Software Design, Modelling and Analysis in UML

Lecture 10: Class Diagrams V

2014-11-27

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lectures:
- associations syntax and semantics

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:
- Associations and OCL
- Btw.: where do we put OCL constraints?
Association Semantics: The System State Aspect
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:

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

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

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

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

- \textbf{do not} belong (in general) to certain objects (in contrast to pointers, e.g.)
- are “first-class citizens” \textbf{next to objects},
- are (in general) \textbf{not} directed (in contrast to pointers).
Only for the course of Lectures 9/10 we change the definition of system states:

**Definition.** Let $\mathcal{D}$ be a structure of the (extended) signature $\mathcal{I} = (\mathcal{T}, \mathcal{C}, V, atr)$. A system state of $\mathcal{I}$ wrt. $\mathcal{D}$ is a pair $(\sigma, \lambda)$ consisting of

- a type-consistent mapping
  \[ \sigma : \mathcal{D}(\mathcal{C}) \leftrightarrow (\text{atr}(\mathcal{C}) \leftrightarrow \mathcal{D}(\mathcal{T})) \]
- a mapping $\lambda$ which assigns each association $\langle r : \langle \text{role}_1 : C_1, \ldots, \text{role}_n : C_n \rangle \rangle \in V$ 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).
Signature:

\[ \mathcal{S} = (\{\text{Int}\}, \{C, D\}, \{x : \text{Int}, \n \). \]

\[ \langle A_C D : \langle c : C, 0..*, +, \{\text{unique}\}, \times, 1 \rangle, \n \). \]

\[ \langle n : D, 0..*, +, \{\text{unique}\}, >, 0 \rangle \rangle \}, \n \). \]

\{C \mapsto \emptyset, D \mapsto \{x\}\})

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

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

\[ \lambda = \{A_C D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}\]
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?
Associations and OCL
**OCL and Associations: Syntax**

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

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

Now becomes

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

If there is

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

Two rows because: order of assoc. ends matters (tech. reason)

Example: \[\text{\begin{tikzpicture}[scale=0.8]
\node (A) at (0,0) {C};
\node (B) at (1,0) {D};
\node (C) at (1.5,0) {\text{if n(stfD) = NO}};
\node (D) at (3,0) {\text{if n(stfD) = OK}};
\draw (A) -- (B);\end{tikzpicture}}\]
**OCL and Associations: Syntax**

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

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

Now becomes

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

if

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

**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.
OCL and Associations Syntax: Example

\[
expr ::= \ldots \mid \text{role}(expr_1) : \tau_C \rightarrow \tau_D \\
\mid \text{role}(expr_1) : \tau_C \rightarrow \text{Set}(\tau_D)
\]

\[
\mu = 0..1 \text{ or } \mu = 1 \quad \text{otherwise}
\]

if

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

Figure 7.21 - Binary and ternary associations [OMG, 2007b, 44].

- context Players inv: size(year(self)) > 0 OK \{ (1p, 2p), = 2/buddy \}
- context Players inv: self.p \rightarrow size > 0 NOT OK \{ (2p, 1p) \}
- context Players inv: self.s.size \rightarrow size > 0 OK
- context Players inv: self.b \rightarrow size > 0 and self.a \rightarrow size > 0 OK
OCL and Associations: Semantics

Recall: (Lecture 03)

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

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

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

Now needed:

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

- We cannot simply write $\sigma(u)(\text{role})$.

Recall: $\text{role}$ is (for the moment) not an attribute of object $u$ (not in $\text{atr}(C)$).

- What we have is $\lambda(r)$ (with $r$, not with $\text{role}$!) — but it yields a set of $n$-tuples, of which some relate $u$ and other some instances of $D$.

- $\text{role}$ denotes the position of the $D$’s in the tuples constituting the value of $r$. 
Assume \( expr_1 : \tau_C \) for some \( C \in \mathcal{C} \). Set \( u_1 := I[expr_1](\sigma, \lambda, \beta) \in \mathcal{D}(\tau_C) \).

\[
\begin{align*}
\mu &= 0.1, \mu = 1 \\
I[\text{role}(expr_1)]((\sigma, \lambda), \beta) &= \begin{cases} 
  u, & \text{if } u_1 \in \text{dom}(\sigma) \text{ and } L(\text{role})(u_1, \lambda) = \{ u \} \\
  \perp, & \text{otherwise}
\end{cases}
\end{align*}
\]

where

\[
L(\text{role})(u, \lambda) = \{(u_1, \ldots, u_n) \in \lambda(r) \mid u \in \{u_1, \ldots, u_n\}\} \downarrow i
\]

if

\[
\langle r : \ldots \langle \text{role}_1 : \_ , \_ , \_ , \_ , \_ \rangle , \ldots \langle \text{role}_n : \_ , \_ , \_ , \_ , \_ \rangle , \ldots \rangle, \text{role} = \text{role}_i.
\]

Given a set of \( n \)-tuples \( A \),

\( A \downarrow i \) denotes the element-wise projection onto the \( i \)-th component.
OCL and Associations Example

$I[\text{role(expr}_1)](\langle \sigma, \lambda \rangle, \beta) := \begin{cases} L(\text{role})(u_1, \lambda) & \text{if } u_1 \in \text{dom}(\sigma) \\ \perp & \text{otherwise} \end{cases}$

$L(\text{role})(u, \lambda) = \{(u_1, \ldots, u_n) \in \lambda(r) \mid u \in \{u_1, \ldots, u_n\}\}$

$S = (\ldots, \{A_{-C-D}: c; c; \ldots, W; D; \ldots\}, \ldots)$

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

$\lambda = \{A_{-C-D} \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$

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

$\lambda = \{A_{-C-D} \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$

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

$\lambda = \{A_{-C-D} \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$

$I[\text{self} \cdot n](\langle \sigma, \lambda \rangle, \{\text{self} \mapsto 1_C\}) = I[I[\text{self} \cdot n](\langle \sigma, \lambda \rangle, \beta)] = \bigwedge(n)(1_C, \lambda) = \{3_D, 7_D\}$

$- \{(v_1, v_2) \in \lambda(A_{-C-D}) \mid 1_C \in \{v_1, v_2\}\}$

$- \{(v_1, v_2) \in \lambda(A_{-C-D}) \mid 1_C \in \{v_1, v_2\}\}$

$- \{(v_1, v_2) \in \lambda(A_{-C-D}) \mid 1_C \in \{v_1, v_2\}\}$

$\downarrow 2 = \{3_D, 7_D\}$
Associations: The Rest
**Recapitulation**: Consider the following association:

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

- **Association name** \( r \) and **role names/types** \( \text{role}_i/C_i \) induce extended system states \( \lambda \).
- **Multiplicity** \( \mu \) is considered in OCL syntax.
- **Visibility** \( \xi \)/**Navigability** \( \nu \): well-typedness.

**Now the rest:**

- **Multiplicity** \( \mu \): we propose to view them as constraints.
- **Properties** \( P_i \): even more typing.
- **Ownership** \( o \): getting closer to pointers/references.
- **Diamonds**: exercise.
Rhapsody Demo
References