Software Design, Modelling and Analysis in UML

Lecture 08: Class Diagrams III

2011-12-06

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
Contents & Goals

Last Lectures:

- Started to discuss “associations”, the general case.

This Lecture:

- **Educational Objectives**: Capabilities for following tasks/questions.
  - Cont’d: Please explain this class diagram with associations.
  - When is a class diagram a good class diagram?
  - What are purposes of modelling guidelines? (Example?)
  - Discuss the style of this class diagram.

- **Content**:
  - Recall association semantics and effect on OCL.
  - Treat “the rest”.
  - Where do we put OCL constraints?
  - Modelling guidelines, in particular for class diagrams (following [Ambler, 2005])
  - Examples: modelling games (made-up and real-world examples)
Recall: Associations and OCL
Recall: 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,
- a visibility,
- a navigability,
- an ownership (not in pictures),
- and possibly a diamond.

Wanted: places in the signature to represent the information from the picture.
Recall: (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 $\tau \in \mathcal{T}$ (as before),
- or is an association of the form

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

where

- $n \geq 2$ (at least two ends),
- $r, \text{ role}_i$ are just names,
- the multiplicity $\mu_i$ is an expression of the form

\[
\mu ::= * \mid N \mid N..M \mid N..* \mid \mu, \mu \quad (N, M \in \mathbb{N})
\]
- $P_i$ 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.
Recall: Associations in General

Recall: We consider associations of the following form:

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

Only these parts are relevant for extended system states:

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

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

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

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

- 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).
Recall: Links in System States

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

Only for the course of this lecture we change the definition of system states:

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

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

- a type-consistent mapping
  \[
  \sigma : \mathcal{D}(\mathcal{C}) \rightarrow (\text{atr}(\mathcal{C}) \rightarrow \mathcal{D}(\mathcal{T})),
  \]

- a mapping \( \lambda \) which assigns each association
  \[
  \langle r : \langle \text{role}_1 : C_1 \rangle, \ldots, \langle \text{role}_n : C_n \rangle \rangle \in V \text{ a relation}
  \]
  \[
  \lambda(r) \subseteq \mathcal{D}(C_1) \times \cdots \times \mathcal{D}(C_n)
  \]

(i.e. a set of type-consistent \( n \)-tuples of identities).
Q: Should it better be

\[ \lambda(c) \leq \text{dom}(c) \] 

(i.e. only alive objects participate in links)

A: choice of lecture: NO

\[ 1 \in C \xrightarrow{n} 2 \text{b}: D \quad \text{P = 5c} \]

(complete)

\( p \) in 2b:D is a dangling reference, 5c is maybe no longer alive.
Example

\[ \langle t \colon \langle \text{ids} : S, \ldots \rangle, \langle \text{sec} : S, \ldots \rangle, \langle \text{tv} : S, \ldots \rangle \rangle \]

\[ \sigma_1 : \{1_s \mapsto \{a, \text{null}, 1_s\}, 2_s \mapsto \{a, \text{null}, 6_s\}, 3_s \mapsto \{a, \text{null}, 7_s\}, 27_s \mapsto \{a, \text{null}, 7_s\} \} \]

\[ \lambda_1 : t \mapsto \{ (1_s, 3_s, 2_s), (1_s, 27_s, 3_s), (2_s, 5_s, 6_s), (3_s, 3_s, 3_s) \} \]  "students 5_s, 6_s left university (k)"

"one student playing all three roles"

"if (k) is not desired, add: context $S$ inv: ids $\neq$ sec and sec $\neq$ tv"
Associations and OCL
**OCL and Associations: Syntax**

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

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

Now becomes

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

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

**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

\[
\text{expr ::= } \ldots \ \mid \ \text{role}(\text{expr}_1) : \tau_C \rightarrow \tau_D \quad \mu = 0..1 \text{ or } \mu = 1 \\
\mid \ \text{role}(\text{expr}_1) : \tau_C \rightarrow \text{Set}(\tau_D) \quad \text{otherwise}
\]

if

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

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

- context Player inv: size(\text{year}) > 0
- \textbf{NOT}: context Player inv: size(\text{p}) > 0
- context Player inv: size(\text{season}) > 0
- \textbf{NOT}: context Player inv: size(\text{goalie}) > 0
**OCL and Associations: Semantics**

**Recall:** (Lecture 03)

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

\[
\begin{align*}
I[r_1(expr_1)](\sigma, \beta) &:= \begin{cases} 
  u, & \text{if } u_1 \in \text{dom}(\sigma) \text{ and } \sigma(u_1)(r_1) = \{u\} \\
  \bot, & \text{otherwise}
\end{cases} \\
I[r_2(expr_1)](\sigma, \beta) &:= \begin{cases} 
  \sigma(u_1)(r_2), & \text{if } u_1 \in \text{dom}(\sigma) \\
  \bot, & \text{otherwise}
\end{cases}
\end{align*}
\]

**Now needed:**

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

- We cannot simply write \( \sigma(u)(role) \).
  
  **Recall:** \( role \) is (for the moment) not an attribute of object \( u \) (not in \( atr(C) \)).

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

- \( role \) denotes the position of the \( D \)'s in the tuples constituting the value of \( r \).
Assume $\text{expr}_1 : \tau_C$ for some $C \in \mathcal{C}$. Set $u_1 := I[\text{expr}_1]((\sigma, \lambda), \beta) \in \mathcal{D}(\tau_C)$.

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

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

where

```
"database lookup"
```

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

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.
\[ \sigma = \{ 1 \mapsto \{ p \mapsto \{2\} \} \\
\{ n \mapsto \{ 3, 4 \} \}, \]
\[ 2 \mapsto \emptyset, \ n \mapsto \emptyset \} \]
OCL and Associations Example

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

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

\[
\sigma = \{1_C \mapsto \emptyset, 3_D \mapsto \{x \mapsto 1\}, 7_D \mapsto \{x \mapsto 2\}\} \\
\lambda = \{A.C.D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}
\]

\[ I[[self \cdot n]]((\sigma, \lambda), \{\text{self} \mapsto 1_C\}) = I[I[n(self)](\sigma, \lambda, \text{self} \mapsto 1_C)] \]

\[ = L(n)(I[I[n(self)](\sigma, \lambda, \text{self} \mapsto 1_C), \lambda) \]

\[ = L(n)(I[I[n(self)](\sigma, \lambda, \text{self} \mapsto 1_C), \lambda) \]

\[ = \{(1_C, 3_D), (1_C, 7_D)\} \cup 2 \]

\[ = \{(3_D, 7_D)\} \]
Associations: The Rest
Recapitulation: Consider the following association:

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

- **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.
References
References

