Software Design, Modelling and Analysis in UML

Lecture 6: Class Diagrams I

2016-11-15

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Course Map

\[ M = (\Sigma, A_{SD}, \rightarrow_{SD}) \]

\[ \phi \in \text{OCL} \]

\[ \pi = (\sigma_0, \epsilon_0) \]

\[ w_0 \]

\[ w_1 \]

\[ \cdots \]

\[ w_n = ((\sigma_i, cons_i, Snd_i))_{i \in \mathbb{N}} \]

\[ G = (N, E, f) \]

\[ \text{UML} \]

\[ \text{Model} \]

\[ \text{Instances} \]

\[ \text{Mathematics} \]

\[ \text{OD} \]

\[ \text{CD, SM} \]

\[ \text{CD, SD} \]
Content

- Stocktaking
- Extended Signatures
- Structures for Extended Signatures
- Semantically Relevant
- Mapping Class Diagrams to Extended Signatures
- What if things are missing?
- (Temporary) Abbreviations
- Stereotypes

UML Class Diagrams: Stocktaking
Recall: Signature vs. Class Diagram

Basic Object System Signature Another Example

\( \mathcal{F} = (T, V, \text{atr}) \) where

- (basic) types \( T \) and classes \( V \) (both finite),
- typed attributes \( V \) from \( T \), or \( C_{0_1}, C_{0_2}, \) or \( C_{0_3} \), for some \( C \in V \),
- \( \text{atr} : V \rightarrow o^2 \) mapping classes to attributes.

Example:

\[ \mathcal{F}_1 = \langle T_1, T_2, T_3, C_1, C_2, C_3, \text{expr} \rangle \]

\[ \mathcal{F}_2 = \langle T_1, T_2, T_3, C_1, C_2, C_3, \text{expr} \rangle \]

That’d Be Too Simple
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 \( \mathcal{S} = (T, C, V, atr) \) where

- \( T \) is a set of (basic) types,
- \( C \) is a finite set of classes \( (C, S_C, a, t) \) where
  - \( S_C \) is a finite (possibly empty) set of stereotypes,
  - \( a \in \mathbb{B} \) is a boolean flag indicating whether \( C \) is abstract, \( a \) is if \( C \) is abstract
  - \( t \in \mathbb{B} \) is a boolean flag indicating whether \( C \) is active,
- \( V \) is a finite set of attributes \( \langle v : T, \xi, \text{expr}0, P_v \rangle \) where
  - \( T \) is a type from \( T \) or \( C_0, C_1, C_\ast \) for some \( C \in C \),
  - \( \xi \in \{ \text{public}, \text{private}, \text{protected}, \text{package} \} \) is the visibility,
  - an initial value expression \( \text{expr}_0 \) given as a word from a language for initial value expressions, e.g. OCL or C++ in the Rhapsody tool; write ‘\( \text{⊲} \)’ to explicitly not give an initial value expression.
  - a finite (possibly empty) set of properties \( P_v \).
- \( atr : C \rightarrow 2^V \) maps each class to its set of attributes.

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

Extended Signature Example

\[ \mathcal{S} = (T, C, V, atr), \quad (C, S_C, a, t) \in C, \quad (v : T, \xi, \text{expr}_0, P_v) \in V \]

\( \mathcal{S} = (\{ \text{Int, Float} \}, \{ (\text{Package} : C, \{ \text{Stereotype}_1, \ldots, \text{Stereotype}_n \}, 0, 0), \}

\( (A, 0, 1, 0), \quad (B, 0, 1, 0), \quad (D, 0, 0, 1) \},

\( \{ (r : \text{Package} : C_0, 1, +, \text{expr}, 0), \quad (s : D_\ast, +, \text{ordered}), \}

\( (v : \text{Int}, -2\,7, 0), \quad (w : \text{Float}, +, \text{readOnly}, \{ \text{ordered} \}), \}

\( (x : \text{Int}, +, \text{ordered}, 0), \quad (y : \text{Int}, +, \text{readOnly}, 0) \}, \)

\( \{ \text{Package} \mapsto \{ r, s, v, w \}, \quad A \mapsto \{ y \}, \quad B \mapsto \emptyset, \quad D \mapsto \{ x \} \} \)
Conventions

- We write $\langle C, S_C, a, t \rangle$ if we want to refer to all aspects of class $C$.
- If the new aspects are irrelevant (for a given context), we simply write $C$, i.e., old definitions (written in terms of $C$) are still valid.
- Similarly, we write $\langle v : T, \xi, expr, P_v \rangle$ if we want to refer to all aspects of attribute $v$.
- Write only $v : T$ or $v$ if details are irrelevant.

Structures of Extended Signatures

Recall:

Definition. A Basic Object System Structure of a Basic Object System Signature $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \mathcal{V}, \text{atr})$ is a domain function $\mathcal{D}$ which assigns to each type a domain, i.e.

- $\tau \in \mathcal{T}$ is mapped to $\mathcal{D}(\tau)$,
- $C \in \mathcal{C}$ is mapped to an infinite set $\mathcal{D}(C)$ of (object) identities.

Note: Object identities only have the "$=$" operation.

- Sets of object identities for different classes are disjoint, i.e.
  \[ \forall C, D \in \mathcal{C} : C \neq D \rightarrow \mathcal{D}(C) \cap \mathcal{D}(D) = \emptyset. \]

- $C^*$ and $C_{0,1}$ for $C \in \mathcal{C}$ are mapped to $2^{\mathcal{D}(C)}$.

We use $\mathcal{D}(\mathcal{C})$ to denote $\bigcup_{C \in \mathcal{C}} \mathcal{D}(C)$; analogously $\mathcal{D}(\mathcal{V})$. 
Structures of Extended Signatures

New:

Definition. An (Object System) Structure of an (Extended Object System) Signature \( \mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr) \) is a domain function \( D \) which assigns to each type a domain, i.e.

- \( \tau \in \mathcal{T} \) is mapped to \( D(\tau) \),
- \( C \in \mathcal{C} \) is mapped to an infinite set \( D(C) \) of (object) identities.

Note: Object identities only have the "\( = \)" operation.

- Sets of object identities for different classes are disjoint, i.e.
  \[ \forall C, D \in \mathcal{C} : C \neq D \rightarrow D(C) \cap D(D) = \emptyset. \]
- \( C, \) and \( C_{0,1} \) for \( C \in \mathcal{C} \) are mapped to \( 2^{D(C)} \).

We use \( D(\mathcal{C}) \) to denote \( \bigcup_{C \in \mathcal{C}} D(C) \); analogously \( D(\mathcal{C}_*) \).

System States of Extended Signatures

Recall:

Definition. Let \( D \) be a basic structure of basic signature \( \mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr) \). A system state of \( \mathcal{S} \) wrt. \( D \) is a type-consistent mapping

\[ \sigma : D(\mathcal{C}) \rightarrow (V \rightarrow (D(\mathcal{T}) \cup D(\mathcal{C}_*))). \]

That is, for each \( u \in D(C), C \in \mathcal{C}, \) if \( u \in \text{dom}(\sigma) \)

- \( \text{dom}(\sigma(u)) = atr(C) \)
- \( \sigma(u)(v) \in D(\tau) \) if \( v : \tau, \tau \in \mathcal{T} \)
- \( \sigma(u)(v) \in D(D_*) \) if \( v : D_{0,1} \) or \( v : D_* \) with \( D \in \mathcal{C} \)

We call \( u \in D(\mathcal{C}) \) alive in \( \sigma \) if and only if \( u \in \text{dom}(\sigma) \).

We use \( \Sigma_D^{\mathcal{S}} \) to denote the set of all system states of \( \mathcal{S} \) wrt. \( D \).
**System States of Extended Signatures**

**New:**

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

A **system state** of $\mathcal{S}$ wrt. $\mathcal{D}$ is a **type-consistent** mapping

$$\sigma : \mathcal{D}(\mathcal{C}) \rightarrow (V \rightarrow (\mathcal{D}(\mathcal{T}) \cup \mathcal{D}(\mathcal{C})))$$

That is, for each $u \in \mathcal{D}(\mathcal{C})$, $C \in \mathcal{C}$, if $u \in \text{dom}(\sigma)$

- $\text{dom}(\sigma(u)) = atr(C)$
- $\sigma(u)(v) \in \mathcal{D}(\tau) \text{ if } v: \tau, \tau \in \mathcal{T}$
- $\sigma(u)(v) \in \mathcal{D}(D_*) \text{ if } v: D_{0,1} \text{ or } v: D_*, \text{ with } D \in \mathcal{C}$
- $\forall \langle C, S_C, I, t \rangle \in \mathcal{C} \cdot \text{dom}(\sigma) \cap \mathcal{D}(C) = \emptyset$

We call $u \in \mathcal{D}(\mathcal{C})$ **alive** in $\sigma$ if and only if $u \in \text{dom}(\sigma)$.

We use $\Sigma_{\mathcal{D}} \mathcal{S}$ to denote the set of all system states of $\mathcal{S}$ wrt. $\mathcal{D}$.

**Semantical Relevance**

- The semantics (or meaning) of an extended object system signature $\mathcal{S}$ wrt. a structure $\mathcal{D}$ is the set of system states $\Sigma_{\mathcal{D}} \mathcal{S}$.
- The semantics (or meaning) of an extended object system signature $\mathcal{S}$ is the set of sets of system states wrt. some structure of $\mathcal{S}$, i.e. the set

$$\{ \Sigma_{\mathcal{D}} \mathcal{S} \mid \mathcal{D} \text{ is structure of } \mathcal{S} \}$$

Which of the following aspects is semantically relevant, i.e. does contribute to the constitution of system states?

**A class**

- has a set of stereotypes, $\times$
- has a name, $\checkmark$
- can be abstract, $\checkmark$
- can be active, $\times$
- has a set of attributes, $\checkmark$
- has a set of operations (later).

**Each attribute has**

- a visibility, $\times$
- a name, a type, $\checkmark$
- a multiplicity, an order, $\times$
- an initial value, and $\times$
- a set of properties, (such as readOnly, ordered, etc.)
From Class Boxes to Extended Signatures

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

\[
\begin{align*}
C(n) &:= \langle C, \{S_1, \ldots, S_k\}, a(n), t(n) \rangle \\
V(n) &:= \{ \langle v_1 : T_1, \xi_1, \text{expr}_1, \{P_{1,1}, \ldots, P_{1,m_1}\} \rangle, \ldots, \langle v_\ell : T_\ell, \xi_\ell, \text{expr}_\ell, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\} \rangle \}
\end{align*}
\]

where

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

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

\[
\begin{align*}
\langle S_1, \ldots, S_k \rangle & \mapsto C \\
\xi_1 & : T_1 = \text{expr}(P_{11}, \ldots, P_{1m_1}) \\
\vdots & \\
\xi_k & : T_k = \text{expr}(P_{k1}, \ldots, P_{km_k}) \\
\end{align*}
\]

\[
C(n) := (C, \{S_1, \ldots, S_k\}, a(n), t(n)) \\
V(n) := \{(v_1 : T_1, \xi_1, \text{expr}^0(P_{11}, \ldots, P_{1m_1})) \ldots, \ (v_\ell : T_\ell, \xi_\ell, \text{expr}^0(P_{\ell1}, \ldots, P_{\ellm_\ell}))\} \\
\text{attr}(n) := \{C \mapsto \{v_1, \ldots, v_\ell\}\}
\]

\[
\mathcal{O} = \{\{\alpha, \text{Float}\}, \langle \text{Type}, \ldots, \text{Type}\rangle, 0, 0\}, \langle A, \emptyset, 1, 0\rangle, \langle B, \emptyset, 1, 0\rangle, \langle D, \emptyset, 0, 1\rangle, \frac{1}{2} < r, \langle \text{std}, \emptyset, \emptyset \rangle, \langle c, \text{readOnly}, \emptyset \rangle, \langle d, \text{unordered}, \emptyset \rangle, \langle y, \text{Int}, \emptyset, \emptyset \rangle, \langle z, \text{Int}, \emptyset, \emptyset \rangle, \langle x, \text{Int}, \emptyset, \emptyset \rangle, \langle \text{Package}, 1, \text{Int}, \text{Int}, \text{Int}, \text{Int}, \emptyset, \emptyset, \emptyset, D \mapsto \emptyset, \emptyset \mapsto \emptyset \}\}
\]

What If Things Are Missing?

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.
From Class Diagrams to Extended Signatures

- We view a class diagram $\mathcal{CD}$ as a graph with nodes $\{n_1, \ldots, n_N\}$ (each "class rectangle" is a node).
- $\mathcal{E}(\mathcal{CD}) := \{C(n_i) \mid 1 \leq i \leq N\}$
- $V(\mathcal{CD}) := \bigcup_{i=1}^{N} V(n_i)$
- $\text{atr}(\mathcal{CD}) := \bigcup_{i=1}^{N} \text{atr}(n_i)$

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

$$\mathcal{F} = \{\mathcal{CD}_1, \ldots, \mathcal{CD}_k\},$$

which induce the following signature:

$$\mathcal{F}(\mathcal{F}) = \left( \mathcal{F}, \bigcup_{i=1}^{k} \mathcal{E}(\mathcal{CD}_i), \bigcup_{i=1}^{k} V(\mathcal{CD}_i), \bigcup_{i=1}^{k} \text{atr}(\mathcal{CD}_i) \right).$$

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

Question: Is $\mathcal{F}(\mathcal{C}, \mathcal{D})$ well-defined?

There are two possible sources for problems:

(1) A class $C$ may appear in multiple class diagrams:

(i)

\[
\begin{array}{ll}
\mathcal{C}D_1 & \mathcal{C}D_2 \\
C & C \\
v : \text{Int} = 0 & w : \text{Int} = 27
\end{array}
\]

(ii)

\[
\begin{array}{ll}
\mathcal{C}D_1 & \mathcal{C}D_2 \\
C & C \\
v : \text{Int} = 0 & v : \text{Int} = 27
\end{array}
\]

Simply forbid the case (ii) – easy syntactical check on diagram.

(2) An attribute $v$ may appear in multiple classes with different type:

\[
\begin{array}{ll}
C & D \\
v : \text{Bool} & v : \text{Int}
\end{array}
\]

Two approaches:

- Require unique attribute names.
  This requirement can easily be established (implicitly, behind the scenes) by viewing $v$ as an abbreviation for
  \[
  C(v) \quad \text{or} \quad D(v)
  \]
  depending on the context. ($C(v) : \text{Bool}$ and $D(v) : \text{Int}$ are then unique.)

- Subtle, formalist’s approach: observe that

  \[
  \langle v : \text{Bool}, \ldots \rangle \quad \text{and} \quad \langle v : \text{Int}, \ldots \rangle
  \]

  are different things in $V$. 

  $\mathcal{D}(\langle v, \text{Bool}, \ldots \rangle)$
  $\mathcal{D}(\langle v, \text{Int}, \ldots \rangle)$
Since we have not yet discussed associations, for now we read:

- $C \rightharpoonup^{r : D_{0,1}}_{0,1} D$
- $C \rightharpoonup^{r : D_*} D$

and

- $C \rightharpoonup^{r : D_{0,1}}_{0,1} D$
- $C \rightharpoonup^{r : D_*} D$

Course Map

- $\varphi \in \text{OCL}$
- $\pi = (\sigma_0, s_0) \xrightarrow{\text{cons}_i, \text{Snd}_i} (\sigma_1, \varepsilon_1) \cdots$
Stereotypes

Stereotypes as Labels or Tags

• What are Stereotypes?
  • Not represented in system states.
  • Not contributing to typing rules / well-formedness.

• Oestereich (2006):
  View stereotypes as (additional) “labelling” (“tags”) or as “grouping”.

• Useful for documentation and model-driven development, e.g. code-generation:
  • Documentation: e.g. layers of an architecture.
    Sometimes, packages (cf. OMG (2011a,b)) are sufficient and “right”.
  • Model Driven Architecture (MDA): later.
Example: Stereotypes for Documentation

- Example: Timing Diagram Viewer
  Schumann et al. (2008)
- Architecture has four layers:
  - core, data layer
  - abstract view layer
  - toolkit-specific view layer/widget
  - application using widget
  Stereotype "=" layer "=" colour.

Other Examples

- Use stereotypes ‘Team_1’, ‘Team_2’, ‘Team_3’ and assign stereotype Team_i to class C if Team_i is responsible for class C.

- Use stereotypes to label classes with licensing information (e.g., LGPL vs. proprietary).

- Use stereotypes ‘Server_A’, ‘Server_B’ to indicate where objects should be stored.

- Use stereotypes to label classes with states in the development process like “under development”, “submitted for testing”, “accepted”.

- etc. etc.

Necessary: a common idea of what each stereotype stands for.
(To be defined / agreed on by the team, not the job of the UML consortium.)
Tell Them What You’ve Told Them...

- **Extended Signatures** allow us to represent aspects like
  - abstract, active, visibility, initial value expression, ...

- Not all of these aspects are **semantically relevant**.

- The only change on **system states**
  is that abstract classes cannot have instances.

- **Class Diagrams** map to **Extended Signatures**, i.e. the meaning of a class diagram
  is the extended signature which it **uniquely** denotes.

- Thus a **Class Diagram** (transitively) denotes a set of system states (given a structure).

- **Stereotypes** are just labels.

References
References


