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:

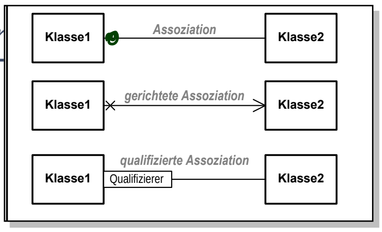
- Started to discuss "associations", the general case.

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



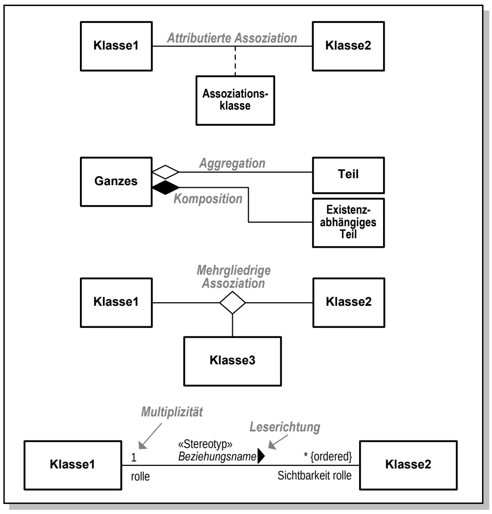
An **association** has

- 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 $\tau \in \mathscr{T}$ (as before),
- **or** is an **association** of the form

one assoc. end

$$\langle r : \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle,$$

assoc. name　role name　Class　set of prop.　navigability

$$\vdots$$

$$\langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle$$

ownership　multiplicity　visibility

where

- $n \geq 2$ (at least two ends),
- r, $role_i$ are just **names**,
- the **multiplicity** μ_i is an expression of the form

$$\mu ::= * \mid N \mid N..M \mid N..* \mid \mu, \mu \qquad (N, M \in \mathbb{N})$$

- 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:

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

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 $P_1 = P_n = \{\texttt{unique}\}$).

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

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

Only for the course of this lecture 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

  dom(σ) = alive objects

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

  object identities

  values for basic type attrs. only

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

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

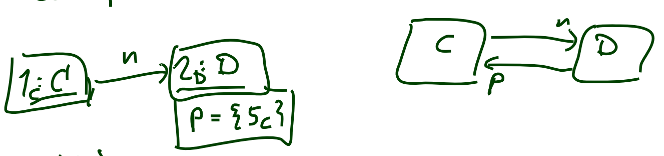
  (i.e. a set of type-consistent n-tuples of identities).

Q: should it better be

$$\lambda(r) \subseteq dom(\sigma)^n \;?$$

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

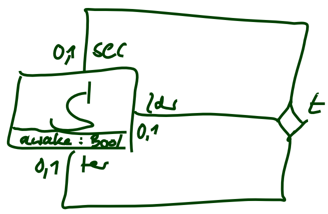
A: choice of lecture: NO



(complete)

↳ p in 2_D : D is a dangling reference, 5_C is maybe no longer alive

Example



$\langle t : \langle ldr : S, \dots \rangle,$
$\langle sec : S, \dots \rangle,$
$\langle ter : S, \dots \rangle \rangle$

$\sigma_1 : \{ 1_S \mapsto \{aw \mapsto 1\}, 2_S \mapsto \{aw \mapsto 0\}, 3_S \mapsto \{aw \mapsto 1\}, 27_S \mapsto \{aw \mapsto 1\} \}$

$\lambda_1 : t \mapsto \{ (1_S, 3_S, 2_S),$
$(1_S, 27_S, 3_S),$
$(2_S, 5_S, 6_S),$ // students $5_S, 6_S$ left university
$(3_S, 3_S, 3_S) \}$ // one student playing all three roles (*)

if (*) is ~~not~~ desired, add:
context S inv : ldr ≠ sec and sec ≠ ter

Associations and OCL

OCL and Associations: Syntax

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

$$\begin{array}{llll} expr ::= \dots & \mid r_1(expr_1) & : \tau_C \to \tau_D & r_1 : D_{0,1} \in atr(C) \\ & \mid r_2(expr_1) & : \tau_C \to Set(\tau_D) & r_2 : D_* \in atr(C) \end{array}$$

Now becomes

$$\begin{array}{llll} expr ::= \dots & \mid role(expr_1) & : \tau_C \to \tau_D & \mu = 0..1 \text{ or } \mu = 1 \\ & \mid role(expr_1) & : \tau_C \to Set(\tau_D) & \text{otherwise} \end{array}$$

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.

– 08 – 2011-12-06 – Sassocol –

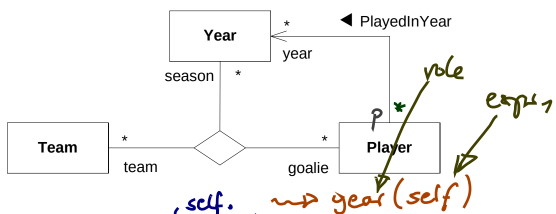
OCL and Associations Syntax: Example

$$\begin{array}{llll} expr ::= \dots & \mid \underline{role}(\underline{expr_1}) & : \tau_C \to \tau_D & \mu = 0..1 \text{ or } \mu = 1 \\ & \mid role(expr_1) & : \tau_C \to Set(\tau_D) & \text{otherwise} \end{array}$$

if

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

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



self: self.

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

– 08 – 2011-12-06 – Sassocol –

OCL and Associations: Semantics

Recall: (Lecture 03)

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

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

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

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

where

"database lookup" assume r is uniquely determined by role

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

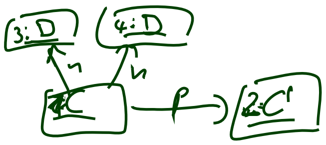
select those tuples where u occurs at some position project onto i-th comp.

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.

OLD:

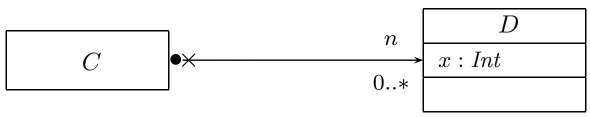


$$\sigma = \{1 \mapsto \{p \mapsto \{2\}\} \cup \{n \mapsto \{3,4\}\},$$
$$2 \mapsto \{p \mapsto \emptyset, n \mapsto \emptyset\}\}$$

OCL and Associations Example

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

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



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

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

$$I[\![self\ .\ n]\!]((\sigma,\lambda),\{self \mapsto 1_C\}) = I[\![n(self)]\!]((\sigma,\lambda),\{self \mapsto 1_C\})$$
$$= L(n)(I[\![self]\!]((\sigma,\lambda),\{self \mapsto 1_C\}),\lambda)$$
$$= L(n)(1_C,\lambda)$$
$$= (\{(1_C,3_D),(1_C,7_D)\}) \downarrow 2$$
$$= \{3_D, 7_D\}$$

Associations: The Rest

The Rest

Recapitulation: Consider the following association:

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

- **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.

References

References

[Ambler, 2005] Ambler, S. W. (2005). *The Elements of UML 2.0 Style*. Cambridge University Press.

[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.