Multi-Level Modelling in the Modelverse

Simon Van Mierlo - simonvanmierlo@uantwerpen.be
Bruno Barroca – bbarroca@cs.mcgill.ca
Hans Vangheluwe – hv@cs.mcgill.ca
Eugene Syriani – syriani@iro.umontreal.ca
Thomas Kühne – thomas.kuehne@ecs.vuw.ac.nz

Motivation

Multi-level modelling is used to overcome limitations of two-level modelling systems (such as UML).

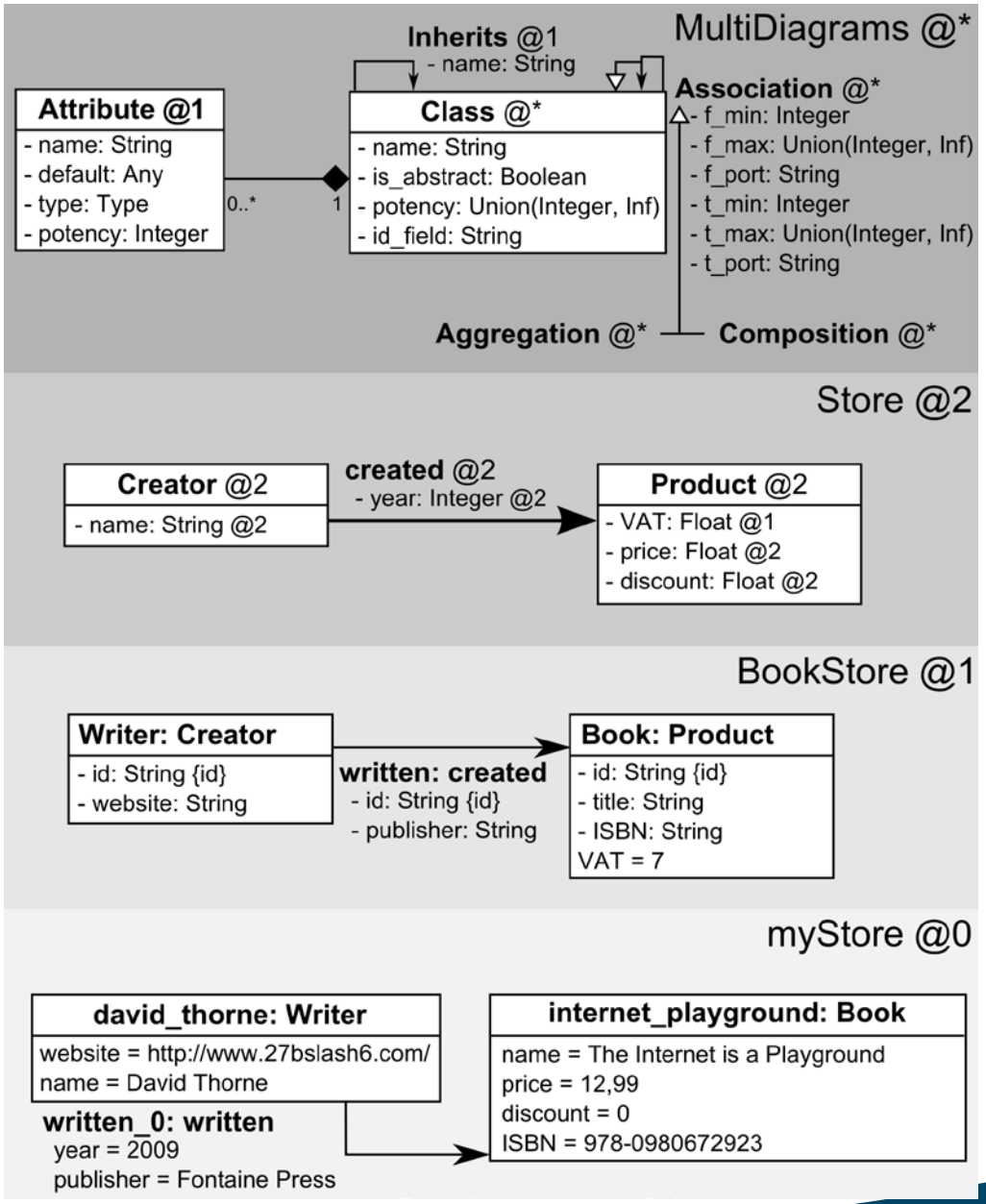
Language designers need to be able to specify multi-level linguistic hierarchies based on the concepts of *deep instantiation* and *deep characterization*.

Current tools often do not always differentiate between the *linguistic* and the *physical* dimension.

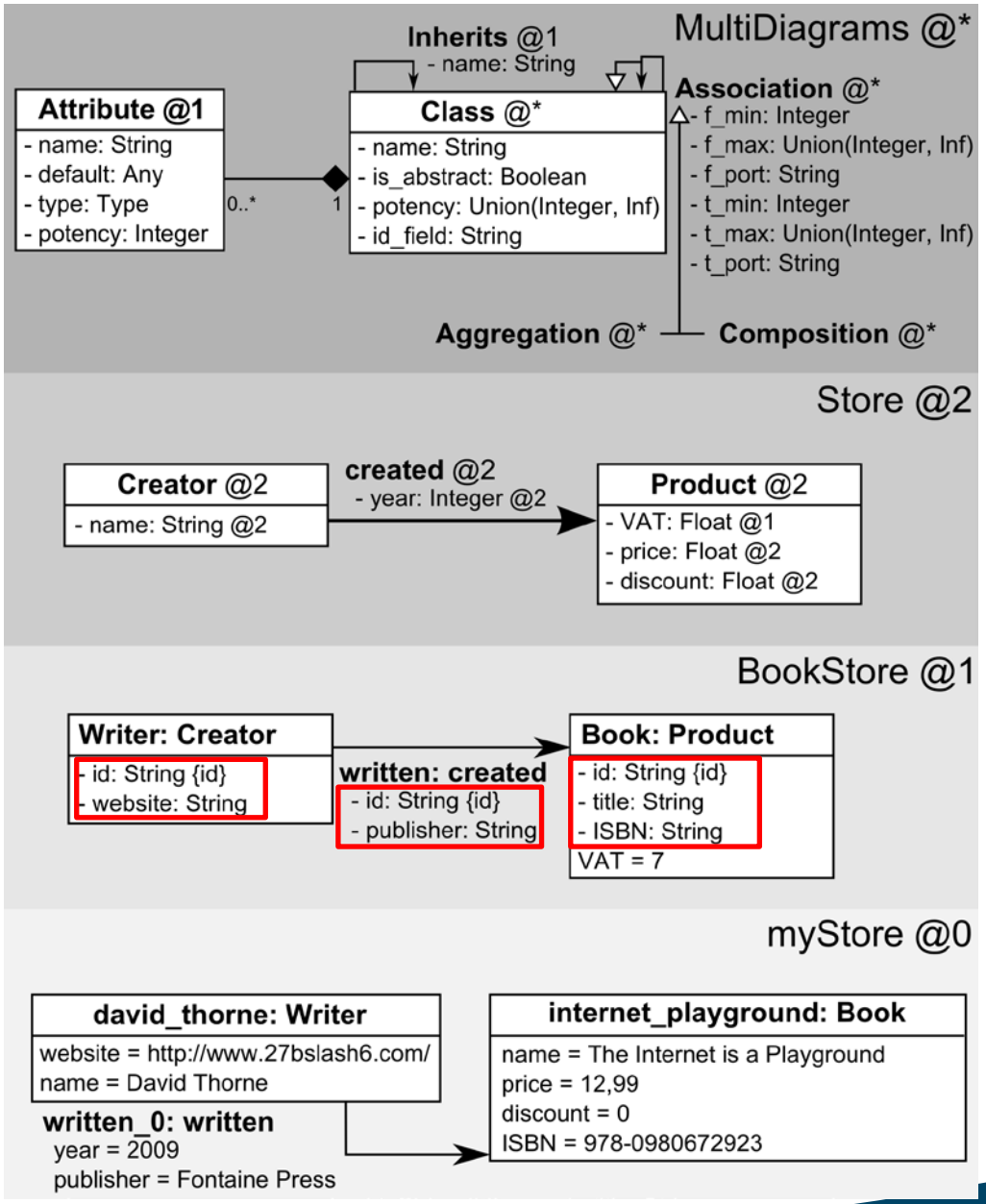
To improve reuse, language designer should be able to reuse (parts of) language definitions.

Contents

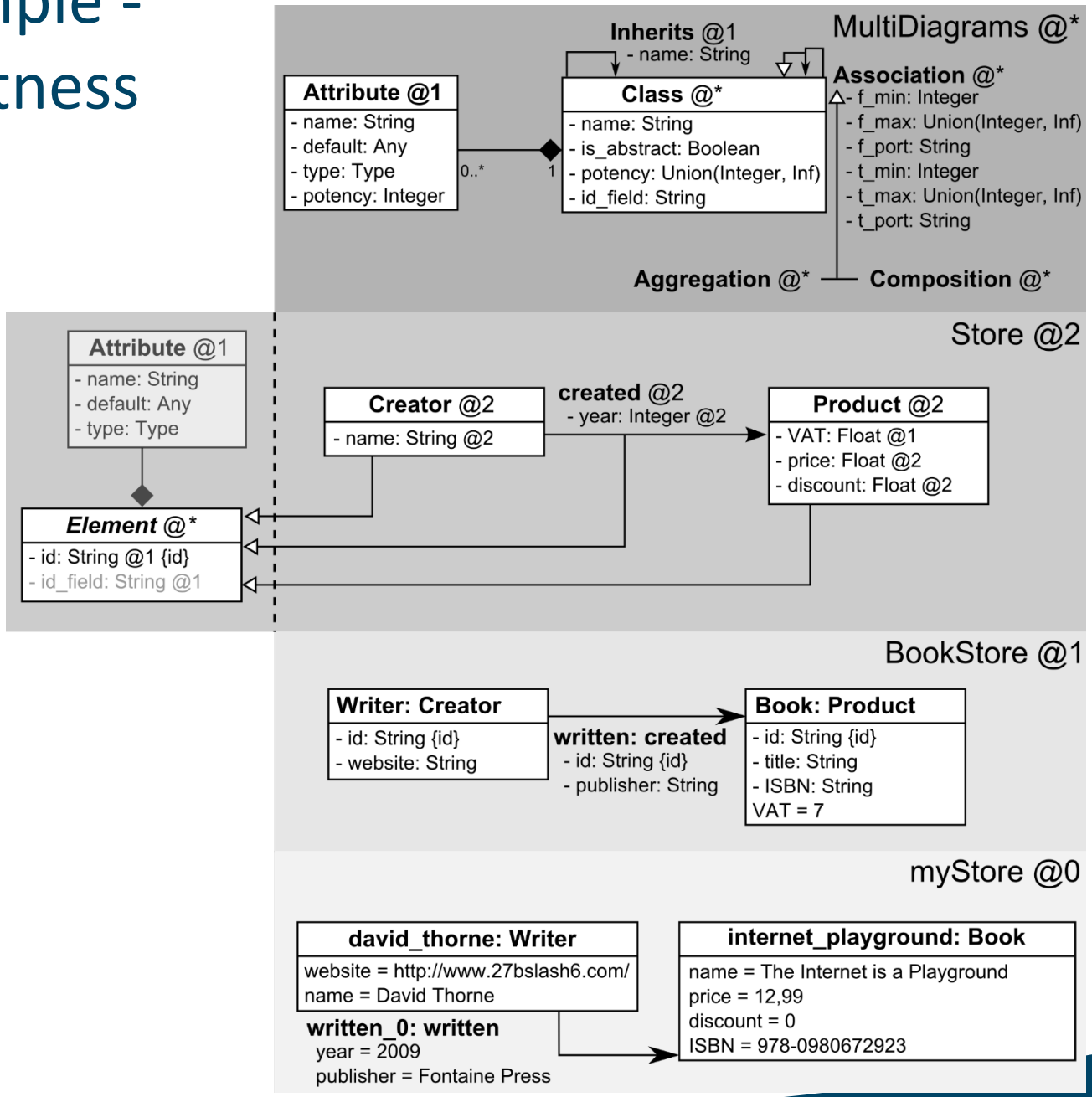
Example


MultiDiagrams @*
Inherits @1
- name: String
Attribute @1
- name: String
- default: Any
- type: Type
- potency: Integer
0..*
1
Class @*
- name: String
- is_abstract: Boolean
- potency: Union(Integer, Inf)
- id_field: String
Association @*
- f_min: Integer
- f_max: Union(Integer, Inf)
- f_port: String
- t_min: Integer
- t_max: Union(Integer, Inf)
- t_port: String
Aggregation @*
Composition @*
Store @2
Creator @2
- name: String @2
created @2
- year: Integer @2
Product @2
- VAT: Float @1
- price: Float @2
- discount: Float @2
BookStore @1
Writer: Creator
- id: String {id}
- website: String
written: created
- id: String {id}
- publisher: String
Book: Product
- id: String {id}
- title: String
- ISBN: String
VAT = 7
myStore @0
david_thorne: Writer
website = http://www.27bslash6.com/
name = David Thorne
written_0: written
year = 2009
publisher = Fontaine Press
internet_playground: Book
name = The Internet is a Playground
price = 12,99
discount = 0
ISBN = 978-0980672923

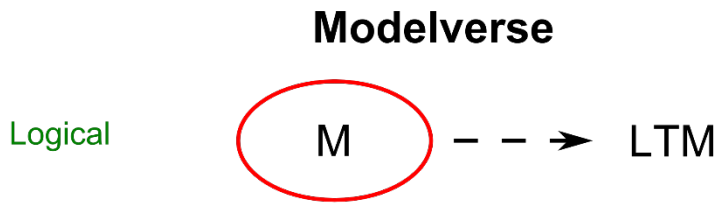
Example - Strictness



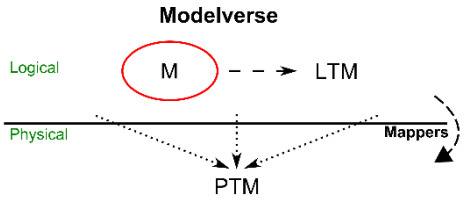
Example - Strictness

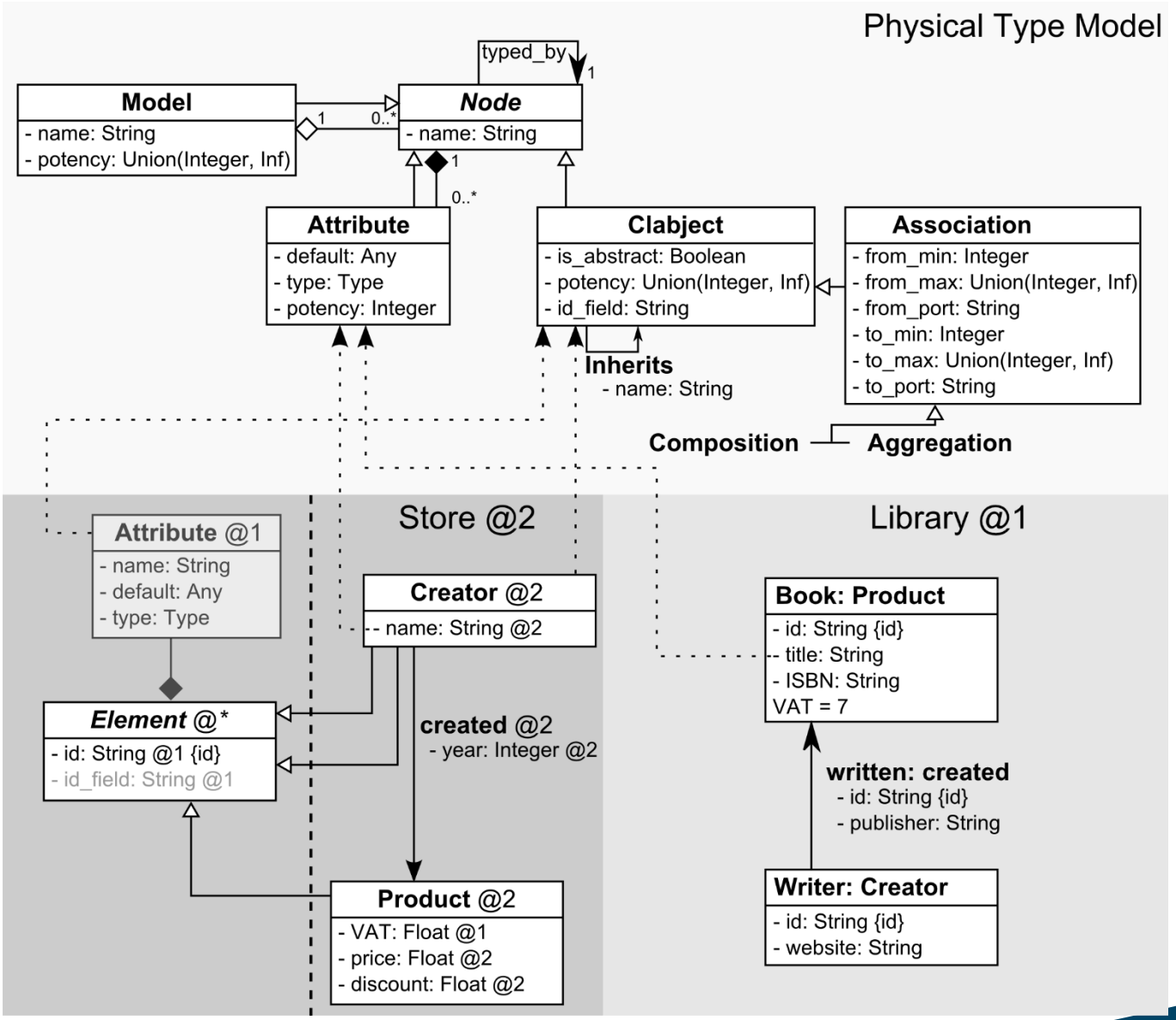


The Modelverse: Architecture



Mappers


Modelverse
Logical
M
LTM
Physical
Mappers
PTM


Physical Type Model
typed_by
1
Model
- name: String
- potency: Union(Integer, Inf)
1
0..*
Node
- name: String
1
0..*
Attribute
- default: Any
- type: Type
- potency: Integer
Clabject
- is_abstract: Boolean
- potency: Union(Integer, Inf)
- id_field: String
Association
- from_min: Integer
- from_max: Union(Integer, Inf)
- from_port: String
- to_min: Integer
- to_max: Union(Integer, Inf)
- to_port: String
Inherits
- name: String
Composition
Aggregation
Store @2
Library @1
Attribute @1
- name: String
- default: Any
- type: Type
Creator @2
- name: String @2
Element @*
- id: String @1 {id}
- id_field: String @1
created @2
- year: Integer @2
Product @2
- VAT: Float @1
- price: Float @2
- discount: Float @2
Book: Product
- id: String {id}
- title: String
- ISBN: String
VAT = 7
written: created
- id: String {id}
- publisher: String
Writer: Creator
- id: String {id}
- website: String

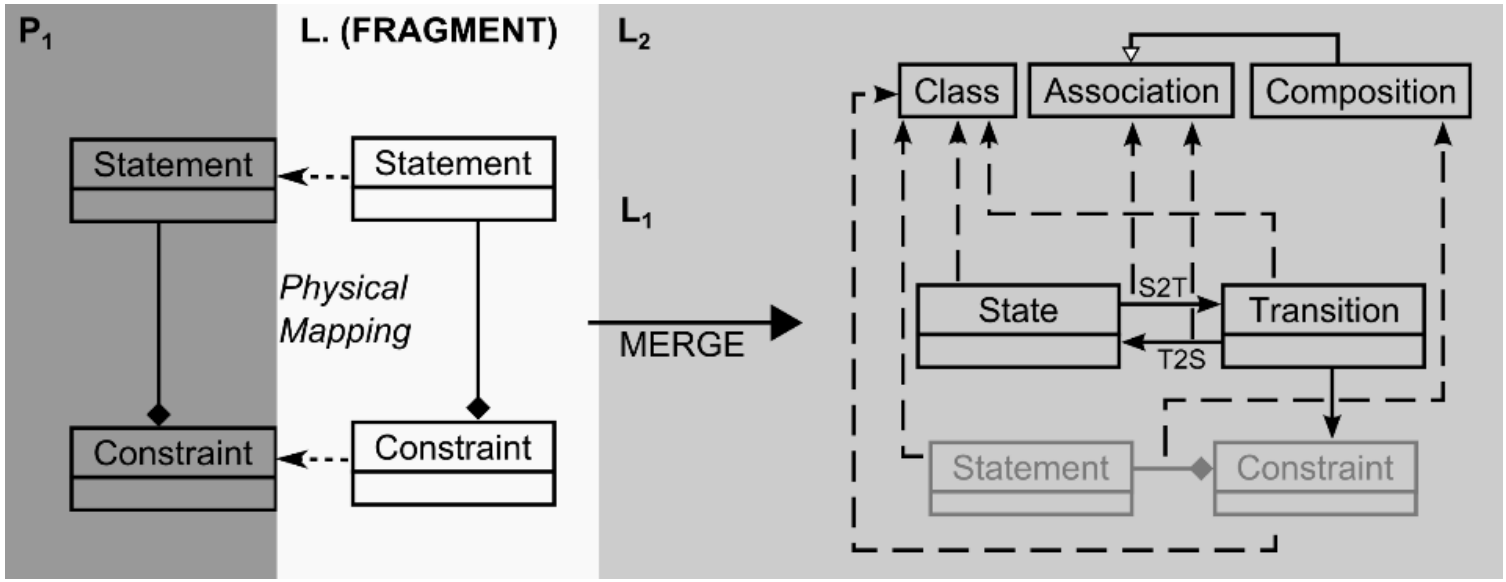
Language Fragments

- Reuse in Languages:
  - Abstract Syntax
  - Concrete Syntax
  - Physical Mapping

- Fragments are “merged” in a linguistic type model.

- Library of Fragments

Language Fragments - Example



Conclusion and Future Work

We've shown how to model in the Modelverse (using the MvK), how the distinction between the logical (linguistic) and physical dimension is made, and how language reuse is possible using fragments.

Future Work:

- Fragments: automatic merging, concrete syntax and semantics, define a library.
- Ontological Conformance
- Representers for distributing the Modelverse.
- Explicit modelling of mappers.

Modelling in the Modelverse - HUTN

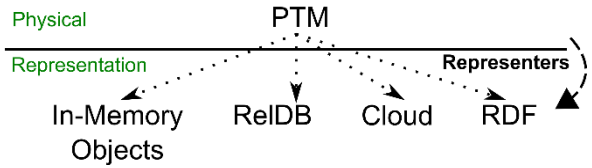
```
package MyFormalisms:
 Model:
  name = 'Store'
  potency = 2

  Class:
   name = 'Element'
   potency = *
   is_abstract = True
   id_field = 'id'

   Attribute:
    name = 'id'
    type = String

   Attribute:
    name = 'id_field'
    type = String

  Class:
   name = 'Product'

   Attribute:
    name = 'VAT'
    type = Float
   potency = 1

   Attribute:
    name = 'price'
    type = Float

   Attribute:
    name = 'discount'
    type = Float

  Inherits:
   name = 'product_i_element'
   from_clabject = 'Product'
   to_clabject = 'Element'

  Class:
   name = 'Creator'

   Attribute:
    name = 'name'
    type = String

  Inherits:
   name = 'creator_i_element'
   from_clabject = 'Creator'
   to_clabject = 'Element'

  Association:
   name = 'created'

   Attribute:
    name = 'year'
    type = Integer

  Inherits:
   name = 'created_i_element'
   from_clabject = 'created'
   to_clabject = 'Element'
```

Listing 1. Textual notation for Model Store

```
package MyFormalisms:
 Store:
  name = 'Library'
  potency = 1

  Product:
   name = 'Book'
   id_field = 'id'
   VAT = 7

   Attribute:
    name = 'id'
    type = String

   Attribute:
    name = 'title'
    type = String

   Attribute:
    name = 'ISBN'
    type = String

  Creator:
   name = 'Writer'
   id_field = 'id'

   Attribute:
    name = 'id'
    type = String

   Attribute:
    name = 'website'
    type = String

  created:
   name = 'written'
   id_field = 'id'

   Attribute:
    name = 'id'
    type = String

   Attribute:
    name = 'publisher'
    type = String
```

Listing 2. Textual notation for Store Library

Representers



- Responsible for instantiation of PTM elements on medium.
  - Objects in Memory
  - Relational Databases
  - RDF
- Allows to represent on most appropriate medium.
- User is not aware of where elements are stored.