Software Design, Modelling and Analysis in UML

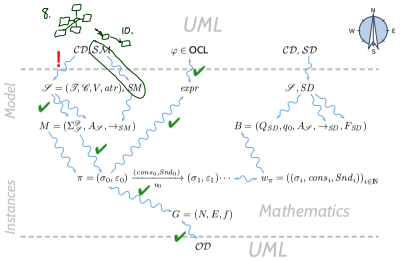
Lecture 8: Class Diagrams III

2016-11-24

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Course Map

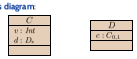


Content

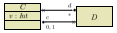
Recall: Plan & Extended Signature

Overview

- **Class diagram**:



Alternative presentation:



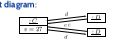
- **Signature**:

$\mathscr{S} = (\{Int\}, \{C, D\}, \{v : Int, d : D_x, e : C_{0,1}\}, \{C \mapsto \{v, d\}, D \mapsto \{e\}\})$

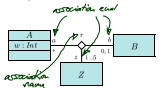
- **Example system state**:

$\sigma = \{1_C \mapsto \{v \mapsto 27, d \mapsto \{5_D, 7_D\}\}, 5_D \mapsto \{e \mapsto \{1_C\}\}, 7_D \mapsto \{e \mapsto \{1_C\}\}\}$

- **Object diagram**:



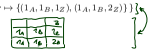
- **Class diagram** (with ternary association):



- **Signature**: extend again to represent
  - **association** r with
    - **association ends** a, b, and z (each with multiplicity, visibility, etc.)
- **Example system state**: (σ, λ)

$\sigma = \{1_A \mapsto \{w \mapsto 13\}, 1_B \mapsto \emptyset, 1_Z \mapsto \emptyset\}$

$\lambda = \{r \mapsto \{(1_A, 1_B, 1_Z), (1_A, 1_B, 2_Z)\}\}$



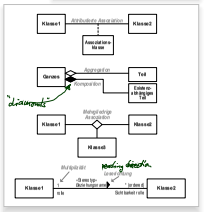
- **Object diagram**: No...

So, 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**,
- and possibly a **diamond**.

Wanted: places in the signature to represent the information from the picture.



Temporarily (Lecture 7 – 9) Extended Signature

Definition. An (Extended) Object System Signature (with Associations) is a quadruple $\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$ where

- ...
- each element of V is
  - either a basic type attribute $\langle v : T, \xi, expr_0, P_0 \rangle$ with $T \in \mathscr{T}$
  - or an association of the form

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

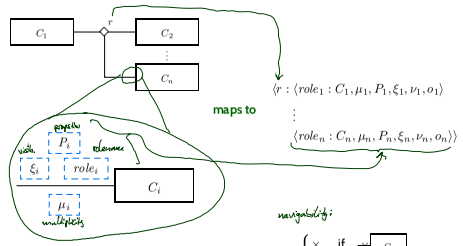
  (ends with multiplicity μ_i, properties P_i, visibility ξ_i, navigability ν_i, ownership o_i, $1 \le i \le n$)
- ...
- $atr : \mathscr{C} \to 2^{\{v \in V \mid v:T,\ T \in \mathscr{T}\}}$ maps classes to basic type (!) attributes.

In other words:

- only basic type attributes "belong" to a class (may appear in $atr(C)$),
- associations are not "owned" by a class (not in any $atr(C)$), but "live on their own".

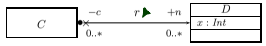
Associations in Class Diagrams

From Association Lines to Extended Signatures



$$o_i = \begin{cases} 1 & \text{, if } \to C_i \\ 0 & \text{, if } - C_i \end{cases} \qquad \nu_i = \begin{cases} \times & \text{, if } \to C_i \\ - & \text{, if } - C_i \\ > & \text{, if } \to C_i \end{cases}$$

Association Example



Signature:

$$\mathscr{S} = \Big(\{\{Int\}, \{C, D\}, \{\langle x : Int, +, \mathbb{M}, \emptyset \rangle, \langle r : \langle n : D, *, \{unique\}, +, \rangle, \langle c : C, 0..*, \{unique\}, -, \times, \gamma \rangle \rangle\}, \{C \mapsto \emptyset, D \mapsto \{x\}\} \Big)$$

What If Things Are Missing?

Most components of associations or association end may be omitted.
For instance (OMG, 2011b, 17), Section 6.4.2, proposes the following rules:

- **Name**: Use

$$A\_\langle C_1 \rangle\_\cdots\_\langle C_n \rangle$$

  if the name is missing.

  Example:

- **Reading Direction**: no default
- **Role Name**: use the class name at that end in lower-case letters

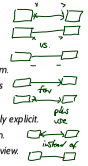
  Example:

  Other convention: (used e.g. by modelling tool Rhapsody)

What If Things Are Missing?

- **Multiplicity**: 1

  In my opinion, it's safer to assume $0..1$ or $*$ (for $0..*$) if there are no fixed, written, agreed conventions ("expect the worst").
- **Properties**: $\emptyset$ (in course: {unique})
- **Visibility**: public
- **Navigability and Ownership**: not so easy. (OMG, 2011b, 4.3)



"Various options may be chosen for showing navigation arrows on a diagram.
In practice, it is often convenient to suppress some of the arrows and crosses and just show exceptional situations:

- *Show all arrows and ×s: Navigation and its absence are made completely explicit.*
- *Suppress all arrows and ×s: No inference can be drawn about navigation.*
  This is similar to any situation in which information is suppressed from a view.
- *Suppress arrows for associations with navigability in both directions, and show arrows only for associations with one-way navigability.*
  In this case, the two-way navigability cannot be distinguished from situations where there is no navigation at all; however, the latter case occurs rarely in practice."

Wait, If Omitting Things...

- ...**is causing so much trouble** (e.g. leading to misunderstanding), why does the standard say "In practice, it is often convenient..."?

  Is it a good idea to trade convenience for **precision / unambiguity**?

  It depends.
  - Convenience as such is a **legitimate goal**.
  - In UML-As-Sketch mode, precision "doesn't matter", so convenience (for writer) can even be a primary goal.
  - In UML-As-Blueprint mode, precision is the primary goal. And misunderstandings are in most cases annoying.

    But: (even in UML-As-Blueprint mode)

    If all associations in your model have multiplicity $*$, then it's probably a good idea not to write all these $*$'s.

    So: tell the reader about your convention and leave out the $*$'s.

Associations: Semantics

Associations in General

Recall: We consider associations of the following form:

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

Only these parts are relevant for extended system states:

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

(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).

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 lectures 5 / 9 we change the definition of system states:

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

A system state of $\mathscr{S}$ wrt. $\mathscr{D}$ is a pair (σ, λ) consisting of (only basic type attributes here)
- a type-consistent mapping (as before)

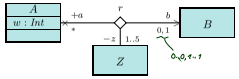
$$\sigma : \mathscr{D}(\mathscr{C}) \nrightarrow (atr(\mathscr{C}) \nrightarrow \mathscr{D}(\mathscr{T})),$$

- a mapping λ which maps each association $\langle r : \langle role_1 : C_1 \rangle, \ldots, \langle role_n : C_n \rangle \rangle \in V$ to 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).

Association / Link Example



Signature:

$\mathscr{S} = (\{Int\}, \{A, Z, B\}, \{ w : Int,$
$\langle r : \langle a : A, 0..*, +, \{unique\}\rangle, \times,$ $0 \rangle,$
$\langle z : Z, 1..5, -, \{unique\}\rangle, -,$ $0 \rangle,$
$\langle b : B, 0..1, r, \{unique\}\rangle, >,$ $0 > \},$
$\{A \mapsto \{w\},\ Z \mapsto \emptyset,\ B \mapsto \emptyset\}\ )$

System state:

$\sigma = \{ a_1 \mapsto \{w \mapsto 27\},$
$a_2 \mapsto \{w \mapsto 13\},$
$a_3 \mapsto \emptyset,$
$z_1 \mapsto \emptyset\},$
$z_2 \mapsto \emptyset,$
$z_3 \mapsto \emptyset,$
$z_4 \mapsto \emptyset,$
$z_5 \mapsto \emptyset,$
$b_1 \mapsto \emptyset\}$

$\in \mathscr{D}(A) \times \mathscr{D}(Z) \times \mathscr{D}(B)$

$\lambda = \{ r \mapsto \{ (a_1, z_3, b_1),$
$(a_1, z_2, z_1),$
$(a_1, z_2, z_5),$
$(a_2, z_3, b_1) \} \}$

$\sim$

a	z	b
a_1	z_2	z_1
a_1	z_2	z_5
a_1	z_3	b_1
a_2	z_3	b_1

$\lambda(r) = \{(z_2, z_3, z_4), (a_1, z_2, b_1)\}$

Associations and OCL

OCL and Associations: Syntax

Recall: OCL syntax as introduced in Lecture 3, interesting part:

$$\begin{array}{rlll} expr ::= \ldots & | \; r_1(expr_1) & : \tau_C \to \tau_D & r_1 : D_{0..1} \in atr(C) \\ & | \; r_2(expr_1) & : \tau_C \to Set(\tau_D) & r_2 : D_* \in atr(C) \end{array}$$

Now becomes

$$\begin{array}{rlll} expr ::= \ldots & | \; role(expr_1) & : \tau_C \to \tau_D & \mu = 0..1 \text{ or } \mu = 1..1 \\ & | \; role(expr_1) & : \tau_C \to Set(\tau_D) & \text{otherwise} \end{array}$$

if there is

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

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

Note:

- Association name as such **does not 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: Semantics

Recall:

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

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

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 $atr(C)$).
- What we have is $\lambda(r)$ (with association name r, not with role name $role$!).
  $$\langle r : \ldots, \langle role : D, \mu, \_, \_, \_, \_ \rangle, \ldots, \langle role' : C, \_, \_, \_, \_, \_ \rangle, \ldots \rangle$$
  But it yields a set of n-tuples, of which **some** relate u and 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 $u_1 := \mathcal{I}[\![expr_1]\!](\sigma, \beta) \in \mathscr{D}(T_C)$.

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

where

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

if

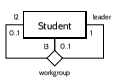
$$\langle r : \langle role_1 : \_, \_, \_, \_, \_ \rangle, \ldots \langle role_n : \_, \_, \_, \_, \_ \rangle, \rangle, \quad role = role_i$$

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

OCL and Associations Semantics: Example

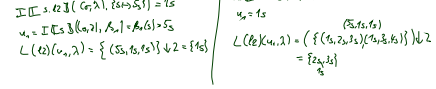
$$\mathcal{I}[\![role(expr_1)]\!]((\sigma, \lambda), \beta) := \begin{cases} u & \text{, if } u_1 \in \mathrm{dom}(\sigma) \text{ and } L(role)(u_1, \lambda) = \{u\} \\ \bot & \text{, otherwise} \end{cases}$$

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



allInstances$_{Student}$->
Exists($s \mid s.l2 = s.l3$)

$\lambda(\text{workgroup}) = \{(1_S, 2_S, 3_S), (1_S, 3_S, 4_S), (5_S, 1_S, 1_S)\}$



Associations: The Rest

The Rest

Recapitulation: Consider the following association:

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

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

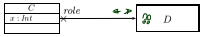
Now the rest:

- **Multiplicity** μ: we propose to view them as constraints.
- **Properties** P_i: even more typing.
- **Ownership** o: getting closer to pointers/references.
- **Diamonds**: exercise.

Navigability

Navigability is treated similar to visibility:
Using names of non-navigable association ends ($\nu = \times$) are **forbidden**.

Example: Given



the following OCL expression is **not well-typed** wrt. navigability,

$$\text{context } D \text{ inv} : role.x > 0$$

The standard says: navigation is...

- '–': ...possible
- '×': ...not possible
- '>': ...efficient

So: In general, UML associations **are different** from pointers / references in general!
But: Pointers / references **can faithfully** be modelled by UML associations.

Multiplicities as Constraints

Recall: Multiplicity is a term of the form $N_1..N_2, \ \ldots \ , N_{2k-1}..N_{2k}$
where $N_i \leq N_{i+1}$ for $1 \leq i \leq 2k$, $\quad N_1, \ldots, N_{2k-1} \in \mathbb{N}, \quad N_{2k} \in \mathbb{N} \cup \{*\}$.

Define $\mu^C_{\mathsf{OCL}}(role) :=$

$$\text{context } C \text{ inv} : (N_1 \leq role\text{->size()} \leq N_2) \text{ or } \ldots \text{ or } (N_{2k-1} \leq role\text{->size()} \underbrace{\leq N_{2k}}_{\text{omit if } N_{2k} = *})$$

for each $\langle r : \ldots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \ldots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \ldots\rangle \in V$ or

$$\langle r : \ldots, \langle role' : C, \_, \_, \_, \_, \_\rangle, \ldots, \langle role : D, \mu, \_, \_, \_, \_\rangle, \ldots\rangle \in V,$$

with $role \neq role'$, if $\mu \neq 0..1, \mu \neq 1..1$, and

$$\mu^C_{\mathsf{OCL}}(role) := \text{context } C \text{ inv} : \text{not(oclIsUndefined}(role))$$

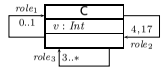
if $\mu = 1..1$.

Note: in n-ary associations with $n > 2$, there is redundancy.

Multiplicities as Constraints Example

$$\mu^C_{\mathsf{OCL}}(role) = \text{context } C \text{ inv} : (N_1 \leq role\text{->size()} \leq N_2) \text{ or } \ldots \text{ or } (N_{2k-1} \leq role\text{->size()} \leq N_{2k})$$

$\mathcal{CD}$:



- $[\text{context } C \text{ inv} : 4 \leq role_2\text{->size()} \leq 4 \text{ or } 17 \leq role_2\text{->size()} \leq 17]$
  $= [\text{context } C \text{ inv} : role_2\text{->size()} = 4 \text{ or } role_2\text{->size()} = 17]$
- $\not\supset [\text{context } C \text{ inv} : 3 \leq role_2\text{->size()}]$

Properties

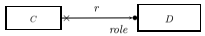
We don't want to cover association **properties** in detail,
only some observations (assume binary associations):

Property	Intuition	Semantical Effect
unique	one object has **at most one** r-link to a single other object	**current setting**
bag	one object may have **multiple** r-links to a single other object	have $\lambda(r)$ yield multi-sets
ordered, **sequence**	an r-link is a **sequence** of object identities (possibly including duplicates)	have $\lambda(r)$ yield sequences

Property	OCL Typing of expression $role(expr)$
unique	$T_D \to Set(T_C)$
bag	$T_D \to Bag(T_C)$
ordered, **sequence**	$T_D \to Seq(T_C)$

For **subsets**, **redefines**, **union**, etc. see (?, 127).

Ownership



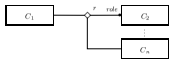
Intuitively it says:

> Association r is **not a "thing on its own"** (i.e. provided by λ),
> but association end '$role$' is **owned** by C (!).
> (That is, it's stored inside C object and provided by σ).

So: if multiplicity of $role$ is $0..1$ or $1..1$, then the picture above is very close to concepts of pointers/references.

Actually, ownership is seldom seen in UML diagrams. Again: if target platform is clear, one may well live without (cf. (OMG, 2011b, 42) for more details).

Not clear to me:



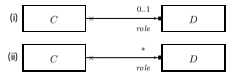
Back to the Main Track

Back to the main track:

Recall: on some earlier slides we said, the extension of the signature is only to study associations in "full beauty".
For the remainder of the course, we should look for something simpler...

Proposal:

- from now on, we only use associations of the form

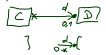


(And we may omit the non-navigability and ownership symbols)

- Form (i) introduces $role : C_{0,1}$, and form (ii) introduces $role : C_*$ in V.
- In both cases, $role \in atr(C)$.
- We drop λ and go back to our nice σ with $\sigma(u)(role) \subseteq \mathcal{P}(D)$.

Tell Them What You've Told Them...

- From class diagrams with (general) **associations**, we obtain **extended signatures**.
- Links (instances of associations) "live on their own" in the λ in extended system states (σ, λ).
- OCL considers **role names**, the **semantics** is (more or less) **straightforward**.
- **The Rest**:
  - **navigability** is treated like visibility.
  - view **multiplicities** as shorthand for **constraints**.
  - properties, ownership, "diamonds": exist.
- **Back to the main track**:
  For simplicity, let's restrict the following discussion to $\underline{C_{0,1}}$ and $\underline{C_*}$ as before (now viewed as abbreviations for particular associations).



References

References

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.