Chapter 3: UML

Content:

• Learn how to draw UML diagrams
• UML is an alternative way to model a database

Next:

• Convert UML and ER diagrams into a database schema
Modeling a small example application: E/R
Modeling a small example application: UML

**Students**
- `StudNr : int`
- `Name : String`
- `Semester : int`
- `GPA() : float`
- `SumWeeklyHours() : short`

**Lectures**
- `LectureNr : int`
- `Title : String`
- `WeeklyHours : int`
- `NumberAttendees() : int`
- `FailureRate() : float`

**Relationships**
- **Students** attends **Lectures**
- **Lectures** requires **Students**

**Notations**
- `1..*` indicates the multiplicity of `Students` attending `Lectures`.
- `*` indicates the multiplicity of `Lectures` required by `Students`.

**Attributes**
- `+Attendee`
- `+Successor`
Data modelling with UML

- UML = Unified Modelling Language
- De facto standard for object oriented software design
- Several diagrams, we focus on class diagrams
Main concepts in UML class diagrams:

- **Classes**: models similar objects according to:
  - Structure (~Attributes)
  - Behavior (~Operations/Methods)
  - ≈ Entities in ER-Diagram

- **Associations**: between classes correspond to relationships
  - Generalization, Aggregation, …
  - ≈ Relationship in ER-Diagram

- **Multiplicities**: for associations
  - 0..* to 0..*, 1 to 1, …
  - ≈ Functionalities in ER-Diagram
Multiplicity

- Every element of **Class A** is associated with at least \( i \) elements of **Class B** and with at most \( j \) elements of **Class B**
- Analogously for the interval \( k..l \)
- Multiplicity is analogously to the functionalities in the ER-Model not to the \((\text{min}, \text{max})\)-Notation: Watch out!
UML Association Types

Association:
• Generic relationship
• Any multiplicity possible

Generalization:
• “Is-a” relationship
• Inheritance in Java/C++
UML Association Types

Aggregation:

- “Part-of” or “has"
- Multiple owners

Composition:

- Special case of Aggregation
- Existence dependent
- Exactly one owner
### Multiplicity: Example 1

- **Human** can have an arbitrary amount of **Dogs**
- **Dog** might have 0 or 1 **Humans**
- Association type: Regular association or aggregation
Multiplicity: Example 2

- A Car has exactly 4 Wheels
- A Wheel belongs to exactly 1 Car

Association type: Composition or aggregation
Multiplicity: Example 3

- A **Lecture** has to have at least 3 **Students**
- A **Student** can attend any number of **Lectures**

- Association type: Aggregation
Multiplicity: Example 4

- A **Square** is a **Shape**

- Association type: Generalization
Association class

Class A
+Att1
+Att2
+ op()

Class C
+Att1
+Att2

Class B
+Att1
+Att2
+ op()

... for attributes of the association
Navigation

<table>
<thead>
<tr>
<th>Students</th>
<th>+Attendee</th>
<th>+Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>+StudNr : int</td>
<td>1..*</td>
<td>*</td>
</tr>
<tr>
<td>+Name : String</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Semester : int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+GPA() : float</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+SumWeeklyHours() : short</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lectures</th>
<th>+LectureNr : int</th>
<th>+Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Title : String</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>+WeeklyHours : int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+NumberAttendees() : int</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+FailureRate() : float</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Arrows: Navigation (Implementation)

No statement on navigation

Navigation from A to B allowed

Navigation from A to B forbidden
Composition

**Students**

- +StudNr: int
- +Name: string
- +Semester: int

- +GPA(): float
- +SumWeeklyHours(): short

**Exams**

- +Examinee
- +Subject
- +move()

- +Grade: decimal
- +Date: date

1 pass

* +Examinee

...
Other useful Diagrams

- Use Case Diagram
- Interaction Diagram
- Sequence Diagram

Search for examples in the internet
Quiz UML

From the Stanford MOOC:


Quiz Q2 + Q5 – Q7
Cheat sheet class diagram


The Account belongs to one Bank. The Bank contains 0 to infinite Accounts.

This type is often use in n to m relationships. The „Account“ chains the „Product“ with the...

The class „KitchenAppliance“ implements the interface.

The account is part of the bank and can't exits without...

The „Child“ extends the „Parent“ and contains every...

The „Customer“ is part of the „Bank“, but the customer...

The „Student“ dependents on the „University“...