From Association Lines to Extended Signatures

Let's assume that each attribute in the course of Lectures 07/08 we assume that each association has the property of being extended.

The extended associations have the form

\[ \text{association} \mapsto \{ \text{attribute} \} \]

The extended associations are not "owned" by a particular class association. Also, only

\[ \text{signature(extended for associations)} \]

\[ \langle \text{attribute} \rangle \mapsto \text{multiplicity} \]

\[ \langle \text{role} \rangle \mapsto \text{visibility} \]

\[ \langle \text{visibility} \rangle \mapsto \text{multiplicity} \]

\[ \text{visibility} \mapsto \text{multiplicity} \]

\[ \text{multiplicity} \mapsto \text{visibility} \]

\[ \text{attribute} \mapsto \text{multiplicity} \]

\[ \text{role} \mapsto \text{visibility} \]

\[ \text{multiplicity} \mapsto \text{visibility} \]

\[ \text{visibility} \mapsto \text{multiplicity} \]

\[ \text{attribute} \mapsto \text{multiplicity} \]

\[ \text{role} \mapsto \text{visibility} \]

\[ \text{multiplicity} \mapsto \text{visibility} \]

\[ \text{visibility} \mapsto \text{multiplicity} \]

\[ \text{attribute} \mapsto \text{multiplicity} \]

\[ \text{role} \mapsto \text{visibility} \]

\[ \text{multiplicity} \mapsto \text{visibility} \]

\[ \text{visibility} \mapsto \text{multiplicity} \]
If Things Are Missing?

Most components of associations or association ends may be omitted. For instance, 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**: $C \langle D \rangle A \langle C \rangle D$ for $C \langle D \rangle$

- **Reading Direction**: nodefault.

- **RoleName**: Use the classname at that end in lowercase letters.
  
  **Example**: $C \langle D \rangle cd$ for $C \langle D \rangle$

- **Other Convention**: (used e.g. by modeling tool Rhapsody)
  
  $C \langle D \rangle d$ for $C \langle D \rangle$

- **Multiplicity**: 1
  
  In my opinion, it's safest to assume $0..1$ or $\ast$ if there are no fixed, written, agreed conventions ("expect the worst").

- **Properties**: $\emptyset$

- **Visibility**: public

- **Navigability and Ownership**: not so easy.

  "Variousoptions may be chosen for showing navigation arrowson 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 anysituation in which information is suppressed from a view.
  - Suppress arrows for associations with navigability in both directions, and show arrowsonly 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...

If things are causing so much trouble (e.g. leading to misunderstanding), why does the standard say "In practice, it is often convenient to suppress some of the arrows and crosses and just show exceptional situations:"

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 $\ast$, then it's probably a good idea not to write all these $\ast$'s. So: tell the reader about it and leave out the $\ast$'s.
Associations in General

Recall: We consider associations of the following form:

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

Only these parts are relevant for extended system states:

\[ \langle r: \langle role_1: C_1, P_1, \xi_1, \nu_1, o_1 \rangle \rangle, \ldots, \langle role_n: C_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle \]

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

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).

Association/Link Example

Signature:
\[ \mathcal{C}_{B} = (\{ \text{Int} \}, \{ C, D \}, \{ x: \text{Int}, \langle A \ C D: \langle c: D, 0..*\text{, \{unique\}, \times, 0 \rangle \rangle, \langle n: C, 0..*\text{, \{unique\}}, \times, 1 \rangle \rangle \}) \]

A system state of \( \mathcal{C}_{B} \) wrt. \( \mathcal{C}_{W} \) is a pair \((\sigma, \lambda)\) consisting of

• a type-consistent mapping \( \sigma: \mathcal{C}_{W}(\mathcal{B}_V) \rightarrow (\mathcal{A}_{\mathcal{C}}(\mathcal{B}_V)) \rightarrow \mathcal{C}_{B} \)
• a mapping \( \lambda \) which assigns each association \( \langle r: \langle role_1: C_1 \rangle \rangle, \ldots, \langle role_n: C_n \rangle \rangle \in \mathcal{V} \) a relation \( \lambda(r) \subseteq \mathcal{C}_{B}(C_1) \times \cdots \times \mathcal{C}_{B}(C_1) \) (i.e. a set of type-consistent \( n \)-tuples of identities).

Extended System States and Object Diagrams

Legitimate question: how do we represent system states such as

\[ \sigma = \{ 1 \ C \rightarrow \emptyset, 3 \ D \rightarrow \{ x \rightarrow 1 \}, 7 \ D \rightarrow \{ x \rightarrow 2 \} \} \]

\[ \lambda = \{ \langle A \ C D \rightarrow \{ (1 \ C, 3 \ D), (1 \ C, 7 \ D) \} \rangle \} \]
OCL and Association Syntax

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

expr ::= ...
| \text{role}(expr1): \tau C \rightarrow \tau D \mu = 0 \ldots 1 \text{ or } \mu = 1 |
| \text{role}(expr1): \tau C \rightarrow \text{Set}(\tau D)

Now becomes:

expr ::= ...
| \text{role}(expr1): \tau C \rightarrow \tau D \mu = 0 \ldots 1 \text{ or } \mu = 1 |
| \text{role}(expr1): \tau C \rightarrow \text{Set}(\tau D)

Note:
• Association name as such doesn’t occur in OCL syntax, role names do.
• expr1 has to denote an object of a class which “participates” in the association.

References