Software Design, Modelling and Analysis in UML

Lecture 6: Class Diagrams I

2015-11-12

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany
$\mathcal{I} = (\mathcal{T}, \mathcal{C}, V, atr), \ SM$

$M = (\Sigma_{\mathcal{I}}, A_{\mathcal{I}}, \rightarrow_{SM})$

$\pi = (\sigma_0, \varepsilon_0) \xrightarrow{(\text{cons}_0, \text{Snd}_0)} u_0 (\sigma_1, \varepsilon_1) \cdots$

$G = (N, E, f)$

$CD, SM$

$\varphi \in \text{OCL}$

$expr$

$CD, SD$

$B = (Q_{SD}, q_0, A_{\mathcal{I}}, \rightarrow_{SD}, F_{SD})$

$w_\pi = ((\sigma_i, \text{cons}_i, \text{Snd}_i))_{i \in \mathbb{N}}$

$\mathcal{O}D$

$\mathcal{I}, SD$

$\mathcal{I}, \mathcal{S}D$

$\mathcal{U}ML$
Contents & Goals

Last Lecture:

- Object Diagrams
  - partial vs. complete; for analysis; for documentation...

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:
  - Study UML syntax.
  - Prepare (extend) definition of signature.
  - Map class diagram to (extended) signature.
  - Stereotypes.
UML Class Diagrams: Stocktaking
$\mathcal{I} = (\mathcal{T}, \mathcal{C}, V, atr)$ where

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

Example:

$\mathcal{I}_2 = (\{\text{U, MyType}\}, \{C, \Box\}, \{x \in \text{U}, x : \text{MyType}, p : \Box, q : C_0, 3\},$

$\{C \mapsto \{p\},$

$\Box \mapsto \{x, p\}\}$

looks like a class diagram...
That’d Be Too Simple

\[
\left\langle \text{Stereotype}_1, \ldots, \text{Stereotype}_n \right\rangle
\]

Package::C

+ \( r : \mathbb{C}_{0,1} = expr \)

\( s : \mathbb{D}_* \{\text{ordered}\} \)

- \( v : \mathbb{I} = 27 \)
  
  \( w : \mathbb{F} \{\text{readOnly}\} \)

\[
\begin{align*}
A & \quad \{A\} \\
y : \mathbb{I} & \\

\end{align*}
\]

\[
\begin{align*}
D & \\
x : \mathbb{I} & \\

\end{align*}
\]

\[
\begin{align*}
B & \\

\end{align*}
\]
What Do We Want / 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 attributes, ✓
- has a set of operations. (✓)

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

- $\mathcal{T}$ is a set of (basic) types,
- $\mathcal{C}$ is a finite set of classes $\langle C, S_C, a, t \rangle$ where
  - $S_C$ is a finite (possibly empty) set of stereotypes,
  - $a \in \mathbb{B}$ is a boolean flag indicating whether $C$ is abstract,
  - $t \in \mathbb{B}$ is a boolean flag indicating whether $C$ is active,
- $V$ is a finite set of attributes $\langle v : T, \xi, expr_0, P_v \rangle$ where
  - $T$ is a type from $\mathcal{T}$, or $C_{0,1}, C_*$ for some $C \in \mathcal{C}$,
  - $\xi \in \{\text{public}, \text{private}, \text{protected}, \text{package}\}$ is the visibility,
  - an initial value expression $expr_0$ given as a word from a language for initial value expressions, e.g. OCL, or C++ in the Rhapsody tool,
  - a finite (possibly empty) set of properties $P_v$.
- $atr : \mathcal{C} \rightarrow 2^V$ maps each class to its set of attributes.

We use $S_\mathcal{C}$ to denote the set $\bigcup_{C \in \mathcal{C}} S_C$ of stereotypes in $\mathcal{S}$. 
Conventions

- We write $\langle C, S_C, a, t \rangle$ if we want to refer to all aspects of $C$.
- If the new aspects are irrelevant (for a given context), we simply write $C$, i.e., old definitions are still valid.
- We write $\langle v : T, \xi, expr_0, P_v \rangle$ if we want to refer to all aspects of $v$.
- Write only $v : T$ or $v$ if details are irrelevant.
- **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 will remain mostly unchanged.
- The other way round: most of the newly added aspects **do not contribute** to the constitution of system states or object diagrams.
Mapping UML Class Diagrams to Extended Signatures
A class box $n$ induces an (extended) signature class as follows:

$$n: \langle \langle S_1, \ldots, S_k \rangle \rangle$$

$$\xi_1 v_1 : T_1 = \text{expr}_0 \{P_{1,1}, \ldots, P_{1,m_1}\}$$

$$\vdots$$

$$\xi_\ell v_\ell : T_\ell = \text{expr}_\ell \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}$$

$$\Downarrow$$

$$C(n) := \langle C, \{S_1, \ldots, S_k\}, a(n), t(n) \rangle$$

$$V(n) := \{\langle v_1 : T_1, \xi_1, \text{expr}_0, \{P_{1,1}, \ldots, P_{1,m_1}\}\rangle, \ldots, \langle v_\ell : T_\ell, \xi_\ell, \text{expr}_\ell, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}\rangle\}$$

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

where

- "abstract" is determined by the font:

$$a(n) = \begin{cases} 
true & \text{if } n = \boxed{C} \text{ or } n = \boxed{C \{A\}} \\
false & \text{otherwise}
\end{cases}$$

- "active" is determined by the frame:

$$t(n) = \begin{cases} 
true & \text{if } n = \boxed{C} \text{ or } n = \boxed{C} \\
false & \text{otherwise}
\end{cases}$$
Example

\[
\langle \langle S_1, \ldots, S_k \rangle \rangle \\
C
\]

\[
\xi_1 \; v_1 : T_1 = \text{expr}_1 \{P_{1,1}, \ldots, P_{1,m_1}\} \\
\vdots \\
\xi_\ell \; v_\ell : T_\ell = \text{expr}_\ell \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}
\]

\[
\Downarrow

C(n) := \langle C, \{S_1, \ldots, S_k\}, a(n), t(n) \rangle
\]

\[
V(n) := \{ \langle v_1 : T_1, \xi_1, \text{expr}_1 \{P_{1,1}, \ldots, P_{1,m_1}\} \rangle, \ldots, \\
\langle v_\ell : T_\ell, \xi_\ell, \text{expr}_\ell \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\} \rangle \}
\]

\[
atr(n) := \{ C \mapsto \{v_1, \ldots, v_\ell\} \}
\]

\[
C(n) = \langle C, \{\text{Stereotype}_1, \ldots, \text{Stereotype}_n\}, \text{false}, \text{false} \rangle
\]

\[
V(n) = \{ \langle r : C_0, \text{true}, \text{expr}, \emptyset \rangle, \\
\langle v : \text{Int}, \text{false}, 27, \emptyset \rangle, \\
\langle s : \text{D}_*, \emptyset \rangle \}
\]
It depends.

- What does the standard say? (OMG, 2011a, 121)
  
  "Presentation Options. The type, visibility, default, multiplicity, property string may be suppressed from being displayed, even if there are values in the model."

- **Visibility**: There is no “no visibility” — an attribute has a visibility in the (extended) signature.
  Some (and we) assume public as default, but conventions may vary.

- **Initial value**: some assume it given by domain (such as “leftmost value“, but what is “leftmost” of $\mathbb{Z}$?).
  Some (and we) understand non-deterministic initialisation if not given.

- **Properties**: probably safe to assume $\emptyset$ if not given at all.
Example Cont’d

\[ \langle S_1, \ldots, S_k \rangle \]

\begin{align*}
C & \ni \\
\xi_1 v_1 & : T_1 = \text{expr}_0^1 \{ P_{1,1}, \ldots, P_{1,m_1} \} \\
& \vdots \\
\xi_\ell v_\ell & : T_\ell = \text{expr}_0^\ell \{ P_{\ell,1}, \ldots, P_{\ell,m_\ell} \}
\end{align*}

\[ \Downarrow \]

\[ C(n) := \langle C, \{ S_1, \ldots, S_k \}, a(n), t(n) \rangle \]

\[ V(n) := \{ \langle v_1 : T_1, \xi_1, \text{expr}_0^1, \{ P_{1,1}, \ldots, P_{1,m_1} \} \rangle, \ldots, \]
\[ \langle v_\ell : T_\ell, \xi_\ell, \text{expr}_0^\ell, \{ P_{\ell,1}, \ldots, P_{\ell,m_\ell} \} \rangle \} \]

\[ \text{atr}(n) := \{ C \mapsto \{ v_1, \ldots, v_\ell \} \} \]

\[ \langle s : D_*^\# +, ? \{ \text{ordered} \} \rangle \]

\[ \Downarrow \]

\[ \langle s : D_\ast^\#, +, \emptyset \{ \text{ordered} \} \rangle \]
From Class Diagrams to Extended Signatures

- We view a **class diagram** $CD$ as a graph with nodes $\{n_1, \ldots, n_N\}$ (each “class rectangle” is a node).

- $C(CD) := \{C(n_i) \mid 1 \leq i \leq N\}$

- $V(CD) := \bigcup_{i=1}^{N} V(n_i)$

- $atr(CD) := \bigcup_{i=1}^{N} atr(n_i)$

- In a **UML model**, we can have **finitely many** class diagrams,

$$CD = \{CD_1, \ldots, CD_k\},$$

which **induce** the following signature:

$$\mathcal{I}(CD) = \left( \mathcal{T}, \bigcup_{i=1}^{k} C(CD_i), \bigcup_{i=1}^{k} V(CD_i), \bigcup_{i=1}^{k} atr(CD_i) \right).$$

(Assuming $\mathcal{T}$ given. In “reality” (i.e. in full UML), we can introduce types in class diagrams, the class diagram then contributes to $\mathcal{T}$. Example: enumeration types.)
Is the Mapping a Function?

**Question:** Is $S(CD)$ well-defined?
References
References


