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

Associations in General

Recall: We consider associations of the following form:

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

Only these parts are relevant for extended system states:

$$\langle r : \langle role_1 : C_1, \_, P_1, \_, \_, \_\rangle, \dots, \langle role_n : C_n, \_, P_n, \_, \_, \_\rangle$$

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

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).

Links in System States

$$\langle r : \langle role_1 : C_1, \_, P_1, \_, \_, \_\rangle, \dots, \langle role_n : C_n, \_, P_n, \_, \_, \_\rangle$$

Only for the course of Lectures 9/10 we change the definition of system states:

Definition. Let $\mathscr{D}$ be a structure of the (extended) signature $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$.

A system state of $\mathscr{S}$ wrt. $\mathscr{D}$ is a pair (σ, λ) consisting of

- a type-consistent mapping

$$\sigma : \mathscr{D}(\mathscr{C}) \nrightarrow (atr(\mathscr{C}) \nrightarrow \mathscr{D}(\mathscr{T})),$$

- a mapping λ which assigns each association $\langle r : \langle role_1 : C_1\rangle, \dots, \langle role_n : C_n\rangle\rangle \in V$ a relation

$$\lambda(r) \subseteq \mathscr{D}(C_1) \times \dots \times \mathscr{D}(C_n)$$

(i.e. a set of type-consistent n-tuples of identities).

Association/Link Example

Signature:

$$\mathscr{S} = (\{Int\}, \{C, D\}, \{x : Int, \langle A\_C\_D : \langle c : C, 0..*, +, \{\mathtt{unique}\}, \times, 1\rangle, \langle n : D, 0..*, +, \{\mathtt{unique}\}, >, 0\rangle\rangle\}, \{C \mapsto \emptyset, D \mapsto \{x\}\})$$

A **system state** of $\mathscr{S}$ (some reasonable $\mathscr{D}$) is (σ, λ) with:

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

$$\lambda = \{A\_C\_D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$$

Extended System States and Object Diagrams

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:

$$expr ::= \dots \quad | \; r_1(expr_1) \;\; : \tau_C \to \tau_D \qquad r_1 : D_{0,1} \in atr(C)$$
$$| \; r_2(expr_1) \;\; : \tau_C \to Set(\tau_D) \qquad r_2 : D_* \in atr(C)$$

Now becomes

$$expr ::= \dots \quad | \; role(expr_1) \;\; : \tau_C \to \tau_D \qquad \mu = 0..1 \text{ or } \mu = 1$$
$$| \; role(expr_1) \;\; : \tau_C \to Set(\tau_D) \qquad \text{otherwise}$$

if there is

$$\langle r : \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots \rangle \in V \text{ or}$$
$$\langle r : \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots \rangle \in V, role \neq role'.$$

"C participates in assoc."

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

Example: C —c— D n(self): NO   n(self): OK

OCL and Associations: Syntax

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

$$expr ::= \dots \quad | \; r_1(expr_1) \;\; : \tau_C \to \tau_D \qquad r_1 : D_{0,1} \in atr(C)$$
$$| \; r_2(expr_1) \;\; : \tau_C \to Set(\tau_D) \qquad r_2 : D_* \in atr(C)$$

Now becomes

$$expr ::= \dots \quad | \; role(expr_1) \;\; : \tau_C \to \tau_D \qquad \mu = 0..1 \text{ or } \mu = 1$$
$$| \; role(expr_1) \;\; : \tau_C \to Set(\tau_D) \qquad \text{otherwise}$$

if

$$\langle r : \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots \rangle \in V \text{ or}$$
$$\langle r : \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots \rangle \in V, role \neq role'.$$

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

$$expr ::= \dots \quad | \; role(expr_1) \;\; : \tau_C \to \tau_D \qquad \mu = 0..1 \text{ or } \mu = 1$$
$$| \; role(expr_1) \;\; : \tau_C \to Set(\tau_D) \qquad \text{otherwise}$$

if

$$\langle r : \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots \rangle \in V \text{ or}$$
$$\langle r : \dots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \dots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \dots \rangle \in V, role \neq role'.$$

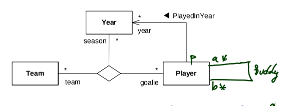


Figure 7.21 - Binary and ternary associations [OMG, 2007b, 44].

- context Player inv: size(year(self)) > 0 OK
- context Player inv: self.p -> size > 0 NOT OK
- context Player inv: self.season -> size > 0 OK
- context Player inv: self.b -> size > 0 and self.a -> size > 0 OK

$\{(1_P, 2_P), \dots\} = \lambda(buddy)$

OCL and Associations: Semantics

Recall: (Lecture 03)

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

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

Now needed:

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

- 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.

OCL and Associations: Semantics Cont'd

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

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

where

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

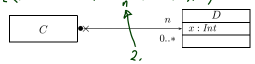
if

$$\langle r : \dots \langle role_1 : \_, \_, \_, \_, \_, \_ \rangle, \dots \langle role_n : \_, \_, \_, \_, \_, \_ \rangle, \dots \rangle, role = role_i.$$

Given a set of n-tuples A,
$A \downarrow i$ denotes the element-wise projection onto the i-th component.

OCL and Associations Example

$$I[\![role(expr_1)]\!]((\sigma, \lambda), \beta) := \begin{cases} L(role)(u_1, \lambda) & \text{, if } u_1 \in \mathrm{dom}(\sigma) \\ \bot & \text{, otherwise} \end{cases}$$
$$L(role)(u, \lambda) = \{(u_1, \dots, u_n) \in \lambda(r) \mid u \in \{u_1, \dots, u_n\}\} \downarrow i$$



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

$$\lambda = \{A\_C\_D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$$

$$I[\![self\ .\ n]\!]((\sigma, \lambda), \{self \mapsto 1_C\})$$

-
-

Associations: The Rest

The Rest

Recapitulation: Consider the following association:

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

- **Association name** r and **role names**/**types** $role_i/C_i$ induce extended system states λ.
- **Multiplicity** μ is considered in OCL syntax.
- **Visibility** ξ/**Navigability** ν: well-typedness.

Now the rest:

- **Multiplicity** μ: 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

[OMG, 2007a] OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.

[OMG, 2007b] OMG (2007b). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.