Using Metamodels for the Definition of Languages

"A Metamodel for SDL-2000 in the Context of Metamodelling ULF"

Joachim Fischer, Michael Piefel, Markus Scheidgen

HUMBOLDT-UNIVERSITÄT ZU BERLIN

Abstract

- Language specification in the world of Model Driven Software Engineering
  - Software is visually notated through graphs
    - Exchange of software artefacts
    - (It is hard to specify concrete syntax)
  - Alignment, relations and transformations between languages
    - Common method for language specification, common “Meta-Metamodel”
  - Software engineering process – Language families
- Position: Grammars are insufficient to do the job. We need metamodels.

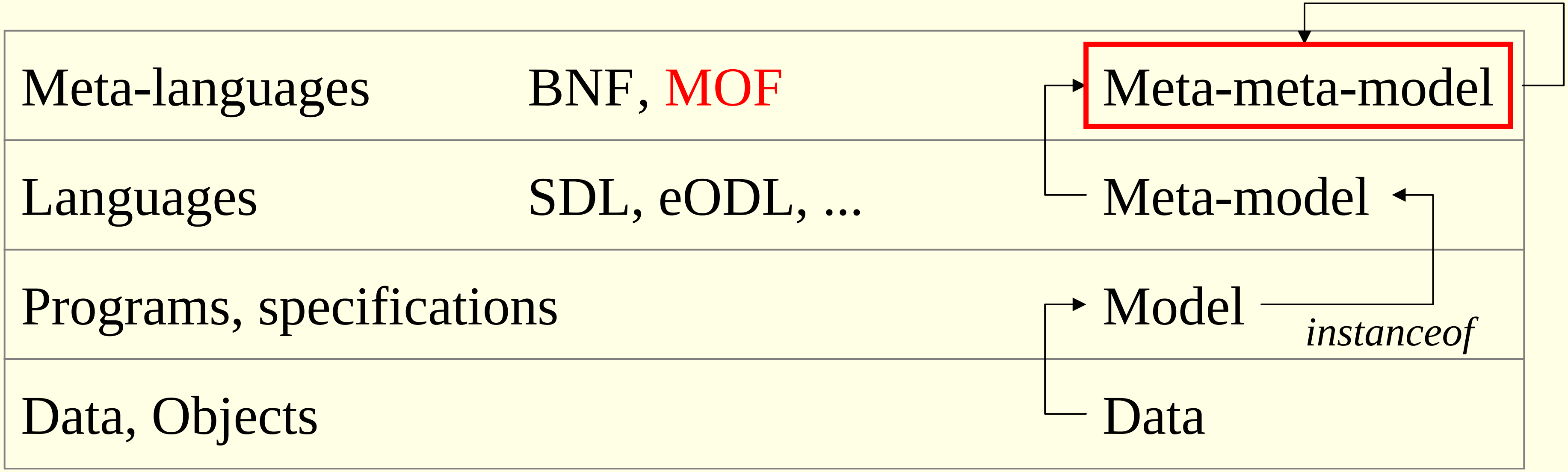
Agenda

- Introduction: Metamodelling
- Our research motivation
- The problem
- The method: „*From Grammars to Metamodels*“
- Conclusions

The Meta-affix

- Etymology: *Meta* Greek affix, engl. beyond
- Continuation of the traditional class-object paradigm
  - An describing meta-element(class) classifies a set of instances(objects). Thus a class is a meta-object.
  - But meta- is a relational affix an can also be applied to classes, and can be used recursively, forming a hierarchy: *Object*, *Class(Meta-object)*, *Meta-class*, *Meta-Meta-class*, *etc.*

Multi-layer architecture



Meta-languages	BNF, MOF	Meta-meta-model
Languages	SDL, eODL, …	Meta-model
Programs, specifications		Model
Data, Objects		Data

- Mostly, when talking metamodelling, ment is a special meta-meta-model
- Object-orientated, UML-class-diagram like, most common is the UML meta-meta-model: MOF-Model

BNF vs. Metamodel

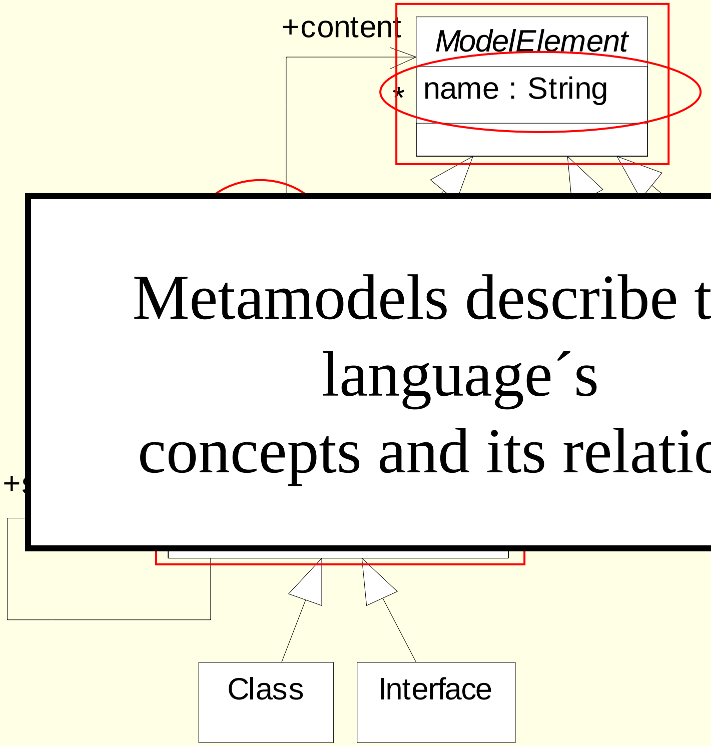
```
package= name

cl

in

id
```

Grammars describe the structure of the
words in a language



Metamodels describe the language´s
concepts and its relations

context ModelElement inv: not oclType()=package implies container->size()=1
context GeneralizableElement inv: supertype->forAll(oclType() = this.oclType())
context Package inv: set{Class, Interface}->includesAll(content->collect(oclType()))
context Interface inv: set{Method}->includesAll(content->collect(oclType()))
context Class inv: set{Method, Attribute}->includesAll(content->collect(oclType()))
inv: supertype.size() <= 1

Motives

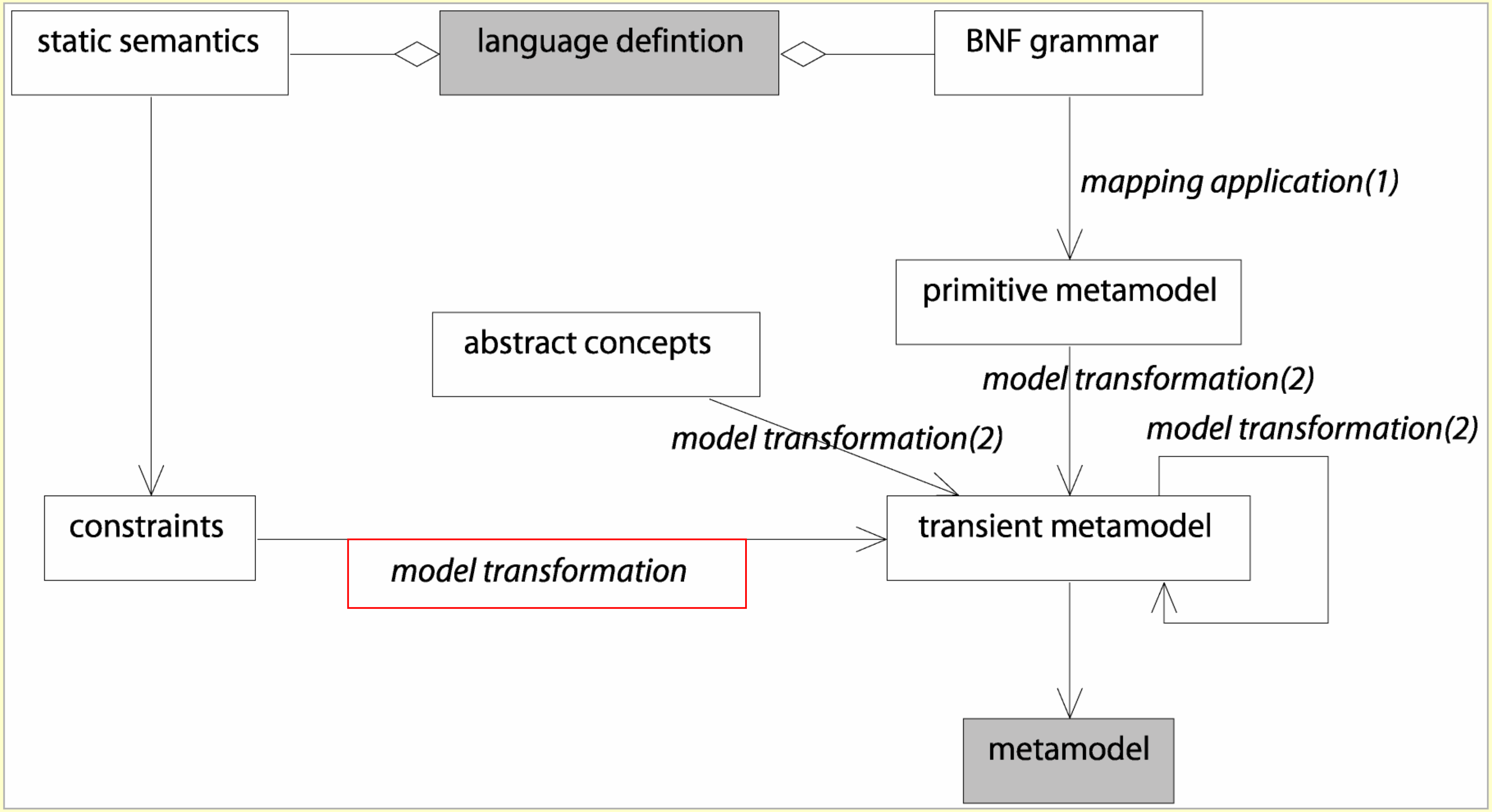
- Model Driven Software Engineering
  - Requires mappings and transformations between languages → a common abstract syntax definition method is required
  - Concrete: transformation from eODL to SDL-2000 to drive projects from structural design to implementation
- Languages families, ULF
  - ITU-T: SDL, MSC, eODL, TTCN, ASN1 unified in ULF
  - Metamodels as a unified syntax definition mechanism

New requirements, Problem

- Language definition requirements, requirements at the Meta-Meta-level
  - Refinable buildings blocks
  - Namespaces, Packages
  - It must be possible to align different models
- Metamodels fulfil these requirements, grammars do not

→ There are many existing grammar based language definitions that a metamodel shall be developed for, concrete need for a SDL metamodel

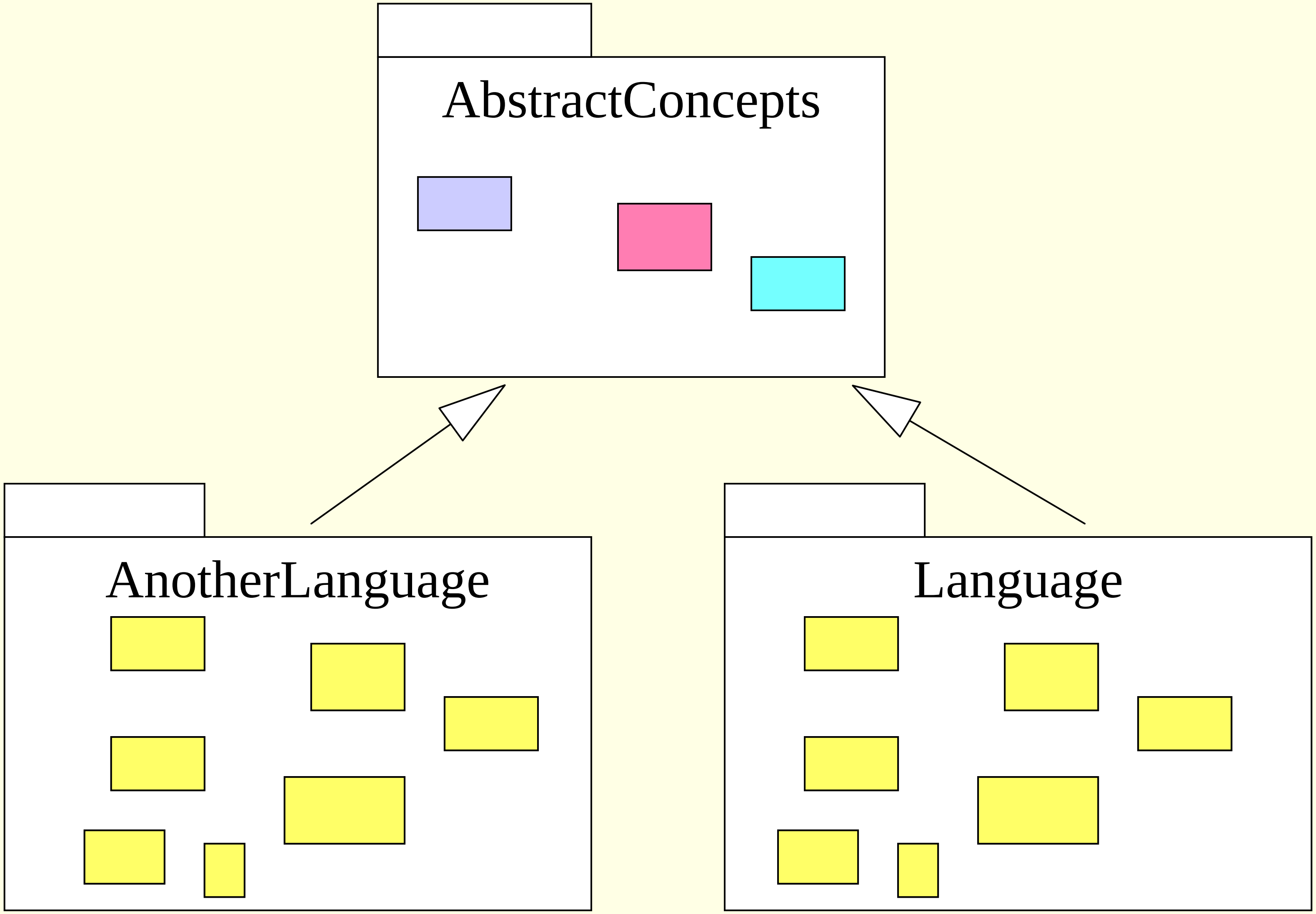
A Method: Metamodels from existing syntax definitions



The method’s characteristics

- Automatic generation of a primitive metamodel
- Use human input for metamodel refinement:
  - Abstract model elements
  - Semantic information about the abstract nature of concepts
- Automated model transformation bases on this input

In the context of multiple languages

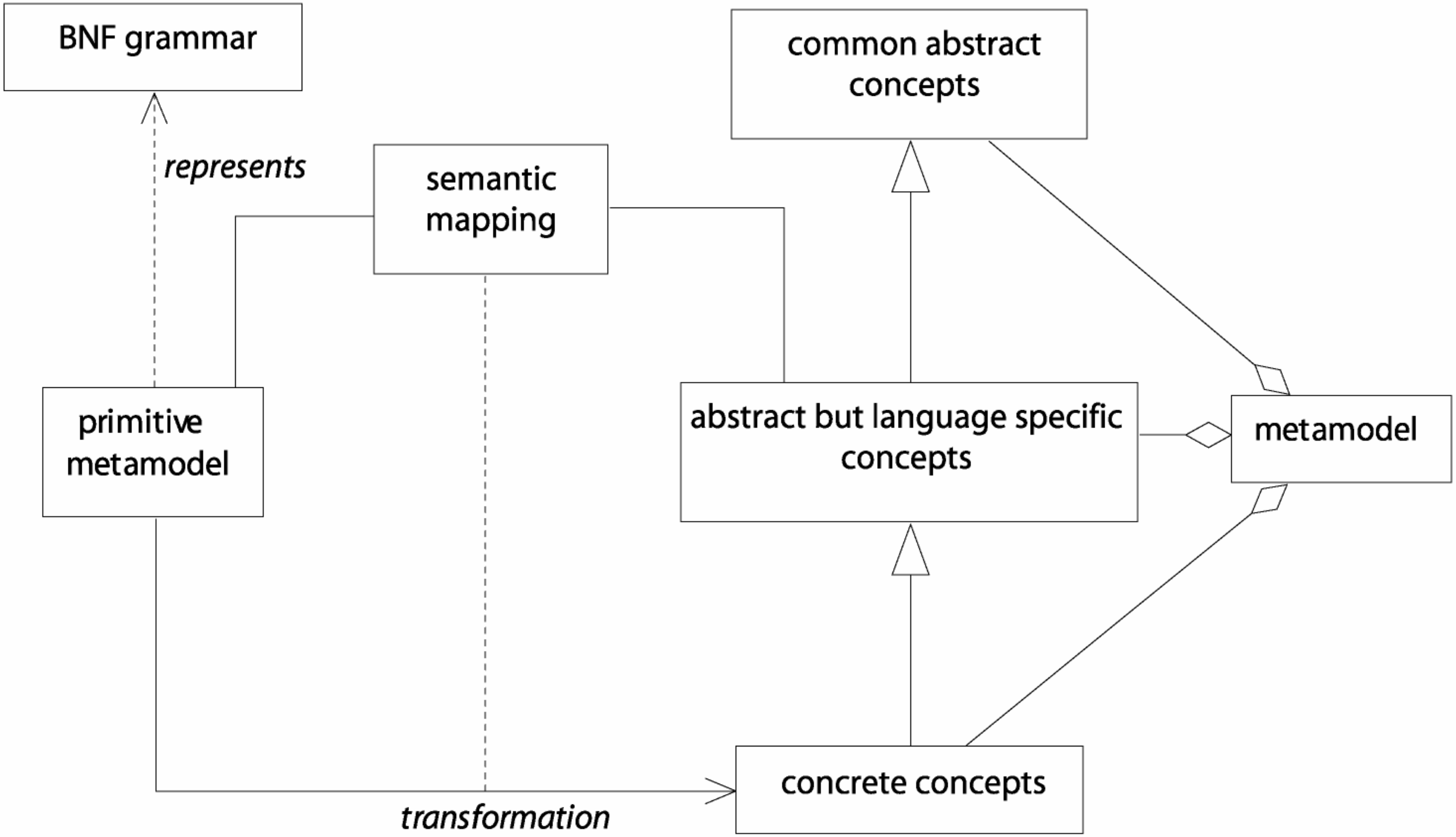


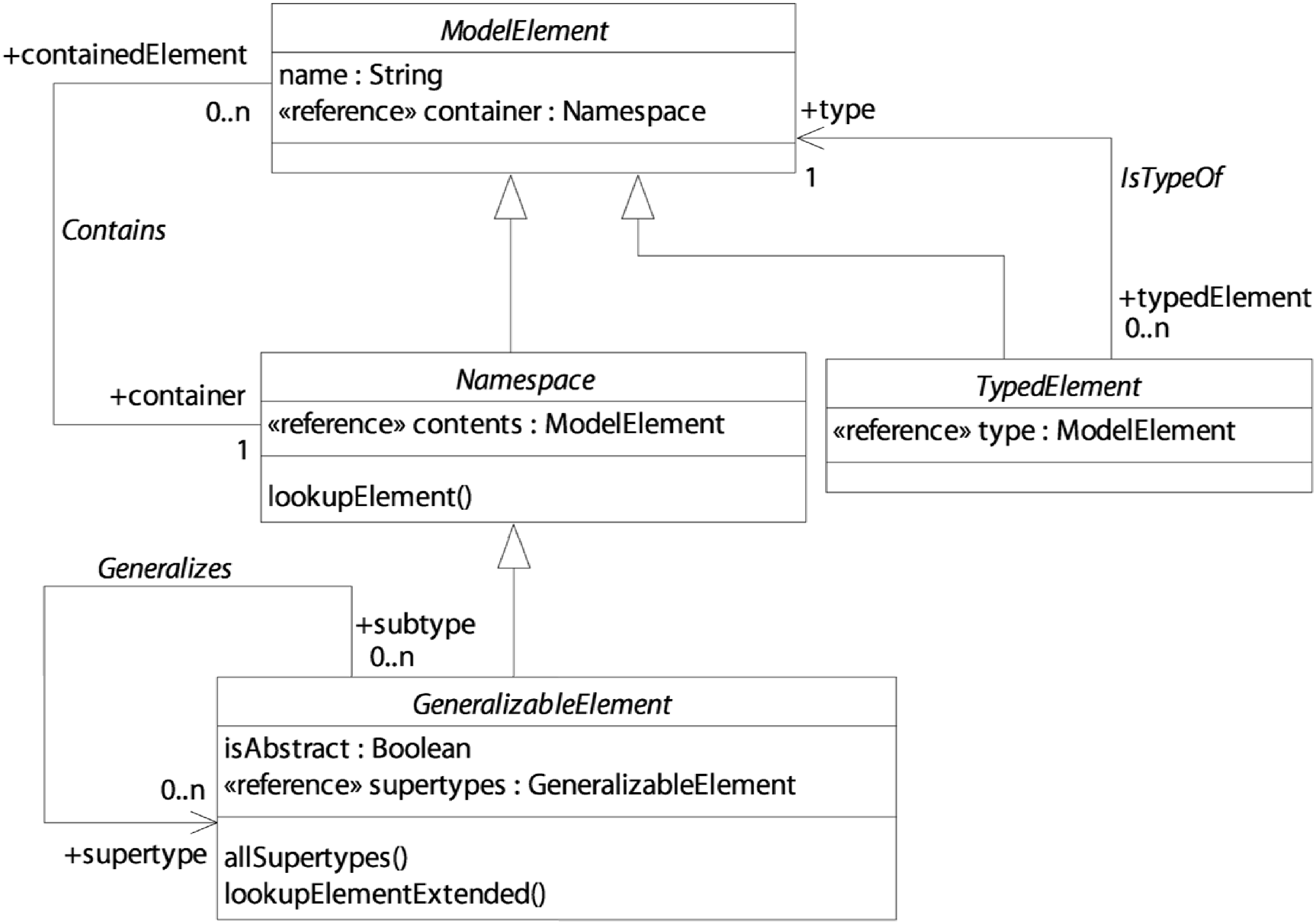
Conclusion

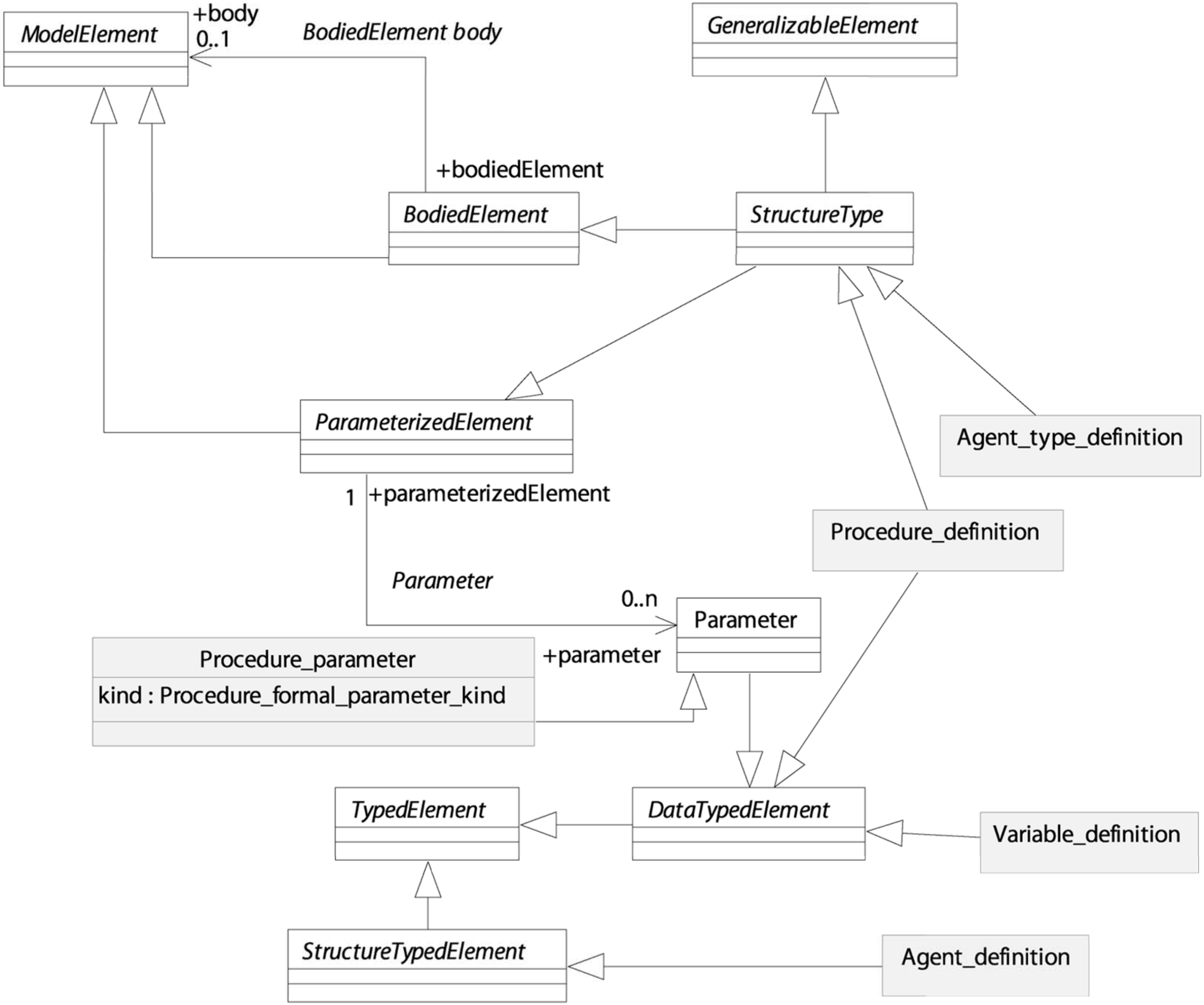
- Semi-automatic method on developing metamodels for existing abstract syntax definitions
- Resulting metamodels are compact, structured, without the use of redundant concept definitions, archived through extensive use of abstract concepts descriptions
- The used abstract concepts, can be used as a shared abstract bases for the definition of multiple languages

Thank you

Reserve

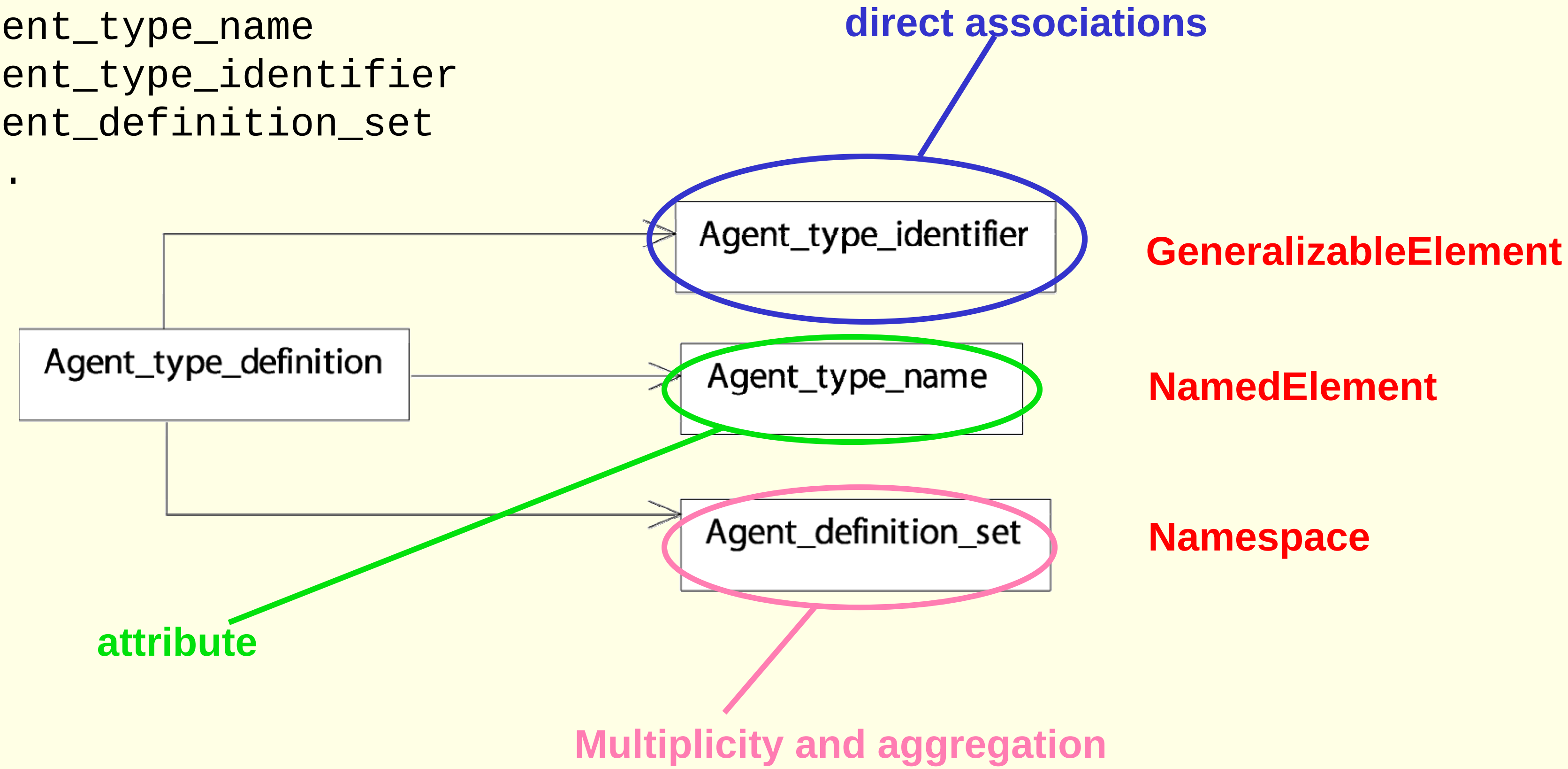

BNF grammar
represents
semantic mapping
common abstract concepts
primitive metamodel
abstract but language specific concepts
metamodel
concrete concepts
transformation


ModelElement
name : String
«reference» container : Namespace
+containedElement
0..n
+type
1
IsTypeOf
Contains
+typedElement
0..n
Namespace
«reference» contents : ModelElement
lookupElement()
+container
1
TypedElement
«reference» type : ModelElement
Generalizes
+subtype
0..n
GeneralizableElement
isAbstract : Boolean
«reference» supertypes : GeneralizableElement
allSupertypes()
lookupElementExtended()
0..n
+supertype


ModelElement
+body
0..1
BodiedElement body
GeneralizableElement
+bodiedElement
BodiedElement
StructureType
ParameterizedElement
Agent_type_definition
1
+parameterizedElement
Procedure_definition
Parameter
0..n
Parameter
+parameter
Procedure_parameter
kind : Procedure_formal_parameter_kind
TypedElement
DataTypedElement
Variable_definition
StructureTypedElement
Agent_definition

Some grammar weeknesses

```
Agent_type_definition ::
    Agent_type_name
    Agent_type_identifier
    Agent_definition_set
    ...
```



Agent_type_definition

Agent_type_identifier

Agent_type_name

Agent_definition_set

direct associations

GeneralizableElement

NamedElement

Namespace

attribute

Multiplicity and aggregation