Software Design, Modelling and Analysis in UML

Lecture 07: Class Diagrams II

2011-11-30

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

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lectures:

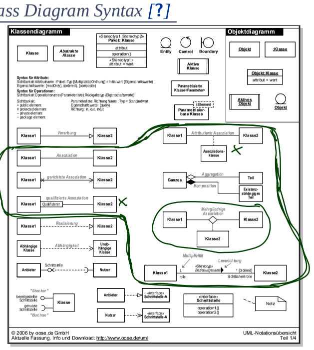
- VL 05: class diagram — except for associations
- VL 06: semantics of visibility within OCL type system

This Lecture:

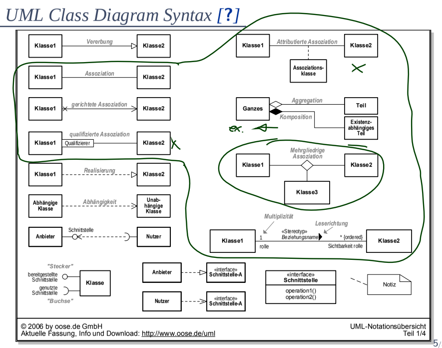
- **Educational Objectives:** Capabilities for following tasks/questions.
  - Please explain this class diagram with associations.
  - Which annotations of an association arrow are semantically relevant?
  - What's a role name? What's it good for?
  - What's "multiplicity"? How did we treat them semantically?
  - 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.
  - Where do we put OCL constraints?

Associations: Syntax

UML Class Diagram Syntax [?]



UML Class Diagram Syntax [?]



UML Class Diagram Syntax [?, 61;43]

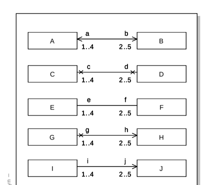


Figure 7.23 - Examples of navigable ends

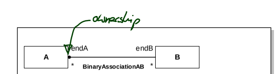


Figure 7.19 - Graphic notation indicating exactly one association end owned by the association

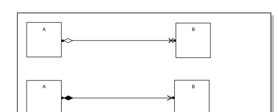


Figure 7.20 - Combining line path graphics

What Do We (Have to) Cover?

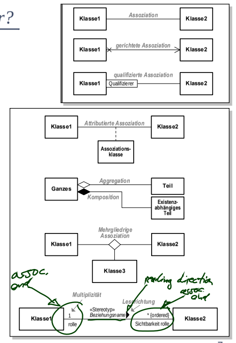
An **association** has

- a **name**,
- a **reading direction**, and
- at least two **ends**. (association)

Each **end** has

- a **role name**,
- a **multiplicity**,
- a set of **properties**, such as **unique**, **ordered**, etc.
- a **qualifier**, (we will not treat)
- 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 07/08 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

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

association name

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** μ_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 07/08 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

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

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.

- r, $role_i$ are just **names**, $C_i \in \mathscr{C}$, $1 \leq i \leq n$,
- the **multiplicity** μ_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: Basic Type Attributes

Also only for the course of ~~this~~ lectures 07/08

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

OLD: C, x : Int —r→ D

$atr(C) = \{x, r\}$

1c : C, x = 27, r = 3D; 3D : D

Formally: we only call

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

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

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

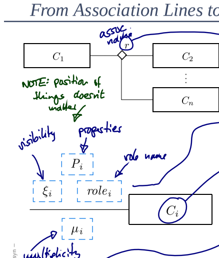
basic type

07/08: σ: 1c : C, x = 27; 3D : D

r : (1c, 3D)

NOW $V = \{\ldots, r : D_{0,1}\}$
NOT: $atr(C) = \{\ldots, r, \ldots\}$
because r not of basic type

From Association Lines to Extended Signatures



reading direction is ignored

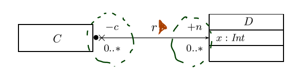
maps to

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

$$o_i = \begin{cases} 1 & \text{, if } C_i \\ 0 & \text{, if } C_i \end{cases} \qquad \nu_i = \begin{cases} \times & \text{, if } C_i \\ - & \text{, if } C_i \\ > & \text{, if } C_i \end{cases}$$

Association Example

< $role_1$: C_1, μ_1, P_1, ξ_1, ν_1, o_1 >



Signature:

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

now: only basic type attrs. are assigned by atr

What If Things Are Missing?

Most components of associations or association end may be omitted.
For instance [?, 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.

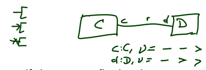
  Example:

- **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

  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. [?, 43]

OLD CONVENTION:

"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 set of **properties**,
- a **navigability**, and
- a **multiplicity**,
- a **visibility**,
- 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:

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

Only these parts are relevant for extended system states:

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

(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

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

Only for the course of lectures 07/08 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 (σ, λ) consisting of

- a type-consistent mapping
$$\sigma : \mathscr{D}(\mathscr{C}) \nrightarrow (atr(\mathscr{C}) \nrightarrow \mathscr{D}(\mathscr{T})),$$
- a mapping λ which assigns each association $\langle r : \langle role_1 : C_1 \rangle, \ldots, \langle role_n : C_n \rangle \rangle \in V$ a relation
$$\lambda(r) \subseteq \mathscr{D}(C_1) \times \cdots \times \mathscr{D}(C_n)$$
(i.e. a set of type-consistent n-tuples of identities).

object identity

NEW: only basic types

set of n-tuples

Association/Link Example

Signature:

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

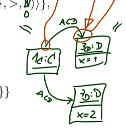
NOT navigability but direction, first/second entry of pair

one instance of class C

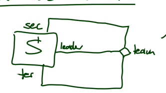
A **system state** of $\mathscr{S}$ (some reasonable $\mathscr{D}$) is (σ, λ) with:

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

(A_C_D)



The order does matter...



$\mathscr{S}$:

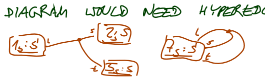
⟨team : ⟨le : S, .. ⟩
⟨sec : S, .. ⟩
⟨te : S, .. ⟩⟩

$\sigma = \{ 1_S \mapsto \emptyset, 2_S \mapsto \emptyset, 5_S \mapsto \emptyset, 7_S \mapsto \emptyset \}$

$\lambda(t) = \{ (5_S, 2_S, 1_S), (1_S, 7_S, 5_S), (7_S, 7_S, 7_S) \}$

$\cap$ $\mathscr{D}(S) \times \mathscr{D}(S) \times \mathscr{D}(S)$

OBJECT DIAGRAM WOULD NEED HYPEREDGES:



WE WILL NOT FORMALLY DEFINE THAT!

20a/.

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**?

- see page 20
- see page 20a

Associations and OCL

OCL and Associations: Syntax

Recall: OCL syntax as introduced in Lecture 03, interesting part:

$$\begin{array}{llll} expr ::= \dots & \mid r_1(expr_1) & : \tau_C \to \tau_D & r_1 : D_{0,1} \in atr(C) \\ & \mid r_2(expr_1) & : \tau_C \to Set(\tau_D) & r_2 : D_* \in atr(C) \end{array}$$

Now becomes

$$\begin{array}{llll} expr ::= \dots & \mid role(expr_1) & : \tau_C \to \tau_D & \mu = 0..1 \text{ or } \mu = 1 \\ & \mid role(expr_1) & : \tau_C \to Set(\tau_D) & \text{otherwise} \end{array}$$

if

$$\langle r : \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots\rangle \in V \text{ or}$$

$$\langle r : \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots\rangle \in V, role \neq role'.$$

Note:

- Association name as such doesn't occur in OCL syntax, role names do.
- $expr_1$ has to denote an object of a class which "participates" in the association.

References