Model

$\mathcal{P} = (\mathcal{T}, \mathcal{C}, V, \text{atr}), SM$

$M = \left( \Sigma_{\mathcal{P}}, A_{\mathcal{P}}, \rightarrow_{SM} \right)$

$\pi = (\sigma_{0}, \epsilon_{0}) \xrightarrow{(cons_{0}, Snd_{0})} u_{0} (\sigma_{1}, \epsilon_{1}) \cdots$

$G = (N, E, f)$

Instances

$\varphi \in \text{OCL}$

$\varphi \in \text{OCL}$

$C_{D}, S_{M}$

$C_{D}, S_{M}$

$B = (Q_{SD}, q_{0}, A_{\mathcal{P}}, \rightarrow_{SD}, F_{SD})$

$w_{\pi} = ((\sigma_{i}, cons_{i}, Snd_{i}))_{i \in \mathbb{N}}$

$\od$

$\text{UML}$

Mathematics
• 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

Let \( \mathcal{S} = (\mathcal{T}, \mathcal{C}, V, \text{atr}) \) where

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

Example:

\[ \mathcal{S}_\lambda = (\text{MyType}_x, \Sigma C, \Pi \bar{x}, \Sigma \times \bar{x}, \lambda, \rho : \bar{x}, \sigma : \Theta, \bar{P} \) }\]

\[ \bar{C} \rightarrow \bar{E} \bar{P}, \overline{\Pi} \rightarrow \bar{E}, \bar{P} \]
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. (later)

Each attribute has

- a visibility,
- a name, a type, (later)
- 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 $\mathcal{I} = (\mathcal{T}, \mathcal{C}, V, atr)$ 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, $(a=1$ iff $C$ 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 $C \in \mathcal{C}$,
  - $\xi \in \{\text{public, private, protected, package}\}$ is the visibility,
    $$:+ \quad :=- \quad :=\# \quad :=\sim$$
  - 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; write ‘$\not\exists$’ to explicitly not give an initial value expression.
  - 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 $\mathcal{I}$. 
\[ \mathcal{I} = (\mathcal{T}, \mathcal{C}, V, \text{atr}), \quad \langle C, S_C, a, t \rangle \in \mathcal{C}, \quad \langle v : T, \xi, \text{expr}_0, P_v \rangle \in V \]
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 \( \langle v : T, \xi, expr_0, P_v \rangle \) if we want to refer to all aspects of attribute \( v \).

- Write only \( v : T \) or \( v \) if details are irrelevant.
Recall:

Definition. A **Basic Object System Structure** of a **Basic Object System Signature** \( \mathcal{I} = (\mathcal{T}, \mathcal{C}, V, atr) \) 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 \( \mathcal{D}(\mathcal{C}_*) \).
**Structures of Extended Signatures**

**New:**

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

- $\tau \in T$ 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.

\[ \forall C, D \in \mathcal{C} : C \neq D \rightarrow \mathcal{D}(C) \cap \mathcal{D}(D) = \emptyset. \]

- $C_{\ast}$ 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 $\mathcal{D}(\mathcal{C}_{\ast})$. 
Recall:

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

$$\sigma : D(C) \rightarrow (V \leftrightarrow (D(\mathcal{T}) \cup D(\mathcal{C}^*))).$$

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

- $\text{dom}(\sigma(u)) = \text{atr}(C)$
- $\sigma(u)(v) \in D(\tau)$ 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 $u \in D(C)$ alive in $\sigma$ if and only if $u \in \text{dom}(\sigma)$.

We use $\Sigma_D$ to denote the set of all system states of $\mathcal{I}$ wrt. $D$. 
New:

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

A system state of \( \mathcal{I} \) wrt. \( \mathcal{D} \) is a type-consistent mapping

\[
\sigma : \mathcal{D}(\mathcal{C}) \ni (V \ni (\mathcal{D}(\mathcal{I}) \cup \mathcal{D}(\mathcal{C}^*))).
\]

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

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

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

We use \( \Sigma_{\mathcal{D}} \) to denote the set of all system states of \( \mathcal{I} \) wrt. \( \mathcal{D} \).
The **semantics** (or meaning) of an extended object system signature $\mathcal{I}$ wrt. a structure $\mathcal{D}$ is the set of system states $\Sigma^\mathcal{D}$.

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

$$\left\{ \Sigma^\mathcal{D} \mid \mathcal{D} \text{ is structure of } \mathcal{I} \right\}.$$

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**, ✗
- has a **name**, ✓
- belongs to a **package**
- can be **abstract**, ✓
- can be **active**, ✗
- has a set of **attributes**, ✓
- has a set of **operations** (later).

**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.
Mapping UML Class Diagrams to Extended Signatures
A class box \( n \) **induces** an (extended) signature class as follows:

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

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

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

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

where

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

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

\[
\begin{align*}
\langle \langle S_1, \ldots, S_k \rangle \rangle \\
\xi_1 v_1 : T_1 = \text{expr}_1 \{P_{1,1}, \ldots, P_{1,m_1}\} \\
\vdots \\
\xi_\ell v_\ell : T_\ell = \text{expr}_\ell \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}
\end{align*}
\]

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

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

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

\[\mathcal{Y} = \{\text{Int, Float}\},\]

\[\langle \langle \text{Package::C, \{Stereotype_1, \ldots, Stereotype_n\}, 0, 0\rangle, \langle A, \emptyset, 1, 0\rangle, \langle B, \emptyset, 1, 0\rangle, \langle D, \emptyset, 0, 1\rangle \rangle,\]

\[\langle r : \text{Package::C}0,1, +, \text{expr}, \emptyset \rangle, \langle s : D, \emptyset, \emptyset, \emptyset, \{\text{ordered}\}\rangle,\]

\[\langle v : \text{Int}, -, 2, \emptyset \rangle, \langle w : \text{Float}, 2, 0, \{\text{readOnly}\}\rangle, \langle y : \text{Int}, ?, 0, 0\rangle,\]

\[\langle x : \text{Int}, ?, 0, 0\rangle \rangle,\]

\[\{\text{Package::C} \mapsto \{r, s, v, w, 3, A \mapsto y, B \mapsto \emptyset, D \mapsto \emptyset\}\}\]
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.
\[ C(n) := \langle C, \{S_1, \ldots, S_k\}, a(n), t(n) \rangle \]

\[ V(n) := \{\langle v_1 : T_1, \xi_1, expr_1^0, \{P_{1,1}, \ldots, P_{1,m_1}\}\rangle, \ldots, \langle v_\ell : T_\ell, \xi_\ell, expr_\ell^0, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}\rangle\} \]

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

\[ \mathcal{S} = (\{Int, Float\}, \{\langle \text{Package::C}, \{\text{Stereotype}_1, \ldots, \text{Stereotype}_n\}\rangle, 0, 0\}, \langle A, \emptyset, 1, 0\rangle, \langle B, \emptyset, 1, 0\rangle, \langle D, \emptyset, 0, 1\rangle, \langle r : \text{Package::C}_{0,1}, +, \text{expr}, \emptyset\rangle, \langle s : D_*, +, \bowtie, \{\text{ordered}\}\rangle, \langle v : \text{Int}, -, 27, \emptyset\rangle, \langle w : \text{Float}, +, \bowtie, \{\text{readOnly}\}\rangle, \langle x : \text{Int}, +, \bowtie, \emptyset\rangle, \langle y : \text{Int}, +, \bowtie, \emptyset\rangle, \{\text{Package::C} \mapsto \{r, s, v, w\}, A \mapsto \{y\}, B \mapsto \emptyset, D \mapsto \{x\}\}) \]
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 $\mathcal{I}(CD)$ well-defined?

There are two possible sources for problems:

1. A class $C$ may appear in multiple class diagrams:

   (i) $CD_1$
   
   $C$
   $v: \text{Int} = 0$

   $CD_2$

   $C$
   $w: \text{Int} = 27$

   (ii) $CD_1$

   $C$
   $v: \text{Int} = 0$

   $CD_2$

   $C$
   $v: \text{Int} = 27$

   Simply forbid the case (ii) – easy syntactical check on diagram.
(2) An attribute \( v \) may appear in multiple classes with different type:

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

**Two approaches:**

- Require **unique** attribute names. This requirement can easily be established (implicitly, behind the scenes) by viewing \( v \) as an abbreviation for

\[
\begin{align*}
\mathbb{D}(C::v) & \\
\mathbb{D}(D::v) &
\end{align*}
\]

depending on the context. \((C::v : \text{Bool} \text{ and } D::v : \text{Int} \text{ are then unique.})\)

- Subtle, formalist's approach: observe that

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

are **different things** in \( V \).
Since we have not yet discussed **associations**, for now we read

- \( C \times D \)

- \( C \times D \)

- \( C \times D \)

- \( C \times D \)

- \( C \times D \)

- \( C \times D \)

- \( C \times D \)

- \( C \times D \)

and
\[ L = (T, C, V, atr), SM \]

\[ M = (\Sigma_L, A_L, \rightarrow_{SM}) \]

\[ \pi = (\sigma_0, \varepsilon_0) \xrightarrow{(cons_0, Snd_0)} u_0 \xrightarrow{(\sigma_1, \varepsilon_1)} \cdots \]

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

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

\[ \varphi \in OCL \]

\[ CD, SM \]

\[ CD, SD \]

\[ UML \]

\[ Mathematics \]

\[ OD \]
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


