MASARYKOVA UNIVERZITA

PV213 Enterprise Information Systems in Practice

01 – Introduction, UML overview

MASARYKOVA UNIVERZITA

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE

esf

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE esf

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost 2007–13

UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Goal of the course

- **Introduce you into real requirements in praxis**

- You should be able to start to work in the enterprise and understand:
  - Different roles (positions) in the enterprise
  - What is expected to know on given position
  - Typical problems you can meet
- Get some feedback from the real world
- Practically oriented (but some “theory” is needed)

Conditions to pass the course

- To pass the course you must do the homework assignment on given theme
  - At least three pages of text, pictures, tables, …
  - Areas of homework will be e.g. from
    - Management of projects
    - Analysis and architecture
    - Testing
- At the latest in the half of the course (end of March 2015) homework assignment will be known
- Deadline for submission is end of May 2015
- You will be informed via mail in the MUNI IS

Further instructions

- If you have any questions please ask
  - Short answers will be answered immediately
  - Longer answers will be answered on the end of each lecture
- On the end of each lecture we can discuss presented theme
- Slides will be available after each lecture

Who we are I?

Ing. Libor Švehlák

- Six lessons in the course
- 15 years of professional experience in IT
- Area of interest: Software architecture and development
- Oracle Certified Master Java EE 5 Enterprise Architect
- Atos IT Solutions and Services
  - International company with 86 000 employees
  - Headquarter in Bezons, France
  - Official IT partner for Olympic Games since 2001
  - Around 400 employees in the Czech Republic

Who we are II?

Ing. et Ing. Aleš Macek

- Six lessons in the course
- 9 years of experience in field of IT
- Area of interest: Project and quality management
- Prince2 Practitioner, Scrum Master
- Honeywell
  - Multinational conglomerate with more than 130 000 employees
  - Headquarters in Morristown, New Jersey, U.S.
  - 4000 employees in Czech Republic
  - Research and development centre in Brno

Content of the course I

01 – Introduction, UML overview
Libor Švehlák, 17.2.2015

02 – Architecture in the standard environment
Libor Švehlák, 24.2.2015

03 – Bid preparation, risk management, team establishment
Aleš Macek, 3.3.2015

04 – Project management, estimations
Aleš Macek, 10.3.2015

Content of the course II

05 – Quality assurance

Aleš Macek, 17.3.2015

06 – Development process

Aleš Macek, 24.3.2015

07 – Cloud computing I

Libor Švehlák, 31.3.2015

08 – Cloud II, Integration of EIS with other systems

Libor Švehlák, 31.3.2015

Content of the course III

09 – Security, Configuration management and tools
Libor Švehlák, 7.4.2015

10 – Deployment, migration, maintenance phase
Libor Švehlák, 7.4.2015

11 – Testing
Aleš Macek, 14.4.2015

12 – Closing, retrospective, final game
Aleš Macek, 21.4.2015

Definition of terms

Enterprise Information System

- Enterprise is a synonym for company, firm, business organization
- Information system is any application or service that support “main” (or supplementary) business of the company
- Keep in mind that information system is there to support business and not vice versa
- It must add some additional value e.g.:
  - Support decision making
  - Increase enterprise efficiency
  - Decrease costs
- This is often in contrast with end customer products – end customers sometimes buy products irrationally

History

- First systems back in 1960s on mainframes in banks, telecommunication companies
- 1970s Personal computers
- 1980s Client/server applications
- 1990s Three and multitier architecture applications
- 2000s Web applications
- 2010s Cloud applications, EIS as a service
  - Everything …as a service

Types of enterprise information systems

ERP – Enterprise Resource Planning

CRM – Customer Relationship Management

SCM – Supply Chain Management

MIS – Management Information System

CMS – Content Management System

KMS – Knowledge Management System

DSS – Decision Support System

GIS – Geographic Information System

And a lot more…

Example – Reservation system - Motivation

In the company there exist some small equipments (mobile phones, tablets, beamers, flip charts, etc.) which can be used by anyone (pool equipment). Till now reservation of these equipments was done on the paper that is maintained by the secretary. This somehow works but causes several problems. Everyone who wants to make a reservation must visit the secretary and check whether equipment is available at given time. This causes inefficient work time and decreased well-being for employees as well. Another problem is that on the end of the month secretary has to rewrite data about the usage of equipments to the reporting system used by management. These data are then used for analysis whether it is required to buy additional items of given equipment.

Example – Reservation system – Brief requirements

It is required to create internal tool for reservation of equipments. Definitions of these equipments (name, quantity, description) is stored in external inventory system (SAP) and they are available via web services. Tool must allow to make a reservation of the equipment between specified dates (and times). Only the author of the reservation or special user (e.g. secretary) can cancel the reservation. To better indicate real reservation requests it must be possible to mark reservation request even when some equipment is already reserved. Tool must support calendar view (which equipment is reserved for today, tomorrow, etc.) and equipment view (for given equipment show when this equipment is/was reserved). Additionally tool must support export of usage data in given period to the management reporting system.

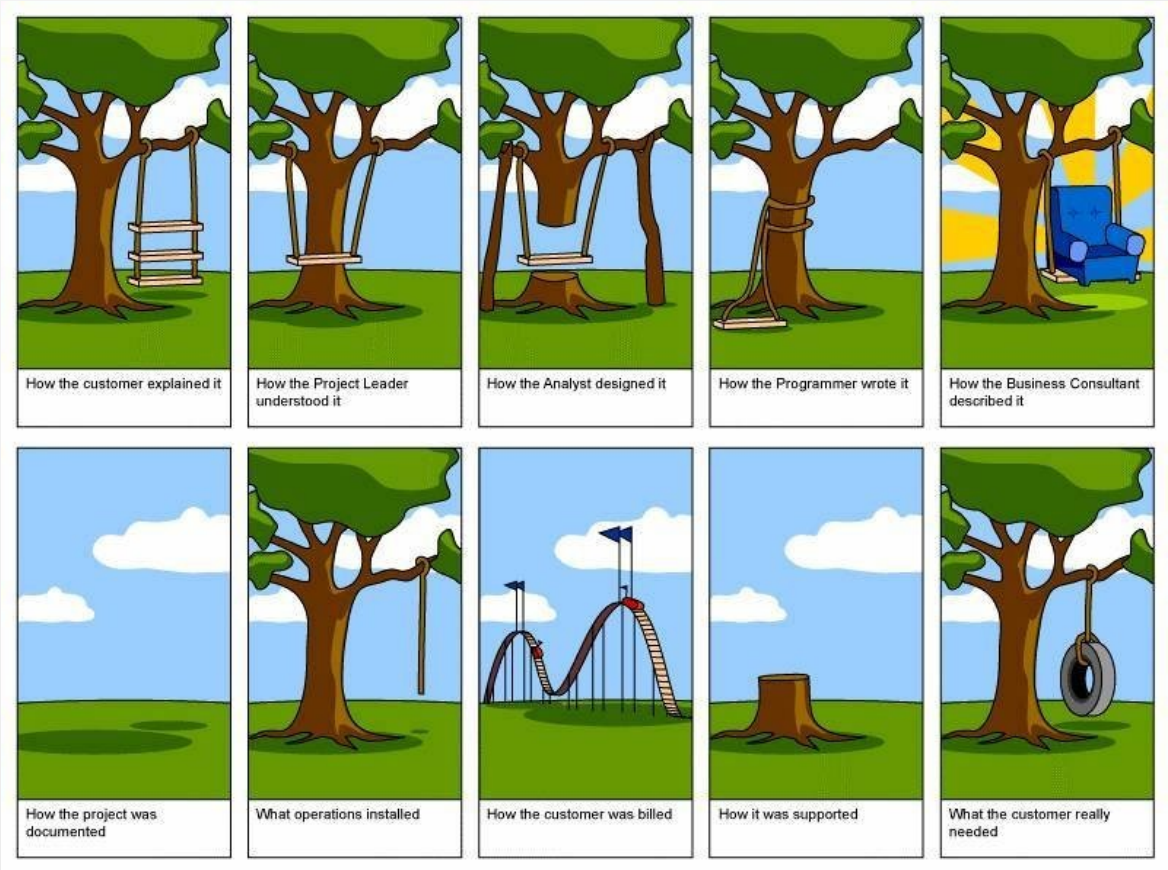
Interesting points to requirements I

- First requirements are often vague, incomplete and sometimes even in opposition. First requirements must be further discussed, analyzed and clarified to create final version of requirements
- Technical details should not be mentioned. If they are present check if they are really required (e.g. interfaces to other systems) or they represent just inappropriate abstraction (author of requirements things implementation should be done this way)
- From requirements must be clear what system must do and also what it must not do (clear responsibilities and borders between systems)
- For some projects there is a special role: Requirement engineer

Interesting points to requirements II

- Customers sometimes doesn't know what they really want. Also requirement specification needs some time and thinking to create a good requirements
- Be sure both sides (customer and your team) understand the problem the same way. Check it!
- At this phase several meetings with customer is the most efficient way. Don't expect everything will be clear within one session with the customer
- Some customers expects that you know everything and their participation is not needed. Explain them that result will be better when they can be part of the whole process
- Don't undervalue requirements. They are basics for further work!

How sometimes projects look like…



Motivation to use Unified Modeling Language (UML)

We have requirements

- How to communicate requirements to the team?
- How to efficiently pass the information between team members?

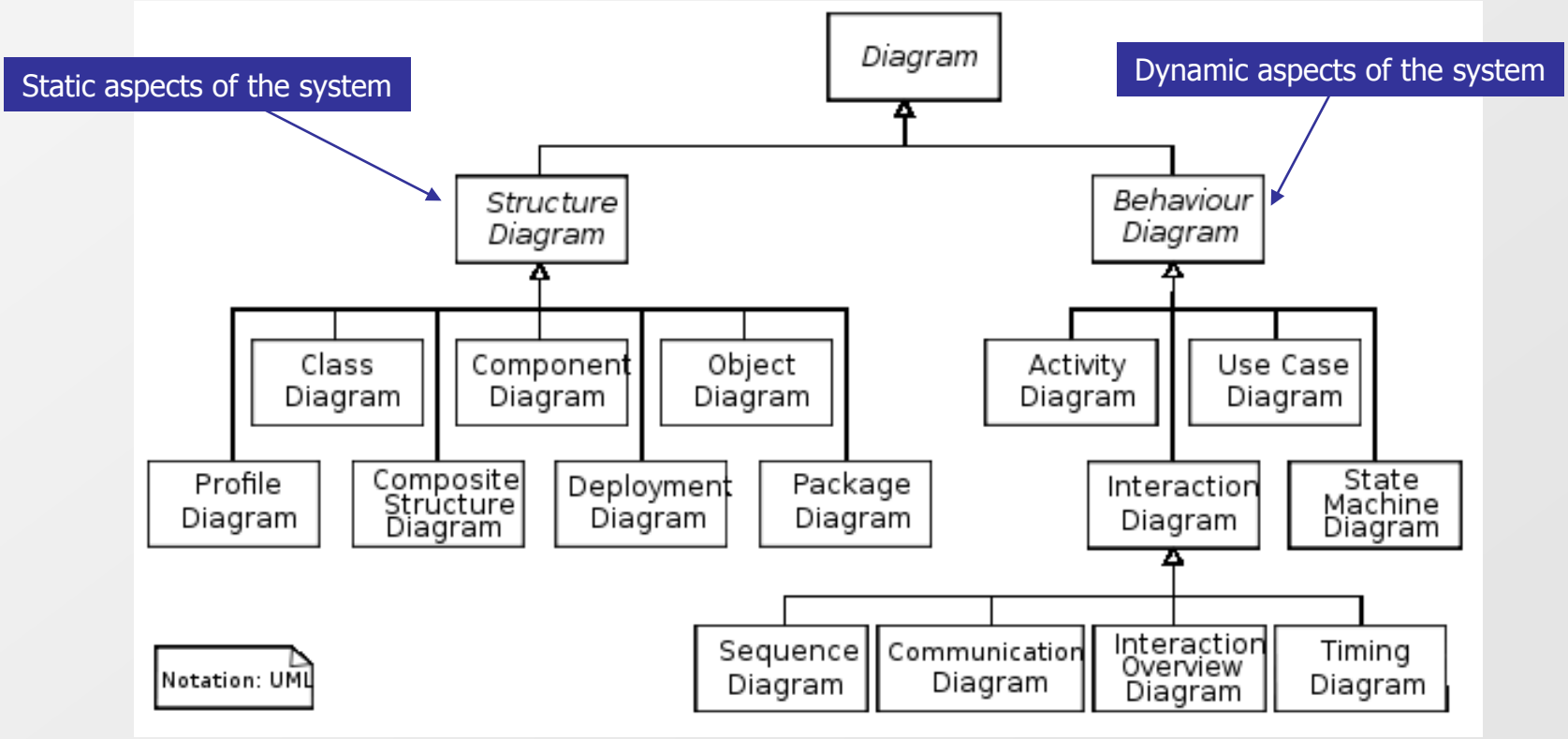
Some standardized language is required

- Answer is Unified Modeling Language (UML)

UML

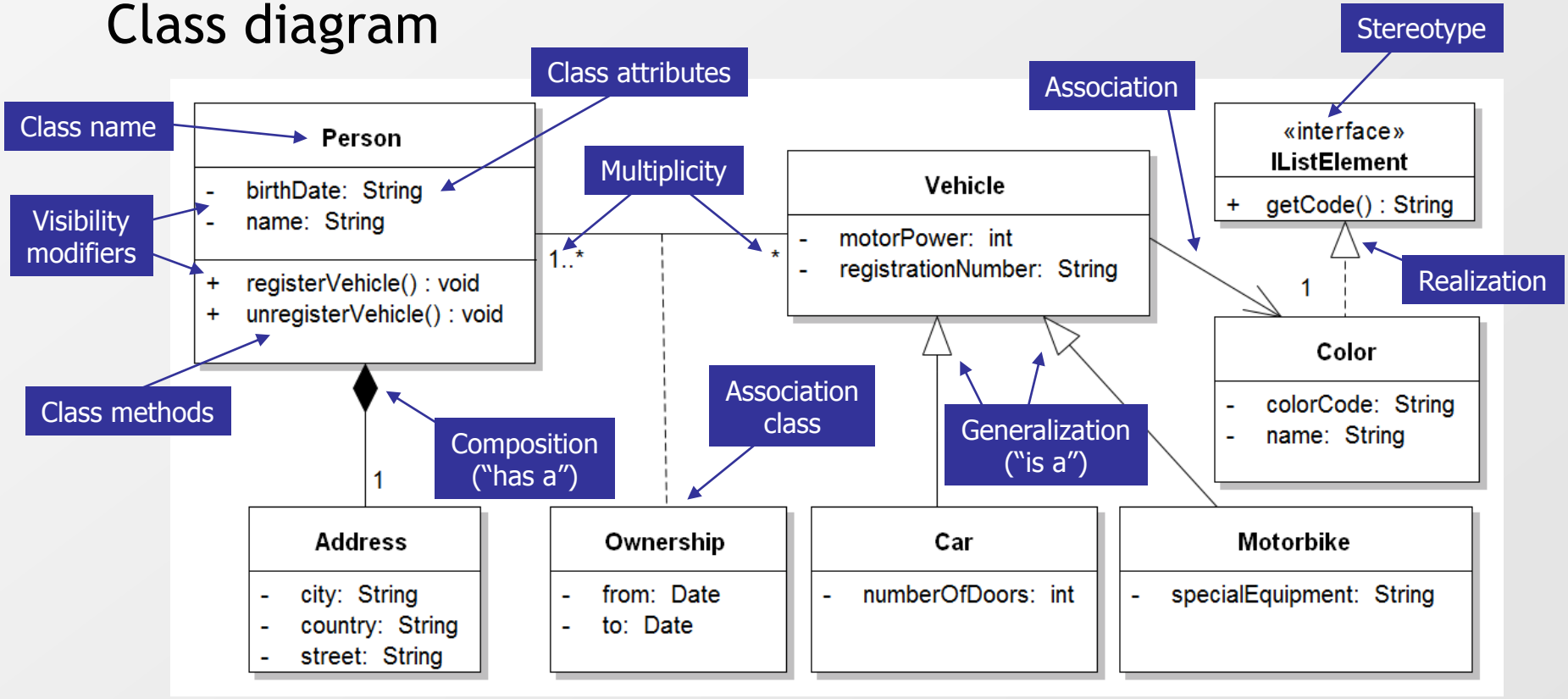
- UML has started in mid of 1990s, standardized in 1997
- UML is graphically oriented (one picture is sometimes more than 1000 of words) but allows to add to pictures notes
- UML allows to express facts in unambiguous form
- UML has several versions. Current released version is 2.4.1 (Aug 2011), version 2.5 is “in process”. Differences are not so important for us
- For different members of the team are important different types of UML diagrams
- UML diagrams alone are not enough. Text descriptions are still needed!

UML diagrams overview



Types of UML diagrams - Structural diagrams I

Class diagram

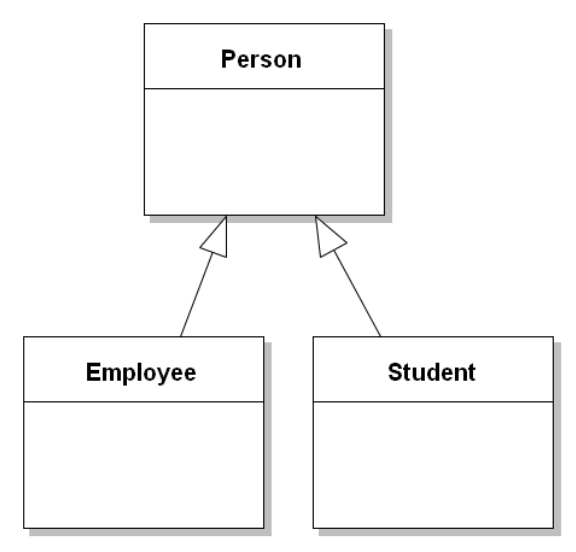


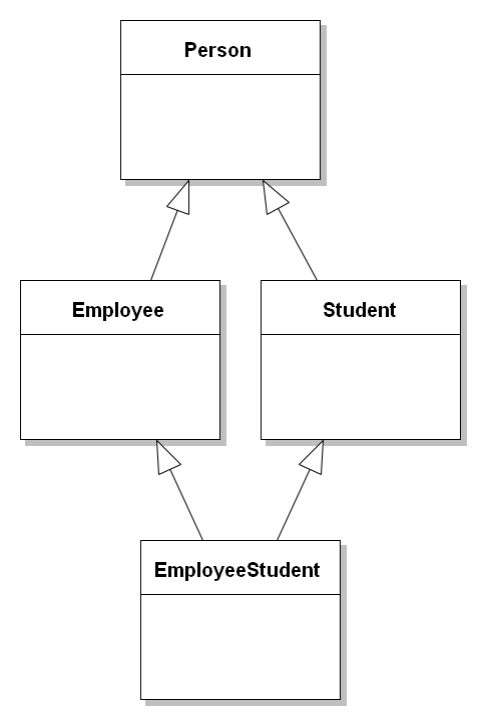
Types of UML diagrams - Structural diagrams II

- Find entities (classes) and relations between entities
  - Find “type” of the relation
    - Simple association
    - Whole / part – Composition (aggregation)
    - Parent / child – Generalization / specialization
  - Check multiplicities – mostly they totally influence how system will behave (totally different behavior)
  - In the first step concentrate on classes and their properties – methods are not so important (but if it helps you mark them)
- Be careful with generalization / specialization
  - It is often misused
  - Simple association or composition is often better

Types of UML diagrams - Structural diagrams III

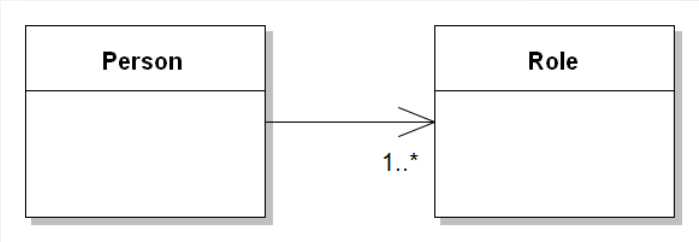
Generalization/specialization – what can be wrong?



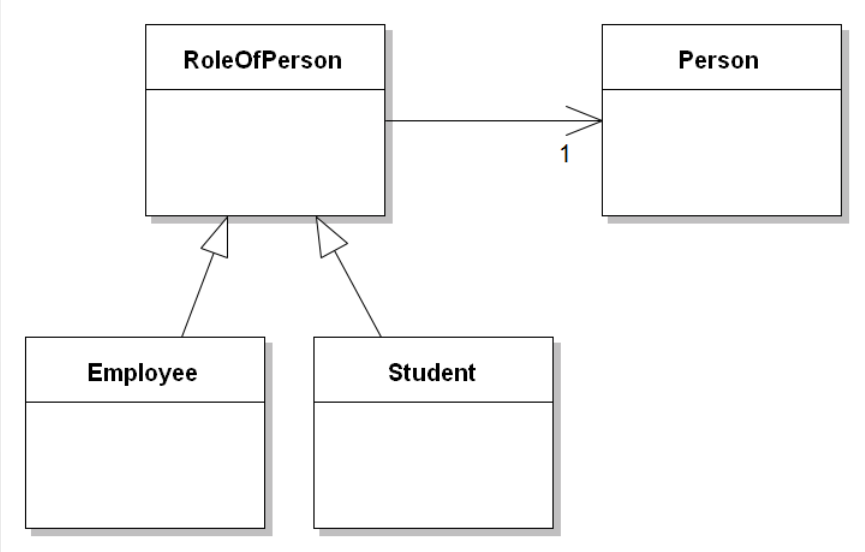


Types of UML diagrams - Structural diagrams IV

Generalization/specialization fix – possible solutions



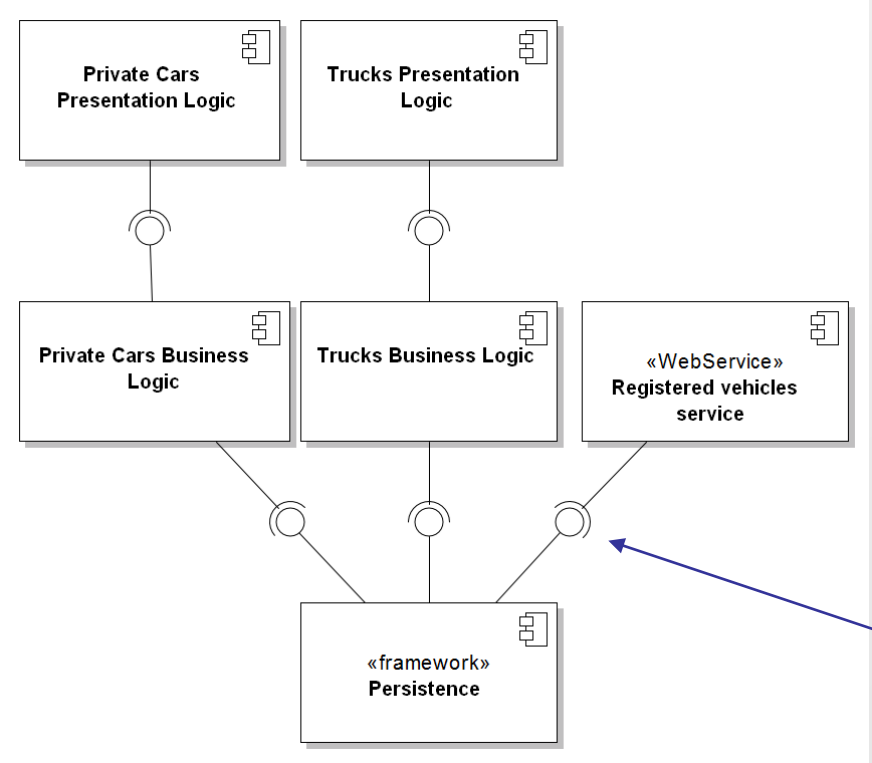
Suitable in a case when we need to only know different roles



Suitable in a case when we need to store for different roles also different attributes

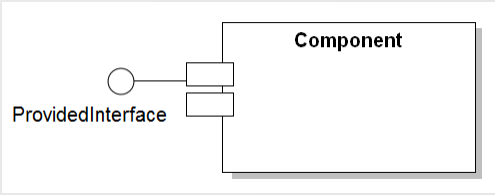
Types of UML diagrams - Structural diagrams V

Component diagram



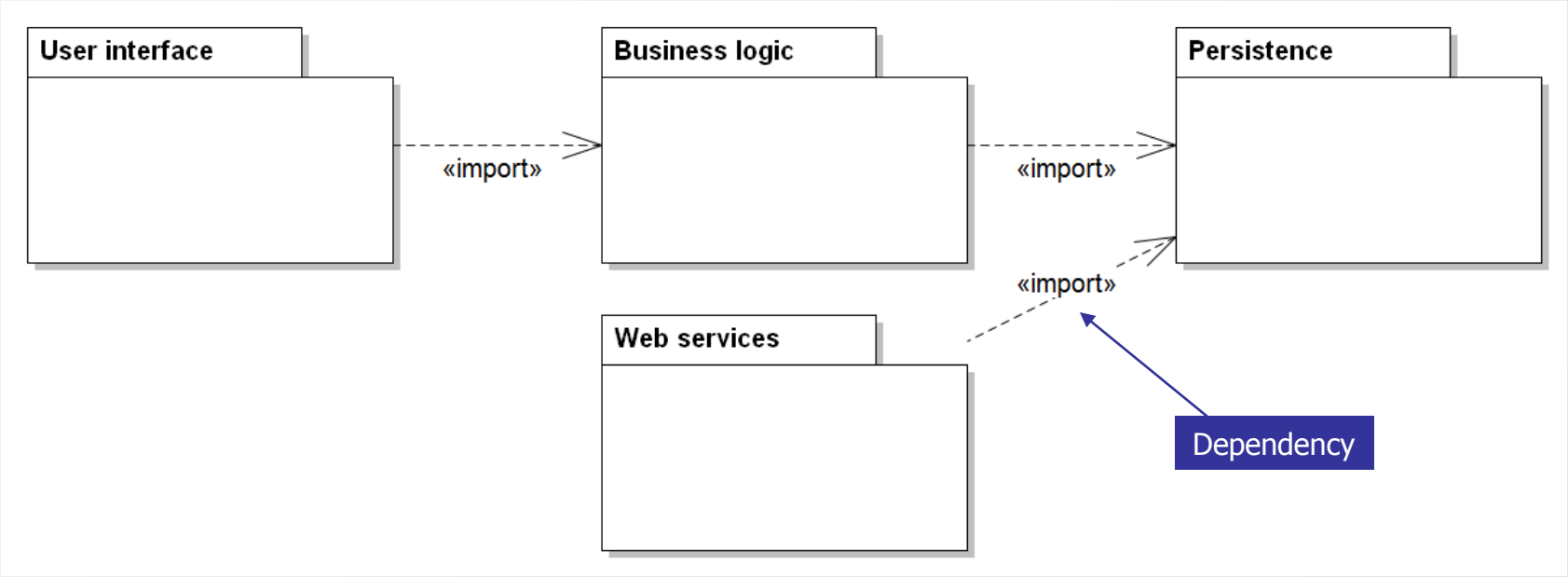
Connector between components

Different notation in UML 1.x



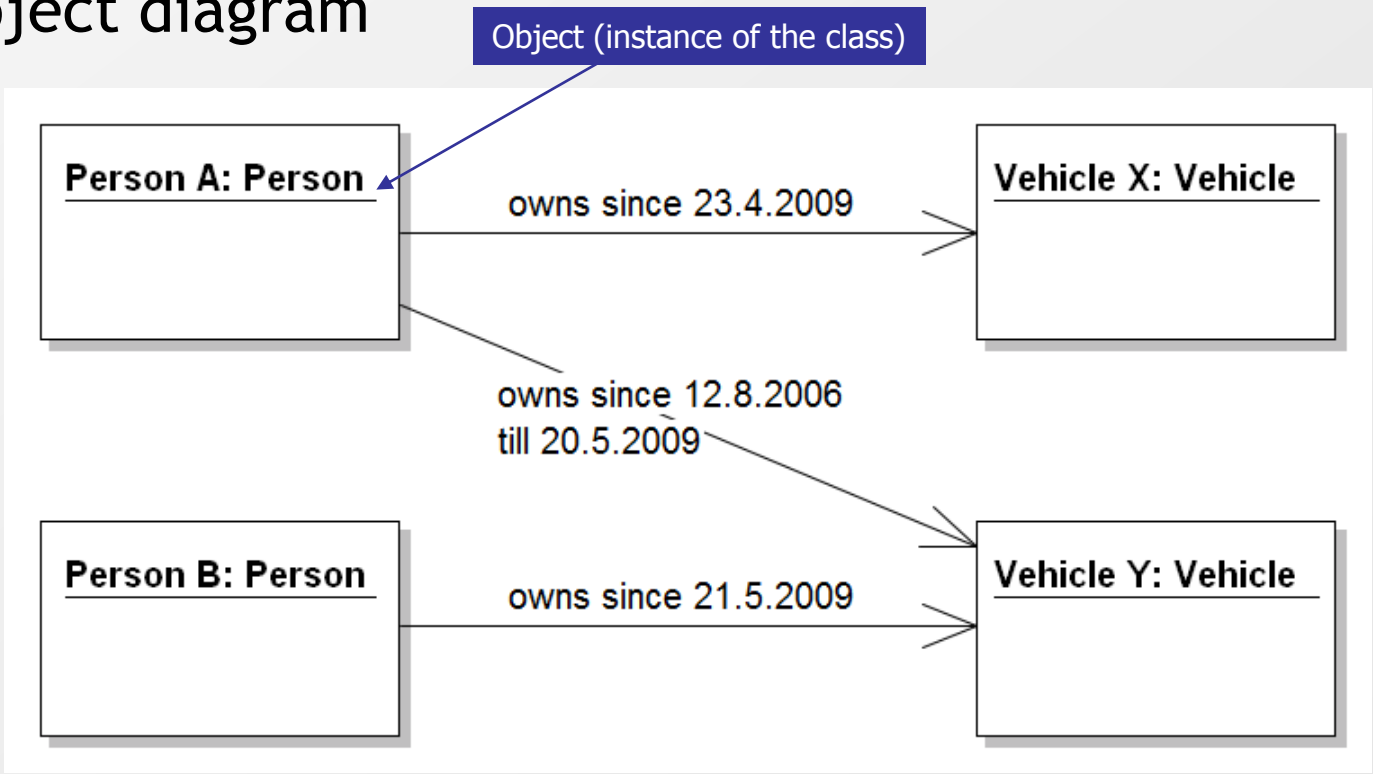
Types of UML diagrams - Structural diagrams VI

Package diagram



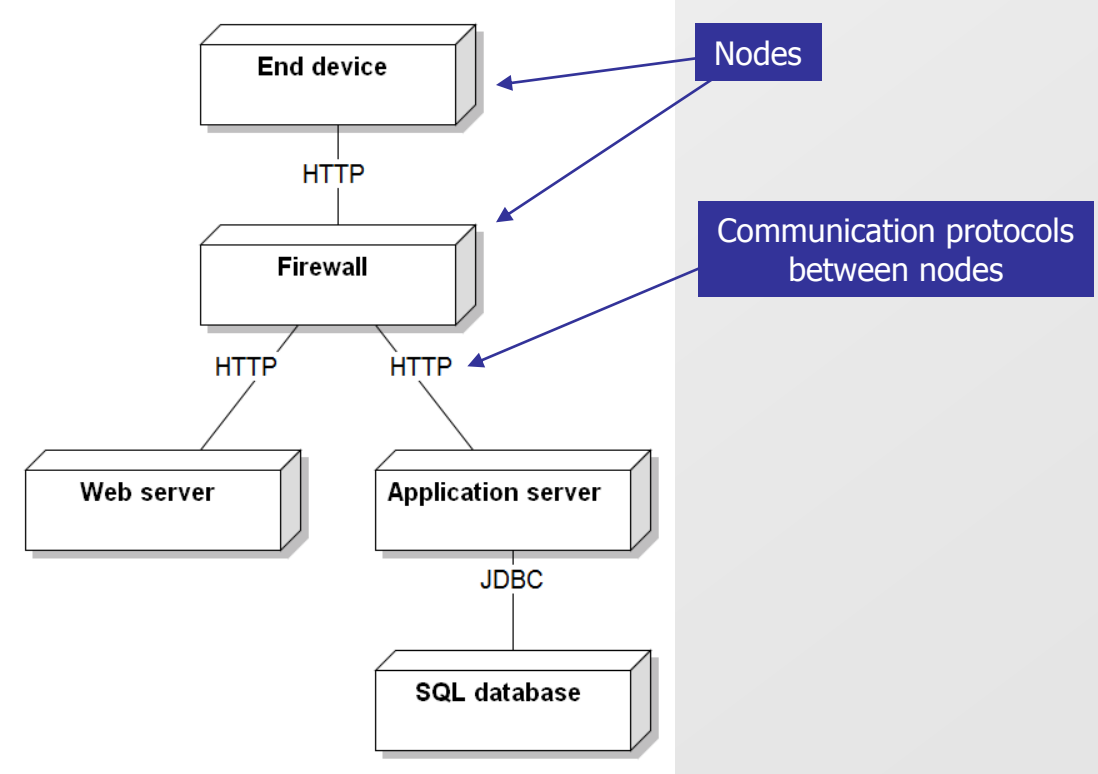
Types of UML diagrams - Structural diagrams VII

Object diagram



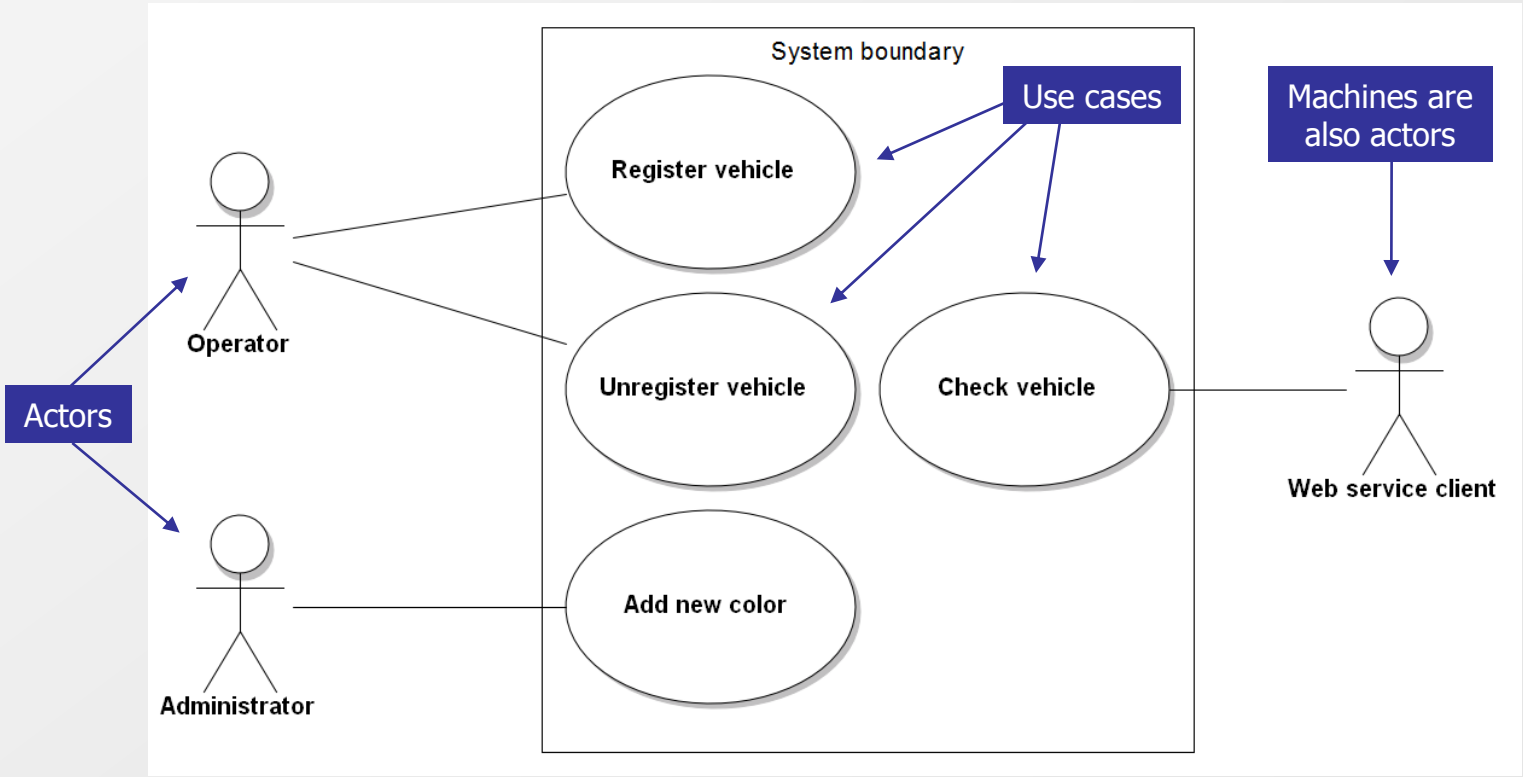
Types of UML diagrams - Structural diagrams VIII

Deployment diagram



Types of UML diagrams - Behavioral diagrams I

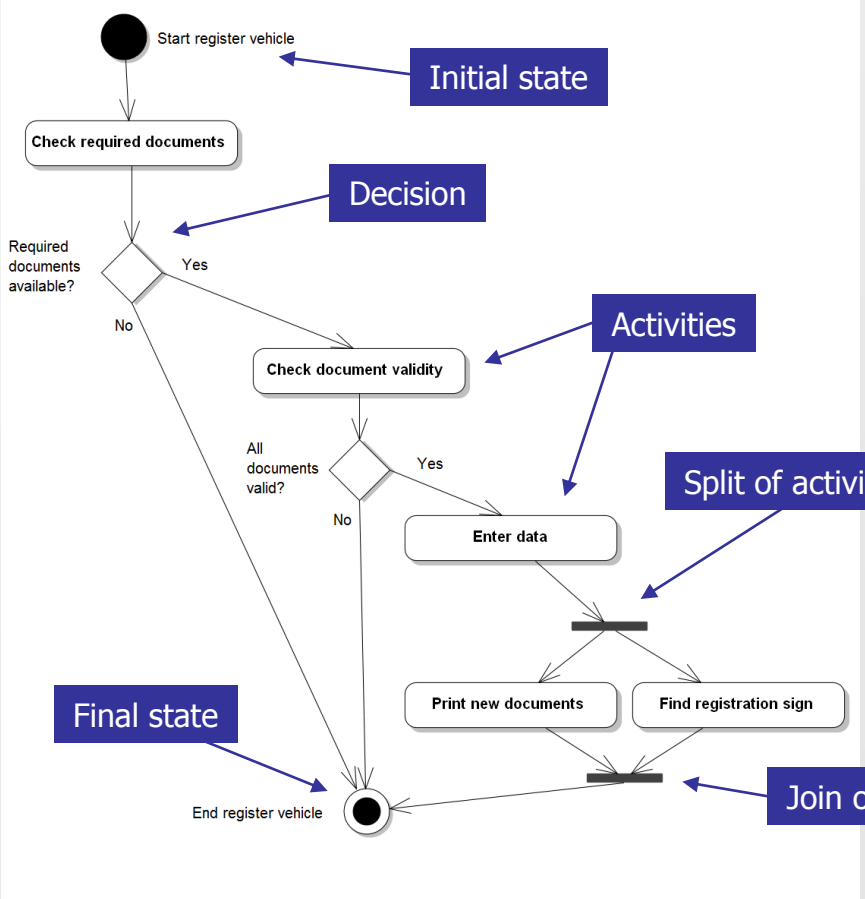
Use case diagram



Types of UML diagrams - Behavioral diagrams II

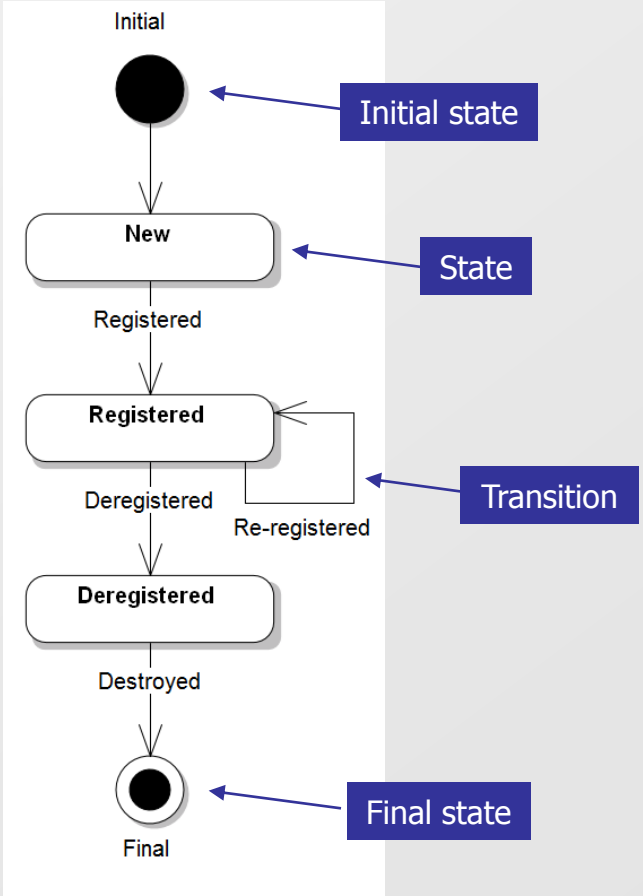
Activity diagram

Process oriented



Types of UML diagrams - Behavioral diagrams III

State Machine Diagram

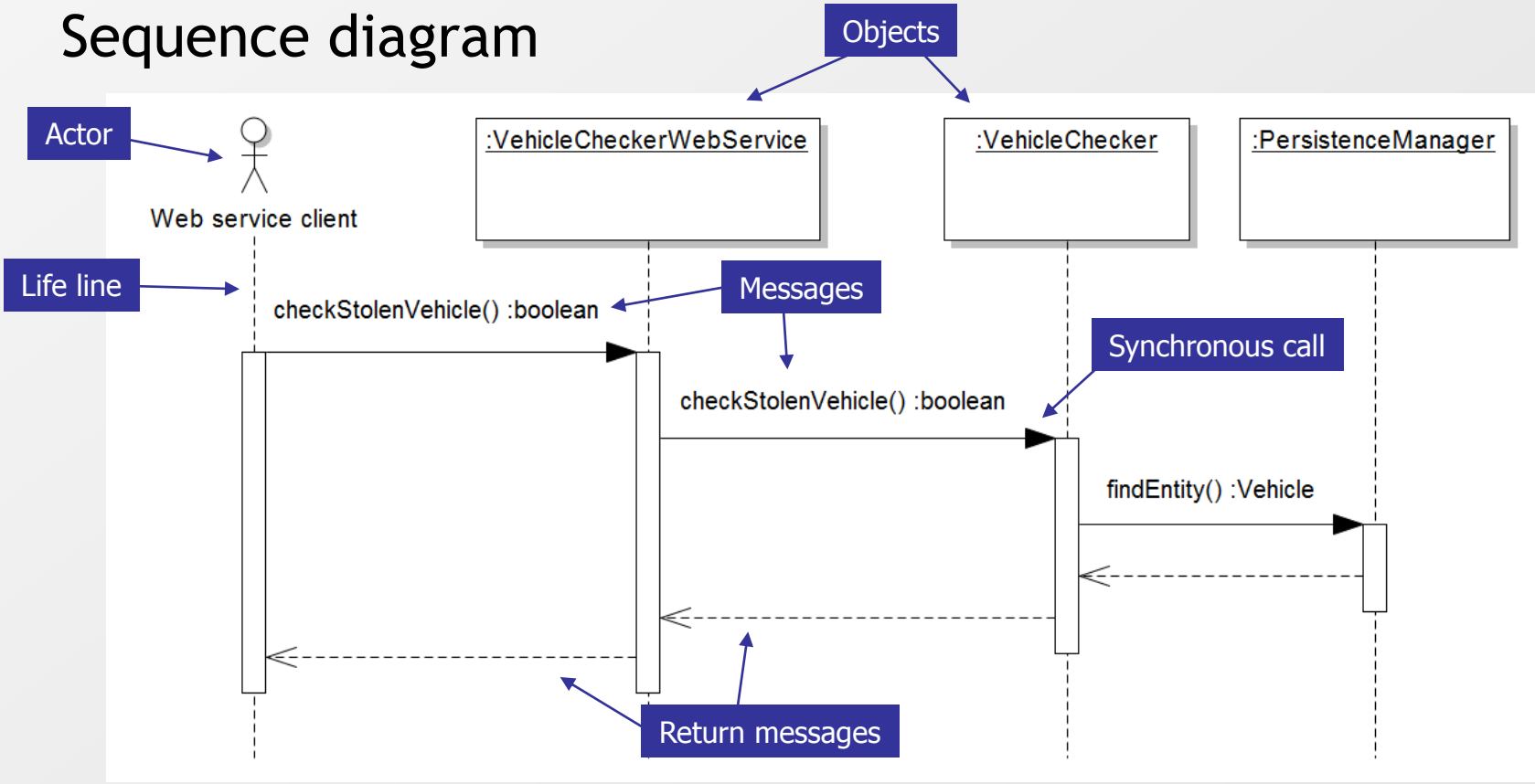


State oriented

Named “State Diagram” in UML 1.x

Types of UML diagrams - Behavioral diagrams IV

Sequence diagram



How to start with transformation from text into diagrams?

How to convert text description into diagrams?

Find **nouns, adjectives** and **verbs** in the text

- Nouns can represent entities
- Adjectives can represent properties of entities
- Verbs can represent actions, processes or relations

Such lists are only preliminary:

- Some items can be redundant
- Some items can be missing
- Sometimes you need to use different wordings (e.g. to eliminate ambiguity)
- Most probably you need to generalize some items

Which diagrams to create in the first step?

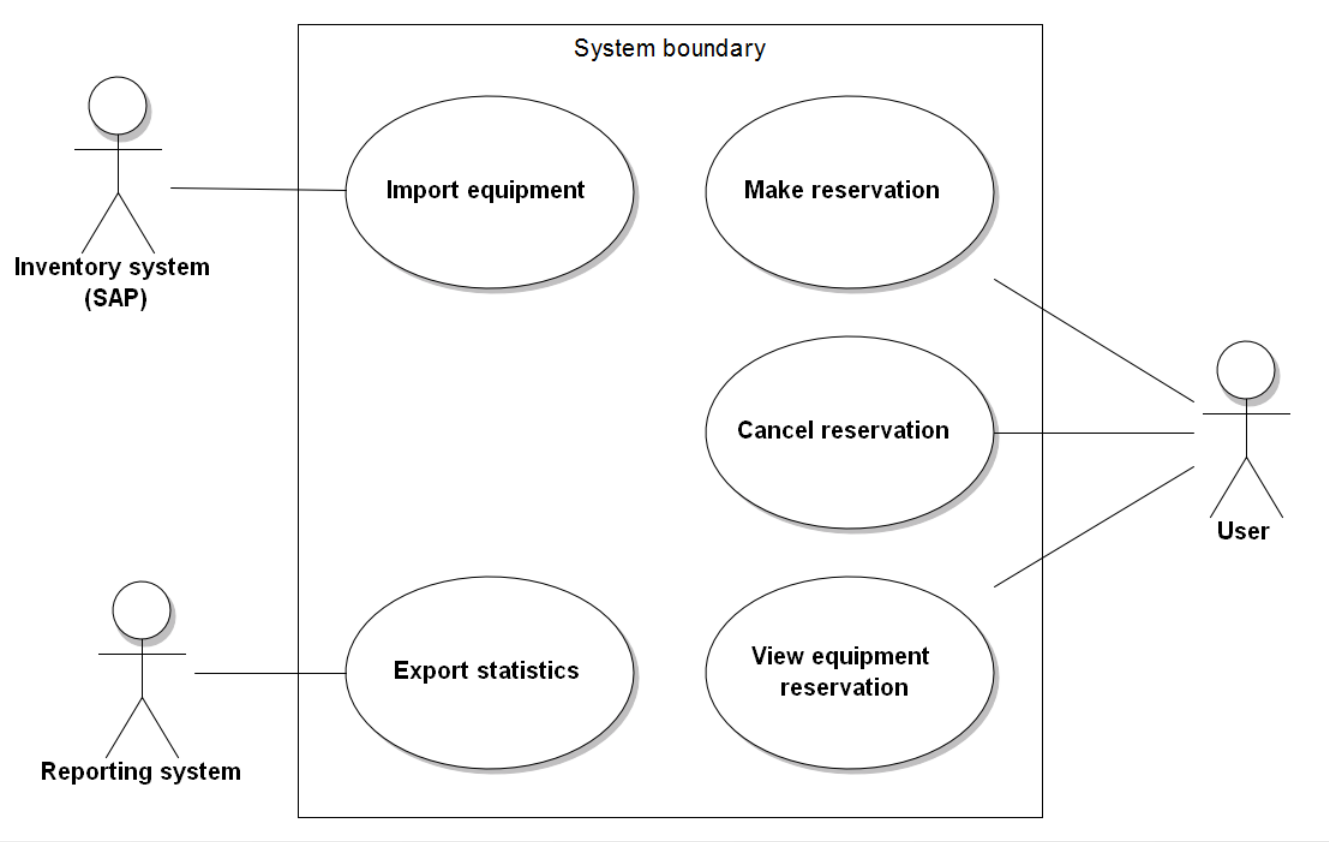
Use case diagram

- Identify main use cases
- Find all actors
- Find boundaries of the system

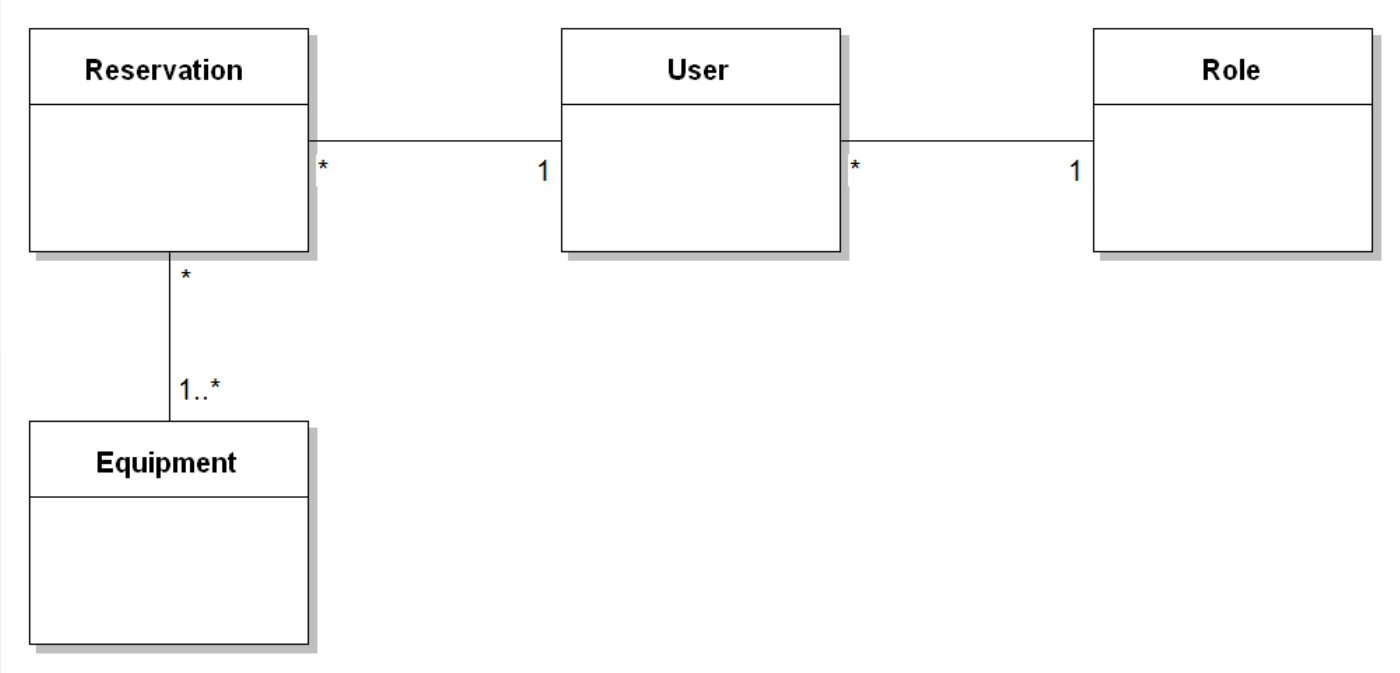
Business Domain Model (BDM)

- Special type of class diagram
- Represents “topmost view“ on the system
- Contains only most important entities representing “core” of the solution
- Important relationships between entities must be covered
  - Identify multiplicities
  - Not needed in this step to specify exact relationships like generalization / specialization or association / composition

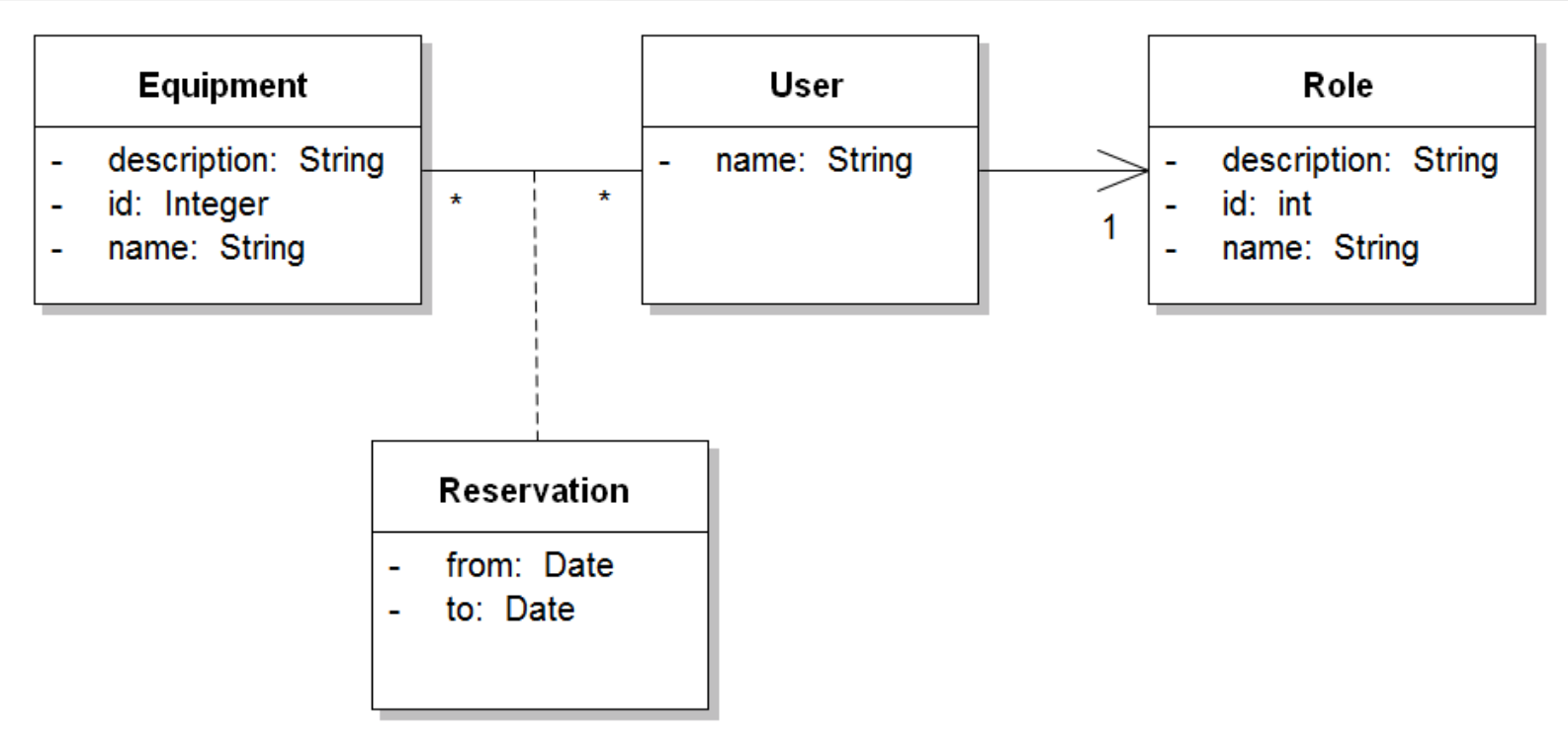
Example - Use Case Diagram for Reservation system



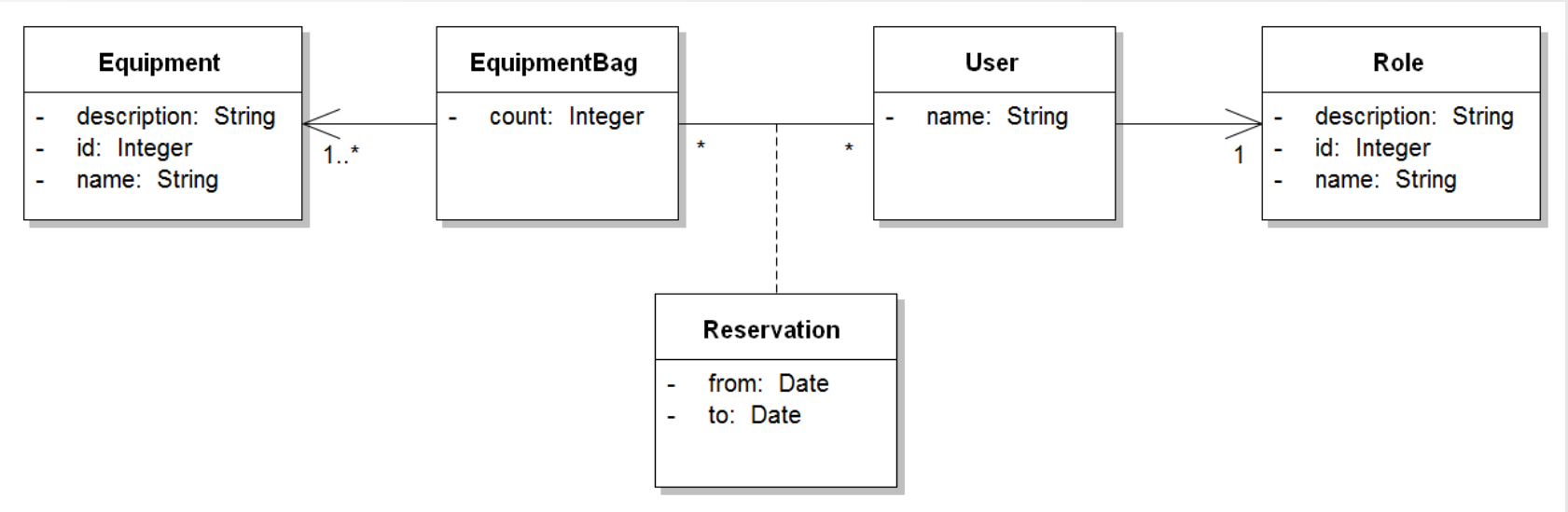
Example – Business Domain Model for Reservation system



Example – Class Model for Reservation system I



Example - Class Model for Reservation system II



Further tips and tricks I

- It is needed to clearly define boundaries (borders) of the system. They will define for what system is responsible. Actors are not part of the system!
- Check that all main use cases are covered. Use cases on top level represents bigger amount of work than sub use cases. If you forget something on top level it can have catastrophic consequences!
- Try to identify missing use cases. Maybe they are missing in description but needed for full functionality of the system.
- Split large use case models into sub use cases or more use case diagrams.
- Check that system will solve problem of customer. The worst situation is when system is implemented but it doesn’t help customer.

Further tips and tricks II

- Think ahead but don't jump into inappropriate level of abstraction (avoid "sidestep to meta-level")
- Start with first version and enhance it in the next iteration after feedback/better know how. Don't expect final results after the first step. Use iterative approach.
- If it is helpful create also other types of diagrams
  - Object diagram to better understand relationships between objects
  - Activity diagram to better understand processes
  - Sequence diagram to better understand use cases when interactions are important

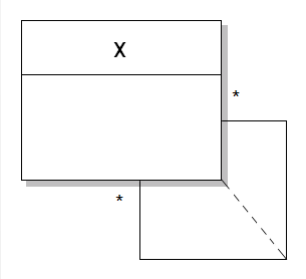
Transformation requires some experience. Don't be frustrated that first results are not ideal!

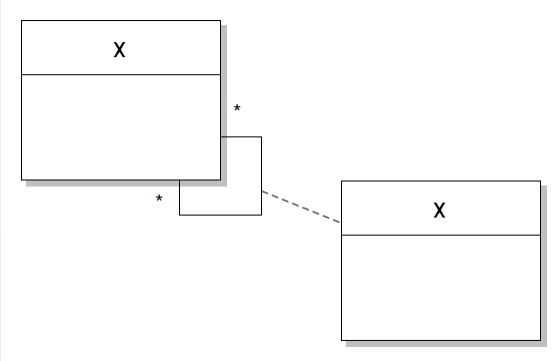
UML and team roles

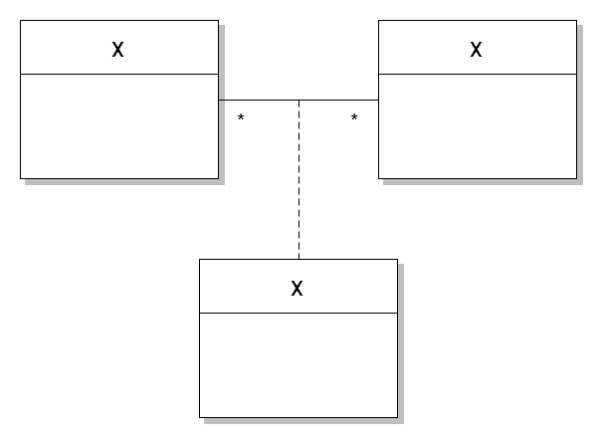
- **Project manager**
  - You should be able to read use case and deployment diagrams
- **Analyst**
  - Create use case, business domain model and activity diagrams
- **Architect**
  - Create deployment, package, component and class diagrams
- **Developer**
  - Create class, sequence, state machine diagrams
- **Tester**
  - Read use case, deployment, activity diagrams

Bonus question

- What can represent following class diagram?







Family (Genealogical) Tree

Děkuji za pozornost.

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE esf

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost 2007-13

UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ