MUNI
FI

lasaris LAB OF SOFTWARE ARCHITECTURES AND INFORMATION SYSTEMS

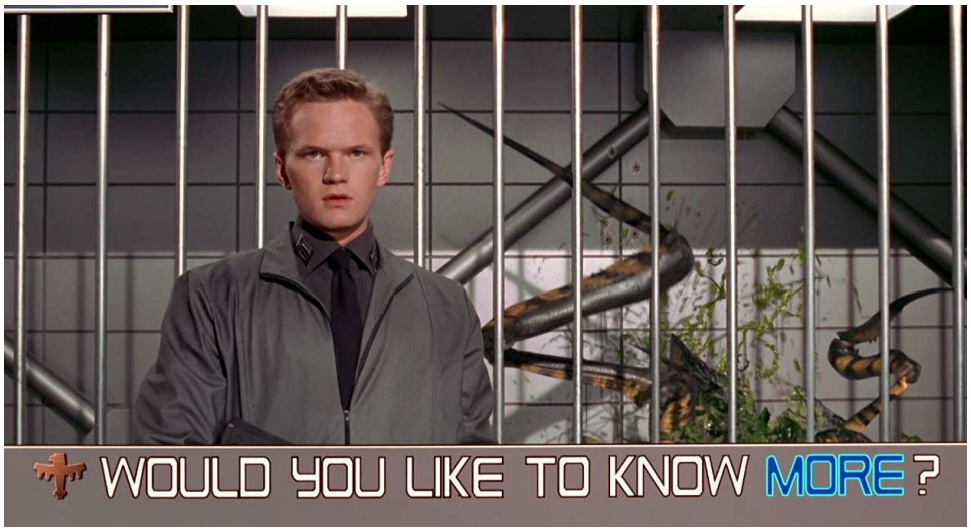
Use Case Diagram II, Textual Specifications

PB007 Software Engineering I

Lukáš Daubner
daubner@mail.muni.cz

Use Case Diagram – There is more…

- Diagram is more expressive
  - Inheritance (generalization) of actors
  - Inheritance (generalization) of use cases
  - “Shared” use case – Include
  - “Modular” use case – Extend

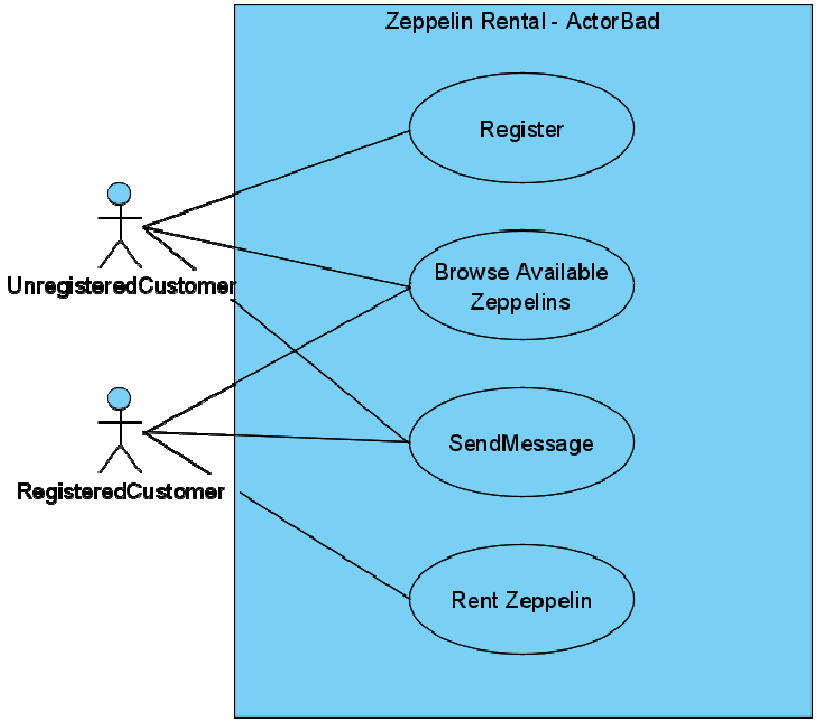


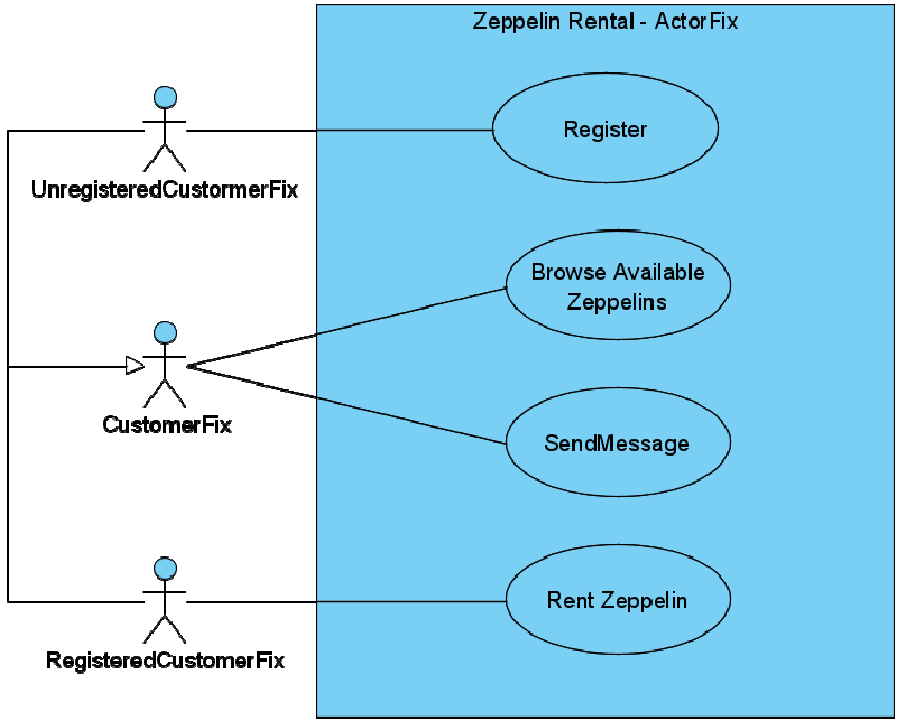
MUNI FI

Actors Generalization

- Relationship between the more general and specialized actors
- For simplifying the diagram
- General actors are often abstract, i.e., not a real role
- A descending actor inherits **all** roles and parent link. Every time it is expected the use of a parent actor, we can use one of the descendants

Actors Generalization – Example



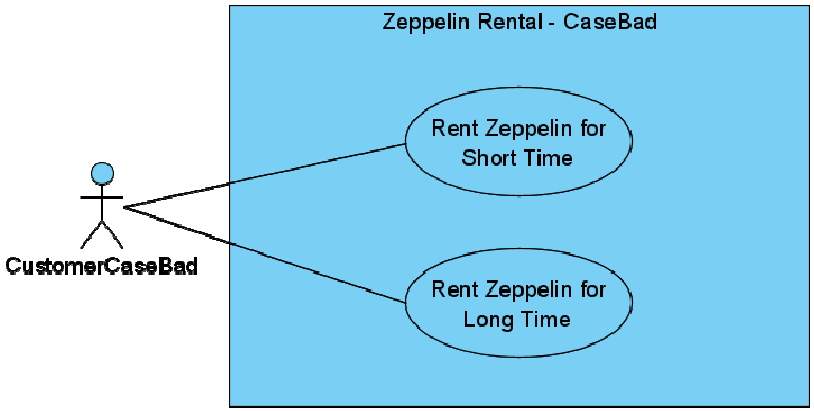


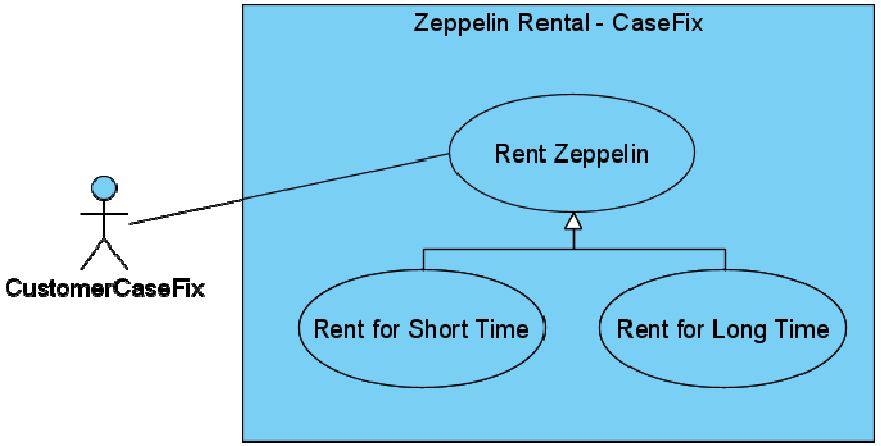
MUNI FI

Use Case Generalization

- relationship between the general and specialized use cases
- A specialized use cases inherits the properties, and add new features by overloading (changing) the inherited properties
  - It cannot overload the parents' extension points
- The textual specification should reflect those changes
- Parental use cases can be abstract (recommended)
  - Either no specification or incomplete specification of the flow of events

Use Case Generalization – Example



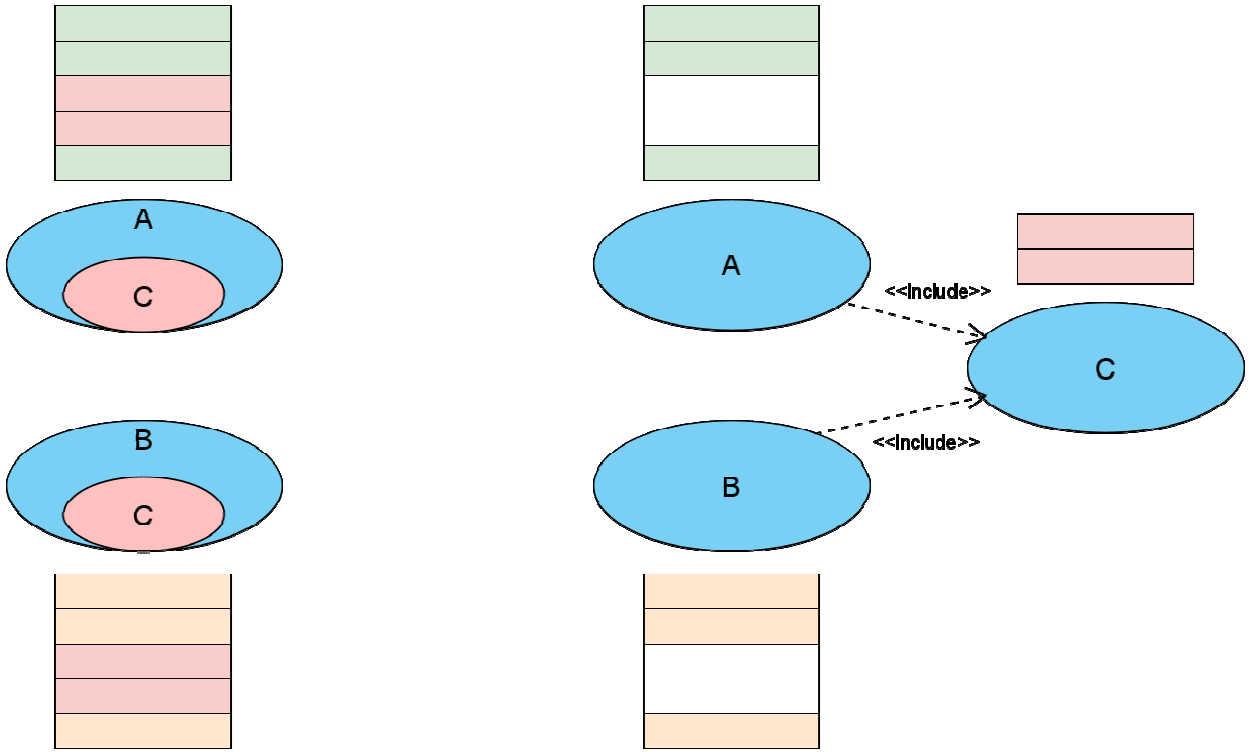


MUNI FI

Include

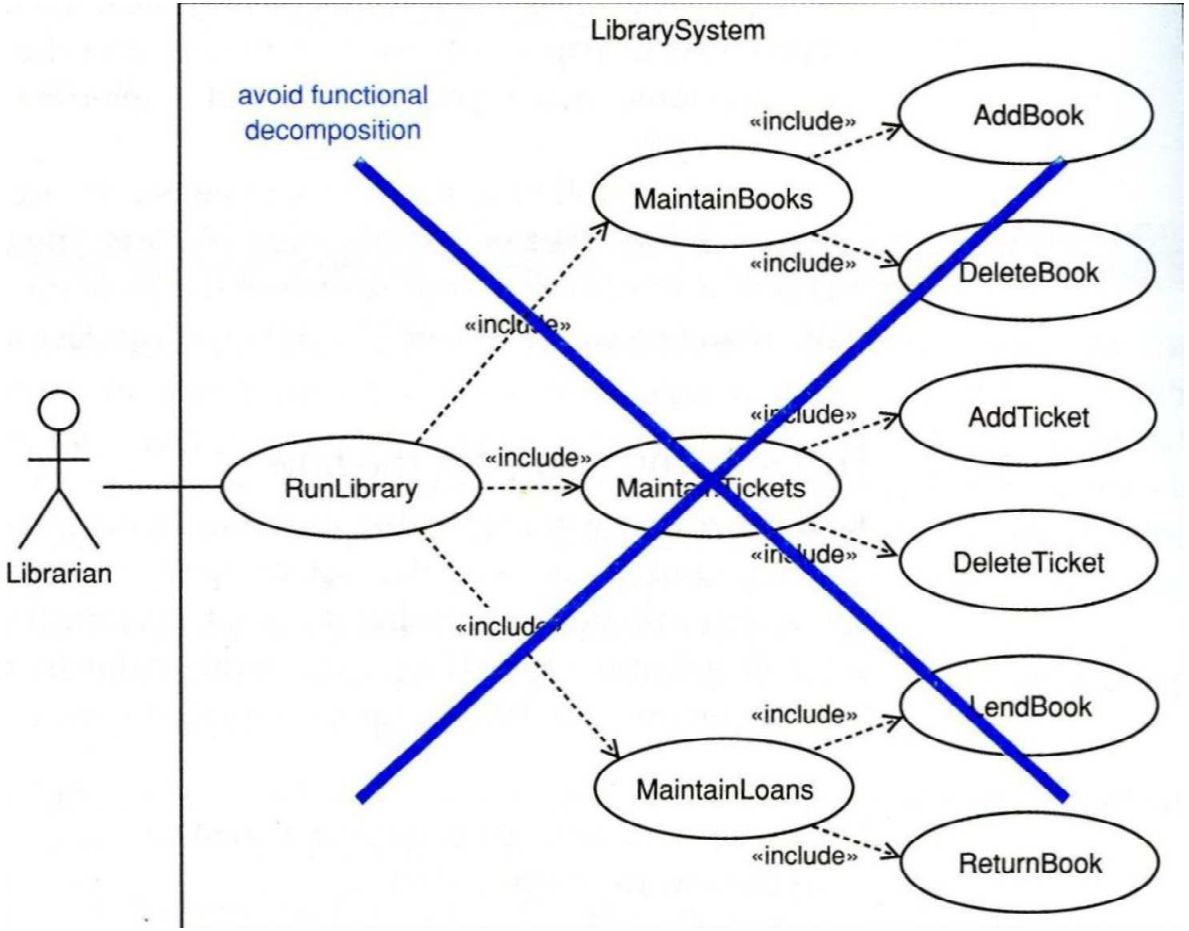
- Specialized type of relationship
- Allows you to allocate repetitive steps in several use cases in separate use case
- The basic use case is incomplete without all the embedded use cases

Include



MUNI FI

Include

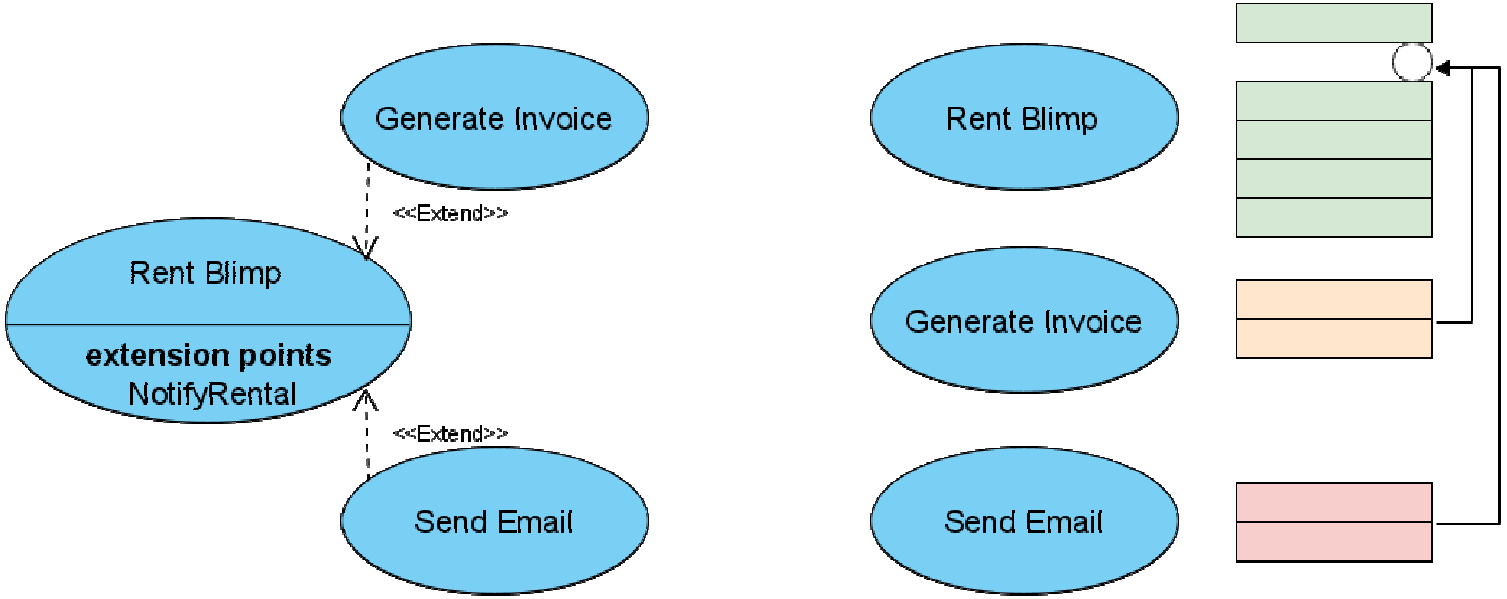


MUNI FI

Extend

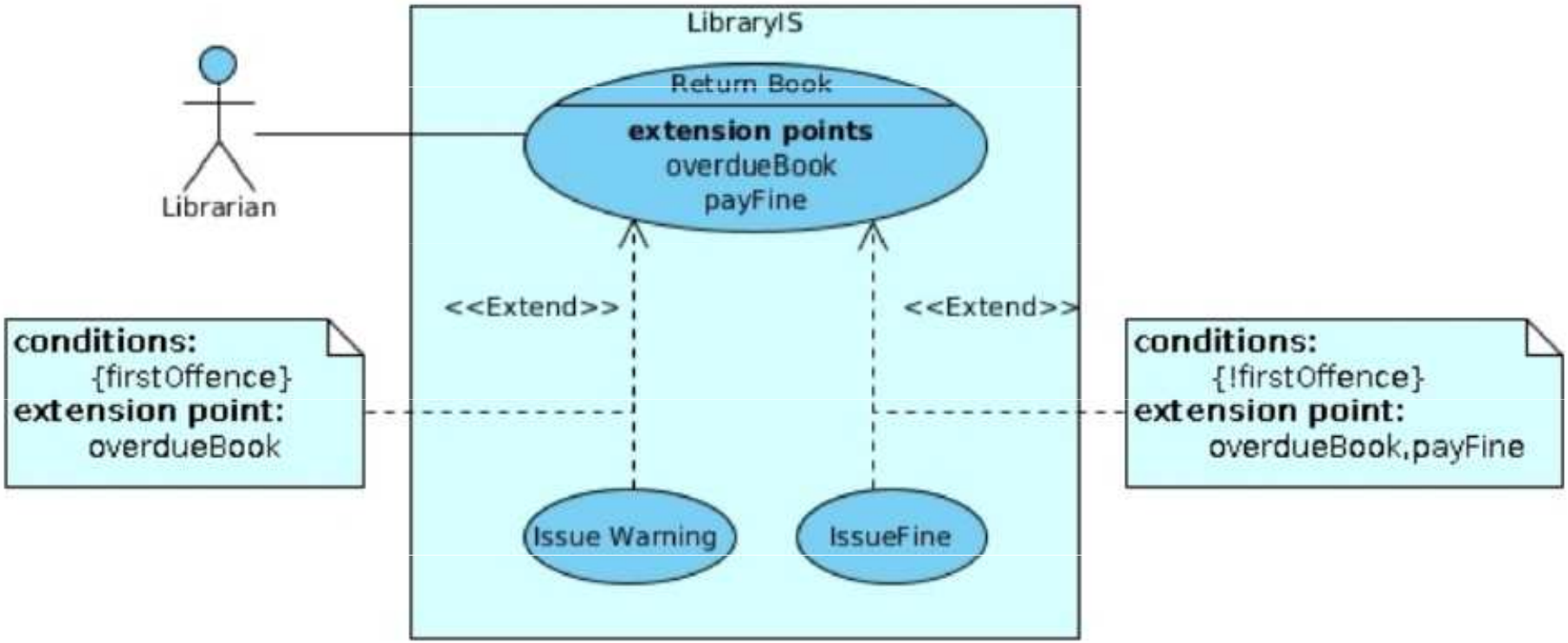
- Specialized type of relationship
- Allows insertion of new behaviour into base use case
  - The extension is transparent for the use case – the base do not know about extensions
- It is inserted at specifically defined points (extension points)
- The extending use cases hooks on the extension points
  - It is possible to define conditions or hook multiple use cases on one extension point

Extend



MUNI FI

Extend



MUNI FI

Textual Specification of Use Cases

- Detailed and structured description of a use case

- It should include:
  - Name and ID
  - Brief description
  - Primary and secondary actors
  - Input and output conditions
  - Main flow of events
  - Alternative flow of events

Flow of Events – Main Flow

- Sequence of steps of interaction with the system in the ideal case
  - (free from errors, interruptions, …)
- Written as: <step id> <actor/system> <action>
- Begins with some action from primary actor
  - Recommended form is: 1. The use case begins when <actor> <action>
- You can use keywords IF, FOR (FOREACH), WHILE

Flow of Events – Example

- 1. Use case starts when Customer opens rental form
- 2. INCLUDE(Find Available Zeppelins)
- 3. System displays zeppelins available for rental
- 4. Customer chooses zeppelin to rent
- 5. WHILE form is not valid
- 5.1 Customer fills personal information
- 6. Customer submits the form
- EXTENSION POINT(NotifyRental)
- 7. System saves rental request

Flow of Events – Alternative Flow

- The same formatting rules as for the main flow
- Represents deviations from the main flow due to errors or interruptions
- It can also be used to capture more complex branching, such as situations that are not exactly known if and when they will occur

In this seminar…

Work, work

- Visual Paradigm demo
- Team work on project
  - Use case diagram II

Task for this week

You gotta do what you gotta do

- Revise the use case diagram and think using the new constructs
  - At least one include/extend should be applicable
- (Wire a brief description for all use cases)
  - ~2 sentences
- Choose 3 use cases and create full textual specification
  - Choose the more complex ones to make more sense
  - Mark them on diagram (colour)