Making Metamodels Aware of Concrete Syntax

Frédéric Fondement, Thomas Baar

Ecole Polytechnique Fédérale de Lausanne (EPFL)

ECMDA

Nuremberg, November 9, 2005

Outline

- Motivation and Introduction
  - Modeling Language Definitions
  - Deficiencies of Informal Concrete Syntax Definitions
  - Visual Language Theory
- Our Approach
- Summary and Future Work

Modeling Language Definitions

- Proliferation of Language Definitions
  - Trend in software engineering: Describe the problem first by a tailored, domain-specific language
- Parts of a definition
  - Abstract syntax → MOF metamodel
  - Concrete syntax → often neglected
  - Semantics → often avoided

All parts of a language definition should be given in standardized format!

Concrete Syntax Definition

The concrete syntax is defined in many cases only informally.

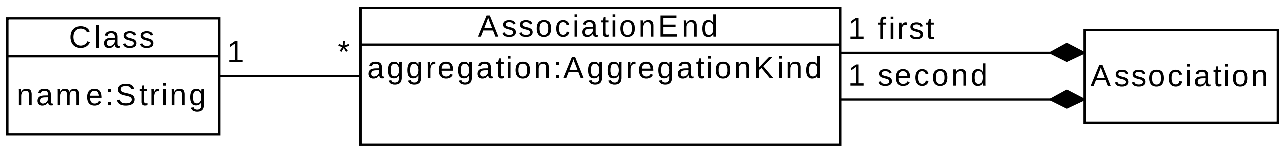
UML1.5, page:3-36, notation for Class:

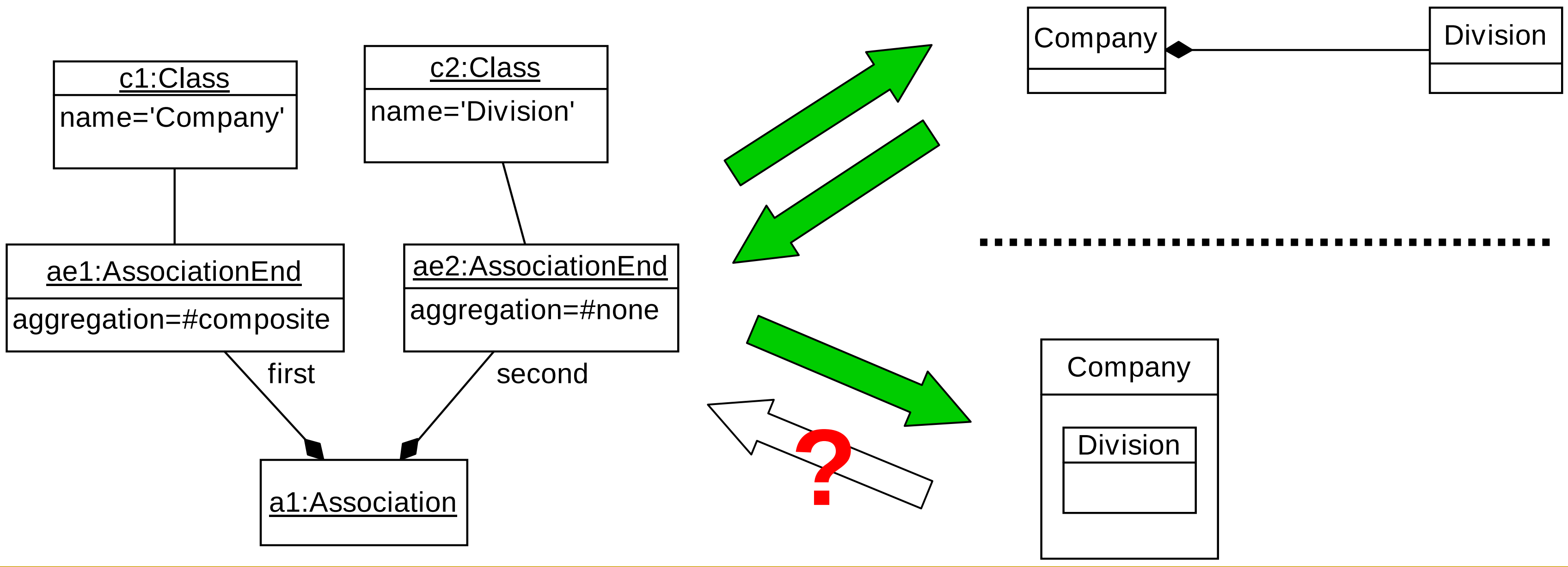
A class is drawn as a solid-outline rectangle with three compartments separated by horizontal lines. The top name compartment holds the class name and other general properties of the class (including stereotype); the middle list compartment holds a list of attributes; the bottom list compartment holds a list of operations.

UML1.5, page 3-81, notation for Composition:

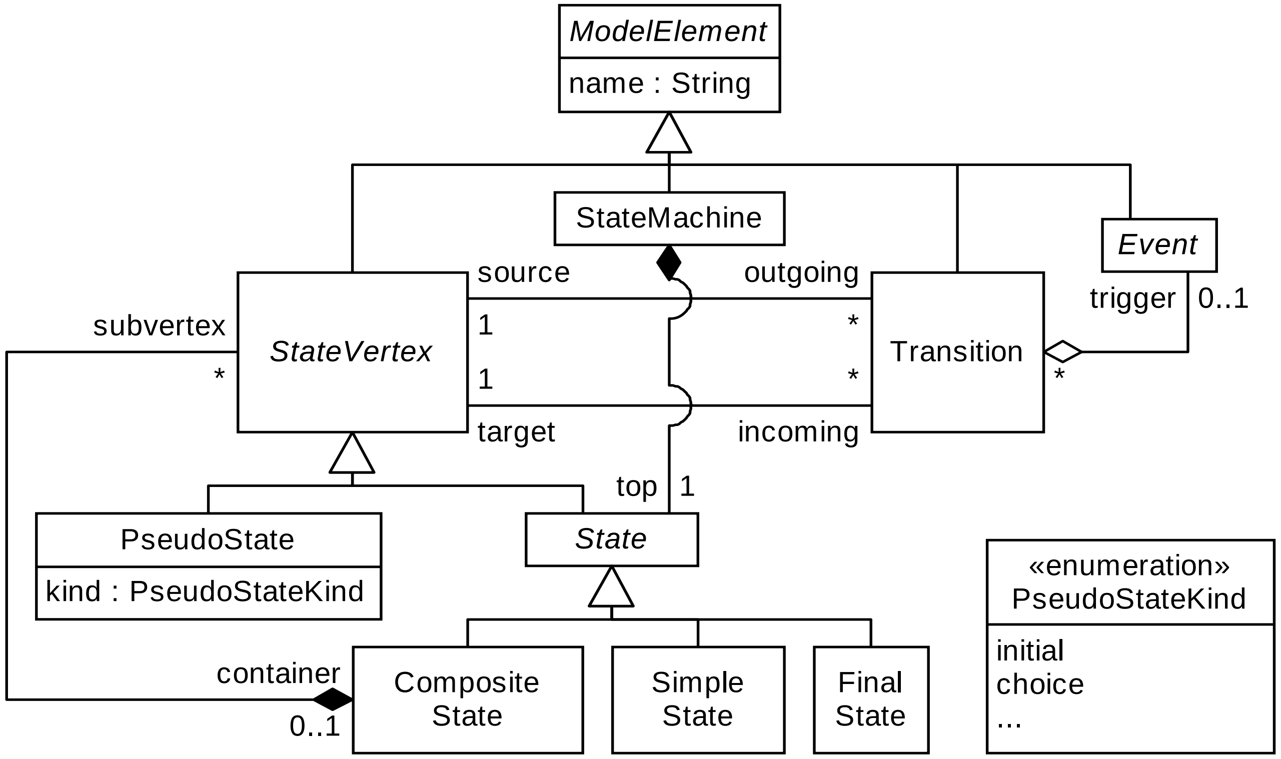
Composition may be shown by a solid filled diamond as an association end adornment.
Instead of using binary association paths using the composition aggregation adornment, composition may be shown by graphical nesting of the symbols of the elements for the parts within the symbol of the element for the whole.

Relationship Concrete-Abstract Syntax

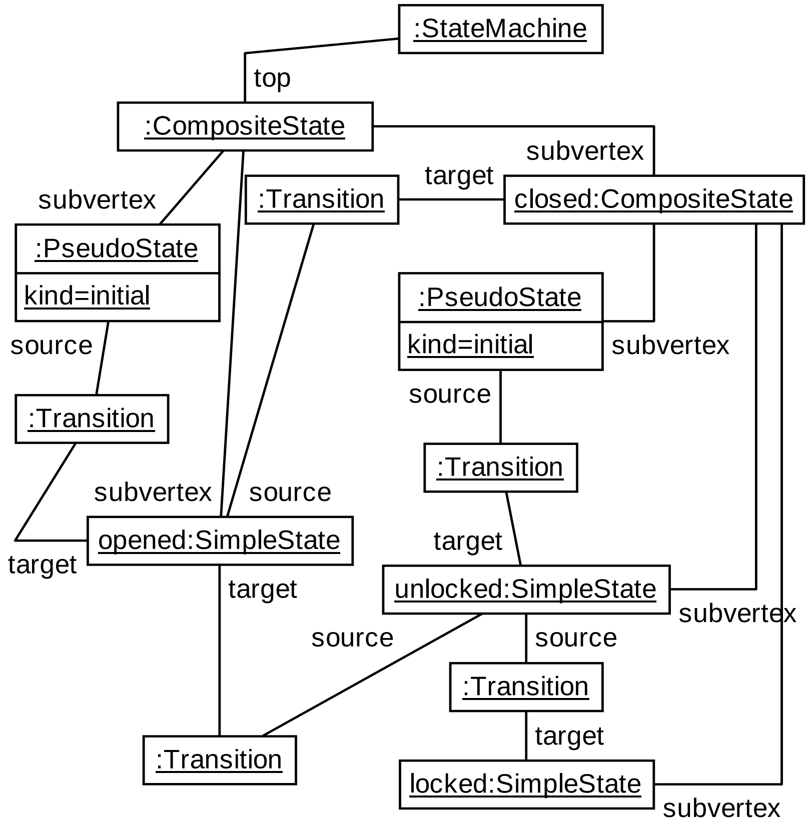


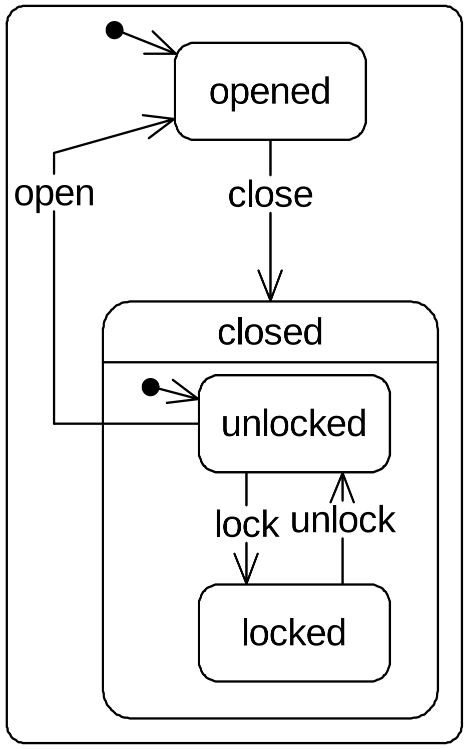


Metamodel for Statecharts



Abstract/Concrete Syntax for Statecharts





Can we make concrete syntax definition as formal and precise that one can decide automatically on the correctness of the graphical rendering?

Visual Language

A diagram (i.e. a visual language sentence) is given by

- Set of visual elements (e.g. rectangles, lines, text)
  - visual elements can be seen as objects having attributes such as `shape`, `color`, (`position`), `attach region`
- Relationship between elements
  - `connectedWith`

    → connection-based language
  - spatial relationships (`right`, `left`, `overlap`, `contain`)

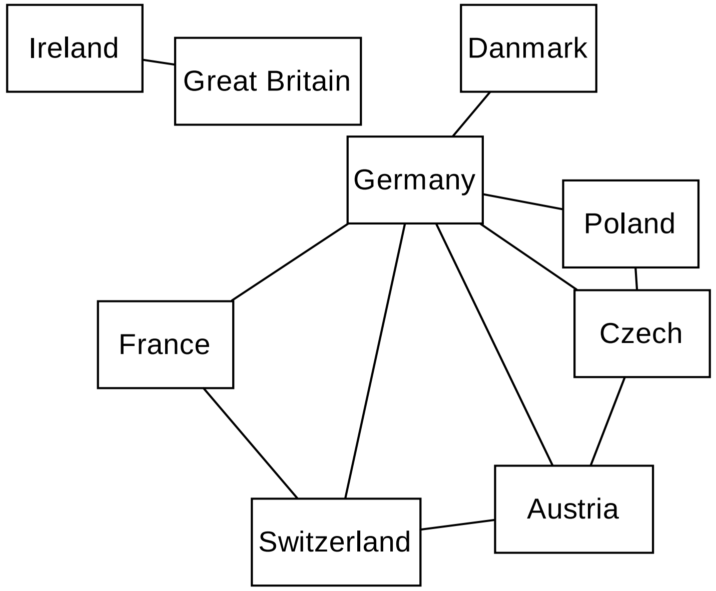
    → geometry-based language

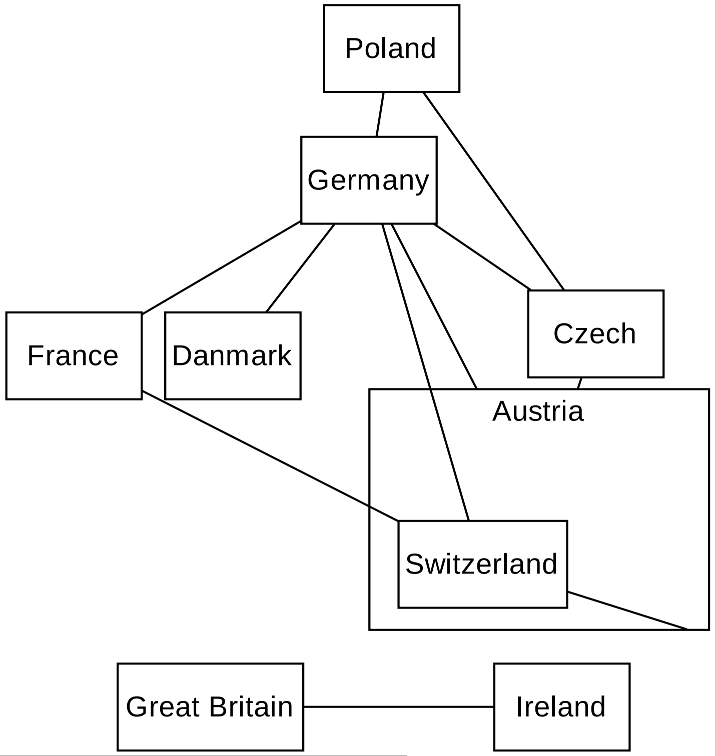
It highly depends from the visual language we want to define *which* of the attributes and relationships are relevant!

VL-Example – Border Diagram

Border Diagram:

- Shows for each country its neighbors
- The only important information are the connecting lines, spatial information do not play any role in this VL.

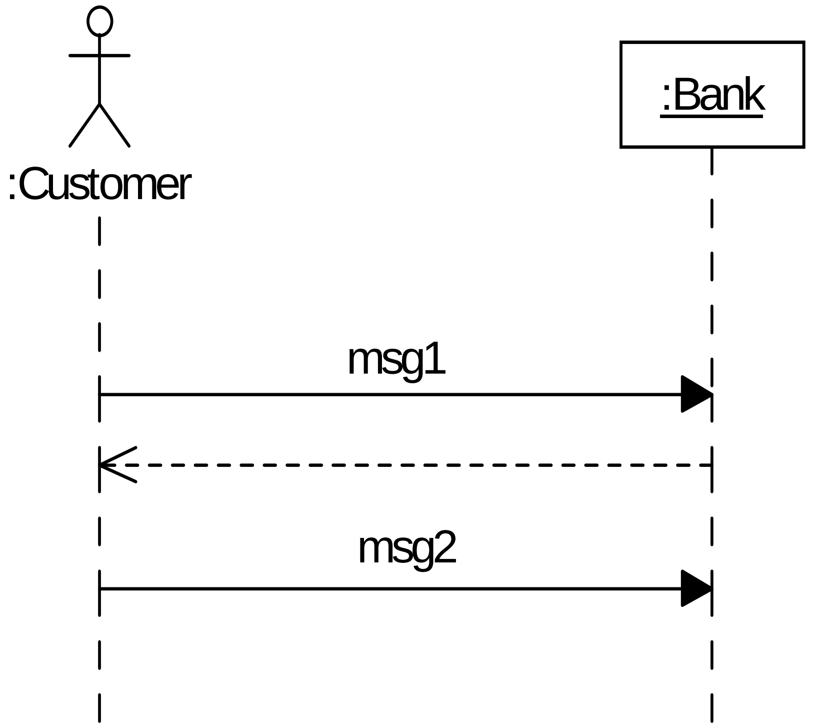




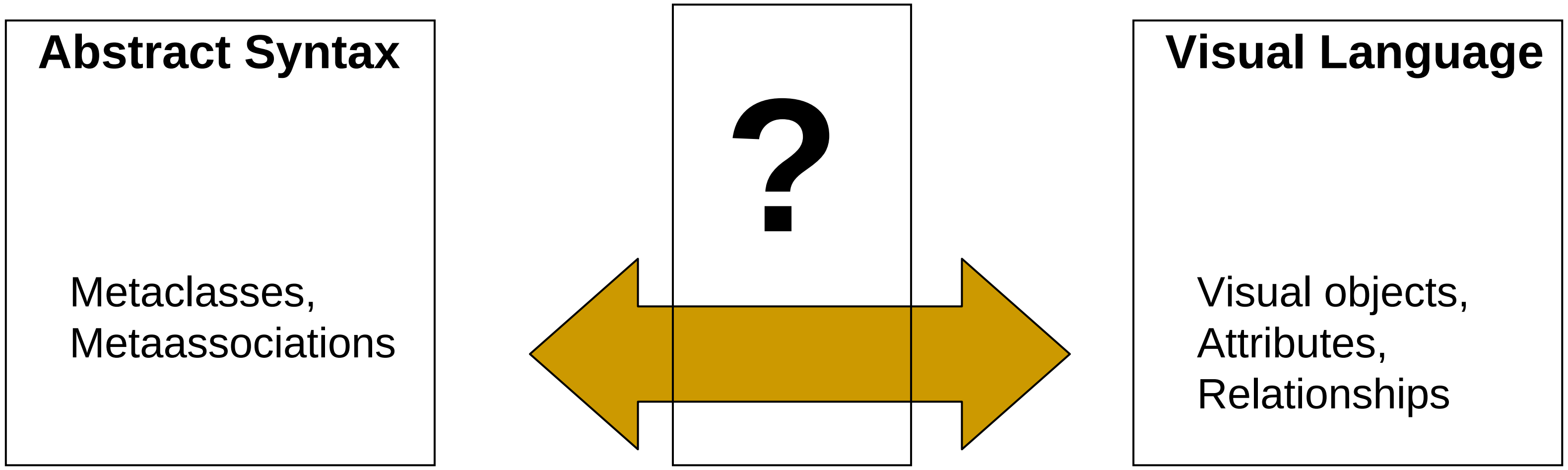
Meaning: Connected countries have direct border

VL Example –Sequence Diagram

In Sequence Diagrams the spatial relationships between message arrows are important because they indicate the ordering of sent messages.



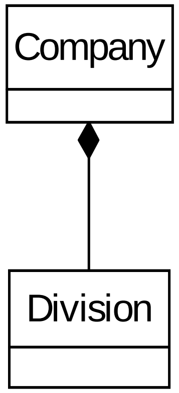
Steps to Define Complete Syntax

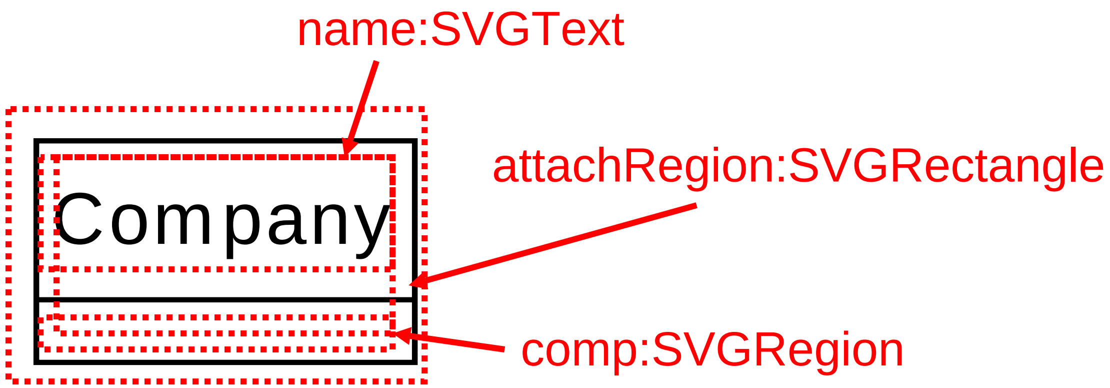


Steps:

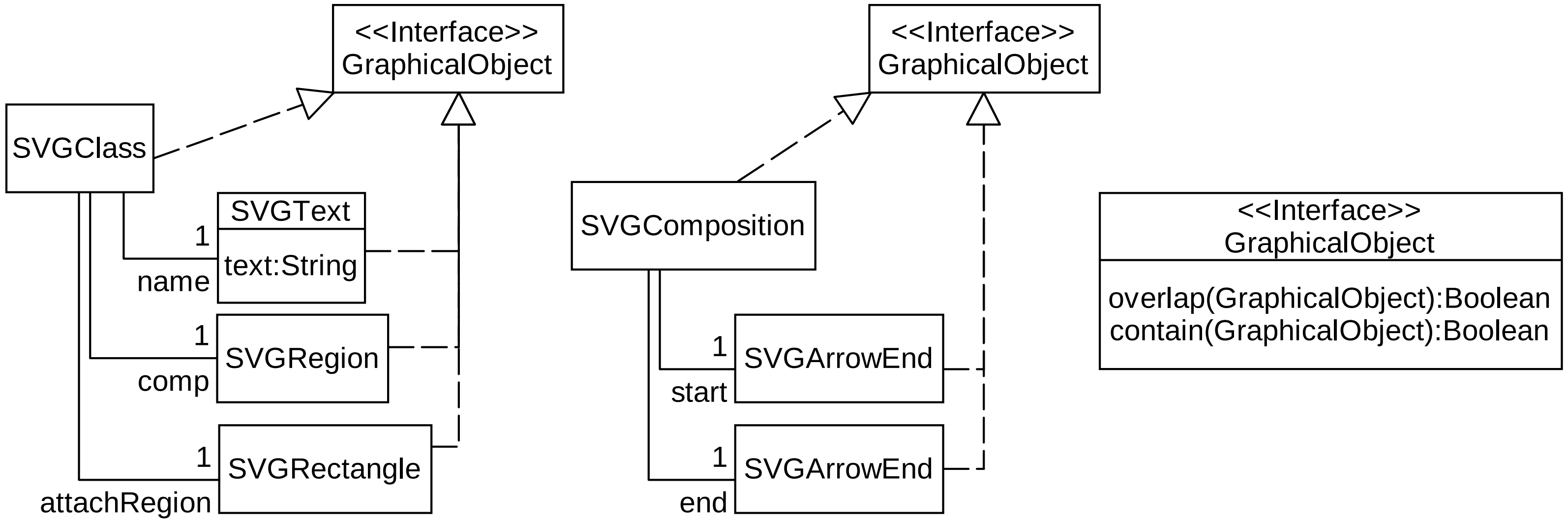
0) Define the abstract syntax
1) Define the visual language
   - define relevant attributes and relationships
2) Define how instances of the metamodel are represented by sentences of visual language

Define a Visual Language



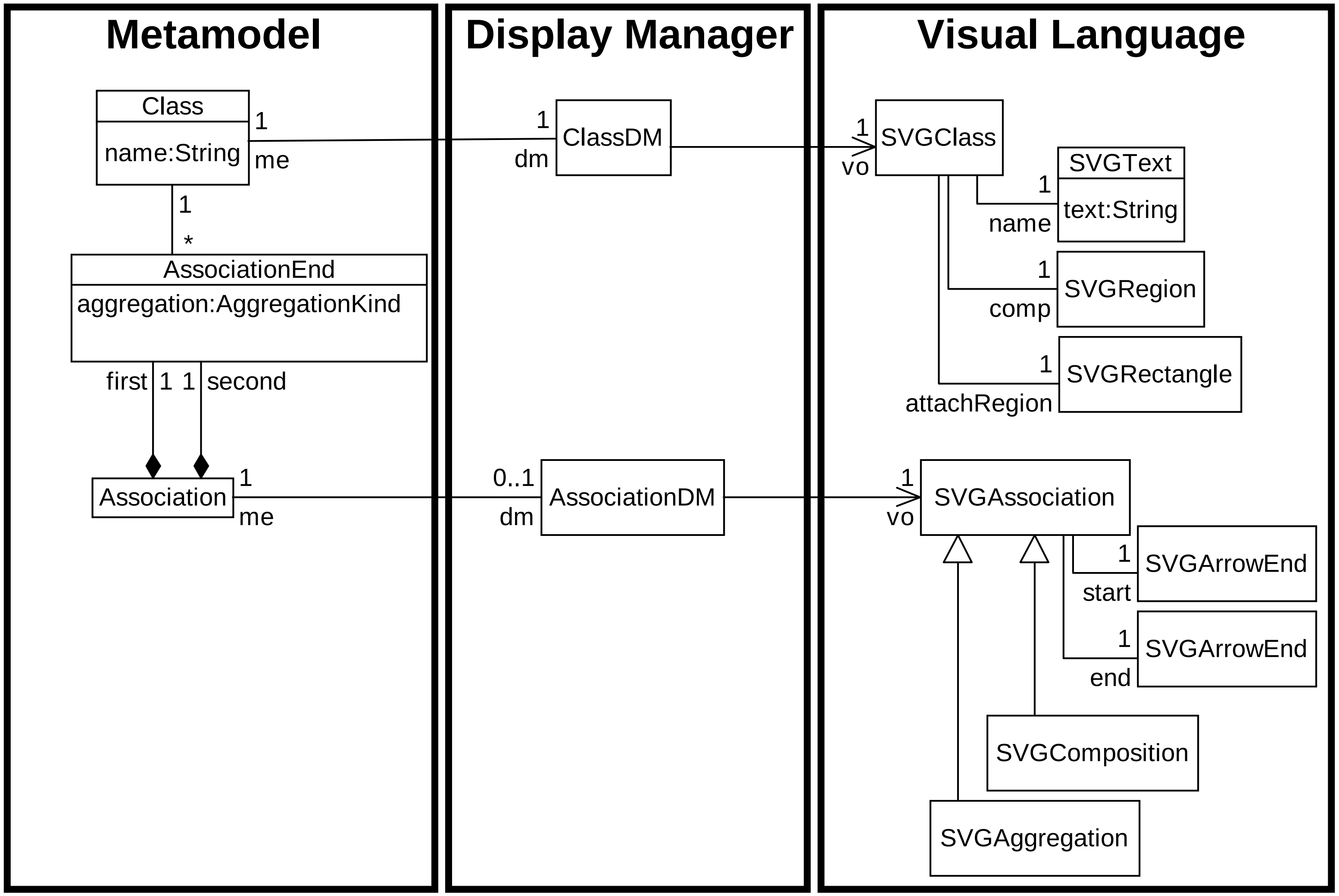


Model of Visual Language:



<<Interface>> GraphicalObject
SVGClass
SVGText
text:String
1
name
1
SVGRegion
comp
1
SVGRectangle
attachRegion
<<Interface>> GraphicalObject
SVGComposition
1
SVGArrowEnd
start
1
SVGArrowEnd
end
<<Interface>> GraphicalObject
overlap(GraphicalObject):Boolean
contain(GraphicalObject):Boolean

Connecting Abstract and Concrete Syntax



Connecting Abstract and Concrete Syntax

The precise description of the relationship between abstract and concrete syntax is given declaratively in terms of OCL constraints.

```
-- the name of the class is the same as the text shown in the text field of the class
      rectangle
context Class inv:
   self.name=self.dm.vo.name.text

-- composition is shown either by adorned association or by nesting
context Association inv:
   self.first.aggregation=#composition implies
   (-- nesting of symbols
    self.first.class.dm.vo.comp.contain(self.second.class.dm.vo) or
    -- composite association
    (self.dm.vo.isKindOf(SVGComposite) and
     self.first.class.dm.vo.attachRegion.overlap(self.dm.vo.start) and
     self.second.class.dm.vo.attachRegion.overlap(self.dm.vo.end)))
```

Reflect ambiguous informal description (= can be used instead)

Using spatial relationships

Summary

- Framework to connect abstract and concrete syntax
  - strict separation between abstract and concrete syntax
  - visual language is represented by
    - relevant attributes of visual elements
    - relevant relationships between visual elements
- Fully declarative description of connection of abstract/concrete syntax
  - usage of OCL

Future work

- Implementation of the framework
- Generation of reference editors
- Finding criteria for well-formedness of abstract/concrete syntax mapping
  - Are all information of a given model (instance of metamodel) represented in a non-ambiguous way?
  - Once the presentation options are fixed, is the mapping from abstract to concrete syntax injective?