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: 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 $\langle v : \tau, \xi, \text{expr}_0, P_v \rangle$ with $\tau \in \mathcal{T}$ (as before),
- or is an association of the form

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

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:

\[
\langle r : \langle \text{role}_1 : \underline{P_1}, \underline{\ldots}, \underline{o_1} \rangle \rangle, \ldots, \langle \text{role}_n : \underline{P_n}, \underline{\ldots}, \underline{o_n} \rangle \rangle
\]

(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).
Recall: Links in System States

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

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 \( \mathcal{I} = (\mathcal{P}, \mathcal{E}, V, \text{atr}) \).

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

- A type-consistent mapping \( \sigma : \mathcal{D}(\mathcal{E}) \rightarrow (\text{atr}(\mathcal{E}) \rightarrow \mathcal{D}(\mathcal{P})) \),
- A mapping \( \lambda \) which assigns each association \( \langle r : \langle \text{role}_1 : C_1, \ldots, \rangle, \ldots, \langle \text{role}_n : C_n, \ldots, \rangle \rangle \in V \) 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).

Q: Should it better be \( \lambda(r) \subseteq \text{dom}(\sigma)^n \) ?

A: Choice of lecture: No

\( \varepsilon \) is a dangling reference, \( \varepsilon \) is maybe no longer alive.
Example

Let:

- \( t_1 \rightarrow \{a_1 \rightarrow \} \),
- \( t_2 \rightarrow \{a_2 \rightarrow \} \),
- \( t_3 \rightarrow \{a_3 \rightarrow \} \),
- \( t_4 \rightarrow \{a_4 \rightarrow \} \)

For any installation \( \lambda ' \),

\( \lambda ' \rightarrow \{4, 1, 2, 3\} \),
\( \lambda ' \rightarrow \{1, 2, 3, 4\} \),
\( \lambda ' \rightarrow \{2, 3, 4\} \)

If (x) is not desired, add:

\[ \text{context \& inv: let \( x = \text{sec} \) and sec \( x = a \) \]
**OCL and Associations: Syntax**

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

\[
expr ::= \ldots \mid r_1(expr_1) : \tau_C \rightarrow \tau_D \quad r_1 : D_{0,1} \in \text{atr}(C) \\
|r_2(expr_1) : \tau_C \rightarrow \text{Set}(\tau_D) \quad r_2 : D_0,1 \in \text{atr}(C)
\]

Now becomes

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

if

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

**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**

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

if

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

---

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

- context Player inv: size(year) > 0
- NOT: context Player inv: size(p) > 0
- context Player inv: size(season) > 0
- NOT: context Player inv: size(goalie) > 0

---
OCL and Associations: Semantics

Recall: (Lecture 03)

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

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

Now needed:

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

- We cannot simply write \( \sigma(u)(role) \).

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

OCL and Associations: Semantics Cont’d

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

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

where

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

if

\[ \langle r : \langle role_1 : \ldots, \ldots, \rangle, \ldots, \langle role_n : \ldots, \ldots, \rangle, \ldots \rangle, role = role_i \].

Given a set of \( n \)-tuples \( A \), \( A \downarrow i \) denotes the element-wise projection onto the \( i \)-th component.
OCL and Associations Example

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

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

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

\[
\lambda = \{A_C.D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}
\]

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

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

\[
= L(\sigma)(\{1_C, \lambda\})
\]

\[
= \{\{1_C, 3_D\}, \{1_C, 7_D\}\} \downarrow 2
\]

\[
= 2 \downarrow 3p.7p
\]
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** \( \xi \) and **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

