

# Software Design, Modelling and Analysis in UML

## Lecture 6: Class Diagrams I

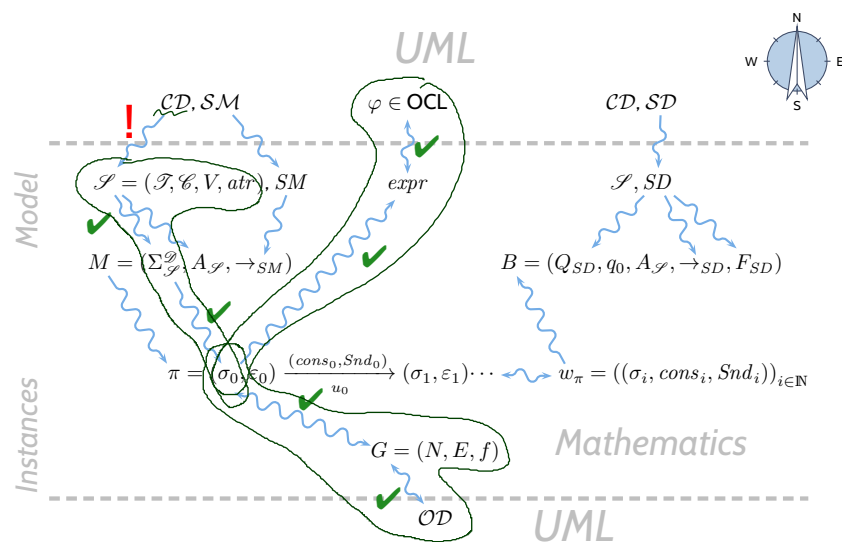
2016-11-15

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

-6-2016-11-15-main-

### Course Map



-6-2016-11-15-main-

## 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

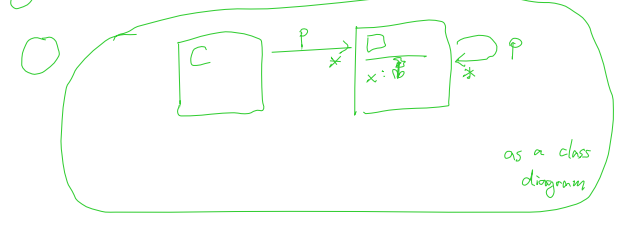
### Basic Object System Signature Another Example

$\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$  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  $C \in \mathcal{C}$ ,
- $atr : \mathcal{C} \rightarrow 2^V$  mapping classes to attributes.

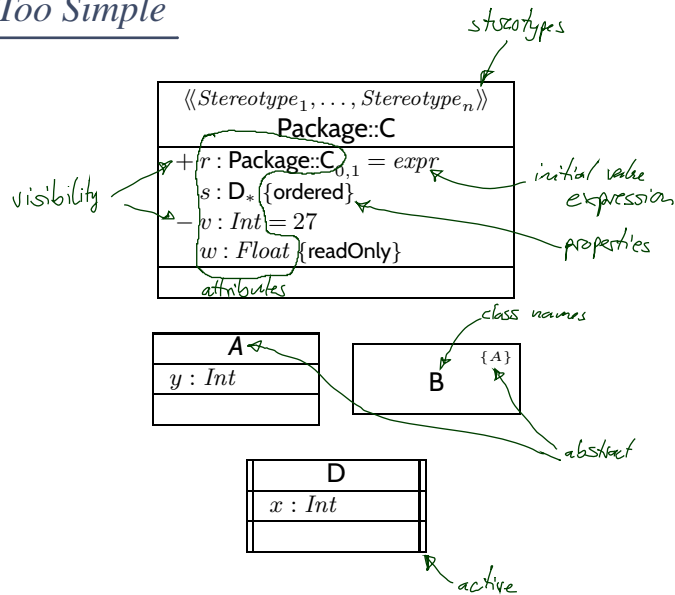
Example:

$\mathcal{S}_1 = (\{\mathbb{R}, \text{MyType}\}, \{C, D\}, \{x: \mathbb{R}, y: \mathbb{R}, z: \mathbb{R}\}, \{C \mapsto \{x, y\}, D \mapsto \{x, y, z\}\})$



as a class diagram

## That'd Be Too Simple



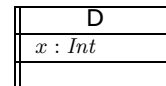
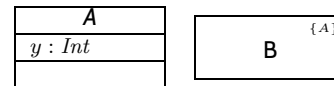
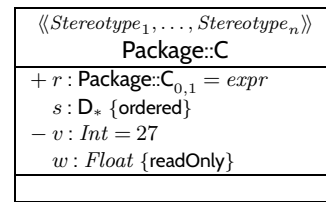
## What Do We Want / Have to Cover?

### 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**. ( $\rightarrow$  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  $\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$  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**. ( $a=1$  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  $C \in \mathcal{C}$ ,
  - $\xi \in \{\underbrace{\text{public}}_{:=+}, \underbrace{\text{private}}_{:= -}, \underbrace{\text{protected}}_{:=\#}, \underbrace{\text{package}}_{:=\sim}\}$  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 '∅' 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.

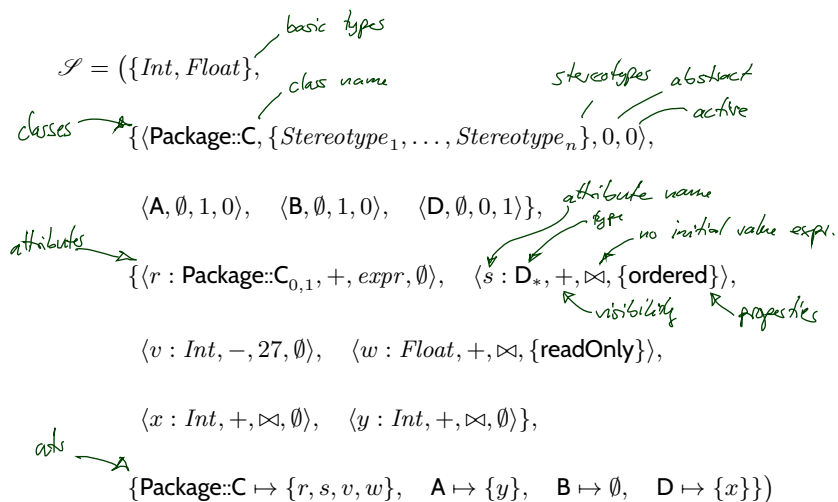
We use  $S_{\mathcal{C}}$  to denote the set  $\bigcup_{C \in \mathcal{C}} S_C$  of stereotypes in  $\mathcal{S}$ .

- 6 - 2016-11-15 - Sonntag -

9/31

## Extended Signature Example

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



- 6 - 2016-11-15 - Sonntag -

10/31

## 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**  $\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$  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**  $\mathcal{S} = (\mathcal{T}, \mathcal{C}, V, atr)$  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  $\mathcal{D}(\mathcal{C}_*)$ .

## System States of Extended Signatures

Recall:

**Definition.** Let  $\mathcal{D}$  be a **basic structure** of **basic 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  $u \in \mathcal{D}(C)$ ,  $C \in \mathcal{C}$ , if  $u \in \text{dom}(\sigma)$

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

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

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

## 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  $u \in \mathcal{D}(C)$ ,  $C \in \mathcal{C}$ , if  $u \in \text{dom}(\sigma)$

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

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  $\mathcal{S}$  wrt.  $\mathcal{D}$ .

-6-2016-11-15 - Sonntag-

13/31

## 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**  $\Sigma_{\mathcal{D}}$ .
- 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

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

$\mathcal{D}_1 : \langle C, \emptyset, 0, 0 \rangle \rightarrow \Sigma_{\mathcal{D}_1} = \Sigma_{\mathcal{D}}$   
 $\mathcal{D}_2 : \langle C, \emptyset, 0, 1 \rangle \rightarrow \Sigma_{\mathcal{D}_2}$   
 (only difference in  $\mathcal{D}_1, \mathcal{D}_2$  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.

-6-2016-11-15 - Sonntag-

14/31



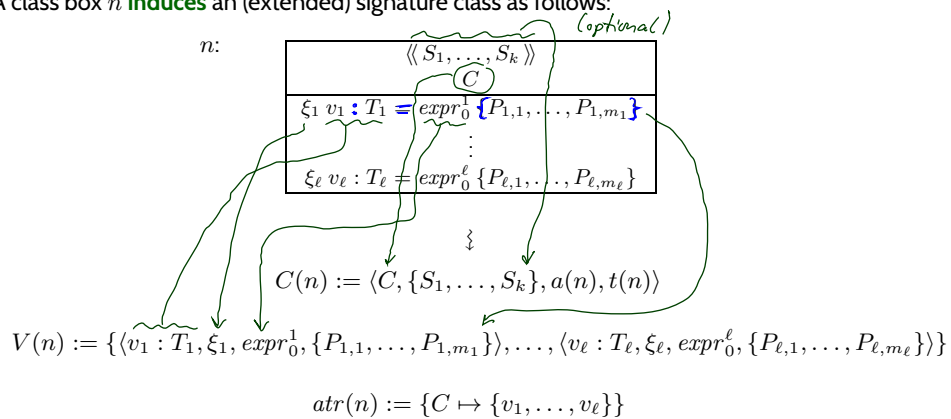
## Mapping UML Class Diagrams to Extended Signatures

-6-2016-11-15-main-

15/31

### From Class Boxes to Extended Signatures

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



where

- "abstract" is determined by the font:

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

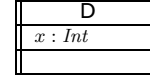
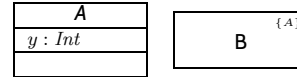
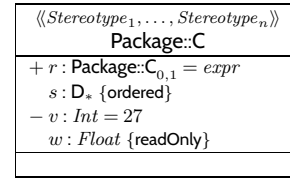
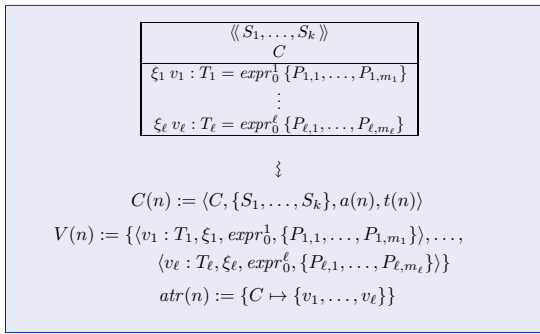
- "active" is determined by the frame:

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

-6-2016-11-15-Schönp -

16/31

## Example



$\mathcal{Y} = \left( \{ \text{Int}, \text{Float} \}, \right.$   
 $\{ \langle \text{Package}::C, \{ \text{Stereotype}_1, \dots, \text{Stereotype}_n \}, 0, 0 \rangle, \langle A, \emptyset, 1, 0 \rangle, \langle B, \emptyset, 1, 0 \rangle, \langle D, \emptyset, 0, 1 \rangle \},$   
 $\{ \langle r : \text{Package}::C_{0,1}, +, \text{expr}, \emptyset \rangle, \langle s : D_*, \{ \text{ordered} \}, \emptyset \rangle,$   
 $\langle v : \text{Int}, -, 27, \emptyset \rangle, \langle w : \text{Float}, \{ \text{readOnly} \}, \emptyset \rangle, \langle y : \text{Int}, \emptyset, \emptyset \rangle,$   
 $\langle x : \text{Int}, \emptyset, \emptyset \rangle \}$   
 $\{ \text{Package}::C \mapsto \{ r, s, v, w \}, A \mapsto \{ y \}, B \mapsto \emptyset, D \mapsto \{ x \} \} \left. \right)$

- 6 - 2016-11-15 - Scdmnp -

17/31

## What If Things Are Missing?

### It depends.

- What does the standard say? (OMG, 2011a, 121)

#### “Presentation Options.

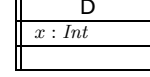
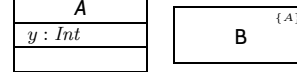
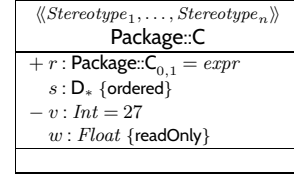
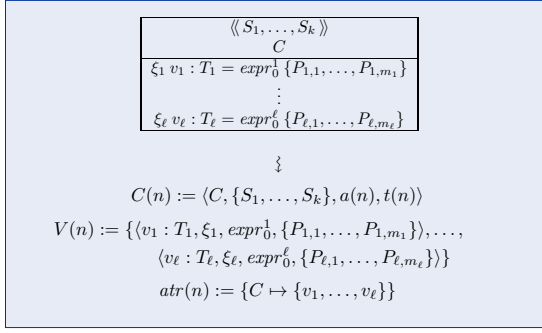
*The type, visibility, default, multiplicity, property string may be suppressed from being displayed, even if there are values in the model.”*

- **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.

- 6 - 2016-11-15 - Scdmnp -

18/31

## Example Cont'd



$$\begin{aligned} \mathcal{S} = & \{ \{ \text{Int}, \text{Float} \}, \\ & \{ \langle \text{Package}::C, \{ \text{Stereotype}_1, \dots, \text{Stereotype}_n \}, 0, 0 \rangle, \\ & \langle \mathbf{A}, \emptyset, 1, 0 \rangle, \langle \mathbf{B}, \emptyset, 1, 0 \rangle, \langle \mathbf{D}, \emptyset, 0, 1 \rangle \}, \\ & \{ \langle r : \text{Package}::C_{0,1}, +, \text{expr}, \emptyset \rangle, \langle s : D_*, +, \bowtie, \{ \text{ordered} \} \rangle, \\ & \langle v : \text{Int}, -, 27, \emptyset \rangle, \langle w : \text{Float}, +, \bowtie, \{ \text{readOnly} \} \rangle, \\ & \langle x : \text{Int}, +, \bowtie, \emptyset \rangle, \langle y : \text{Int}, +, \bowtie, \emptyset \rangle \}, \\ & \{ \text{Package}::C \mapsto \{ r, s, v, w \}, \mathbf{A} \mapsto \{ y \}, \mathbf{B} \mapsto \emptyset, \mathbf{D} \mapsto \{ x \} \} \end{aligned}$$

- 6 - 2016-11-15 - Scdmp -

19/31

## From Class Diagrams to Extended Signatures

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

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

$$\mathcal{CD} = \{CD_1, \dots, CD_k\},$$

which **induce** the following signature:

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

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

- 6 - 2016-11-15 - Scdmp -

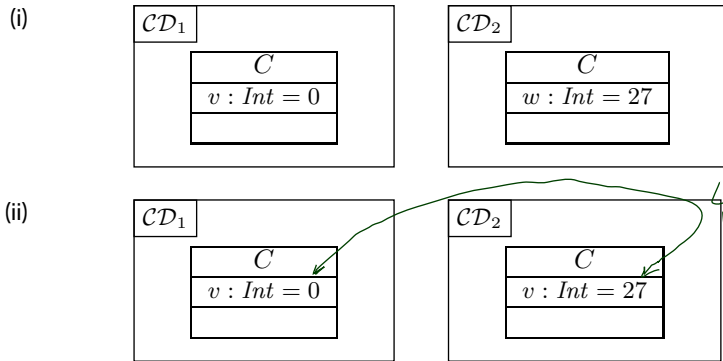
20/31

## Is the Mapping a Function?

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

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

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



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

- 6 - 2016-11-15 - Scdmap -

21/31

## Is the Mapping a Function?

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



**Two approaches:**

- Require **unique** attribute names.

This requirement can easily be established (implicitly, behind the scenes) by viewing  $v$  as an abbreviation for

$$\underline{C::v} \text{ or } \underline{D::v} \quad \mathcal{D}(C::v) \dots \quad \mathcal{D}(D::v) \dots$$

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

- Subtle, formalist's approach: observe that

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

are **different things** in  $V$ .

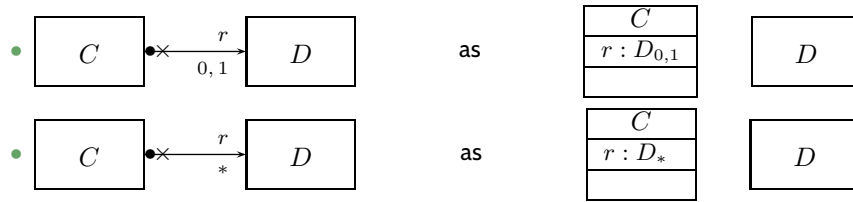
$$\mathcal{D}(\langle v : Bool, \dots \rangle) \quad \mathcal{D}(\langle v : Int, \dots \rangle)$$

- 6 - 2016-11-15 - Scdmap -

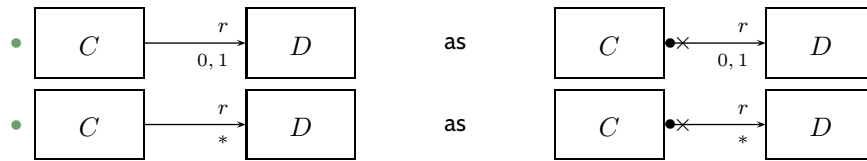
22/31

## Abbreviations

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

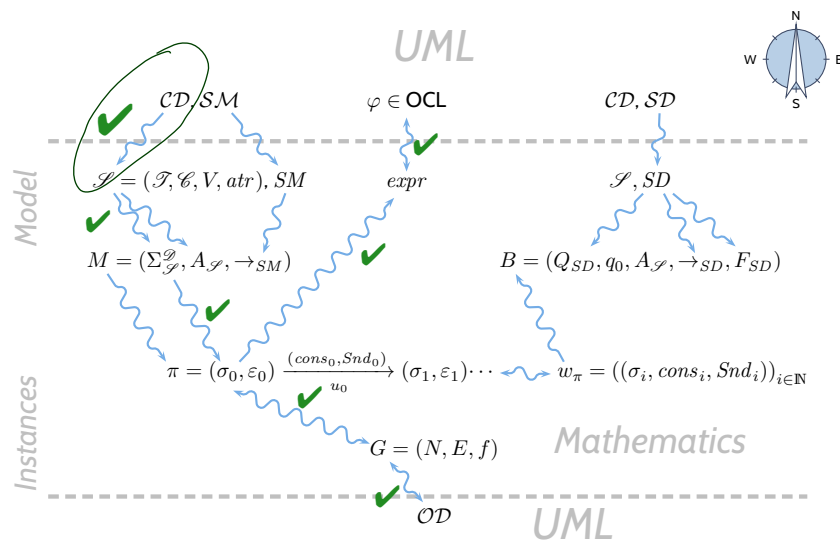


and



- 6 - 2016-11-15 - Scdmhp -

## Course Map



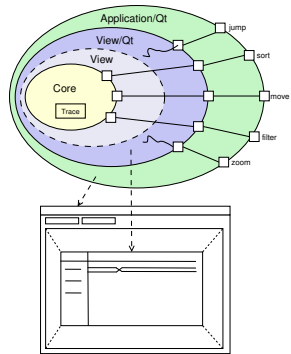
- 6 - 2016-11-15 - math -

# Stereotypes

## Stereotypes as Labels or Tags

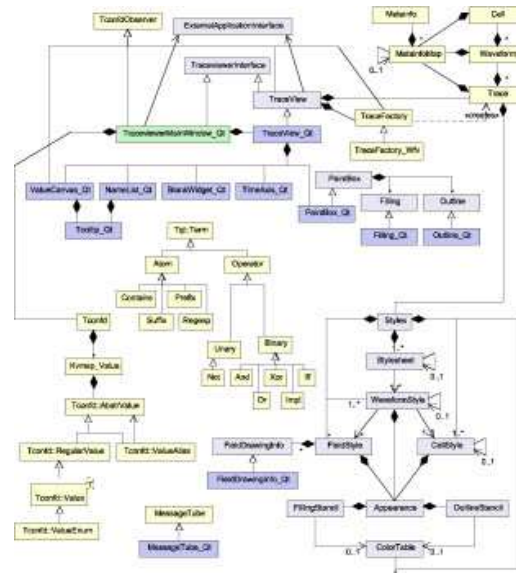
- What are Stereotypes?
  - **Not** represented in system states.
  - **Not** contributing to typing rules / well-formedness.
- [Oestereich \(2006\)](#):  
View stereotypes as (additional) “labelling” (“tags”) or as “grouping”.
- Useful for documentation and model-driven development, e.g. code-generation:
  - **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.

- 6 - 2016-11-15 - Stereo -



27/31

## Other Examples

- Use stereotypes 'Team<sub>1</sub>', 'Team<sub>2</sub>', 'Team<sub>3</sub>' and assign stereotype Team<sub>i</sub> to class C if Team<sub>i</sub> is responsible for class C.
- Use stereotypes to label classes with licensing information (e.g., LGPL vs. proprietary).
- Use stereotypes 'Server<sub>A</sub>', 'Server<sub>B</sub>' 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.)

- 6 - 2016-11-15 - Stereo -

28/31

## *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.