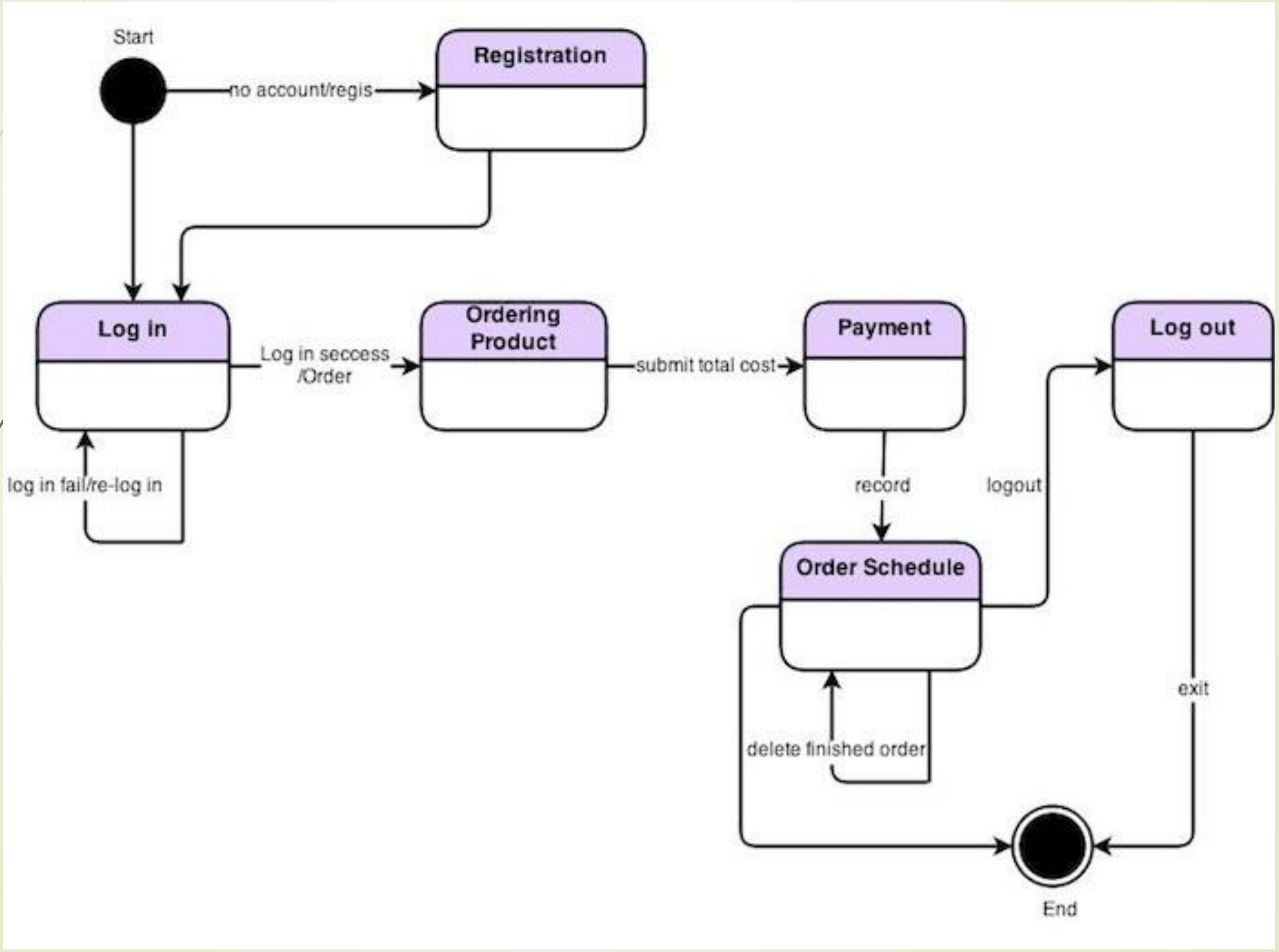
Activity diagram Ordering system



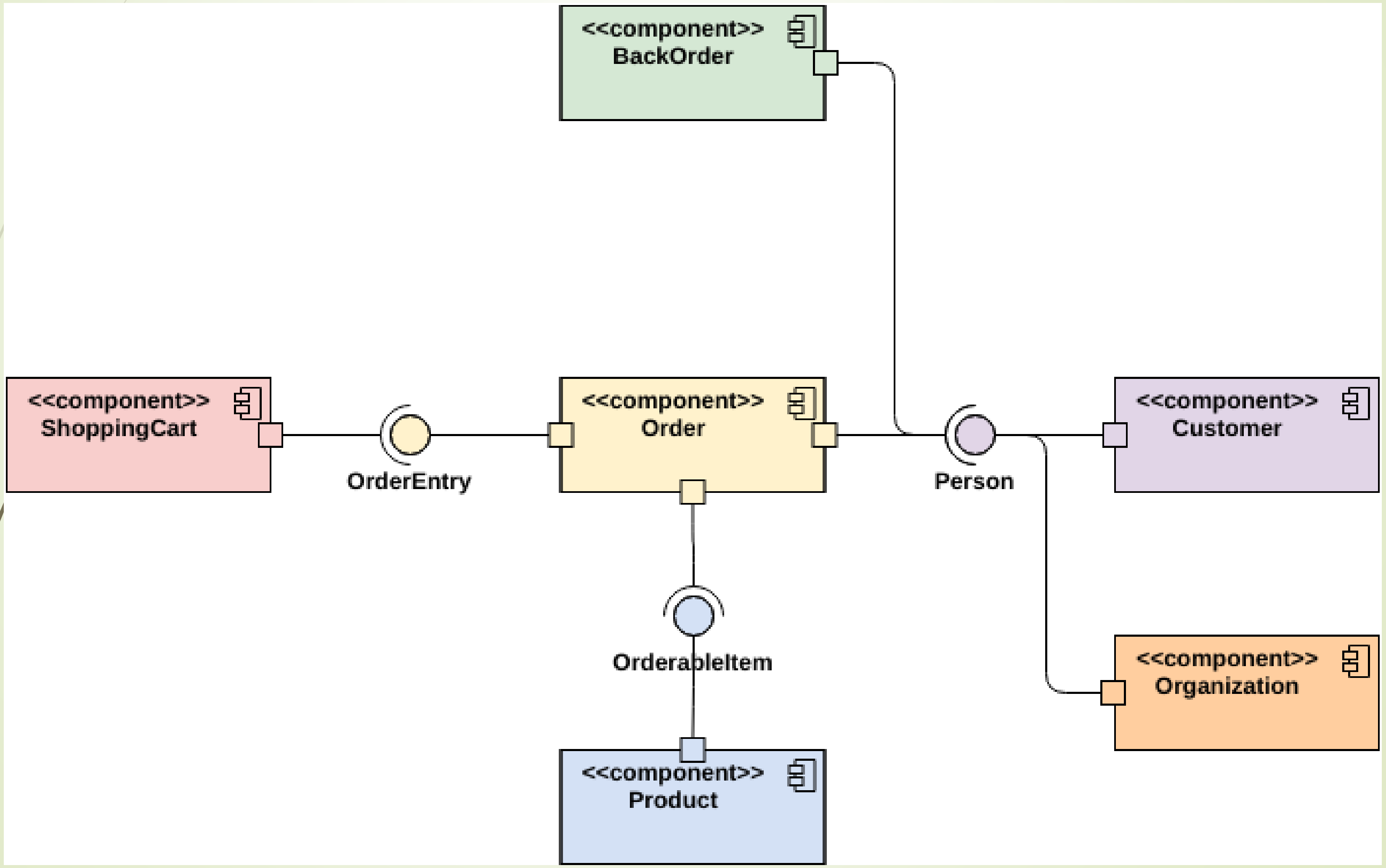
Sequence diagram



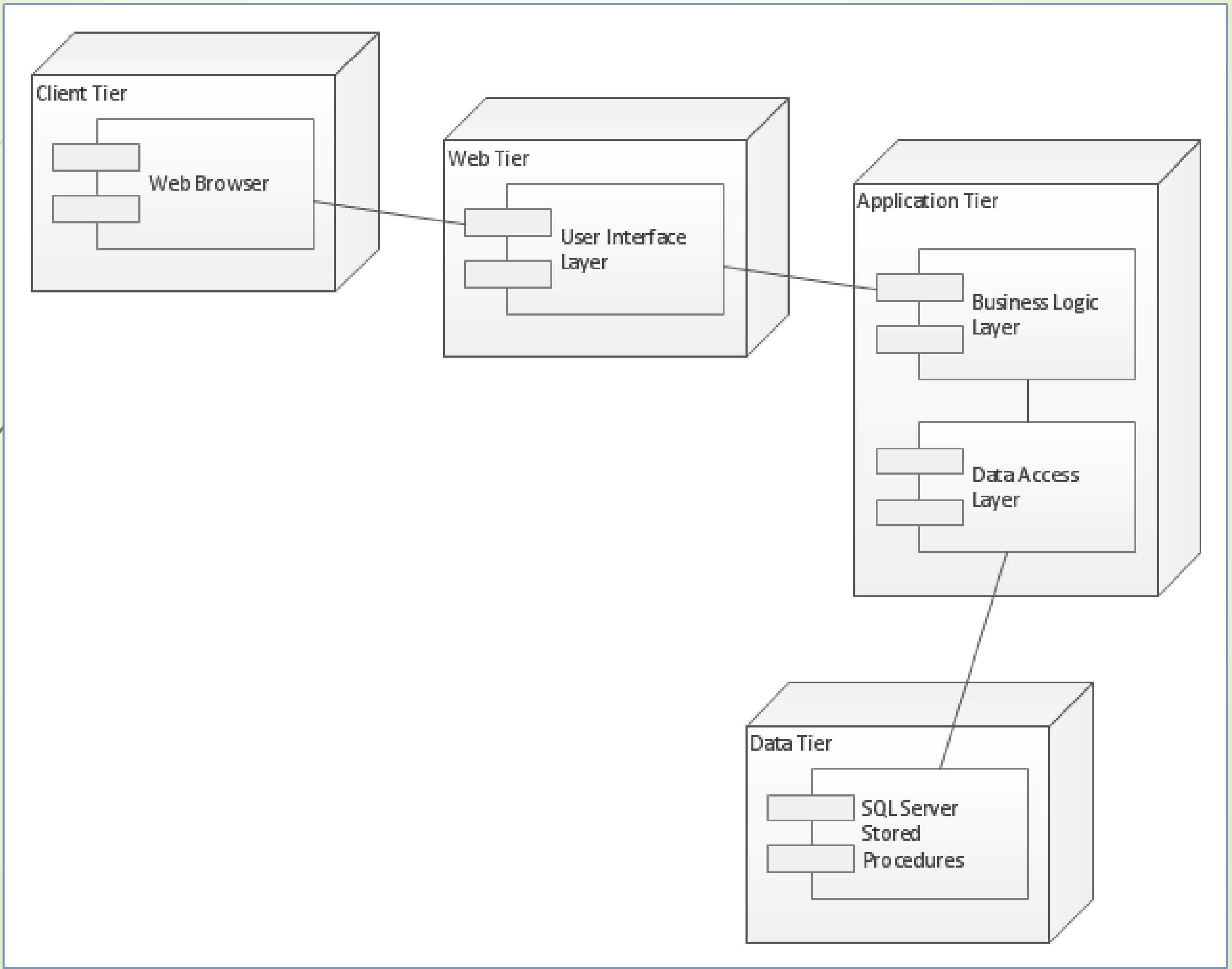
State diagram



Component diagram

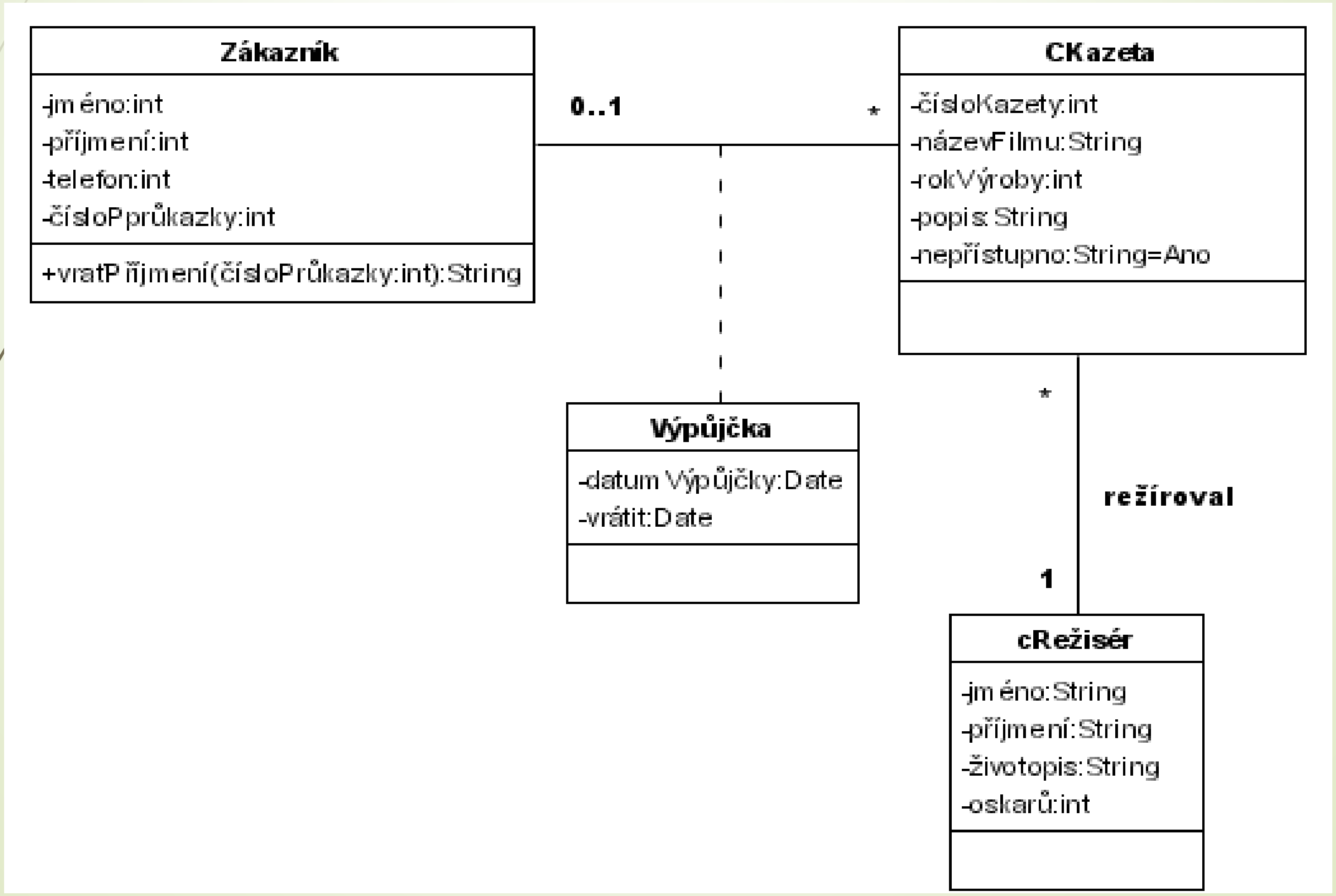


Deployment diagram



Class diagram – diagram tříd

Diagram tříd zobrazuje statický pohled na systém, zejména třídy (class) jako typy objektů, obsah tříd a statické vztahy, které mezi nimi existují.

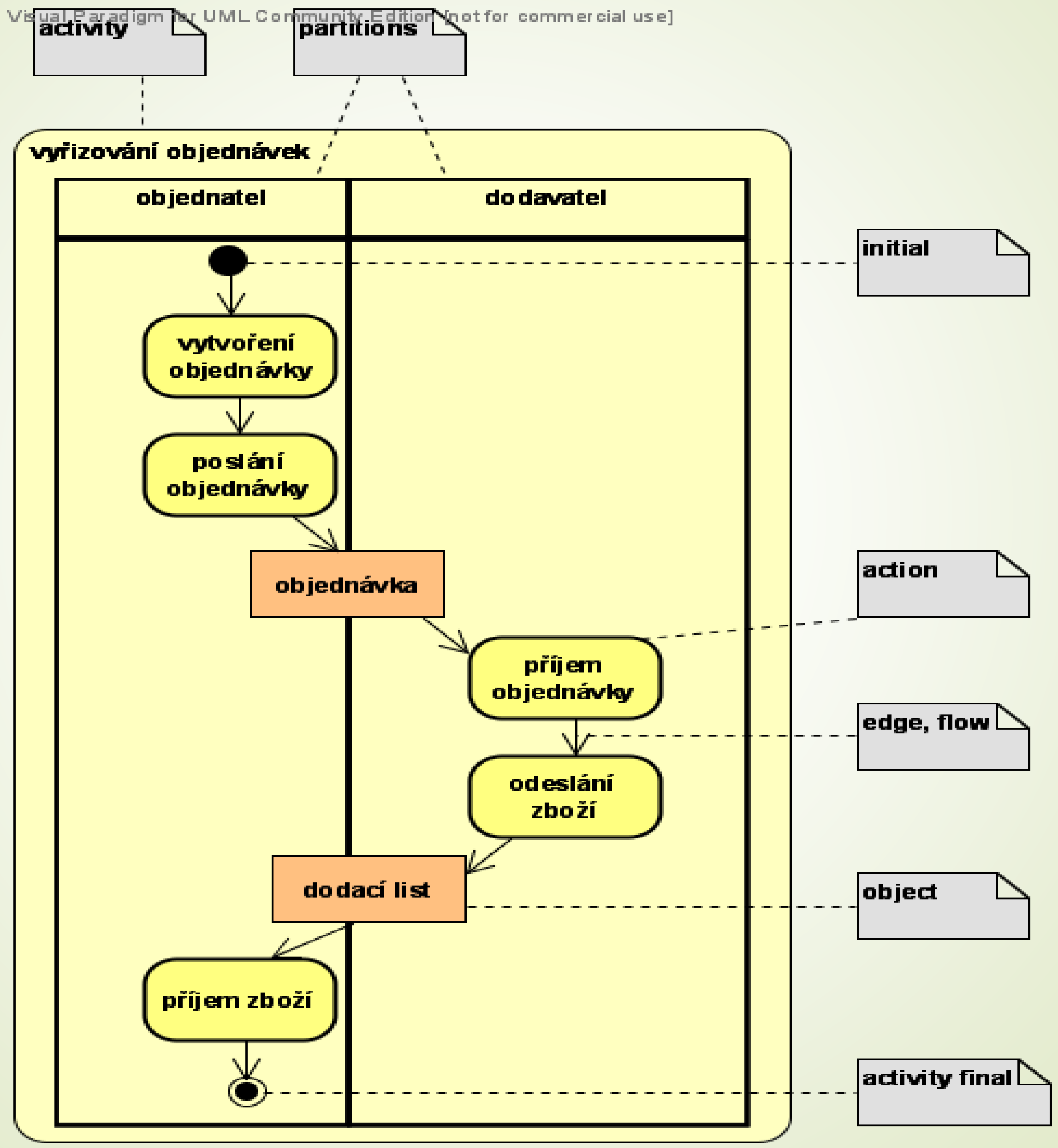


Activity diagram

Diagram aktivit se používá pro popis dynamických aspektů systému.

- Jde o jakýsi *flowchart* - znázorňuje tok řízení z aktivity do aktivity. Používá se také k modelování obchodních (business) procesů a workflow.
- Diagram aktivit se soustřeďuje spíše na proces výpočtu než na objekty účastnící se výpočtu (i když i objekty mohou být znázorněny jako prvek aktivity).

Activity diagram

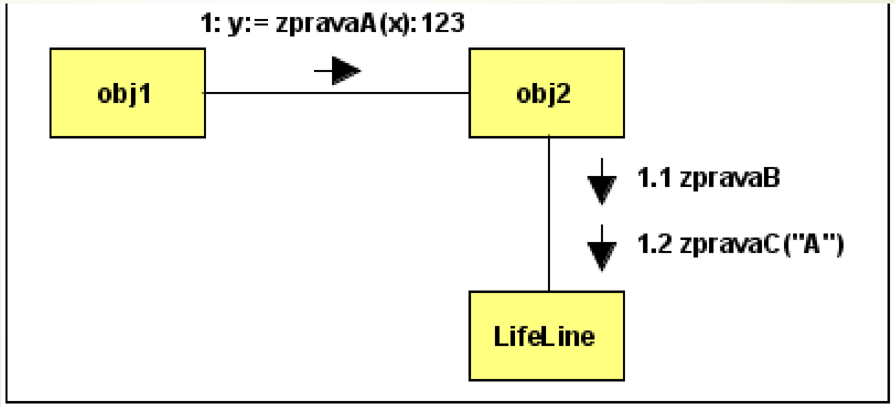


Collaboration diagram (komunikační) Interaction diagram (UML 2.0)

Diagram komunikací ukazuje objekty (přesněji : části kompozitní struktury nebo role ve spolupráci(collaboration)) a spojení a zprávy, které si objekty posílají. Čas zde nevystupuje jako zvláštní dimenze, proto musí být sekvence zpráv a spouběžnost threadů určena pomocí čísel sekvencí

Kolaborační diagramy a diagramy sekvenční jsou izomorfní

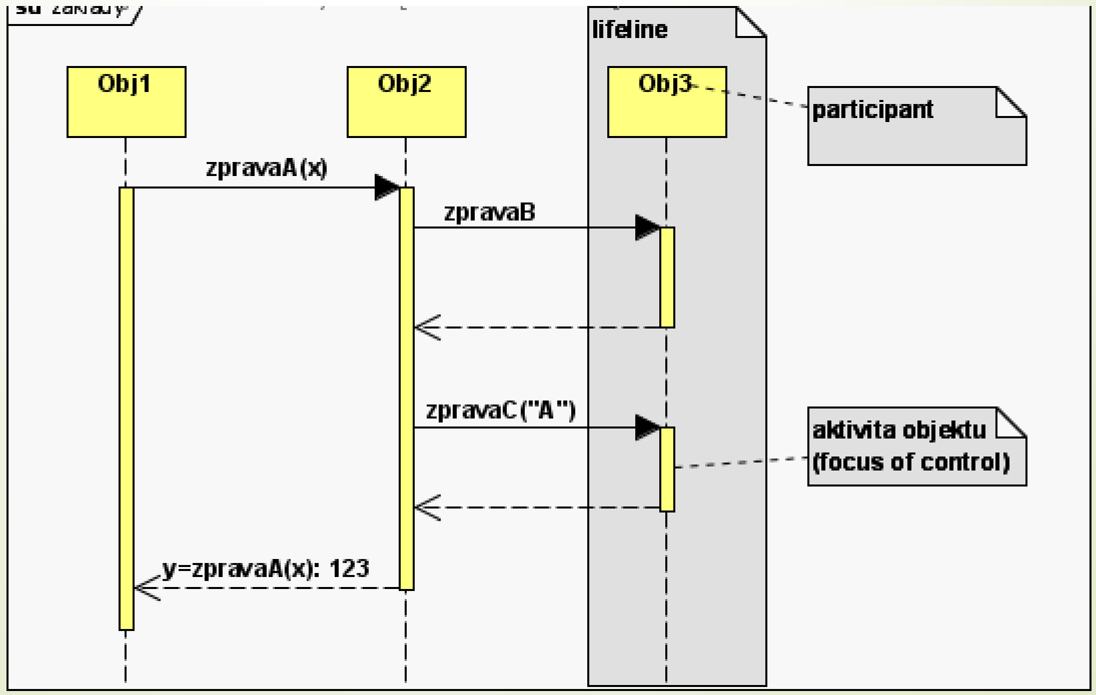
Použití diagramu komunikací bývá vhodnější v těch případech, kde chceme zdůraznit strukturální aspekty spolupráce, tj. ukázat hlavně **kdo s kým komunikuje** - jsou méně vhodné pro zdůraznění časových souvislostí interakcí.



Sequence diagram (sekvenční)

Objekty si mohou posílat zprávy. Sekvenční diagram zobrazuje jejich časovou posloupnost.

Použití sekvenčního diagramu bývá vhodnější v těch případech, kde jsou důležité časové souvislosti interakcí, ovšem nevidíme v něm zobrazené vztahy mezi objekty.

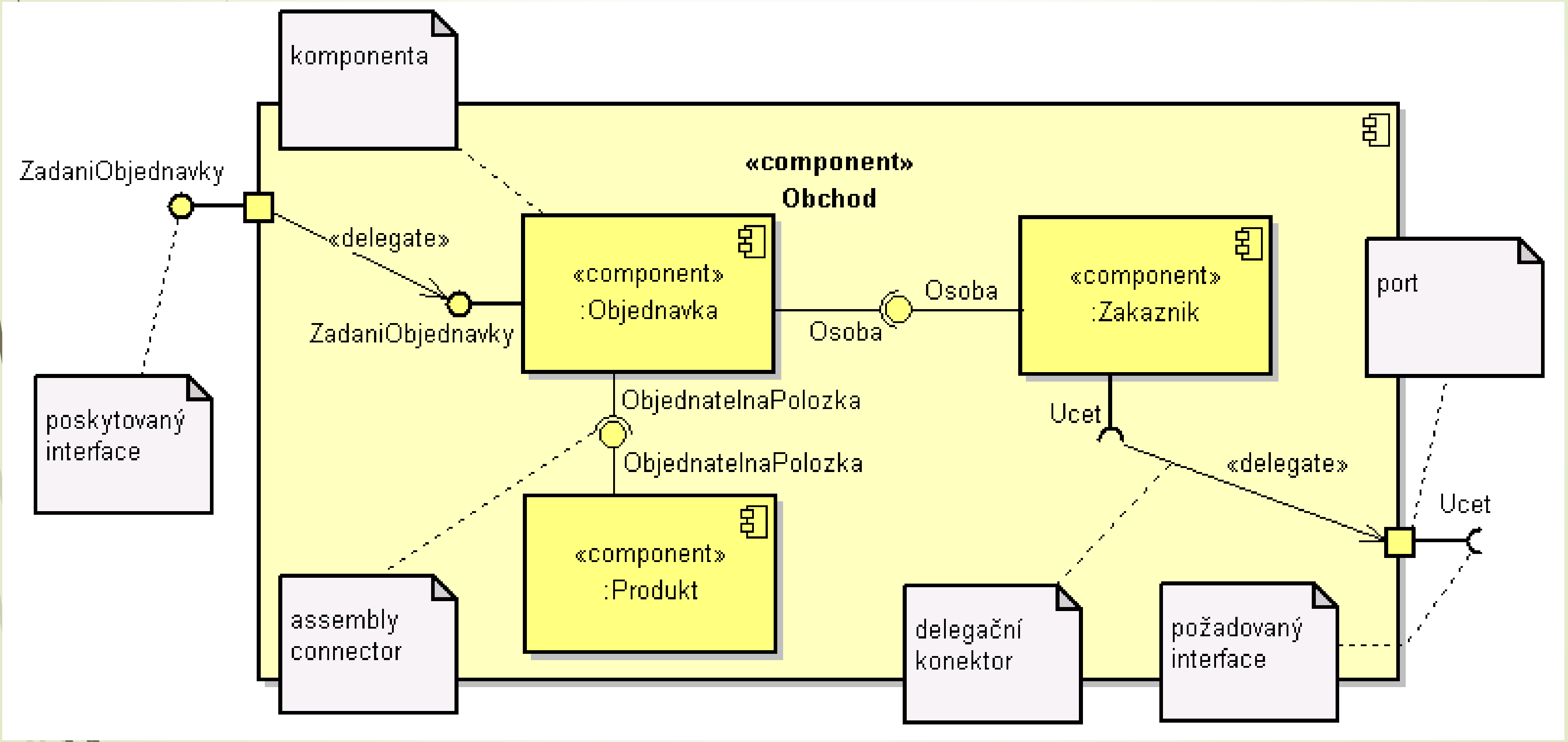


Component diagram (komponent)

Diagram komponent znázorňuje komponenty použité v systému, tj. **logické komponenty** (např. business k., procesní k.) či **fyzické komponenty** (např. EJB k., CORBA k., .NET k. atd.), může pomoci zejména tam, kde používáme vývoj založený na komponentách a kde struktura systému je založena na komponentách.

Komponenta je modulární část systému, která zapouzdřuje svůj obsah (tj. zapouzdřuje stav a chování i více klasifikátorů) a jejíž projev je nahraditelný (tj. komponenty, poskytující ekvivalentní funkcionalitu založenou na kompatibilitě jejich interfejsů, mohou být libovolně zaměňovány, a to buď už v čase tvorby designu, nebo až za běhu cílového systému). Komponenta je specializací strukturované třídy.

Component diagram (komponent)



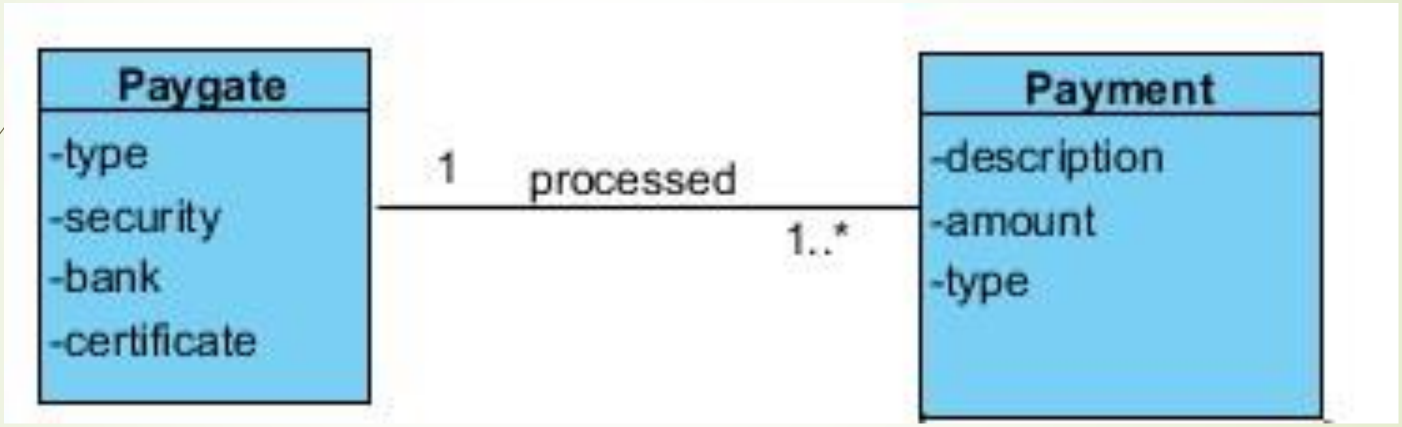
Úkoly na příště

- Nachystat Artefakt 1 – Vize k odsouhlasení projektu (dokument popisující systém z pohledu zákazníka).
- Zopakovat si UML

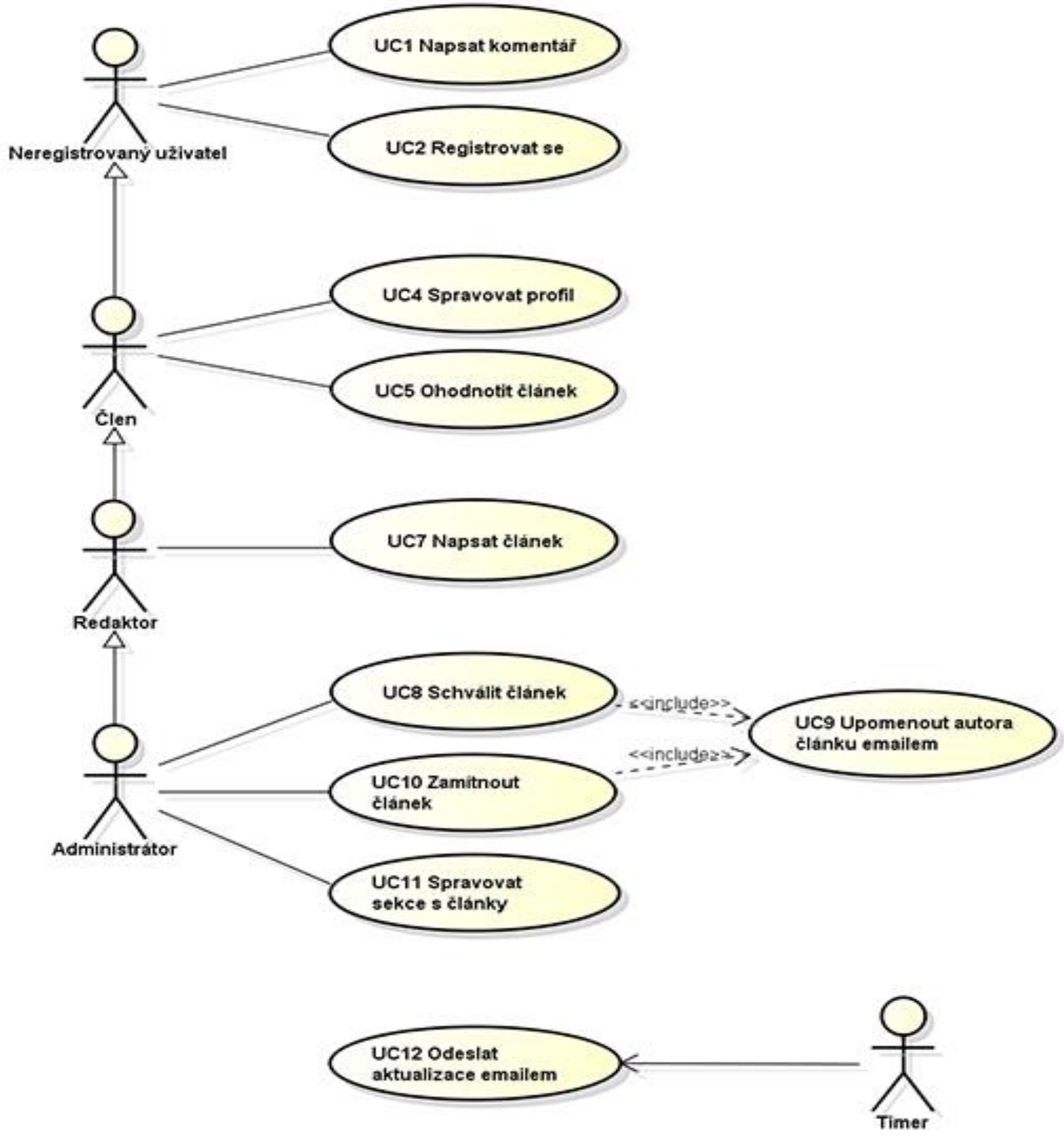
Programování: úkol z přednášky

Navrhnout minimálně dvě třídy s asociací pro semestrální úlohu a připravit jednoduchou implementaci od formuláře po uložená data.

Příklad 2 tříd s asociací



Redakční systém



Use Case – případy užití

UC1 - Napsat komentář

Use case umožňuje okomentovat článek v redakčním systému.

Aktéři: Uživatel, Systém

Podmínky pro spuštění: Uživatel se musí nacházet na příslušném článku.

Základní tok

1. Systém vygeneruje formulář a obrázek se 4mi náhodnými, orotovanými písmeny.
2. Uživatel vyplní text zprávy, své celé jméno a email. Dále opíše text z obrázku do připraveného pole a formulář odešle.
3. Systém zvaliduje data od uživatele.
4. Systém uloží zprávu.

Alternativní tok 1

1.1 Pokud je uživatel registrovaný, neprochází kontrolou pomocí opsání kontrolního obrázku (catptcha).

Alternativní tok 2

2.1 Pokud uživatel zadal neplatný vstup nebo vstupy, systém na skutečnost uživatele upozorní a nedovolí zprávu odeslat.
2.2 Uživatel opraví neplatný vstup/vstupy a tok pokračuje na 2. bodu základního toku

Podmínky pro dokončení: Nový komentář bude korektně uložen v databázi.