# Software Design, Modelling and Analysis in UML

## Lecture 09: Class Diagrams III

*2014-11-25*

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany



## Contents & Goals

**Last Lectures:**

- completed class diagrams... except for associations

**This Lecture:**

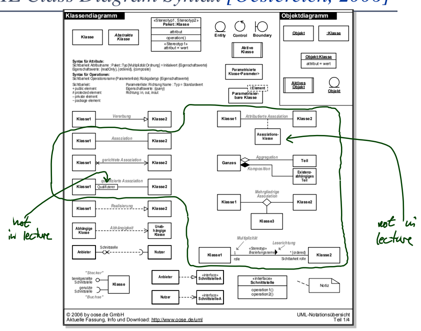
- **Educational Objectives:** Capabilities for following tasks/questions.
  - Please explain this class diagram with associations.
  - Which annotations of an association arrow are semantically relevant?
  - What's a role name? What's it good for?
  - What is "multiplicity"? How did we treat them semantically?
  - What is "reading direction", "navigability", "ownership", . . . ?
  - What's the difference between "aggregation" and "composition"?
- **Content:**
  - Study concrete syntax for "associations".
  - (**Temporarily**) extend signature, define mapping from diagram to signature.
  - Study effect on OCL.
  - Btw.: where do we put OCL constraints?



## Associations: Syntax

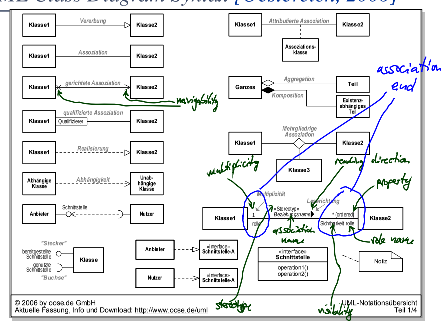


## UML Class Diagram Syntax [Oestereich, 2006]






## UML Class Diagram Syntax [Oestereich, 2006]






## UML Class Diagram Syntax [OMG, 2007b, 61;43]

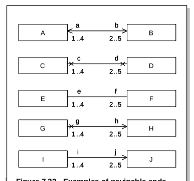


Figure 7.23 - Examples of navigable ends

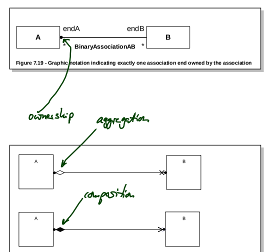



Figure 7.19 - Graphic notation indicating exactly one association end owned by the association

Figure 7.20 - Combining line path graphics

## What Do We (Have to) Cover?

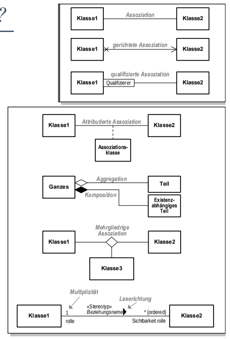
An **association** has

- ! a **name**,
- ✓ a **reading direction**, and ← *just a hint to the reader of the diagram*
- ! at least two **ends**.
- ! *a set of stereotypes*

Each **end** has

- ! a **role name**,
- ! a **multiplicity**,
- ! a set of **properties**, such as **unique**, **ordered**, etc.
- – a **qualifier**, *(not in lecture)*
- ! a **visibility**,
- ! a **navigability**,
- ! an **ownership**,
- ! and possibly a **diamond**. *(exercises)*

**Wanted**: places in the signature to represent the information from the picture.






## (Temporarily) Extend Signature: Associations

**Only for the course of Lectures 9/10** we assume that each attribute in $V$

- **either** is $\langle v : \tau, \xi, expr_0, P_v\rangle$ with $\tau \in \mathscr{T}$ (as before),
- **or** is an **association** of the form

$$\begin{aligned}\langle r : \quad & \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1\rangle, \\ & \vdots \\ & \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n\rangle\rangle\end{aligned}$$

*assoc. name* (→ $r$), *assoc. end* (→ $\langle role_n : \ldots\rangle$), *the class where this association end is located* (→ $C_n$)



## (Temporarily) Extend Signature: Associations

**Only for the course of Lectures 9/10** we assume that each attribute in $V$

- **either** is $\langle v : \tau, \xi, expr_0, P_v\rangle$ with $\tau \in \mathscr{T}$ (as before),
- **or** is an **association** of the form

$$\begin{aligned}\langle r : \quad & \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1\rangle, \\ & \vdots \\ & \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n\rangle\rangle\end{aligned}$$

where

- $n \geq 2$ (at least two ends),
- $r$, $role_i$ are just **names**, $C_i \in \mathscr{C}$, $1 \leq i \leq n$,
- the **multiplicity** $\mu_i$ is an expression of the form

$$\mu ::= * \mid N \mid N..M \mid N..* \mid \mu, \mu \qquad (N, M \in \mathbb{N})$$

- $P_i$ is a set of **properties** (as before),
- $\xi \in \{+, -, \#, \sim\}$ (as before),
- $\nu_i \in \{\times, -, >\}$ is the **navigability**,
- $o_i \in \mathbb{B}$ is the **ownership**.



## (Temporarily) Extend Signature: Associations

**Only for the course of Lectures 9/10** we assume that each attribute in $V$

- **either** is $\langle v : \tau, \xi, expr_0, P_v\rangle$ with $\tau \in \mathscr{T}$ (as before),
- **or** is an **association** of the form

$$\langle r : \quad \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1\rangle,$$

**Alternative syntax** for multiplicities:

$$\mu ::= N..M \mid N..* \mid \mu, \mu \qquad (N, M \in \mathbb{N} \cup \{*\})$$

and define $*$ and $N$ as abbreviations.

**Note**: $N$ could abbreviate $0..N$, $1..N$, or $N..N$. We use last one.

$$\mu ::= * \mid N \mid N..M \mid N..* \mid \mu, \mu \qquad (N, M \in \mathbb{N})$$

- $P_i$ is a set of **properties** (as before),
- $\xi \in \{+, -, \#, \sim\}$ (as before),
- $\nu_i \in \{\times, -, >\}$ is the **navigability**,
- $o_i \in \mathbb{B}$ is the **ownership**.



## (Temporarily) Extend Signature: Basic Type Attributes

**Also only** for the course of this lecture

- we only consider **basic type attributes** to "belong" to a class (to appear in $atr(C)$),
- **associations** are not "owned" by a particular class (do not appear in $atr(C)$), but live on their own.

**Formally**: we only call

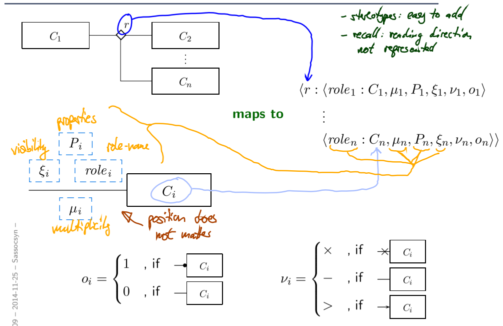
$$(\mathscr{T}, \mathscr{C}, V, atr)$$

a **signature (extended for associations)** if

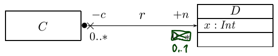
$$atr : \mathscr{C} \to 2^{\{v \in V \mid v : \tau, \tau \in \mathscr{T}\}}.$$



## From Association Lines to Extended Signatures



## Association Example




**Signature**:

$\mathscr{S} = (\{Int\}, \{C, D\}, \{\langle x : Int, +, *, \emptyset \rangle,$
$\langle r : \langle c : C, 0..*, \emptyset, -, \times, 1 \rangle,$
$\langle n : D, 0..1, \emptyset, +, >, 0 \rangle \rangle\},$
$\{C \mapsto \emptyset$
$D \mapsto \{x\}\})$ ← only basic type attributes here



## What If Things Are Missing?

Most components of associations or association end may be omitted.
For instance [OMG, 2007b, 17], Section 6.4.2, proposes the following rules:

- **Name**: Use

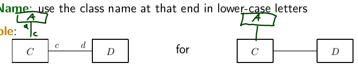
$$A\_\langle C_1 \rangle\_ \cdots \_\langle C_n \rangle$$

if the name is missing.

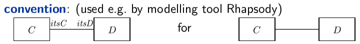
**Example**:


- **Reading Direction**: no default.
- **Role Name**: use the class name at that end in lower-case letters

**Example**:




**Other convention**: (used e.g. by modelling tool Rhapsody)






## What If Things Are Missing?

- **Multiplicity**: 1 0..∗

  In my opinion, it's safer to assume $0..1$ or $*$ if there are no fixed, written, agreed conventions ("expect the worst").
- **Properties**: $\emptyset$
- **Visibility**: public
- **Navigability and Ownership**: not so easy. [OMG, 2007b, 43]

*"Various options may be chosen for showing navigation arrows on a diagram.*
*In practice, it is often convenient to suppress some of the arrows and crosses and just show exceptional situations:*

- *Show all arrows and x's. Navigation and its absence are made completely explicit.*
- *Suppress all arrows and x's. No inference can be drawn about navigation.*
  *This is similar to any situation in which information is suppressed from a view.*
- *Suppress arrows for associations with navigability in both directions, and show arrows only for associations with one- way navigability.*
  *In this case, the two-way navigability cannot be distinguished from situations where there is no navigation at all; however, the latter case occurs rarely in practice."*



## Wait, If Omitting Things...

- ...**is causing so much trouble** (e.g. leading to misunderstanding), why does the standard say "**In practice, it is often convenient**..."?

  Is it a good idea to trade **convenience** for **precision/unambiguity**?

  **It depends**.
  - Convenience as such is a legitimate goal.
  - In UML-As-Sketch mode, precision "doesn't matter", so convenience (for writer) can even be a primary goal.
  - In UML-As-Blueprint mode, **precision** is the **primary goal**.
    And misunderstandings are in most cases annoying.

    **But**: (even in UML-As-Blueprint mode)
    If all associations in your model have multiplicity $*$,
    then it's probably a good idea not to write all these $*$'s.
    **So**: tell the reader about it and leave out the $*$'s.



## Association Semantics



## Overview

**What's left? Named** association with at least two typed **ends**, each having

- a **role name**,
- a **multiplicity**,
- a set of **properties**,
- a **visibility**,
- a **navigability**, and
- an **ownership**.

**The Plan:**

- Extend **system states**, introduce so-called **links** as instances of associations — depends on **name** and on **type** and **number** of ends.
- Integrate **role name** and **multiplicity** into **OCL syntax/semantics**.
- Extend **typing rules** to care for **visibility** and **navigability**
- Consider **multiplicity** also as part of the **constraints** set $Inv(\mathcal{CD})$.
- **Properties**: for now assume $P_v = \{\texttt{unique}\}$.
- **Properties** (in general) and **ownership**: later.

## Association Semantics: The System State Aspect





## Associations in General

**Recall**: We consider associations of the following form:

$$\langle r : \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1\rangle, \dots, \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n\rangle\rangle$$

Only these parts are relevant for extended system states:

$$\langle r : \langle role_1 : C_1, \_, P_1, \_, \_, \_\rangle, \dots, \langle role_n : C_n, \_, P_n, \_, \_, \_\rangle$$

(recall: we assume $P_1 = P_n = \{\texttt{unique}\}$).

The UML standard thinks of associations as **n-ary relations** which "**live on their own**" in a system state.

That is, **links** (= association instances)

- **do not** belong (in general) to certain objects (in contrast to pointers, e.g.)
- are "first-class citizens" **next to objects**,
- are (in general) **not** directed (in contrast to pointers).





## Links in System States

$$\langle r : \langle role_1 : C_1, \_, P_1, \_, \_, \_\rangle, \dots, \langle role_n : C_n, \_, P_n, \_, \_, \_\rangle$$

**Only for the course of Lectures 9/10** we change the definition of system states:

**Definition.** Let $\mathscr{D}$ be a structure of the (extended) signature $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$.

A system state of $\mathscr{S}$ wrt. $\mathscr{D}$ is a pair $(\sigma, \lambda)$ consisting of

- a type-consistent mapping

$$\sigma : \mathscr{D}(\mathscr{C}) \nrightarrow (atr(\mathscr{C}) \nrightarrow \mathscr{D}(\mathscr{T})),$$

- a mapping $\lambda$ which assigns each association $\langle r : \langle role_1 : C_1\rangle, \dots, \langle role_n : C_n\rangle\rangle \in V$ a relation

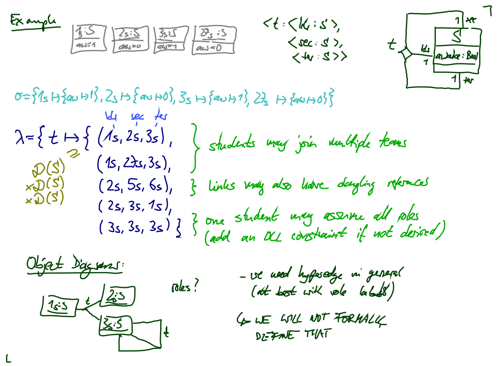
$$\lambda(r) \subseteq \mathscr{D}(C_1) \times \dots \times \mathscr{D}(C_n)$$

(i.e. a set of type-consistent $n$-tuples of identities).

values for basic types only








## Association/Link Example


**Signature**:

$$\mathscr{S} = (\{Int\}, \{C, D\}, \{x : Int, \langle A\_C\_D : \langle c : C, 0..*, +, \{\texttt{unique}\}, \times, 1\rangle, \langle n : D, 0..*, +, \{\texttt{unique}\}, >, 0\rangle\rangle\}, \{C \mapsto \emptyset, D \mapsto \{x\}\})$$

A **system state** of $\mathscr{S}$ (some reasonable $\mathscr{D}$) is $(\sigma, \lambda)$ with:

$$\sigma = \{1_C \mapsto \emptyset, 3_D \mapsto \{x \mapsto 1\}, 7_D \mapsto \{x \mapsto 2\}\}$$

$$\lambda = \{A\_C\_D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$$





## Extended System States and Object Diagrams

**Legitimate question**: how do we represent system states such as

$$\sigma = \{1_C \mapsto \emptyset, 3_D \mapsto \{x \mapsto 1\}, 7_D \mapsto \{x \mapsto 2\}\}$$
$$\lambda = \{A\_C\_D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$$

as **object diagram**?

# References



[Oestereich, 2006] Oestereich, B. (2006). *Analyse und Design mit UML 2.1, 8. Auflage*. Oldenbourg, 8. edition.

[OMG, 2007a] OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.

[OMG, 2007b] OMG (2007b). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.