UML Model Instances

$\phi \in \text{OCL expr}$

$\pi = (\sigma_0, \varepsilon_0) \xrightarrow{-(\text{cons}_0, \text{Snd}_0)} u_0 \cdots w \pi = ((\sigma_i, \text{cons}_i, \text{Snd}_i))_{i \in \mathbb{N}}$

$G = (N, E, f)$

$\text{Mathematics}$

$\text{UML}$

Content

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

Extended Signature Example

What Do We Want / Have to Cover?

Extended Signature

Basic Object System Signature

of a

Structures of Extended Signatures

Extended Signature Example

What Do We Want / Have to Cover?

Extended Signature

Basic Object System Signature

of a

Structures of Extended Signatures

Extended Signature Example

What Do We Want / Have to Cover?
From Class Boxes to Extended Signatures

The set of system states \( \Delta \) is a domain function \( \Delta: \Sigma \rightarrow \mathbb{N} \), which assigns to each type a domain function \( D \) of system state \( \Delta \). This is mapped to an infinite set \( \Delta(\Sigma) \) which assigns to each type a domain function \( D \) of system state \( \Delta \).

\( \Delta(\Sigma) \) is semantically relevant if \( \Delta(\Sigma) \neq \emptyset \), i.e., \( \Delta(\Sigma) \) is a non-empty set.

The set of objects for different classes are disjoint, i.e.,

\[
C \cap D = \emptyset.
\]

Each \( C \) has a set of object identities for different classes, i.e.,

\[
C = \{ u \mid \text{is alive} \}.
\]

\( \Delta \) is a domain function \( \Delta: \Sigma \rightarrow \mathbb{N} \), which assigns to each type a domain function \( D \) of system state \( \Delta \).

\( \Delta(\Sigma) \) is semantically relevant if \( \Delta(\Sigma) \neq \emptyset \), i.e., \( \Delta(\Sigma) \) is a non-empty set.

The set of objects for different classes are disjoint, i.e.,

\[
C \cap D = \emptyset.
\]

Each \( C \) has a set of object identities for different classes, i.e.,

\[
C = \{ u \mid \text{is alive} \}.
\]
Some (and we) understand if not given. 

Require Two approaches

(i) A may appear in $C \{ i \leq m \}$ as a graph with nodes $v_1, v_2, ..., v_n$ as default, but conventions may vary.

(ii) $\mathbb{V}$ := $\mathbb{V} \{ v \}$ := $\{ \langle \{ \emptyset \}, \langle \{ \emptyset \} \rangle \}$. Some (and we) assume it as a graph with nodes $v_1, v_2, ..., v_n$ as default, but conventions may vary.

$\mathbb{V}$ := $\mathbb{V} \{ v \}$ := $\{ \langle \{ \emptyset \}, \langle \{ \emptyset \} \rangle \}$. Some (and we) assume it as a graph with nodes $v_1, v_2, ..., v_n$ as default, but conventions may vary.

$\mathbb{V}$ := $\mathbb{V} \{ v \}$ := $\{ \langle \{ \emptyset \}, \langle \{ \emptyset \} \rangle \}$. Some (and we) assume it as a graph with nodes $v_1, v_2, ..., v_n$ as default, but conventions may vary.

$\mathbb{V}$ := $\mathbb{V} \{ v \}$ := $\{ \langle \{ \emptyset \}, \langle \{ \emptyset \} \rangle \}$. Some (and we) assume it as a graph with nodes $v_1, v_2, ..., v_n$ as default, but conventions may vary.

$\mathbb{V}$ := $\mathbb{V} \{ v \}$ := $\{ \langle \{ \emptyset \}, \langle \{ \emptyset \} \rangle \}$. Some (and we) assume it as a graph with nodes $v_1, v_2, ..., v_n$ as default, but conventions may vary.

$\mathbb{V}$ := $\mathbb{V} \{ v \}$ := $\{ \langle \{ \emptyset \}, \langle \{ \emptyset \} \rangle \}$. Some (and we) assume it as a graph with nodes $v_1, v_2, ..., v_n$ as default, but conventions may vary.

$\mathbb{V}$ := $\mathbb{V} \{ v \}$ := $\{ \langle \{ \emptyset \}, \langle \{ \emptyset \} \rangle \}$. Some (and we) assume it as a graph with nodes $v_1, v_2, ..., v_n$ as default, but conventions may vary.
Since we have not yet discussed associations, for now we read

\[ C D \times r_{0,1} \]

as

\[ C r_{0,1} D \]

and

\[ C D \times r_{0,1} \]

as

\[ C D_{0,1} \]

• 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

Core View
Application/Qt
Trace
sort
move
filter
jump
zoom
View/Qt

• 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.)

• 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


