Software Design, Modelling and Analysis in UML

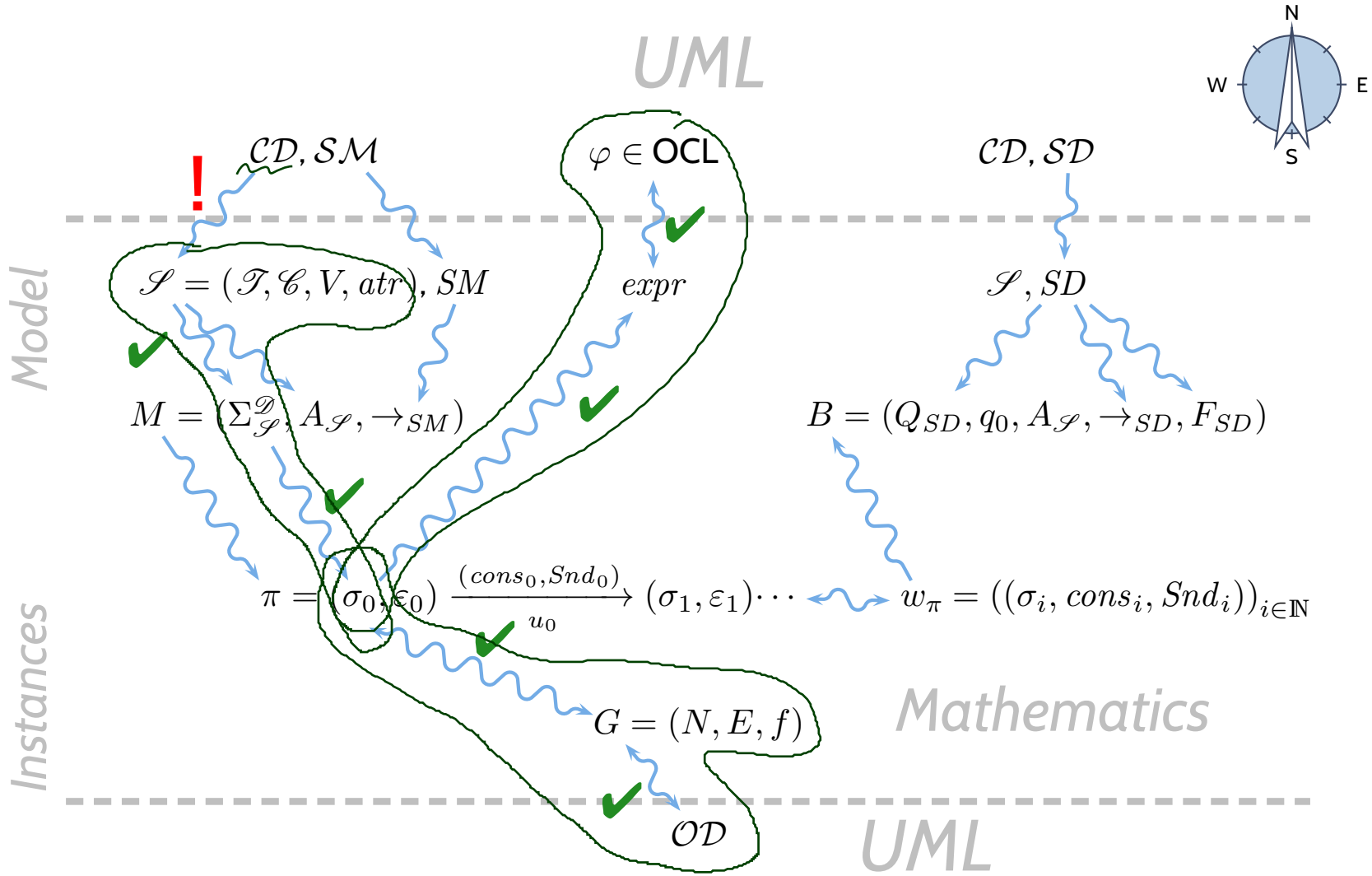
Lecture 6: Class Diagrams I

2016-11-15

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Course Map


UML
N
W
E
S
$\mathcal{CD}, \mathcal{SM}$
$\varphi \in \mathsf{OCL}$
$\mathcal{CD}, \mathcal{SD}$
Model
$\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr), SM$
$expr$
$\mathscr{S}, SD$
$M = (\Sigma^{\mathscr{D}}_{\mathscr{S}}, A_{\mathscr{S}}, \rightarrow_{SM})$
$B = (Q_{SD}, q_0, A_{\mathscr{S}}, \rightarrow_{SD}, F_{SD})$
$\pi = (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \cdots$
$w_\pi = ((\sigma_i, cons_i, Snd_i))_{i \in \mathbb{N}}$
Instances
$G = (N, E, f)$
Mathematics
$\mathcal{OD}$
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**

UML Class Diagrams: Stocktaking

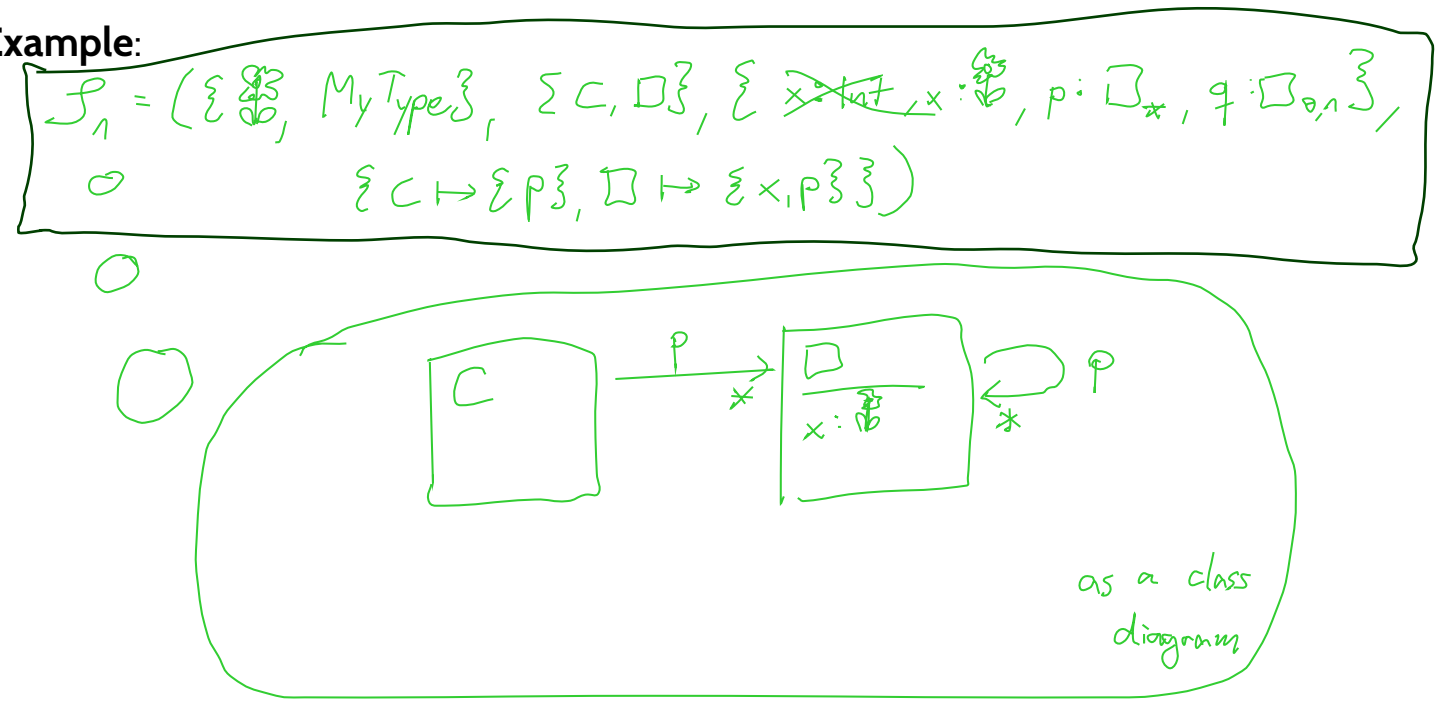
Recall: Signature vs. Class Diagram

Basic Object System Signature Another Example

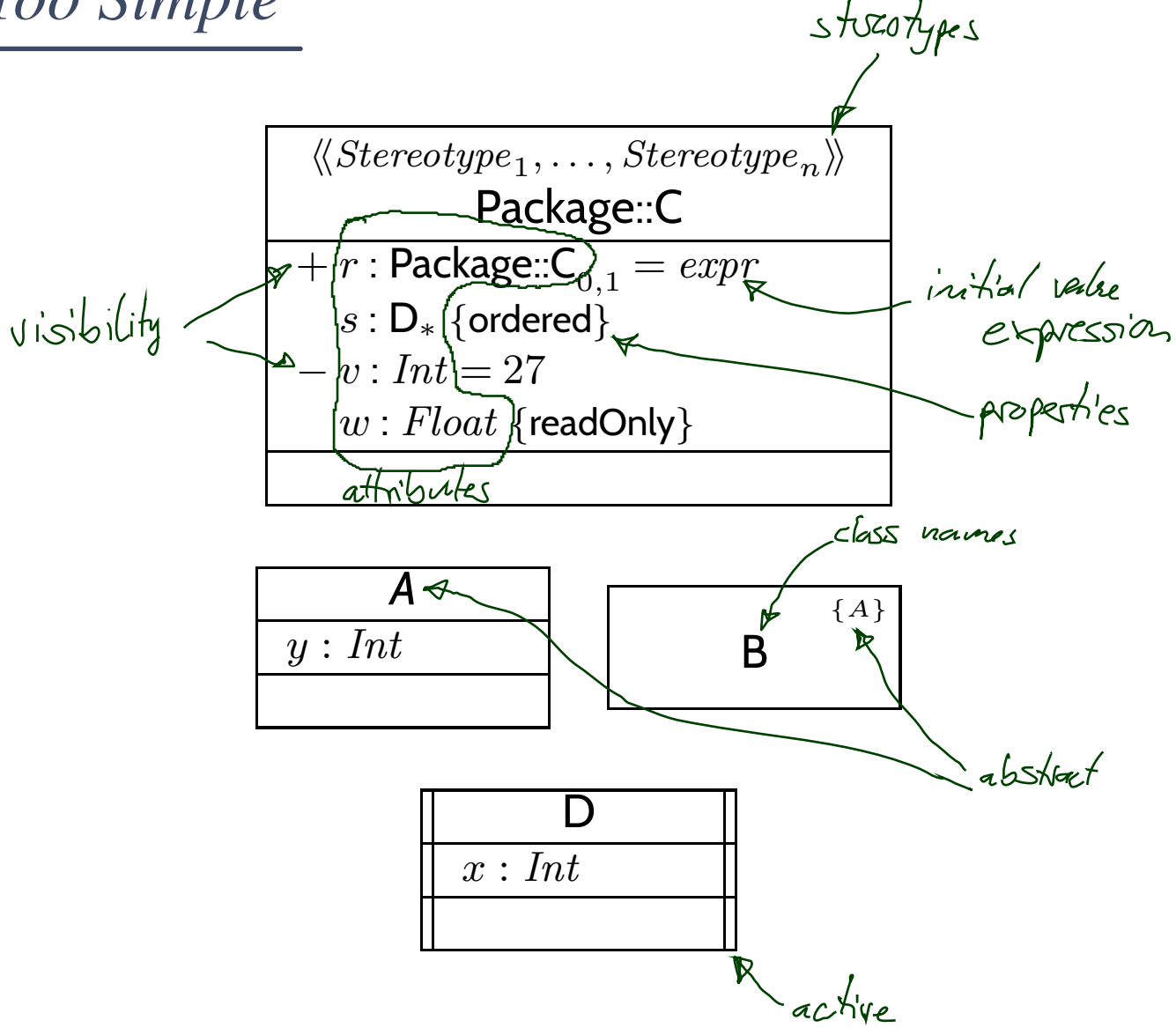
$\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr)$ where

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

Example:



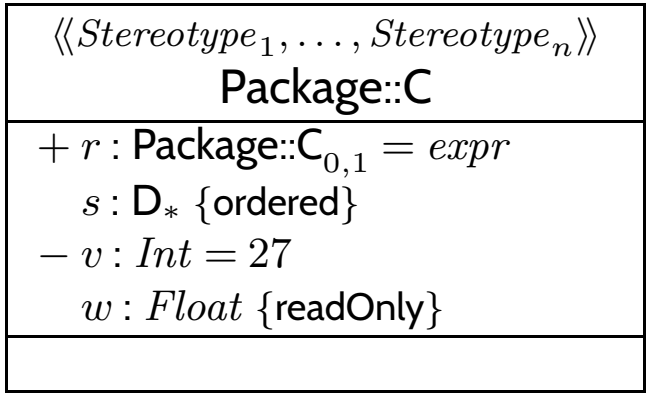
That'd Be Too Simple

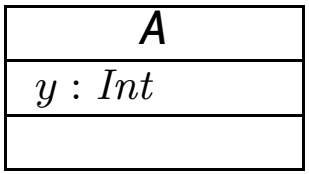


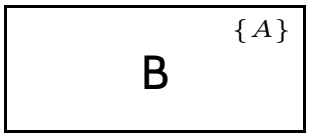
What Do We Want / Have to Cover?

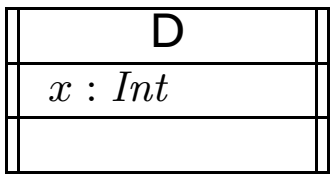
A **class**

- 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**. (→ 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

Extended Signature

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

- $\mathscr{T}$ is a set of (basic) types,
- $\mathscr{C}$ 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 $\mathscr{T}$, or $C_{0,1}, C_*$ for some $C \in \mathscr{C}$,
  - $\xi \in \{\underbrace{\text{public}}_{:=+}, \underbrace{\text{private}}_{:=-}, \underbrace{\text{protected}}_{:=\#}, \underbrace{\text{package}}_{:=\sim}\}$ 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;
    write '$\bowtie$' to explicitly **not** give an initial value expression.
  - a finite (possibly empty) set of **properties** P_v.
- $atr : \mathscr{C} \to 2^V$ maps each class to its set of attributes.

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

Extended Signature Example

$$\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr), \qquad \langle C, S_C, a, t\rangle \in \mathscr{C}, \qquad \langle v : T, \xi, expr_0, P_v\rangle \in V$$

basic types

class name

stereotypes

abstract

active

classes

$$\mathscr{S} = \big(\{Int, Float\},$$

$$\{\langle \mathsf{Package::C}, \{Stereotype_1, \ldots, Stereotype_n\}, 0, 0\rangle,$$

attribute name

type

no initial value expr.

attributes

$$\langle \mathsf{A}, \emptyset, 1, 0\rangle, \quad \langle \mathsf{B}, \emptyset, 1, 0\rangle, \quad \langle \mathsf{D}, \emptyset, 0, 1\rangle\},$$

$$\{\langle r : \mathsf{Package::C}_{0,1}, +, expr, \emptyset\rangle, \quad \langle s : \mathsf{D}_*, +, \bowtie, \{\mathsf{ordered}\}\rangle,$$

visibility

properties

$$\langle v : Int, -, 27, \emptyset\rangle, \quad \langle w : Float, +, \bowtie, \{\mathsf{readOnly}\}\rangle,$$

$$\langle x : Int, +, \bowtie, \emptyset\rangle, \quad \langle y : Int, +, \bowtie, \emptyset\rangle\},$$

ops

$$\{\mathsf{Package::C} \mapsto \{r, s, v, w\}, \quad \mathsf{A} \mapsto \{y\}, \quad \mathsf{B} \mapsto \emptyset, \quad \mathsf{D} \mapsto \{x\}\}\big)$$

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**.

Structures of Extended Signatures

Recall:

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

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

  Note: Object identities only have the "$=$" operation.
- Sets of object identities for different classes are disjoint, i.e.

$$\forall\, C, D \in \mathscr{C} : C \neq D \rightarrow \mathscr{D}(C) \cap \mathscr{D}(D) = \emptyset.$$

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

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

Structures of Extended Signatures

New:

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

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

  Note: Object identities only have the "$=$" operation.
- Sets of object identities for different classes are disjoint, i.e.

$$\forall\, C, D \in \mathscr{C} : C \neq D \rightarrow \mathscr{D}(C) \cap \mathscr{D}(D) = \emptyset.$$

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

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

System States of Extended Signatures

Recall:

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

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

$$\sigma : \mathscr{D}(\mathscr{C}) \nrightarrow (V \nrightarrow (\mathscr{D}(\mathscr{T}) \cup \mathscr{D}(\mathscr{C}_*))).$$

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

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

We call $u \in \mathscr{D}(\mathscr{C})$ alive in σ if and only if $u \in \mathrm{dom}(\sigma)$.

We use $\Sigma^{\mathscr{D}}_{\mathscr{S}}$ to denote the set of all system states of $\mathscr{S}$ wrt.$\mathscr{D}$.

System States of Extended Signatures

New:

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

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

$$\sigma : \mathscr{D}(\mathscr{C}) \nrightarrow (V \nrightarrow (\mathscr{D}(\mathscr{T}) \cup \mathscr{D}(\mathscr{C}_*))).$$

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

- $\mathrm{dom}(\sigma(u)) = atr(C)$
- $\sigma(u)(v) \in \mathscr{D}(\tau)$ if $v : \tau, \tau \in \mathscr{T}$
- $\sigma(u)(v) \in \mathscr{D}(D_*)$ if $v : D_{0,1}$ or $v : D_*$ with $D \in \mathscr{C}$
- $\forall \langle C, S_C, \mathbb{1}, t \rangle \in \mathscr{C} \bullet \mathrm{dom}(\sigma) \cap \mathscr{D}(C) = \emptyset$.

abstract

We call $u \in \mathscr{D}(\mathscr{C})$ alive in σ if and only if $u \in \mathrm{dom}(\sigma)$.

We use $\Sigma^{\mathscr{D}}_{\mathscr{S}}$ to denote the set of all system states of $\mathscr{S}$ wrt.$\mathscr{D}$.

Semantical Relevance

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

$$\{\Sigma^{\mathscr{D}}_{\mathscr{S}} \mid \mathscr{D} \text{ is structure of } \mathscr{S}\}.$$

$\mathscr{S}_1 : \langle C, \emptyset, 0, 0 \rangle \rightsquigarrow \Sigma^{\mathscr{D}}_{\mathscr{S}_1} =$
$\mathscr{S}_2 : \langle C, \emptyset, 0, 1 \rangle \rightsquigarrow \Sigma^{\mathscr{D}}_{\mathscr{S}_2}$
(only difference in $\mathscr{S}_1, \mathscr{S}_2$ is activeness of C)

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).

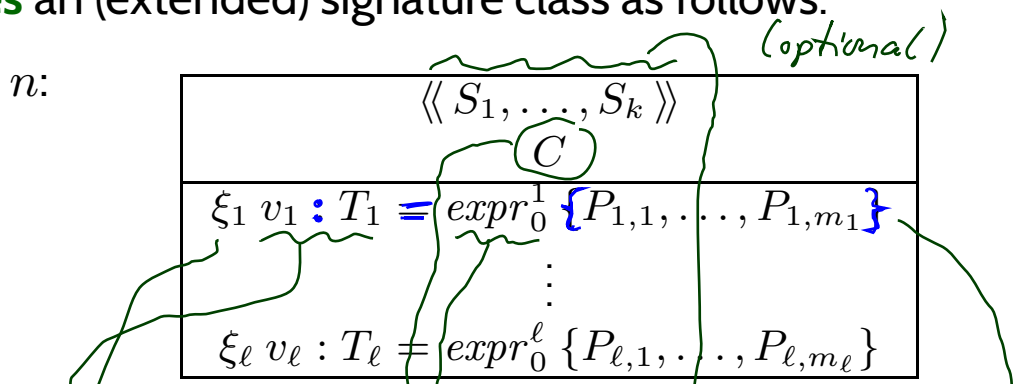
Each **attribute** has

- 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

From Class Boxes to Extended Signatures

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



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

$$V(n) := \{\langle v_1 : T_1, \xi_1, expr_0^1, \{P_{1,1}, \ldots, P_{1,m_1}\}\rangle, \ldots, \langle v_\ell : T_\ell, \xi_\ell, expr_0^\ell, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}\rangle\}$$

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

where

- "abstract" is determined by the font:

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

- "active" is determined by the frame:

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

Example

$\langle\!\langle S_1, \ldots, S_k \rangle\!\rangle$ C
$\xi_1\, v_1 : T_1 = expr_0^1\, \{P_{1,1}, \ldots, P_{1,m_1}\}$ $\vdots$ $\xi_\ell\, v_\ell : T_\ell = expr_0^\ell\, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}$

$$\wr$$

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

$$V(n) := \{\langle v_1 : T_1, \xi_1, expr_0^1, \{P_{1,1}, \ldots, P_{1,m_1}\}\rangle, \ldots,$$
$$\langle v_\ell : T_\ell, \xi_\ell, expr_0^\ell, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}\rangle\}$$

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

$\langle\!\langle Stereotype_1, \ldots, Stereotype_n \rangle\!\rangle$ Package::C
$+\ r$: Package::$C_{0,1} = expr$ s : D_* {ordered} $-\ v : Int = 27$ $w : Float$ {readOnly}

A
$y : Int$

$\{A\}$ B

D
$x : Int$

$\mathcal{Y} = ($ {Int, Float},
{ ⟨ Package::C, {Stereotype$_1$, ..., Stereotype$_n$}, 0, 0⟩, ⟨A, ∅, 1, 0⟩, ⟨B, ∅, 1, 0⟩, ⟨D, ∅, 0, 1⟩ },
{ ⟨ r : Package::$C_{0,1}$, +, expr, ∅ ⟩, ⟨ s : D_*, ?, ⋈, {ordered} ⟩,
⟨ v : Int, −, 27, ∅ ⟩, ⟨ w : Float, ?, ⋈, {readOnly} ⟩, ⟨ y : Int, ?, ⋈, ∅ ⟩,
⟨ x : Int, ?, ⋈, ∅ ⟩ },
{ Package::C ↦ {r, s, v, w}, A ↦ {y}, B ↦ ∅, D ↦ {x} })

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.

Example Cont'd

$\langle\!\langle S_1, \ldots, S_k \rangle\!\rangle$ C
$\xi_1\, v_1 : T_1 = expr_0^1\, \{P_{1,1}, \ldots, P_{1,m_1}\}$ $\vdots$ $\xi_\ell\, v_\ell : T_\ell = expr_0^\ell\, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}$

$$\wr$$

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

$$V(n) := \{\langle v_1 : T_1, \xi_1, expr_0^1, \{P_{1,1}, \ldots, P_{1,m_1}\}\rangle, \ldots,$$
$$\langle v_\ell : T_\ell, \xi_\ell, expr_0^\ell, \{P_{\ell,1}, \ldots, P_{\ell,m_\ell}\}\rangle\}$$

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

$\langle\!\langle Stereotype_1, \ldots, Stereotype_n \rangle\!\rangle$ Package::C
$+\ r$: Package::$C_{0,1} = expr$ $\quad s$: D_* {ordered} $-\ v : Int = 27$ $\quad w : Float$ {readOnly}

A
$y : Int$

{A} B

D
$x : Int$

$$\mathscr{S} = \big(\{Int, Float\},$$

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

$$\{\langle r : \text{Package::C}_{0,1}, +, expr, \emptyset\rangle, \quad \langle s : \text{D}_*, +, \bowtie, \{\text{ordered}\}\rangle,$$
$$\langle v : Int, -, 27, \emptyset\rangle, \quad \langle w : Float, +, \bowtie, \{\text{readOnly}\}\rangle,$$
$$\langle x : Int, +, \bowtie, \emptyset\rangle, \quad \langle y : Int, +, \bowtie, \emptyset\rangle\},$$

$$\{\text{Package::C} \mapsto \{r, s, v, w\}, \quad \text{A} \mapsto \{y\}, \quad \text{B} \mapsto \emptyset, \quad \text{D} \mapsto \{x\}\}\big)$$

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).
  - $\mathscr{C}(\mathcal{CD}) := \{C(n_i) \mid 1 \leq i \leq N\}$
  - $V(\mathcal{CD}) := \bigcup_{i=1}^{N} V(n_i)$
  - $atr(\mathcal{CD}) := \bigcup_{i=1}^{N} atr(n_i)$

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

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

  which **induce** the following signature:

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

  (Assuming $\mathscr{T}$ given. In "reality" (i.e. in full UML), we can introduce types in class diagrams, the class diagram then contributes to $\mathscr{T}$. Example: enumeration types.)

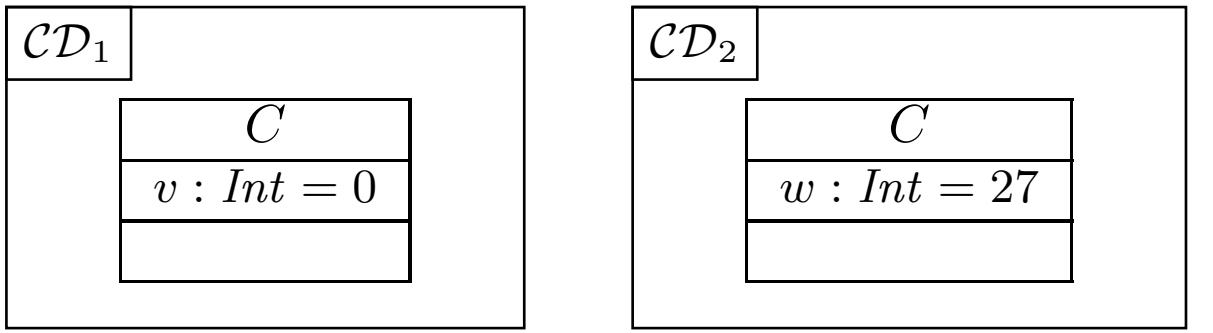
Is the Mapping a Function?

Question: Is $\mathscr{S}(\mathscr{CD})$ **well-defined**?

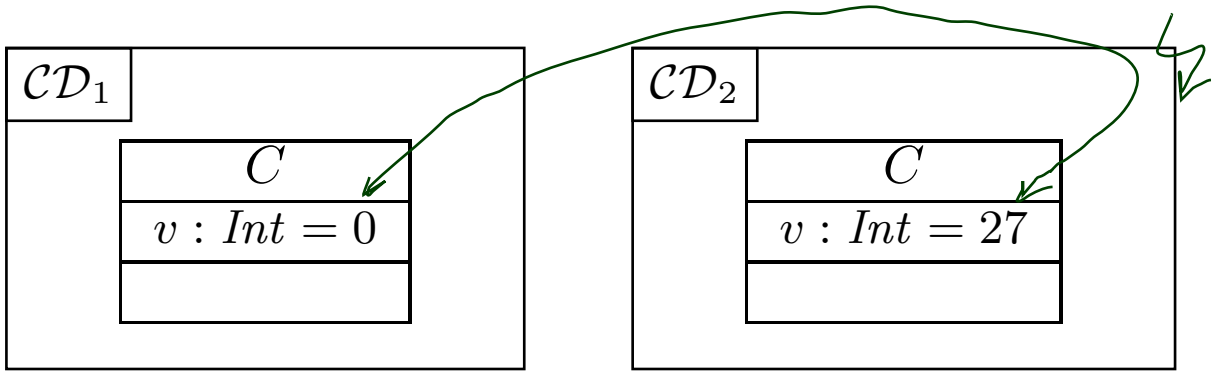
There are two possible **sources for problems**:

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

(i)



(ii)



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

Is the Mapping a Function?

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

C
$v : Bool$

D
$v : Int$

Two approaches:

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

  $C::v$ or $D::v$

  depending on the context. ($C::v : Bool$ and $D::v : Int$ are then unique.)

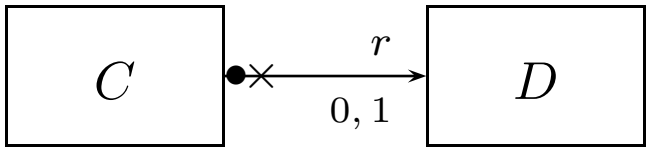
- Subtle, formalist's approach: observe that

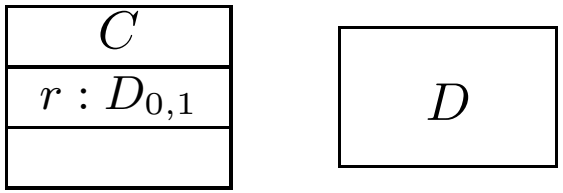
  $$\langle v : Bool, \ldots \rangle \quad \text{and} \quad \langle v : Int, \ldots \rangle$$

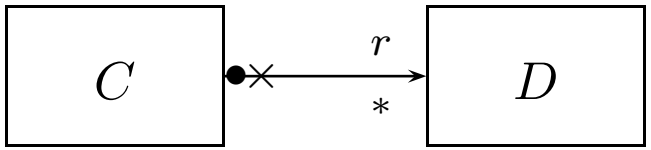
  are **different things** in V.

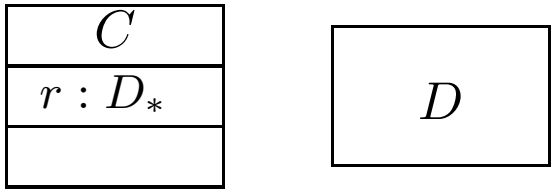
Abbreviations

Since we have not yet discussed **associations**, for now we read

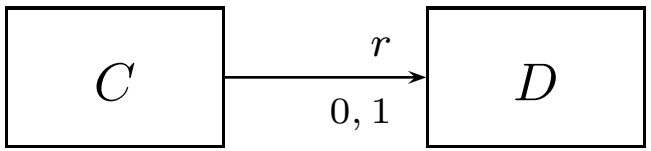
- 

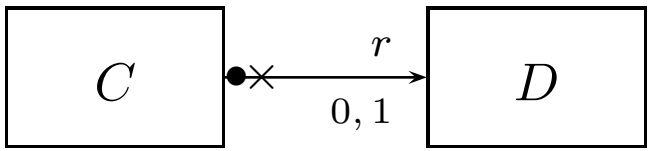
as


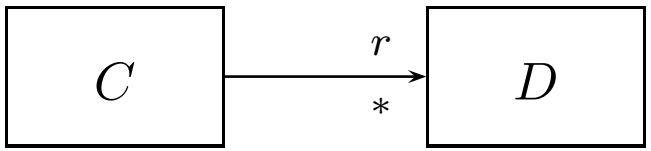
- 

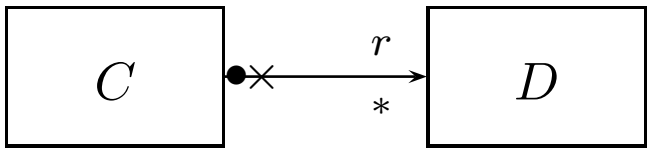
as


and

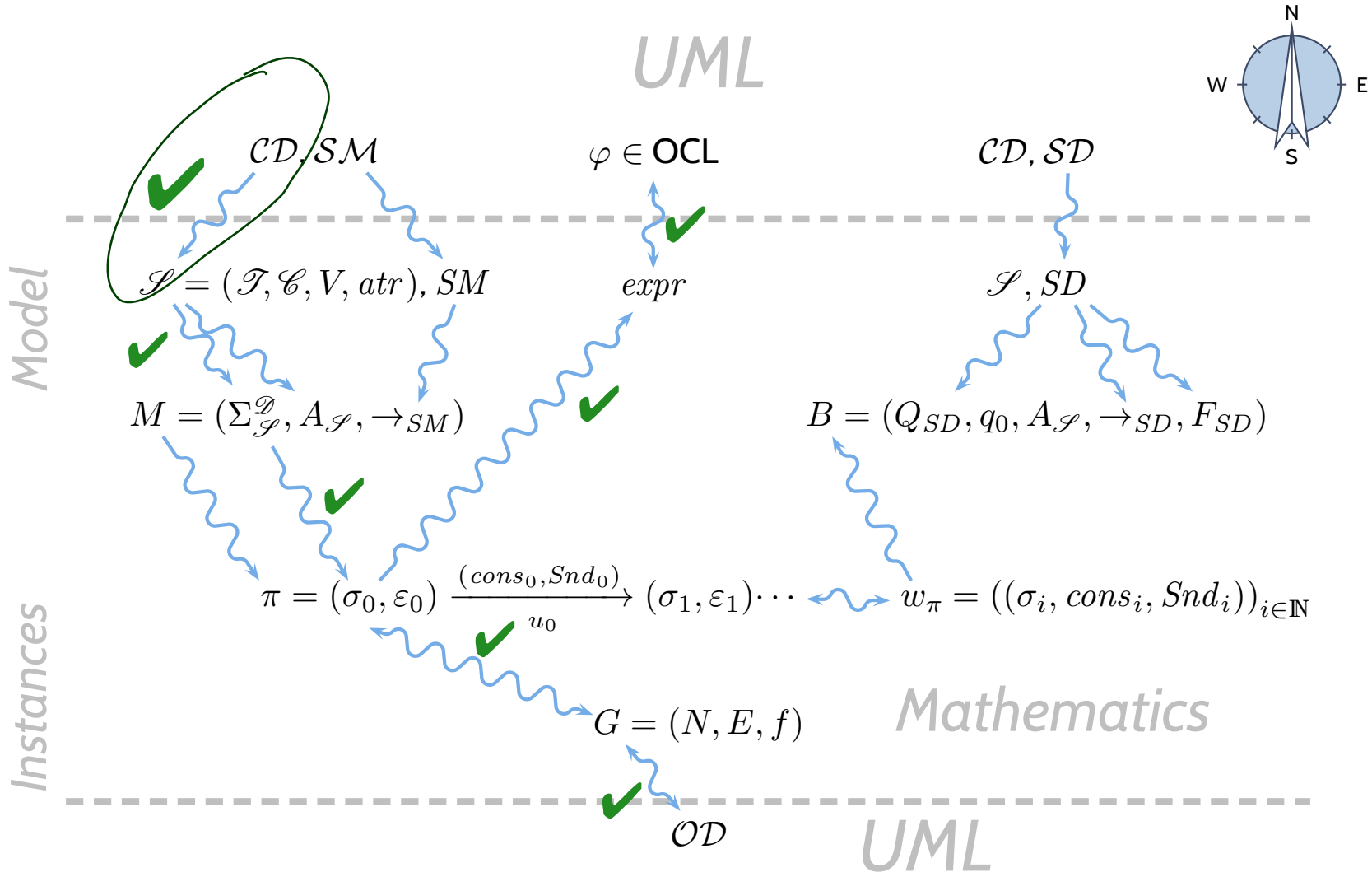
- 

as


- 

as


Course Map


UML
N
W
E
S
$\mathcal{CD}, \mathcal{SM}$
$\varphi \in \mathsf{OCL}$
$\mathcal{CD}, \mathcal{SD}$
Model
$\mathscr{S} = (\mathscr{T}, \mathscr{C}, V, atr), SM$
$expr$
$\mathscr{S}, SD$
$M = (\Sigma^{\mathscr{D}}_{\mathscr{S}}, A_{\mathscr{S}}, \rightarrow_{SM})$
$B = (Q_{SD}, q_0, A_{\mathscr{S}}, \rightarrow_{SD}, F_{SD})$
$\pi = (\sigma_0, \varepsilon_0) \xrightarrow[u_0]{(cons_0, Snd_0)} (\sigma_1, \varepsilon_1) \cdots$
$w_\pi = ((\sigma_i, cons_i, Snd_i))_{i \in \mathbb{N}}$
Instances
$G = (N, E, f)$
Mathematics
$\mathcal{OD}$
UML

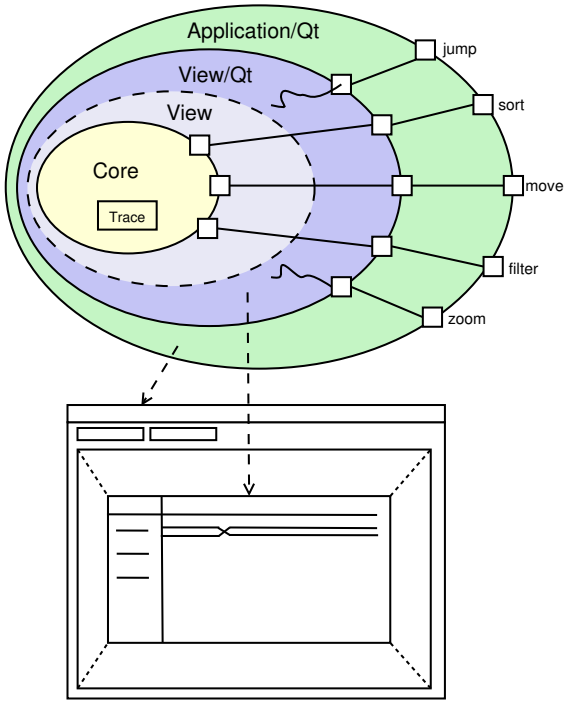
Stereotypes

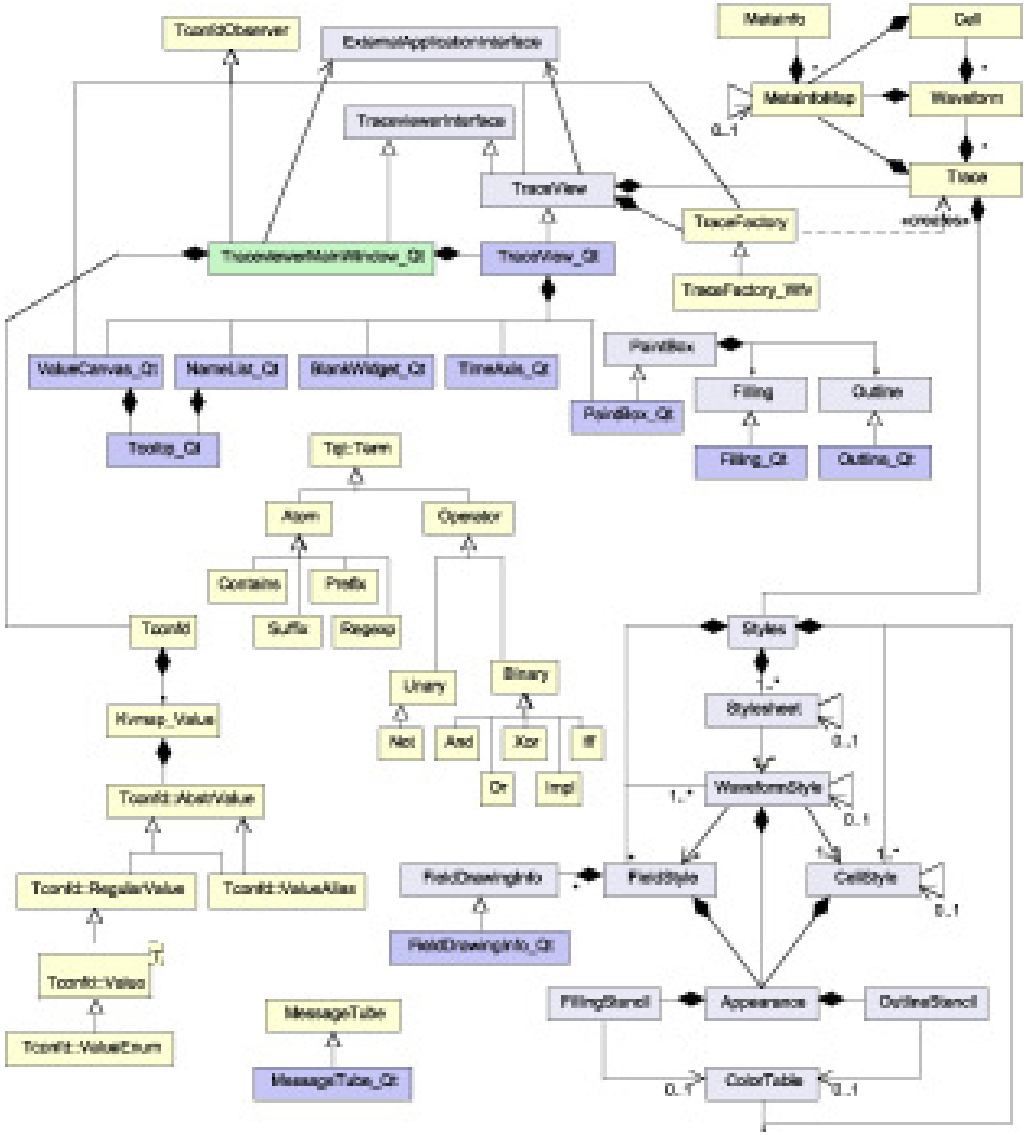
Stereotypes as Labels or Tags

- 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

Oestereich, B. (2006). *Analyse und Design mit UML 2.1, 8. Auflage*. Oldenbourg, 8. edition.

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.

Schumann, M., Steinke, J., Deck, A., and Westphal, B. (2008). Traceviewer technical documentation, version 1.0. Technical report, Carl von Ossietzky Universität Oldenburg und OFFIS.