

Software Design, Modelling and Analysis in UML

Lecture 06: Class Diagrams I

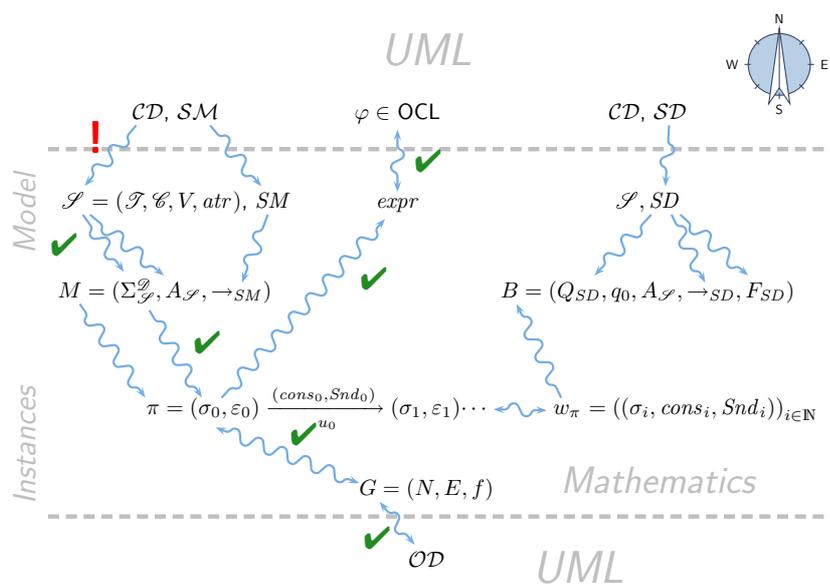
2013-11-11

Prof. Dr. Andreas Podelski, Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

-06 - 2013-11-11 - main -

Course Map



-06 - 2013-11-11 - Spreim -

Contents & Goals

Last Lecture:

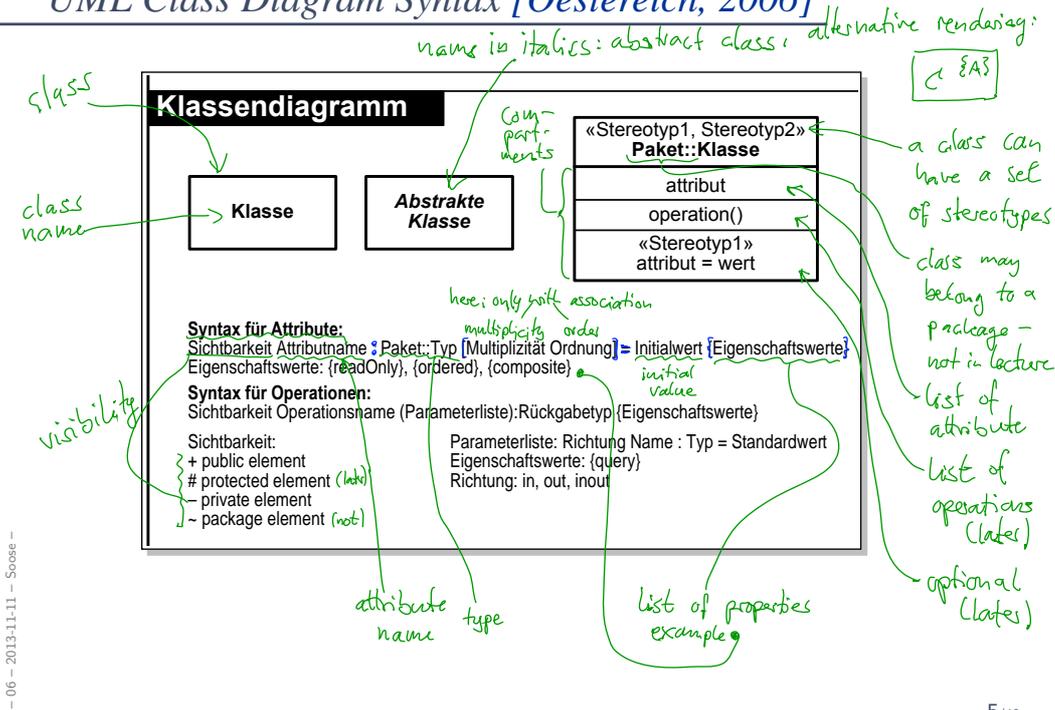
- OCL Semantics
- Object Diagrams

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
 - What is a class diagram?
 - For what purposes are class diagrams useful?
 - Could you please map this class diagram to a signature?
 - Could you please map this signature to a class diagram?
- **Content:**
 - Study UML syntax.
 - Prepare (extend) definition of signature.
 - Map class diagram to (extended) signature.
 - Stereotypes – for documentation.

UML Class Diagrams: Stocktaking

UML Class Diagram Syntax [Oestereich, 2006]



-06 - 2013-11-11 - Soose -

What Do We (Have to) Cover?

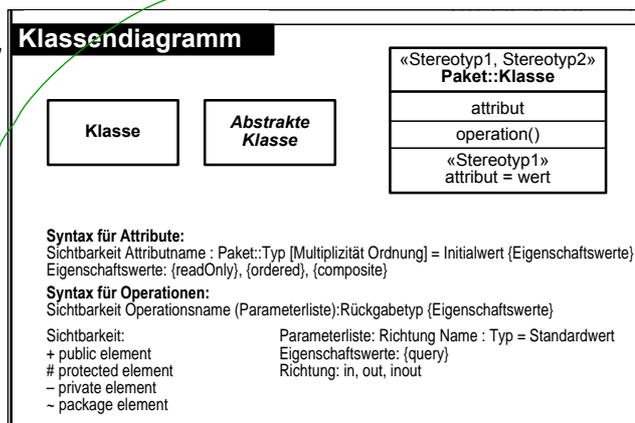
A class

- has a set of **stereotypes**,
- has a **name**,
- NOT # • belongs to a **package**,
- can be **abstract**,
- can be **active**,
- later # • has a set of **operations**,
- has a set of **attributes**.

Each attribute has

- a **visibility**,
- a **name**, a **type**,
- NOT # • 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.



-06 - 2013-11-11 - Soose -

Extended Signature

Recall: Signature

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

Too abstract to represent class diagram, e.g. no “place” to put class **stereotypes** or attribute **visibility**.

So: **Extend** definition for classes and attributes: Just as attributes already have types, we will assume that

- classes have (among other things) **stereotypes** and
- attributes have (in addition to a type and other things) a **visibility**.

Extended Classes

From now on, we assume that each class $C \in \mathcal{C}$ has:

- a finite (possibly empty) set S_C of **stereotypes**,
- a boolean flag $a \in \mathbb{B}$ indicating whether C is **abstract**,
- a boolean flag $t \in \mathbb{B}$ indicating whether C is **active**.

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

(Alternatively, we could add a set St as 5-th component to \mathcal{S} to provides the stereotypes (names of stereotypes) to choose from. But: too unimportant to care.)

Convention:

- We write

$$\langle C, S_C, a, t \rangle \in \mathcal{C}$$

when we want to refer to all aspects of C .

- If the new aspects are irrelevant (for a given context), we simply write $C \in \mathcal{C}$ i.e. old definitions are still valid.

Extended Attributes

- From now on, we assume that each attribute $v \in V$ has (in addition to the type):

- a **visibility**

$$\xi \in \underbrace{\{\text{public}\}}_{:=+}, \underbrace{\{\text{private}\}}_{:= -}, \underbrace{\{\text{protected}\}}_{:=\#}, \underbrace{\{\text{package}\}}_{:=\sim}$$

- an **initial value** $expr_0$ given as a word from **language for initial values**, e.g. OCL expressions.
(If using Java as **action language** (later) Java expressions would be fine.)
- a finite (possibly empty) set of **properties** P_v .
We define P_v analogously to stereotypes.

Convention:

- We write $\langle v : \tau, \xi, expr_0, P_v \rangle \in V$ when we want to refer to all aspects of v .
- Write only $v : \tau$ or v if details are irrelevant.

And?

- **Note:**

All definitions we have up to now **principally still apply** as they are stated in terms of, e.g., $C \in \mathcal{C}$ — which still has a meaning with the extended view.

For instance, system states and object diagrams remain mostly unchanged.

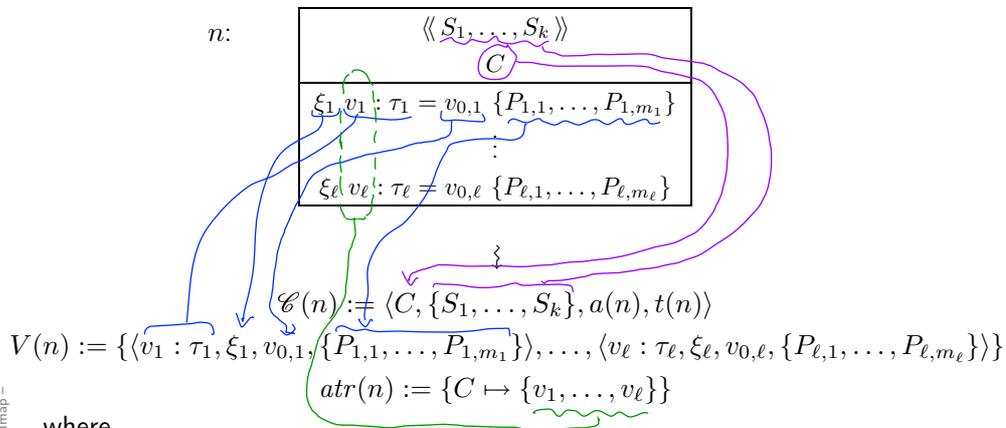
- **The other way round:** **most** of the newly added aspects **don't contribute** to the constitution of system states or object diagrams.

- Then what **are** they useful for...?
- First of all, to represent class diagrams.
- And then we'll see.

Mapping UML CDs to Extended Signatures

From Class Boxes to Extended Signatures

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



where

- “abstract” is determined by the font:
- “active” is determined by the frame:

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

- 06 - 2013-11-11 - Scimap -

13/40

What If Things Are Missing?

C
$v : Int$

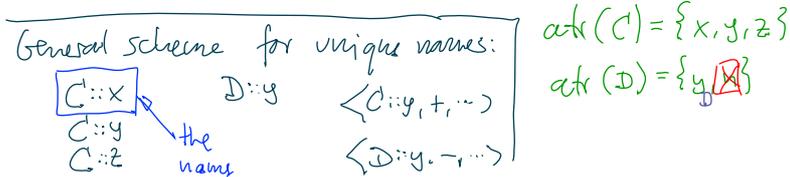
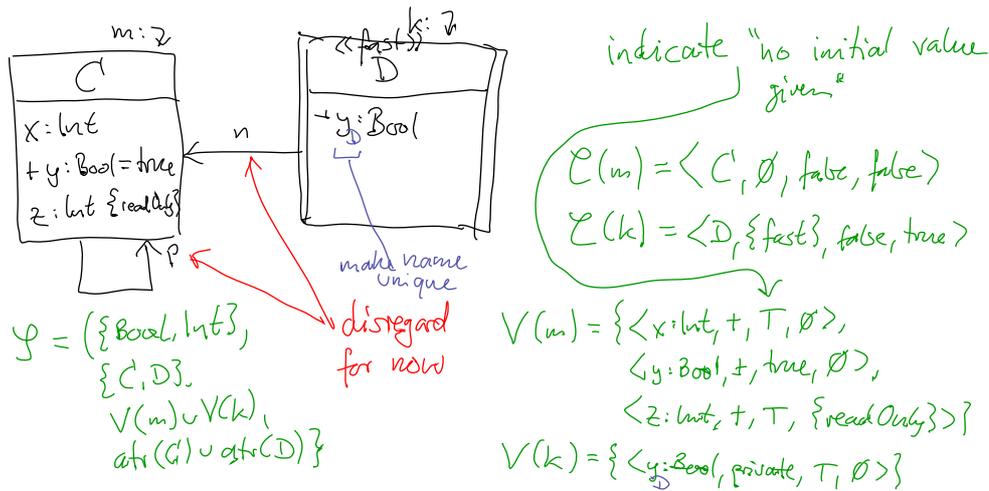
- For instance, what about the box above?
- v has **no visibility**, **no initial value**, and (strictly speaking) **no properties**.

It depends.

- What does the standard say? [OMG, 2007a, 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**.
- **Properties:** probably safe to assume \emptyset if not given at all.

- 06 - 2013-11-11 - Scimap -

14/40



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).
 - $\mathcal{E}(CD) := \bigcup_{i=1}^N \{ \mathcal{E}(n_i) \mid n_i \in CD \}$
 - $V(CD) := \bigcup_{i=1}^N V(n_i)$
 - $atr(CD) := \bigcup_{i=1}^N atr(n_i)$
- In a **UML model**, we can have **finitely many** class diagrams,

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

which **induce** the following signature:

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

(Assuming \mathcal{T} given. In "reality", we can introduce types in class diagrams, the class diagram then contributes to \mathcal{T} .)

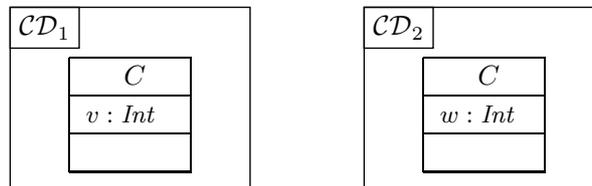
Is the Mapping a Function?

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

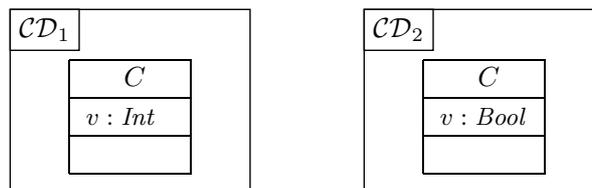
Two possible **sources for problems**:

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

(i)



(ii)



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

- 06 - 2013-11-11 - Scimap -

16/40

Is the Mapping a Function?

- (2) An **attribute** v may appear in **multiple** **classes**:



Two approaches:

- Require **unique** attribute names.

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

$$C::v \text{ or } D::v$$

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

- Subtle, formalist's approach: observe that

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

are **different things** in V . But we don't follow that path...

- 06 - 2013-11-11 - Scimap -

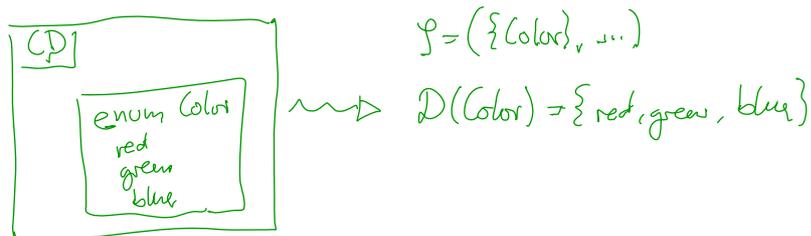
17/40

Class Diagram Semantics

Semantics

- The semantics of a set of **class diagrams** \mathcal{CD} first of all is the induced (extended) **signature** $\mathcal{S}(\mathcal{CD})$.
- The **signature** gives rise to a set of **system states** given a **structure** \mathcal{D} .
- Do we need to redefine/extend \mathcal{D} ? **No.**

(Would be different if we considered the definition of enumeration types in class diagrams. Then the domain of an enumeration type τ , i.e. the set $\mathcal{D}(\tau)$, would be determined by the class diagram, and not free for choice.)



Semantics

- The semantics of a set of **class diagrams** $\mathcal{C}\mathcal{D}$ first of all is the induced (extended) **signature** $\mathcal{S}(\mathcal{C}\mathcal{D})$.
- The **signature** gives rise to a set of **system states** given a **structure** \mathcal{D} .
- Do we need to redefine/extend \mathcal{D} ? **No.**
(Would be different if we considered the definition of enumeration types in class diagrams. Then the domain of an enumeration type τ , i.e. the set $\mathcal{D}(\tau)$, would be determined by the class diagram, and not free for choice.)
- What is the effect on $\Sigma_{\mathcal{D}}^{\mathcal{D}}$? **Little.**

For now, we only **remove** abstract class instances, i.e.

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

is now **only** called **system state** if and only if, for all $\langle C, S_C, l, t \rangle \in \mathcal{C}$,

$$\text{dom}(\sigma) \cap \mathcal{D}(C) = \emptyset.$$

With $a = 0$ as default “abstractness”, the earlier definitions apply directly. We’ll revisit this when discussing inheritance.

C is abstract



What About The Rest?

- **Classes:**
 - **Active:** not represented in σ .
Later: relevant for behaviour, i.e., how system states evolve over time.
 - **Stereotypes:** in a minute.
- **Attributes:**
 - **Initial value:** not represented in σ .
Later: provides an initial value as effect of “creation action”.
 - **Visibility:** not represented in σ .
Later: viewed as additional **typing information** for well-formedness of system transformers; and with inheritance.
 - **Properties:** such as readOnly, ordered, composite (**Deprecated** in the standard.)
 - readOnly — **later** treated similar to visibility.
 - ordered — too fine for our representation.
 - composite — cf. lecture on associations.

Stereotypes

Stereotypes as Labels or Tags

- So, a class is

$$\langle C, S_C, a, t \rangle$$

with a the abstractness flag, t activeness flag, and S_C a set of **stereotypes**.

- What are Stereotypes?

- **Not** represented in system states.
- **Not** contributing to typing rules.
(cf. **later** lecture on type theory for UML)

- [Oestereich, 2006]:

View stereotypes as (additional) "**labelling**" ("tags") or as "**grouping**".

Useful for documentation and MDA.

- **Documentation**: e.g. layers of an architecture.
Sometimes, packages (cf. the standard) are sufficient and "right".
- **Model Driven Architecture** (MDA): **later**.

References

References

- [Oestereich, 2006] Oestereich, B. (2006). *Analyse und Design mit UML 2.1*, 8. Auflage. Oldenbourg, 8. edition.
- [OMG, 2007a] OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.
- [OMG, 2007b] OMG (2007b). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.
- [Schumann et al., 2008] 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.