Software Design, Modelling and Analysis in UML

Lecture 06: Class Diagrams I

2014-11-11

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Course Map
Contents & Goals

Last Lecture:
- OCL Semantics
- Object Diagrams

This Lecture:
- Educational Objectives: Capabilities for following tasks/questions.
  - What is a class diagram?
  - For what purposes are class diagrams useful?
  - Could you please map this class diagram to a signature?
  - Could you please map this signature to a class diagram?

- Content:
  - Final notes on object diagrams.
  - Study UML syntax.
  - Prepare (extend) definition of signature.
  - Map class diagram to (extended) signature.
  - Stereotypes – for documentation.

The Other Way Round
If we only have a picture as below, we typically assume that it's meant to be an object diagram wrt. some signature and structure.

\[ u_1 : C \quad \quad u_2 : C \quad \quad u_3 : D \]

In the example, we can conclude that the author is referring to some signature \( \mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{V}, \text{atr}) \) with at least

- \( \{ C, D \} \subseteq \mathcal{C} \)
- \( T \subseteq \mathcal{T} \)
- \( \{ x : C, p, \zeta, \omega \in T \} \subseteq \mathcal{V} \)
- \( \{ x \} \subseteq \text{at}(C) \)
- \( \{ p, z \} \subseteq \text{at}(D) \)

and a structure with

- \( \emptyset \subseteq \text{O}(T) \)
- \( \{ u, v, z \} \subseteq \text{O}(C) \)
- \( \{ u \} \subseteq \text{O}(D) \)

Example: Object Diagrams for Documentation
Example: Data Structure [Schumann et al., 2008]

Example: Illustrative Object Diagram [Schumann et al., 2008]
**UML Class Diagram Syntax** [Oestereich, 2006]
### What Do We (Have to) Cover?

**A class**
- has a set of **stereotypes**,
- has a **name**,
- belongs to a **package**,  
- can be **abstract**,  
- can be **active**,  
- has a set of **operations**,  
- has a set of **attributes**.

Each **attribute** has
- a **visibility**,  
- a **name**, a **type**,  
- a **multiplicity**, an **order**  
- an **initial value**, and  
- a set of **properties**, such as **readOnly**, **ordered**, etc.

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

\[ \mathcal{I} = (\mathcal{F}, \mathcal{C}, V, \text{atr}) \]

where

- (basic) **types** \( \mathcal{F} \) and **classes** \( \mathcal{C} \), (both finite),
- **typed attributes** \( V, \tau \) from \( \mathcal{F} \) or \( C_{0,1} \) or \( C_s \), \( C \in \mathcal{C} \),
- \( \text{atr} : \mathcal{C} \rightarrow 2^V \) mapping classes to attributes.

**Too abstract** to represent class diagram, e.g. no “place” to put class **stereotypes** or attribute **visibility**.

So: **Extend** definition for classes and attributes: Just as attributes already have types, we will assume that

- classes have (among other things) **stereotypes** and
- attributes have (in addition to a type and other things) a **visibility**.

**Extended Classes**

From now on, we assume that each class \( C \in \mathcal{C} \) has:

- a finite (possibly empty) set \( S_C \) of **stereotypes**,
- a boolean flag \( a \in \mathbb{B} \) indicating whether \( C \) is **abstract**, \( \text{(active if abstract)} \)
- a boolean flag \( t \in \mathbb{B} \) indicating whether \( C \) is **active**.

We use \( S_{\mathcal{C}} \) to denote the set \( \bigcup_{C \in \mathcal{C}} S_C \) of stereotypes in \( \mathcal{I} \).

(Alternatively, we could add a set \( S t \) as 5-th component to \( \mathcal{I} \) to provides the stereotypes (names of stereotypes) to choose from. But: too unimportant to care.)

**Convention:**

- We write
  \[ \langle C, S_C, a, t \rangle \in \mathcal{C} \]
  when we want to refer to all aspects of \( C \).
- If the new aspects are irrelevant (for a given context),
  we simply write \( C \in \mathcal{C} \) i.e. old definitions are still valid.
Extended Attributes

- From now on, we assume that each attribute $v \in V$ has (in addition to the type):
  - a visibility
    $$\xi \in \{\text{public, private, protected, package}\}$$
  - an initial value $expr_0$ given as a word from language for initial values, e.g. OCL expressions.
    (If using Java as action language (later) Java expressions would be fine.)
  - a finite (possibly empty) set of properties $P_v$.
    We define $P_v$ analogously to stereotypes.

Convention:
- We write $\langle v : \tau, \xi, expr_0, P_v \rangle \in V$ when we want to refer to all aspects of $v$.
- Write only $v : \tau$ or $v$ if details are irrelevant.

And?

- Note:
  All definitions we have up to now principally still apply as they are stated in terms of, e.g., $C \in \mathcal{C}$ — which still has a meaning with the extended view.
  For instance, system states and object diagrams remain mostly unchanged.

- The other way round: most of the newly added aspects don’t contribute to the constitution of system states or object diagrams.

- Then what are they useful for...
  - First of all, to represent class diagrams.
  - And then we’ll see.
From Class Boxes to Extended Signatures

A class box $n$ induces an (extended) signature class as follows:

$$V(n) := \{(v_1 : \tau_1, \xi_1, v_0, \{P_{1,1}, \ldots, P_{1,m_1}\}), \ldots, (v_\ell : \tau_\ell, \xi_\ell, v_0, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\})\}$$

$\text{attr}(n) := \{C \mapsto \{v_1, \ldots, v_\ell\}\}$

where

- "abstract" is determined by the font:
  $$a(n) = \begin{cases} 
  \text{true} & \text{if } n = \boxed{C} \text{ or } n = \boxed{C(n)} \\
  \text{false} & \text{otherwise} 
  \end{cases}$$

- "active" is determined by the frame:
  $$t(n) = \begin{cases} 
  \text{true} & \text{if } n = \boxed{C} \text{ or } n = \boxed{C} \\
  \text{false} & \text{otherwise} 
  \end{cases}$$

