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?
Overview

- **Class diagram:**
  
  ![Class diagram](image)

- **Alternative presentation:**
  
  ![Alternative presentation](image)

- **Signature:**
  
  $$\mathcal{I} = (\{\text{Int}\}, \{C, D\}, \{v : \text{Int}, d : D_*, c : C_{0,1}\}, 
  \{C \mapsto \{v, d\}, D \mapsto \{c\}\})$$

- **Example system state:**
  
  $$\sigma = \{1_C \mapsto \{v \mapsto 27, d \mapsto \{5_D, 7_D\}\}, 
  5_D \mapsto \{c \mapsto \{1_C\}\}, 7_D \mapsto \{c \mapsto \{1_C\}\}\}$$

- **Object diagram:**
  
  ![Object diagram](image)

- **Class diagram** (with ternary association):
  
  ![Class diagram](image)

- **Signature:** extend again;
  
  represent **association** $r$
  
  with **association ends** $a$, $b$, and $z$
  
  (each with multiplicity, visibility, etc.)

- **Example system state:**
  
  $$(\sigma, \lambda) \xrightarrow{3_2} \sigma \xrightarrow{2_2} \emptyset$$
  
  $$\sigma = \{1_A \mapsto \{w \mapsto 13\}, 1_B \mapsto \emptyset, 1_Z \mapsto \emptyset\}$$
  
  $$\lambda = \{r \mapsto \{(1_A, 1_B, 1_Z)\}\}$$
Overview

- **Class diagram:**

```
  C
  \( v : \text{Int} \)
  \( d : D^* \)

  D
  \( c : C_{0,1} \)
```

Alternative presentation:

```
  C
  \( v : \text{Int} \)

  \( c \)
  \( 0,1 \)

  \( d \)
```

- **Signature:**

\[ \mathcal{S} = (\{\text{Int}\}, \{C, D\}, \{v : \text{Int}, d : D^*, c : C_{0,1}\}, \{C \mapsto \{v, d\}, D \mapsto \{c\}\}) \]

- **Example system state:**

\[ \sigma = \{1_C \mapsto \{v \mapsto 27, c \mapsto \{5_D, 7_D\}\}, 5_D \mapsto \{d \mapsto \{1_C\}\}, 7_D \mapsto \{d \mapsto \{1_C\}\}\} \]

- **Object diagram:**

```
  \( C \)
  \( v = 27 \)

  \( D \)
  \( d \)
  \( c \)
```

- **Class diagram** (with ternary association):

```
  A
  \( w : \text{Int} \)

  B
  \( a \)
  \( b \)

  Z
  \( r \)
  \( z \)
  \( 1.5 \)
  \( 0,1 \)
```

- **Signature:** extend again;
  represent association \( r \)
  with association ends \( a, b, \) and \( z \)
  (each with multiplicity, visibility, etc.)

- **Example system state:**

\[ \sigma = \{1_A \mapsto \{w \mapsto 13\}, 1_B \mapsto \emptyset, 1_Z \mapsto \emptyset\} \]

\[ \lambda = \{r \mapsto \{(1_A, 1_B, 1_Z)\}\} \]

- **Object diagram:** No...
Plan

(i) Study association syntax.

(ii) Extend signature accordingly.

(iii) Define \((\sigma, \lambda)\) system states with

- **objects** in \(\sigma\) (instances of classes),
- **links** in \(\lambda\) (instances of associations).

(iv) Change syntax of OCL to refer to association ends.

(v) Adjust interpretation \(I\) accordingly.

(vi) ... go back to the special case of \(C_{0,1}\) and \(C_*\) attributes.

- **Class diagram** (with ternary association):

- **Signature**: extend again; represent association \(r\) with association ends \(a, b,\) and \(z\) (each with multiplicity, visibility, etc.)

- **Example system state**:

  \[
  \sigma = \{ 1_A \mapsto \{ w \mapsto 13 \}, 1_B \mapsto \emptyset, 1_Z \mapsto \emptyset \} \\
  \lambda = \{ r \mapsto \{(1_A, 1_B, 1_Z)\} \} 
  \]

- **Object diagram**: No...
Associations: Syntax
UML Association Syntax

Oestereich (2006)

Klasse1

Assoziation

Klasse2

gerichtete Assoziation

Klasse1

qualifizierte Assoziation

Klasse2

Klasse1

Attributierte Assoziation

Klasse2

Assoziationsklasse

Klasse1

Ganzes

Aggregation

Klasse2

Teil

Existenz-abhängiges Teil

Klasse1

Mehrgliedrige Assoziation

Klasse3

Klasse1

Attribut

Klasse2

Multiplizität

Leserichtung

Klasse1

Sichtbarkeit rolle

Klasse2

* (ordered)

rolle

«Stereotyp» Beziehungsname
More Association Syntax *(OMG, 2011b, 61;43)*

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

Figure 7.20 - Combining line path graphics

Figure 7.23 - Examples of navigable ends
So, What Do We (Have to) Cover?

An association has

- a name,
- a reading direction, and
- at least two ends.

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. (exercise)

Wanted: places in the signature to represent the information from the picture.
(Temporarily) Extend Signature: Associations

Only for the course of Lectures 08/09 we assume that each element in $V$ is

- either a **basic type attribute** $\langle v : T, \xi, \text{expr}_0, P_v \rangle$ with $T \in \mathcal{T}$ (as before),
- or an **association** of the form

$$
\begin{align*}
\langle r : \langle \text{role}_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \\
\vdots \\
\langle \text{role}_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle
\end{align*}
$$

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

$$
\mu ::= N..M \mid N..* \mid \mu, \mu
$$

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

- $N$ for $N..N$,
- $*$ for $0..*$ (use with care!)
(Temporarily) Extend Signature: Associations

Only for the course of Lectures 08/09 we assume that each element in $V$ is

- **either a basic type attribute** $\langle v : T, \xi, expr_0, P_v \rangle$ with $T \in \mathcal{T}$ (as before),
- **or an association** of the form

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

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

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

- $P_i$ 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**.

**Multiplicity abbreviations:**

- $N$ for $N..N$,
- $*$ for $0..*$ (use with care!)
From Association Lines to Extended Signatures

\[ r : \langle \text{role}_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle \]

maps to

\[ \langle \text{role}_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \]

\[ o_i = \begin{cases} 1 & \text{, if } C_i \\ 0 & \text{, if } \end{cases} \]

\[ \nu_i = \begin{cases} \times & \text{, if } C_i \\ - & \text{, if } C_i \\ > & \text{, if } C_i \end{cases} \]
Association Example

![Diagram of association example]

Signature:

\[ \mathcal{I} = (\{\text{Int}\}, \{C, D\}, \{x : \text{Int}\}, \langle r : \langle c : C, 0..*, \{\text{unique}\}, -, x, 1 \rangle \rangle, \langle n : D, 0..*, \{\text{unique}\}, +, >, 0 \rangle \rangle, \{C \mapsto \emptyset, D \mapsto \{x\}\} ) \]
What If Things Are Missing?

Most components of associations or association end may be omitted. For instance (OMG, 2011b, 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**: 
  \[
  \begin{array}{c}
  \text{C} \\
  \end{array}
  \begin{array}{c}
  \text{D} \\
  \end{array}
  \]
  \[ A_{\text{C-D}} \]
  for
  \[
  \begin{array}{c}
  \text{C} \\
  \end{array}
  \begin{array}{c}
  \text{D} \\
  \end{array}
  \]

- **Reading Direction**: no default.

- **Role Name**: use the class name at that end in lower-case letters

  **Example**:
  \[
  \begin{array}{c}
  \text{C} \\
  \end{array}
  \begin{array}{c}
  \text{D} \\
  \end{array}
  \]
  \[ c_d \]
  for
  \[
  \begin{array}{c}
  \text{C} \\
  \end{array}
  \begin{array}{c}
  \text{D} \\
  \end{array}
  \]

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

  **Example**:
  \[
  \begin{array}{c}
  \text{C} \\
  \end{array}
  \begin{array}{c}
  \text{D} \\
  \end{array}
  \]
  \[ itsC \text{ itsD} \]
  for
  \[
  \begin{array}{c}
  \text{C} \\
  \end{array}
  \begin{array}{c}
  \text{D} \\
  \end{array}
  \]
**What If Things Are Missing?**

- **Multiplicity**: 1

  In my opinion, it’s safer to assume 0..1 or ∗ (for 0..∗) if there are no fixed, written, agreed conventions (“expect the worst”).

- **Properties**: ∅

- **Visibility**: public

- **Navigability and Ownership**: not so easy. (OMG, 2011b, 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.”
...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 your convention and leave out the *’s.
**Definition.** An (Extended) Object System Signature (with Associations) is a quadruple $\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, \text{atr})$ where

- ... 
- each element of $V$ is
  - either a **basic type attribute** $\langle v : T, \xi, \text{expr}_0, P_v \rangle$ with $T \in \mathcal{T}$
  - or an association of the form
    $$\langle r : \langle \text{role}_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \ldots, \langle \text{role}_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle$$
- ... 
- $\text{atr} : \mathcal{C} \rightarrow 2\{v \in V \mid v : T, T \in \mathcal{T} \}$ maps each class to its set of **basic type (!)** attributes.

In other words:
- only **basic type attributes** “belong” to a class (may appear in $\text{atr}(C)$),
- **associations** are not “owned” by a particular class (do not appear in any $\text{atr}(C)$), but “live on their own”. 

---

**Temporarily (Lecture 8/9) Extended Signature**

---

- 8 – 2015-11-26 – Sassocsyn – 15/34
Associations: Semantics
Recall: We consider associations of the following form:

\[
\langle r : \langle \text{role}_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \ldots, \langle \text{role}_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle
\]

Only these parts are relevant for extended system states:

\[
\langle r : \langle \text{role}_1 : C_1, -, P_1, -, -, - \rangle, \ldots, \langle \text{role}_n : C_n, -, P_n, -, -, - \rangle \rangle
\]

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

The UML standard thinks of associations as \textit{n-ary relations} which \textit{“live on their own”} in a system state.

That is, \textit{links} (= association instances)

- \textbf{do not} belong (in general) to certain objects (in contrast to pointers, e.g.)
- are “first-class citizens” \textit{next to objects},
- are (in general) \textbf{not} directed (in contrast to pointers).
Only for the course of lectures 8 / 9 we change the definition of system states:

**Definition.** Let $\mathcal{D}$ be a structure of the (extended) signature with associations $\mathcal{I} = (\mathcal{T}, \mathcal{C}, V, \text{atr})$.

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

- a type-consistent mapping (as before)
  \[
  \sigma : \mathcal{D}(\mathcal{C}) \leftrightarrow (\text{atr}(\mathcal{C}) \leftrightarrow \mathcal{D}(\mathcal{T})),
  \]
- a mapping $\lambda$ which maps each association $\langle r : \langle \text{role}_1 : C_1 \rangle, \ldots, \langle \text{role}_n : C_n \rangle \rangle \in V$ to a relation
  \[
  \lambda(r) \subseteq \mathcal{D}(C_1) \times \cdots \times \mathcal{D}(C_n)
  \]
  (i.e. a set of type-consistent $n$-tuples of identities).
Association / Link Example

Signature:
\[ S = (\{b, c\}, \{A, B, Z\}, \{w: Int, \langle a: A, 0..1, r, \{unique\}, 0\rangle, \langle b: B, 0..1, r, \{unique\}, 0\rangle, \langle z: Z, 1..5, r, \{unique\}, 0\rangle\}) \]

System state:
\[ \sigma: \{1_A \mapsto \{w \mapsto 0\}, 2_A \mapsto \{w \mapsto 13\}, 3_A \mapsto \{w \mapsto 23\}, 10_B \mapsto \emptyset, 11_B \mapsto \emptyset, 22_B \mapsto \emptyset, 23_B \mapsto \emptyset\} \]

\[ \lambda: \{r \mapsto \{(1_A, 10_B, 23_Z), (2_A, 10_B, 23_Z), (1_A, 11_B, 23_Z), (2_A, 13_B, 29_Z)\}\} \]
Associations and OCL
Recall: OCL syntax as introduced in Lecture 3, interesting part:

\[
expr ::= \ldots \mid r_1(expr_1) : \tau_C \rightarrow \tau_D \quad r_1 : D_{0,1} \in atr(C)
\]

\[
\mid r_2(expr_1) : \tau_C \rightarrow Set(\tau_D) \quad r_2 : D_* \in atr(C)
\]
OCL and Associations: Syntax

Recall: OCL syntax as introduced in Lecture 3, interesting part:

\[
expr ::= \ldots \mid r_1(expr_1) : \tau_C \rightarrow \tau_D \quad r_1 : D_{0,1} \in atr(C') \\
| r_2(expr_1) : \tau_C \rightarrow \text{Set}(\tau_D) \quad r_2 : D_\ast \in atr(C)
\]

Now becomes

\[
expr ::= \ldots \mid \text{role}(expr_1) : \tau_C \rightarrow \tau_D \quad \mu = 0..1 \text{ or } \mu = 1..1 \\
| \text{role}(expr_1) : \tau_C \rightarrow \text{Set}(\tau_D) \quad \text{otherwise}
\]

if there is

\[
\langle r : \ldots, \langle \text{role} : D, \mu, \ldots \rangle, \ldots, \langle \text{role}' : C, \ldots \rangle, \ldots \rangle \in V \text{ or }
\langle r : \ldots, \langle \text{role}' : C, \ldots \rangle, \ldots, \langle \text{role} : D, \mu, \ldots \rangle, \ldots \rangle \in V, \quad \text{role} \neq \text{role}'.
\]

Note:

- Association name as such does not occur in OCL syntax, role names do.
- \(expr_1\) has to denote an object of a class which “participates” in the association.
**OCL and Associations Syntax: Example**

\[
expr ::= \ldots \mid \text{role}(expr_1) : \tau_C \rightarrow \tau_D \quad \mu = 0..1 \text{ or } \mu = 1.
\]

if there is

\[
\langle r : \ldots, \langle \text{role} : D, \mu, \ldots \rangle, \ldots, \langle \text{role}' : C, \ldots \rangle, \ldots \rangle \in V \text{ or }
\langle r : \ldots, \langle \text{role}' : C, \ldots \rangle, \ldots, \langle \text{role} : D, \mu, \ldots \rangle, \ldots \rangle \in V, \text{role} \neq \text{role}'.
\]

* Figure 7.21 - Binary and ternary associations (OMG, 2011b, 44).

*context Players inv: \quad \text{team} \rightarrow \forall \text{tl}(\text{tl}.\text{hasLicense})*
Recall:

Assume $expr_1 : \tau_C$ for some $C \in \mathcal{C}$. Set $u_1 := I[expr_1](\sigma, \beta) \in \mathcal{D}(T_C)$.

- $I[r_1(expr_1)](\sigma, \beta) := \begin{cases} u & \text{if } u_1 \in \text{dom}(\sigma) \text{ and } \sigma(u_1)(r_1) = \{u\} \\ \bot & \text{otherwise} \end{cases}$

- $I[r_2(expr_1)](\sigma, \beta) := \begin{cases} \sigma(u_1)(r_2) & \text{if } u_1 \in \text{dom}(\sigma) \\ \bot & \text{otherwise} \end{cases}$

Now needed:

$I[role(expr_1)]((\sigma, \lambda), \beta)$

- We cannot simply write $\sigma(u)(\text{role})$.
  
  Recall: role is (for the moment) not an attribute of object $u$ (not in $\text{attr}(C)$).

- What we have is $\lambda(r)$ (with association name $r$, not with role name role!).

  $$\langle r : \ldots, \langle \text{role} : D, \mu, \ldots, \ldots \rangle, \ldots, \langle \text{role}' : C, \ldots, \ldots, \ldots \rangle, \ldots \rangle$$

  But it yields a set of $n$-tuples, of which some relate $u$ and some instances of $D$.

- role denotes the position of the $D$’s in the tuples constituting the value of $r$. 

References
