Type Systems for Domain-specific Languages

ATPS Workshop 2013

Reiner Jung Christian Schneider Wilhelm Hasselbring

Christian-Albrechts-Universität zu Kiel
Institut für Informatik

26.02.2013

Large DSL Project MENGES [MEG09]

- Goal: DSL for railway control centers
- DSLs for various purposes
- Target languages for PLC [IEC03]
- But one XML output file (PLCOpen.xml)

Large DSL Project MENGES [MEG09]

- Goal: DSL for railway control centers
- DSLs for various purposes
- Target languages for PLC [IEC03]
- But one XML output file (PLCOpen.xml)

Agile grammar development

1. Domain analysis
2. Language development
3. Evaluation with users and case studies
4. Improving domain knowledge
5. goto 2

Problems

- Development of semantic checks

Problems

- Development of semantic checks
- Mapping of source to target language
  - data structures and
  - expressions

Problems

- Development of semantic checks
- Mapping of source to target language
  - data structures and
  - expressions
- Generator composition

Problems

- Development of semantic checks
- Mapping of source to target language
  - data structures and
  - expressions
- Generator composition

Result

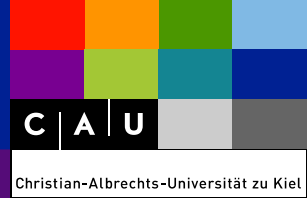
- Incomplete generators

Problems

- Development of semantic checks
- Mapping of source to target language
  - data structures and
  - expressions
- Generator composition

Result

- Incomplete generators
- Unmaintainable generator after 2 years



Problems

- Development of semantic checks
- Mapping of source to target language
  - data structures and
  - expressions
- Generator composition

Result

- Incomplete generators
- Unmaintainable generator after 2 years
- Language evolution complicated

Question How could we improve generator development?

Question How could we improve generator development?

Idea

- Look into compiler development techniques
- Adapt techniques for DSL development
- Integrate with Xtext [AF11]

Xtext/TS [1]

- Type system framework
- DSL to write type checker
- Compact notation
- Limited typing features

[1] http://code.google.com/a/eclipselabs.org/p/xtext-typesystem/
[2] http://xsemantics.sourceforge.net/

Xtext/TS [1]

- Type system framework
- DSL to write type checker
- Compact notation
- Limited typing features

XSemantics [2]

- DSL to define type systems
- Rich syntax
- Compact notation
- Uses type system terminology (judgement, rule, . . .)

[1] http://code.google.com/a/eclipselabs.org/p/xtext-typesystem/
[2] http://xsemantics.sourceforge.net/

Features

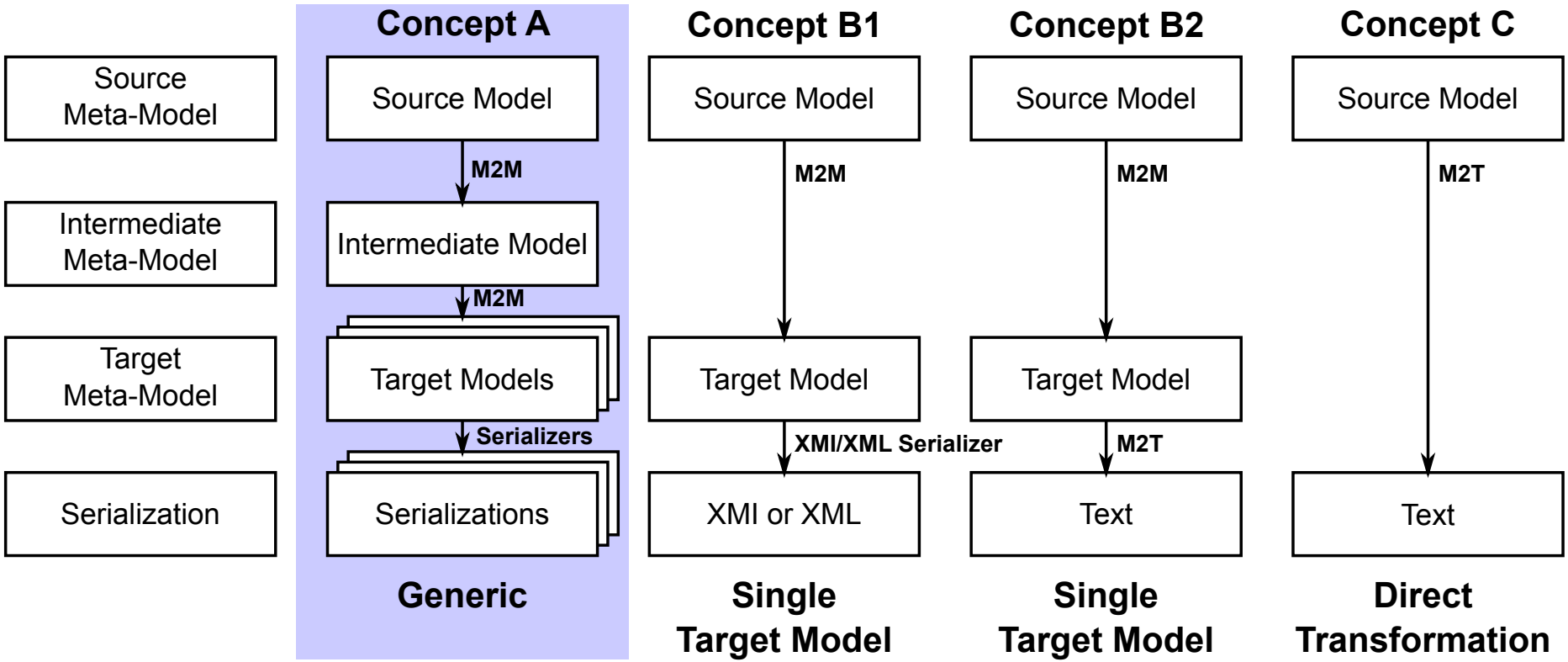
- Xtext-based expression framework
- Uses Java type system
- Provides Java type checking
- Provides partial code generation

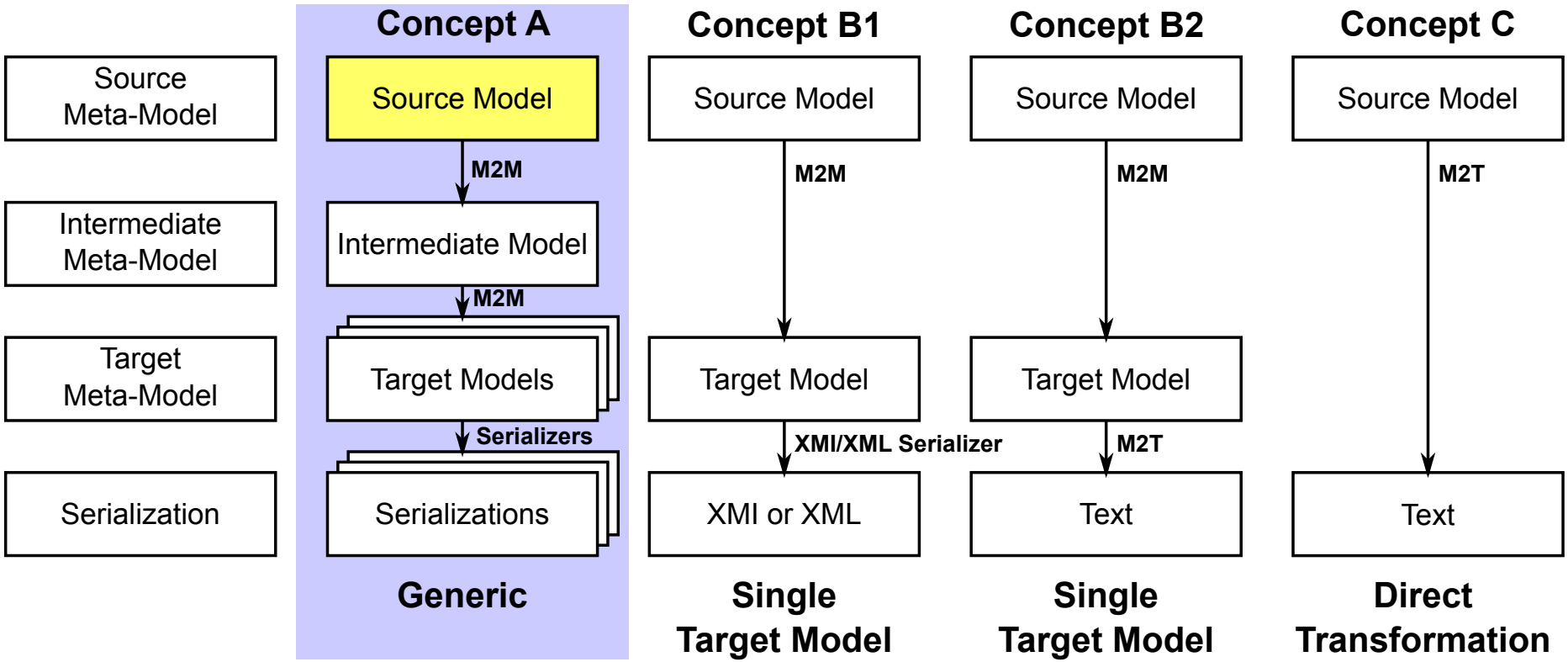
Features

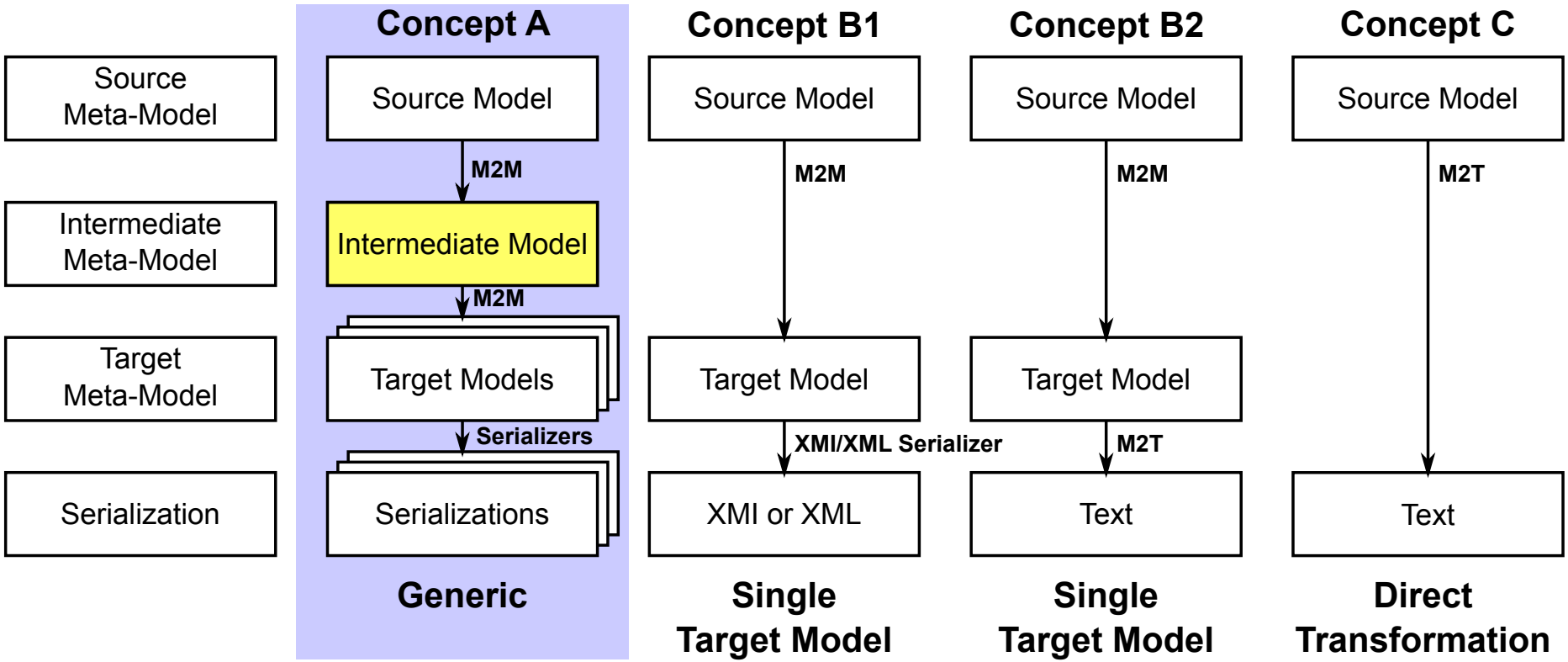
- Xtext-based expression framework
- Uses Java type system
- Provides Java type checking
- Provides partial code generation

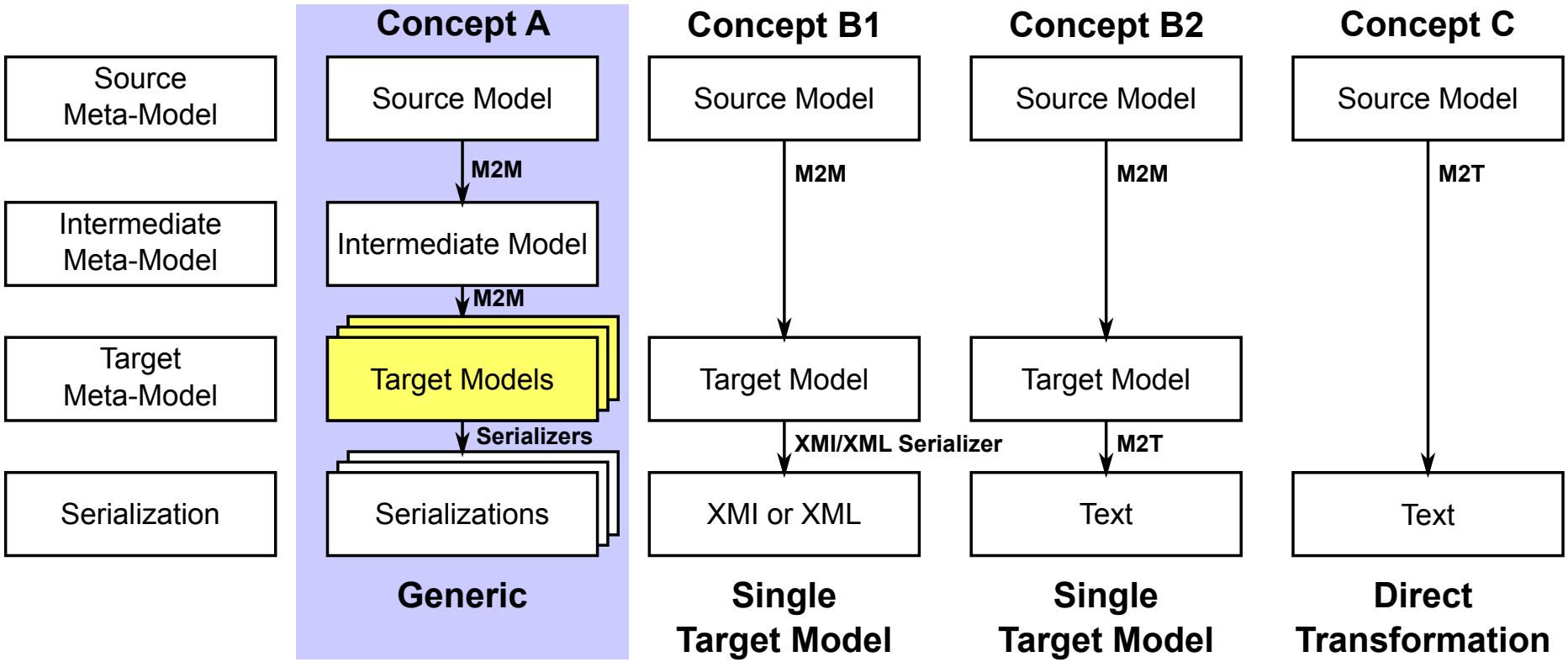
Limitations

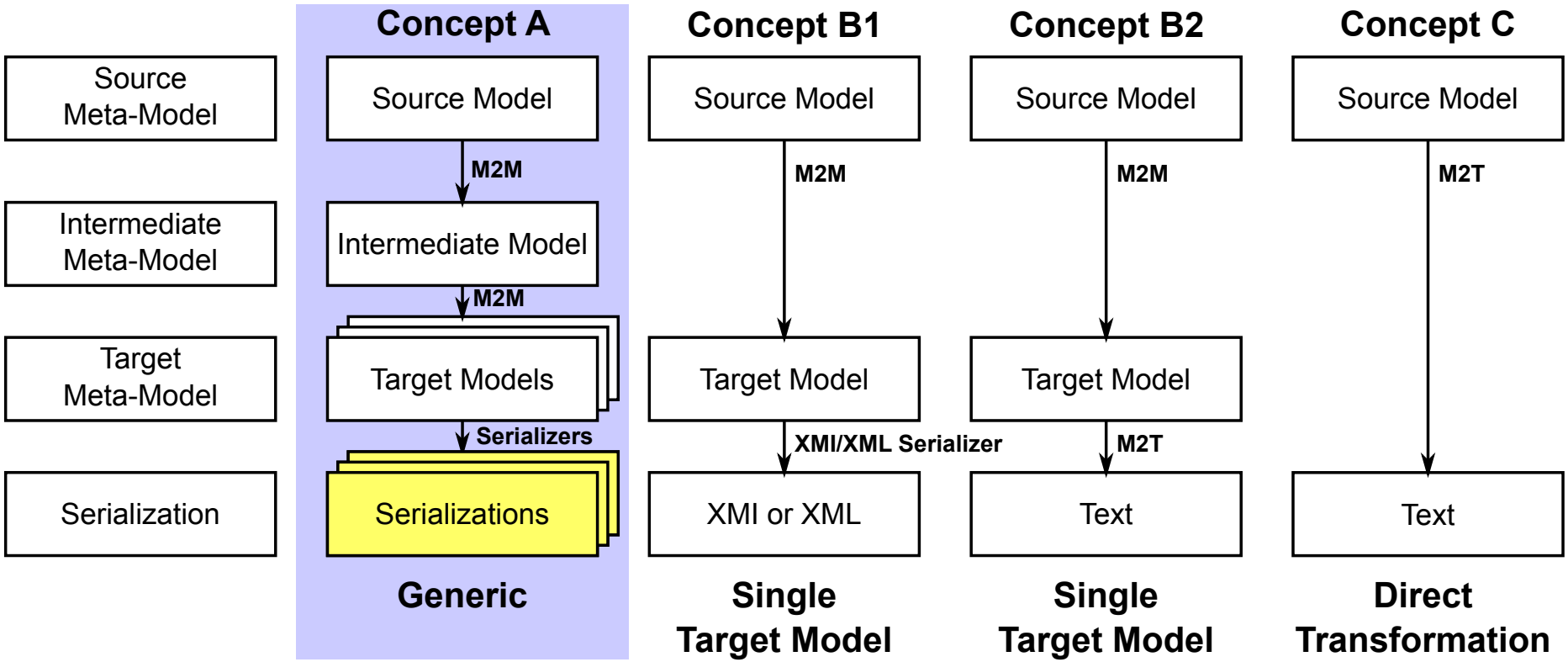
- Works only with Java
- Non Java-types must be added by hand

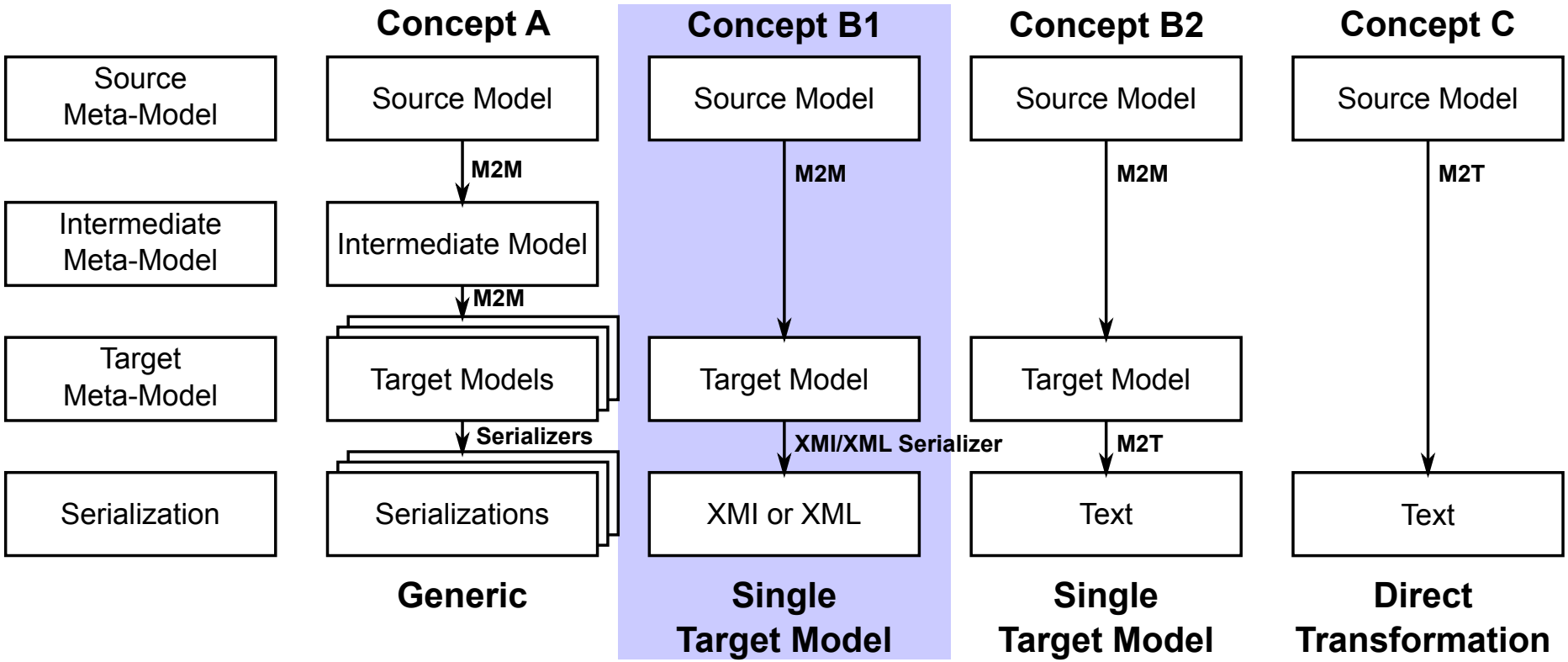

Source Meta-Model
Intermediate Meta-Model
Target Meta-Model
Serialization
Concept A
Source Model
M2M
Intermediate Model
M2M
Target Models
Serializers
Serializations
Generic
Concept B1
Source Model
M2M
Target Model
XMI/XML Serializer
XMI or XML
Single Target Model
Concept B2
Source Model
M2M
Target Model
M2T
Text
Single Target Model
Concept C
Source Model
M2T
Text
Direct Transformation

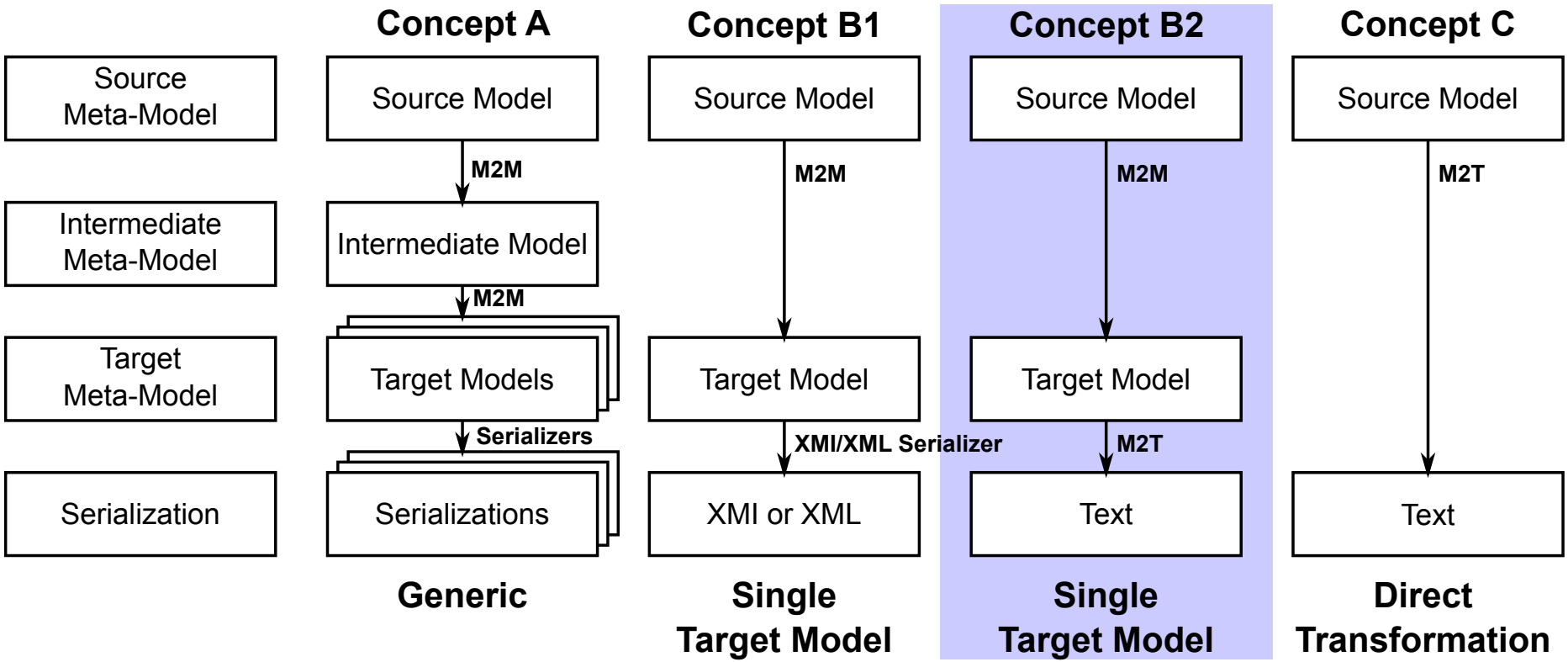

Source Meta-Model
Intermediate Meta-Model
Target Meta-Model
Serialization
Concept A
Source Model
M2M
Intermediate Model
M2M
Target Models
Serializers
Serializations
Generic
Concept B1
Source Model
M2M
Target Model
XMI/XML Serializer
XMI or XML
Single Target Model
Concept B2
Source Model
M2M
Target Model
M2T
Text
Single Target Model
Concept C
Source Model
M2T
Text
Direct Transformation

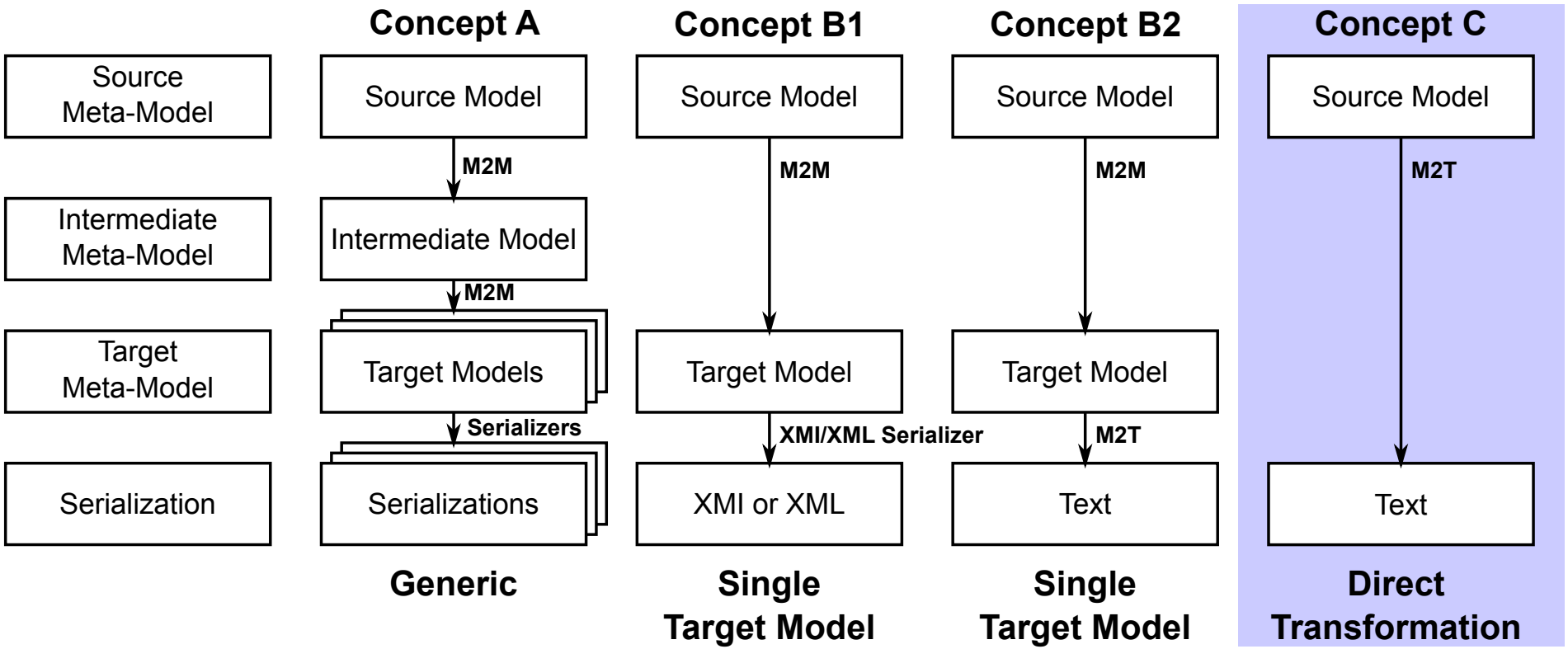

Source Meta-Model
Intermediate Meta-Model
Target Meta-Model
Serialization
Concept A
Source Model
M2M
Intermediate Model
M2M
Target Models
Serializers
Serializations
Generic
Concept B1
Source Model
M2M
Target Model
XMI/XML Serializer
XMI or XML
Single Target Model
Concept B2
Source Model
M2M
Target Model
M2T
Text
Single Target Model
Concept C
Source Model
M2T
Text
Direct Transformation


Source Meta-Model
Intermediate Meta-Model
Target Meta-Model
Serialization
Concept A
Source Model
M2M
Intermediate Model
M2M
Target Models
Serializers
Serializations
Generic
Concept B1
Source Model
M2M
Target Model
XMI/XML Serializer
XMI or XML
Single Target Model
Concept B2
Source Model
M2M
Target Model
M2T
Text
Single Target Model
Concept C
Source Model
M2T
Text
Direct Transformation


Source Meta-Model
Intermediate Meta-Model
Target Meta-Model
Serialization
Concept A
Source Model
M2M
Intermediate Model
M2M
Target Models
Serializers
Serializations
Generic
Concept B1
Source Model
M2M
Target Model
XMI/XML Serializer
XMI or XML
Single Target Model
Concept B2
Source Model
M2M
Target Model
M2T
Text
Single Target Model
Concept C
Source Model
M2T
Text
Direct Transformation

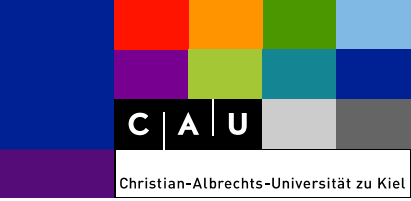

Concept A
Concept B1
Concept B2
Concept C
Source Meta-Model
Intermediate Meta-Model
Target Meta-Model
Serialization
Source Model
M2M
Intermediate Model
M2M
Target Models
Serializers
Serializations
Source Model
M2M
Target Model
XMI/XML Serializer
XMI or XML
Source Model
M2M
Target Model
M2T
Text
Source Model
M2T
Text
Generic
Single Target Model
Single Target Model
Direct Transformation


Concept A
Concept B1
Concept B2
Concept C
Source Meta-Model
Intermediate Meta-Model
Target Meta-Model
Serialization
Source Model
M2M
Intermediate Model
M2M
Target Models
Serializers
Serializations
Source Model
M2M
Target Model
XMI/XML Serializer
XMI or XML
Source Model
M2M
Target Model
M2T
Text
Source Model
M2T
Text
Generic
Single Target Model
Single Target Model
Direct Transformation


Concept A
Concept B1
Concept B2
Concept C
Source Meta-Model
Intermediate Meta-Model
Target Meta-Model
Serialization
Source Model
M2M
Intermediate Model
M2M
Target Models
Serializers
Serializations
Source Model
M2M
Target Model
XMI/XML Serializer
XMI or XML
Source Model
M2M
Target Model
M2T
Text
Source Model
M2T
Text
Generic
Single Target Model
Single Target Model
Direct Transformation

- **Type**
- **TypeReference:** reference = Type, (remainder=TypeReference)?

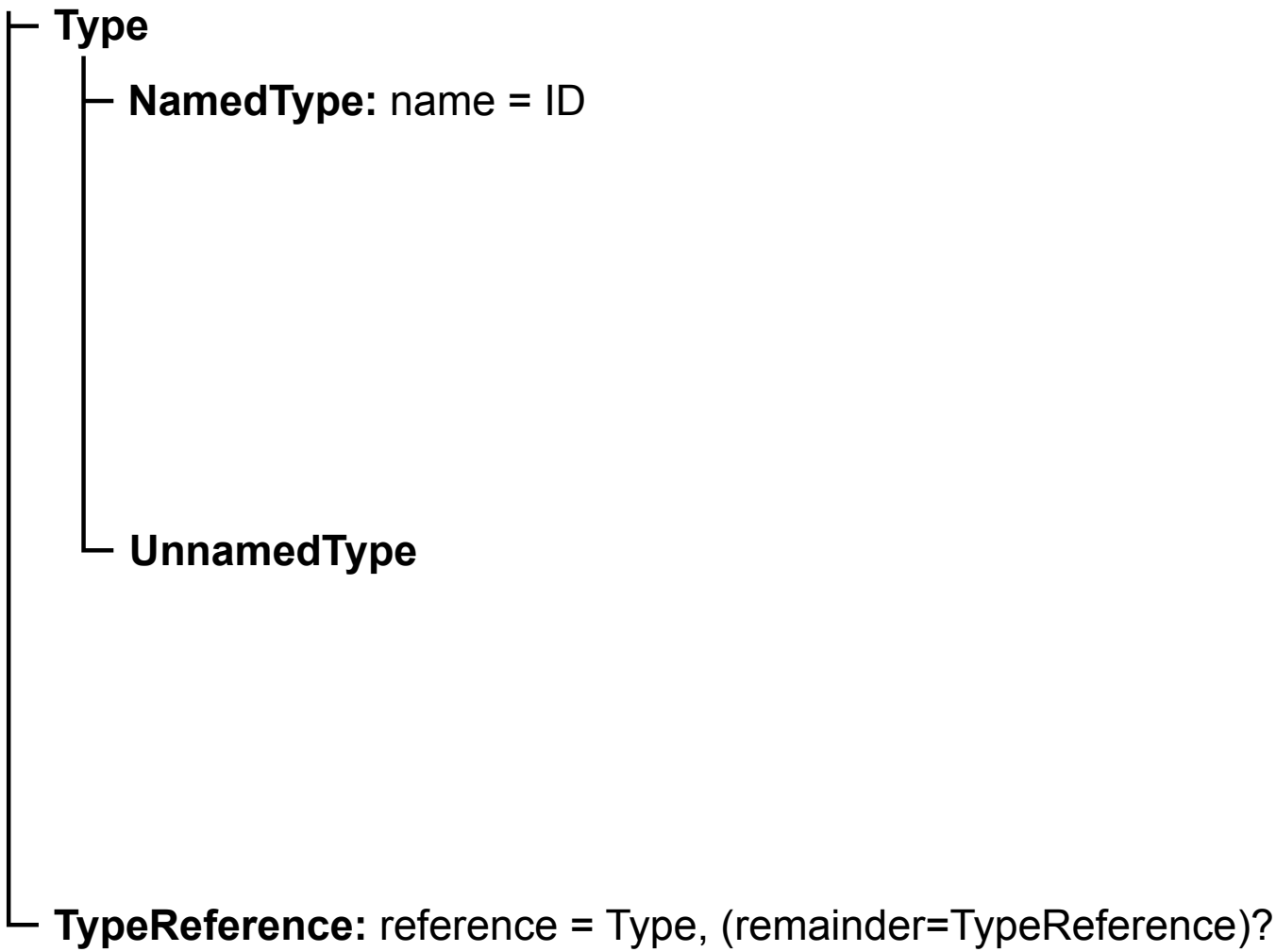
- **Type**
- **TypeReference:** reference = Type, (remainder=TypeReference)?



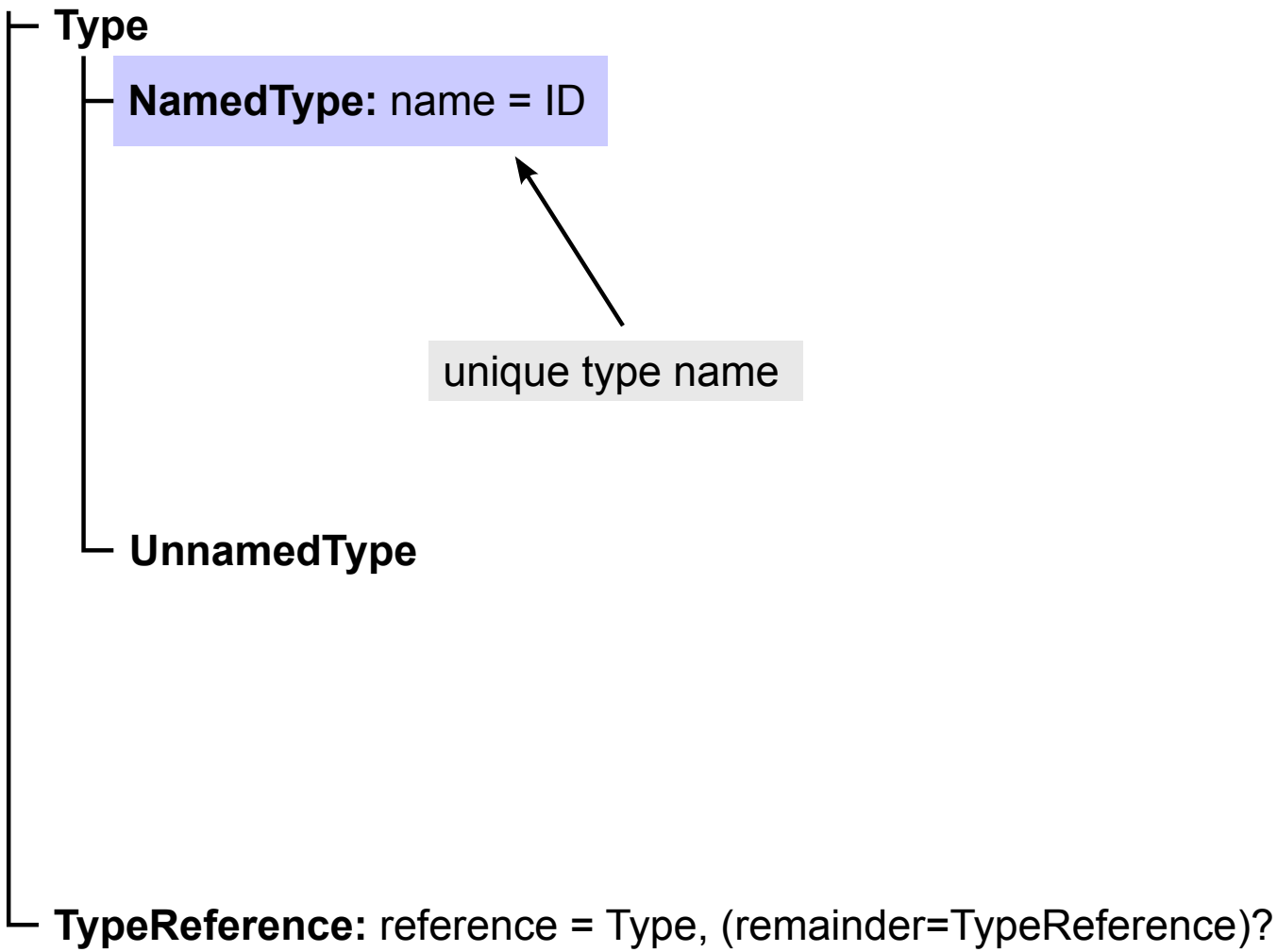
- **Type**
- **TypeReference:** reference = Type, (remainder=TypeReference)?

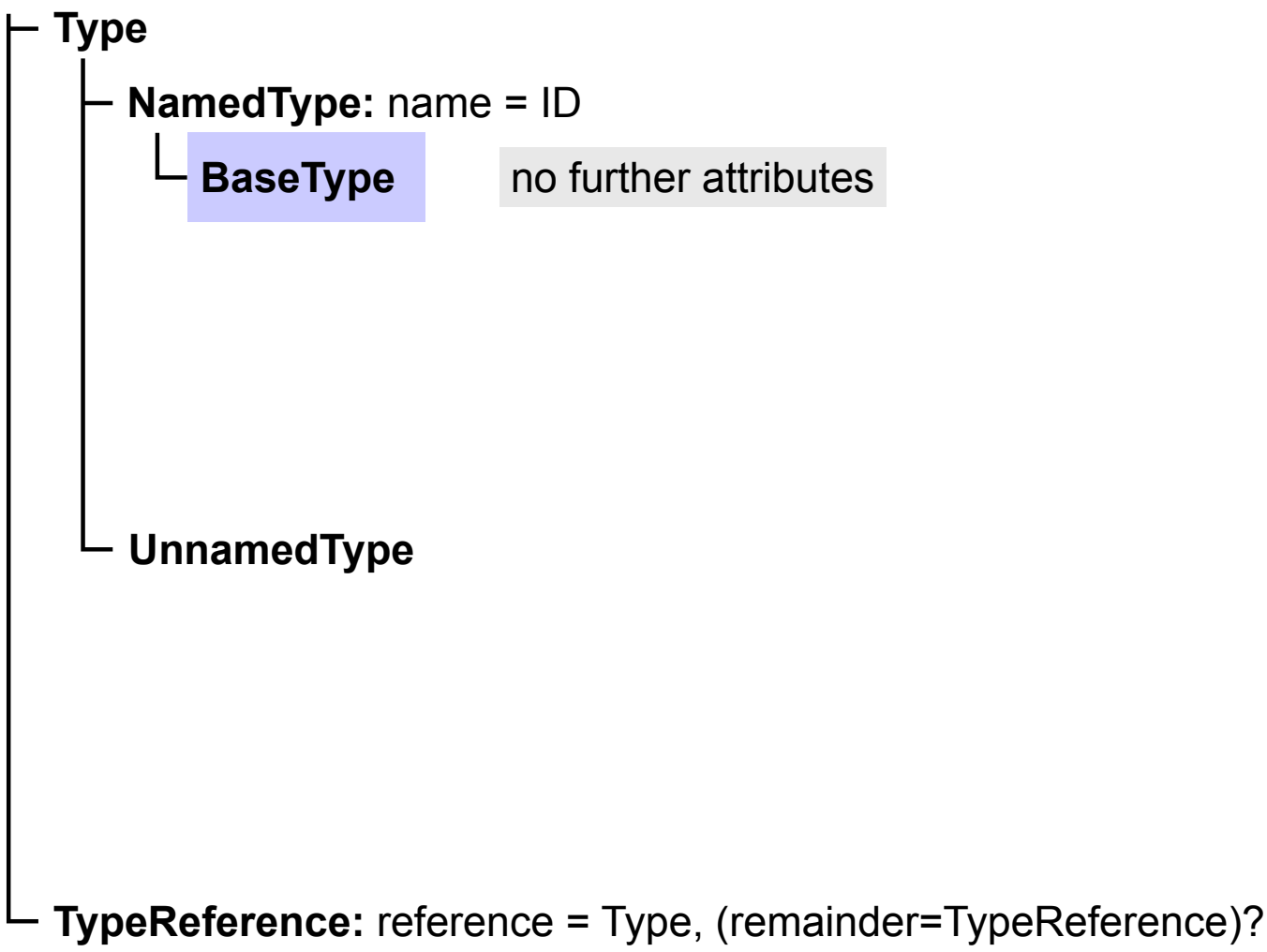
- **Type**
- **TypeReference:** reference = Type, (remainder=TypeReference)?

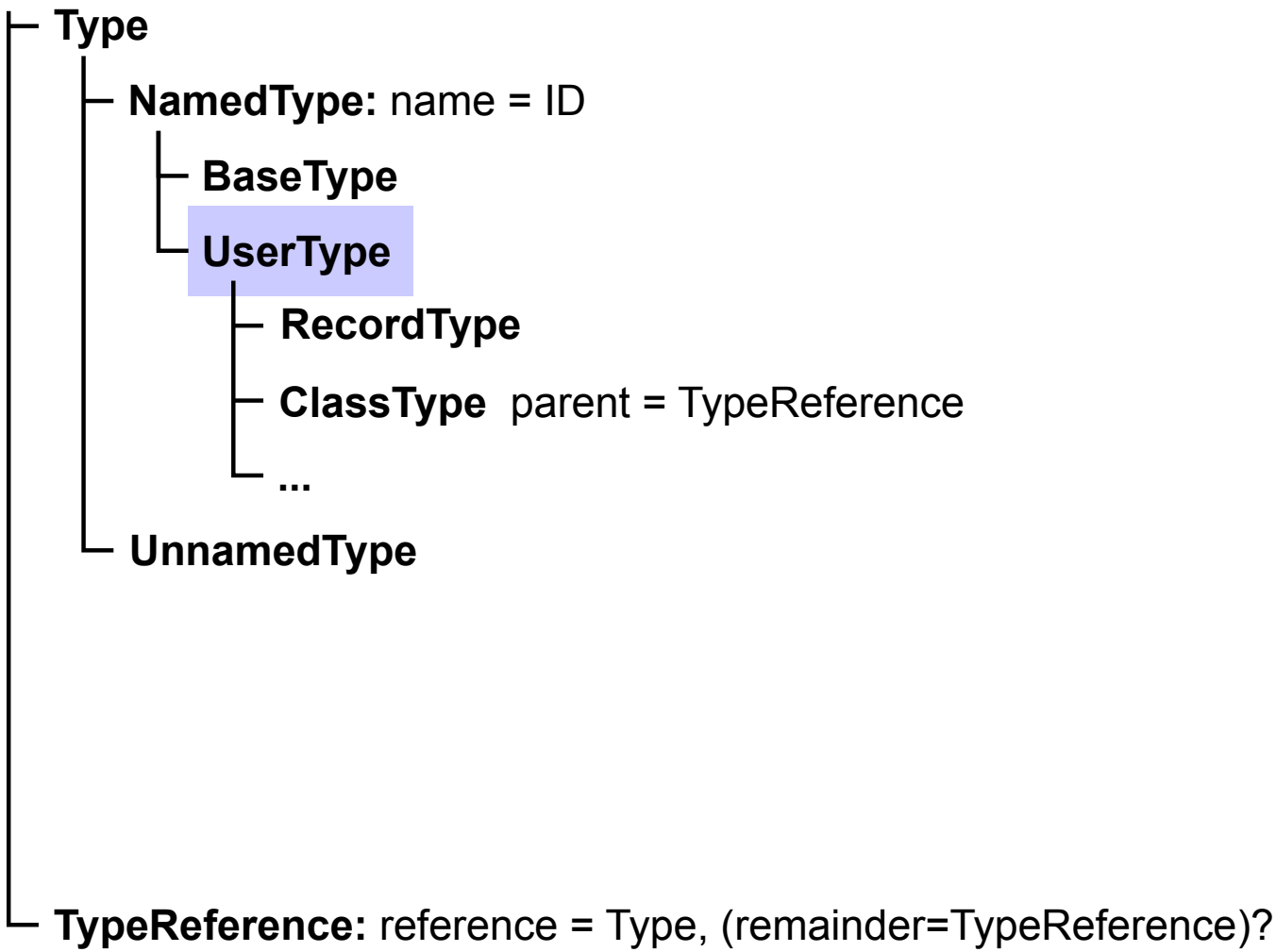
for nested type structures

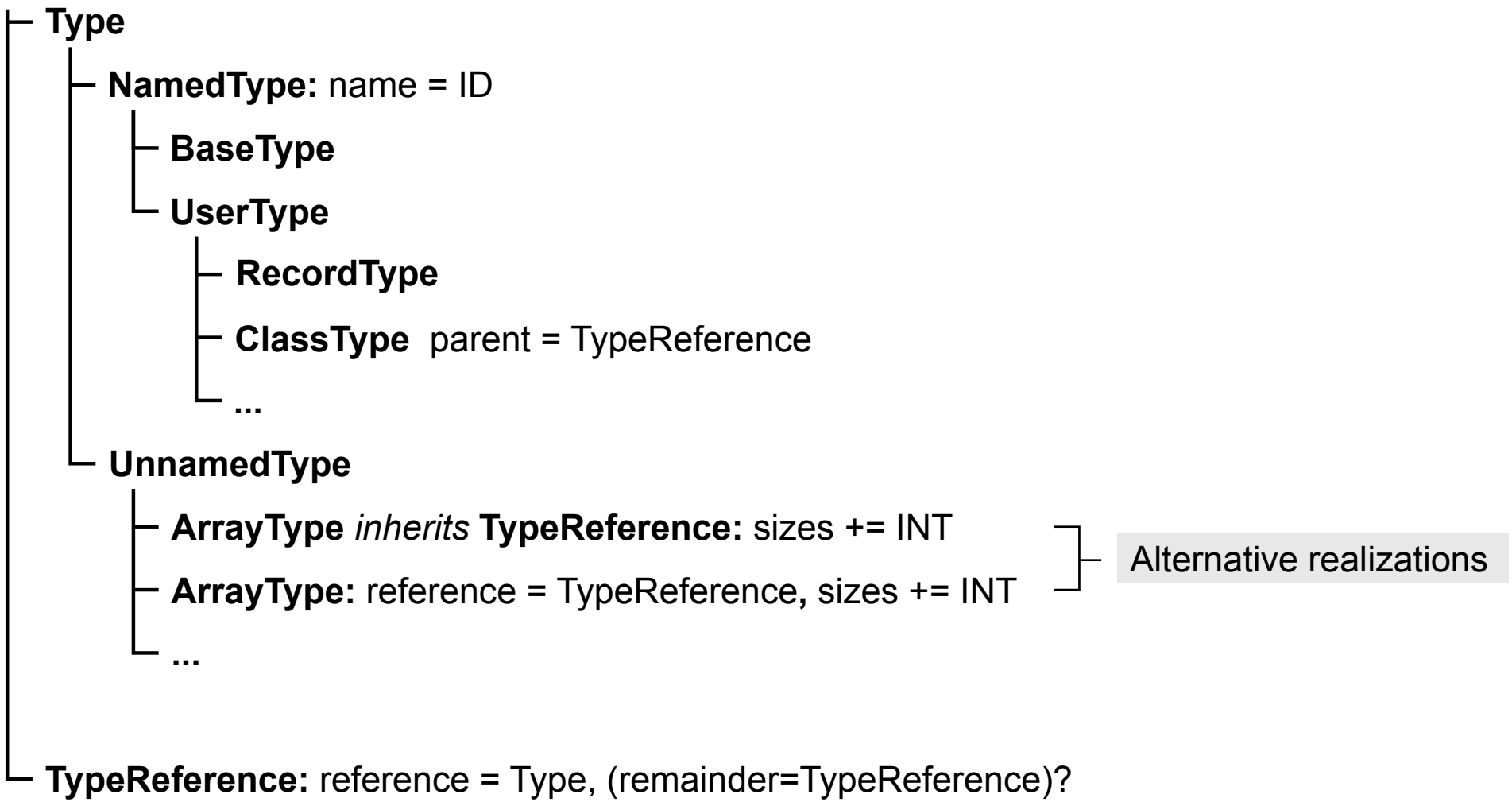

Type
NamedType: name = ID
UnnamedType
TypeReference: reference = Type, (remainder=TypeReference)?

- **Type**
  - **NamedType:** name = ID
  - **UnnamedType**
- **TypeReference:** reference = Type, (remainder=TypeReference)?


Type
NamedType: name = ID
unique type name
UnnamedType
TypeReference: reference = Type, (remainder=TypeReference)?


Type
NamedType: name = ID
BaseType
no further attributes
UnnamedType
TypeReference: reference = Type, (remainder=TypeReference)?


Type
NamedType: name = ID
BaseType
UserType
RecordType
ClassType parent = TypeReference
...
UnnamedType
TypeReference: reference = Type, (remainder=TypeReference)?


Type
NamedType: name = ID
BaseType
UserType
RecordType
ClassType parent = TypeReference
...
UnnamedType
ArrayType inherits TypeReference: sizes += INT
ArrayType: reference = TypeReference, sizes += INT
...
Alternative realizations
TypeReference: reference = Type, (remainder=TypeReference)?

Enumeration Base type declaration

```
public enum BaseTypes {
    BOOLEAN, INT, FLOAT, STRING; /* base types */

    public String lowerCaseName() {
        return this.name().toLowerCase();
    }
}
```

From our example application

`http://build.se.informatik.uni-kiel.de/de.cau.cs.se.lad.git`

Enumeration Base type declaration

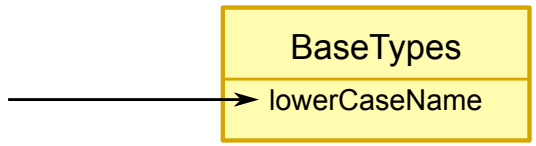
```
public enum BaseTypes {
    BOOLEAN, INT, FLOAT, STRING; /* base types */

    public String lowerCaseName() {
        return this.name().toLowerCase();
    }
}
```

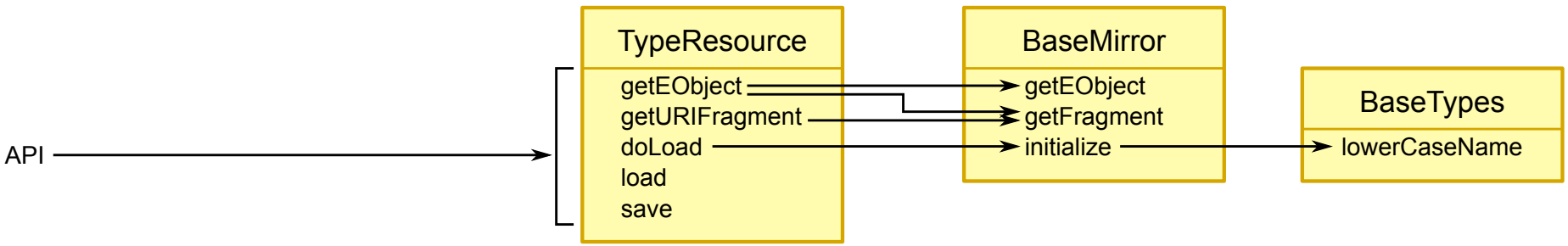
From our example application

`http://build.se.informatik.uni-kiel.de/de.cau.cs.se.lad.git`

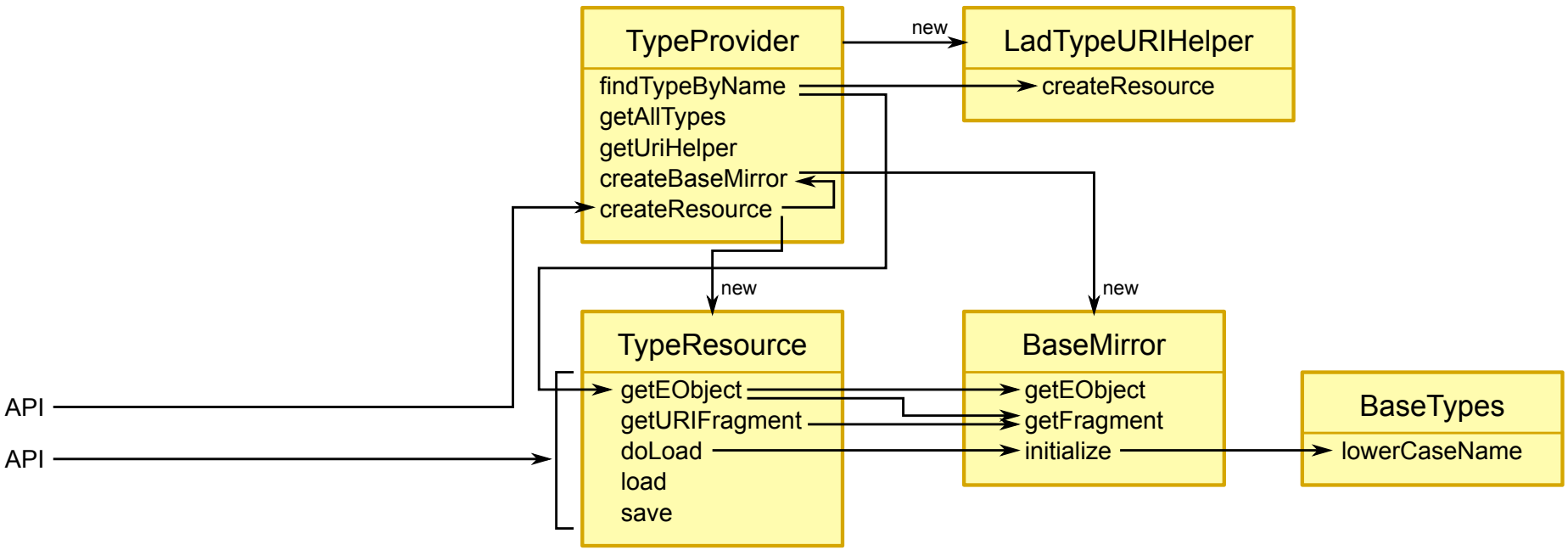
Embedding into Xtext



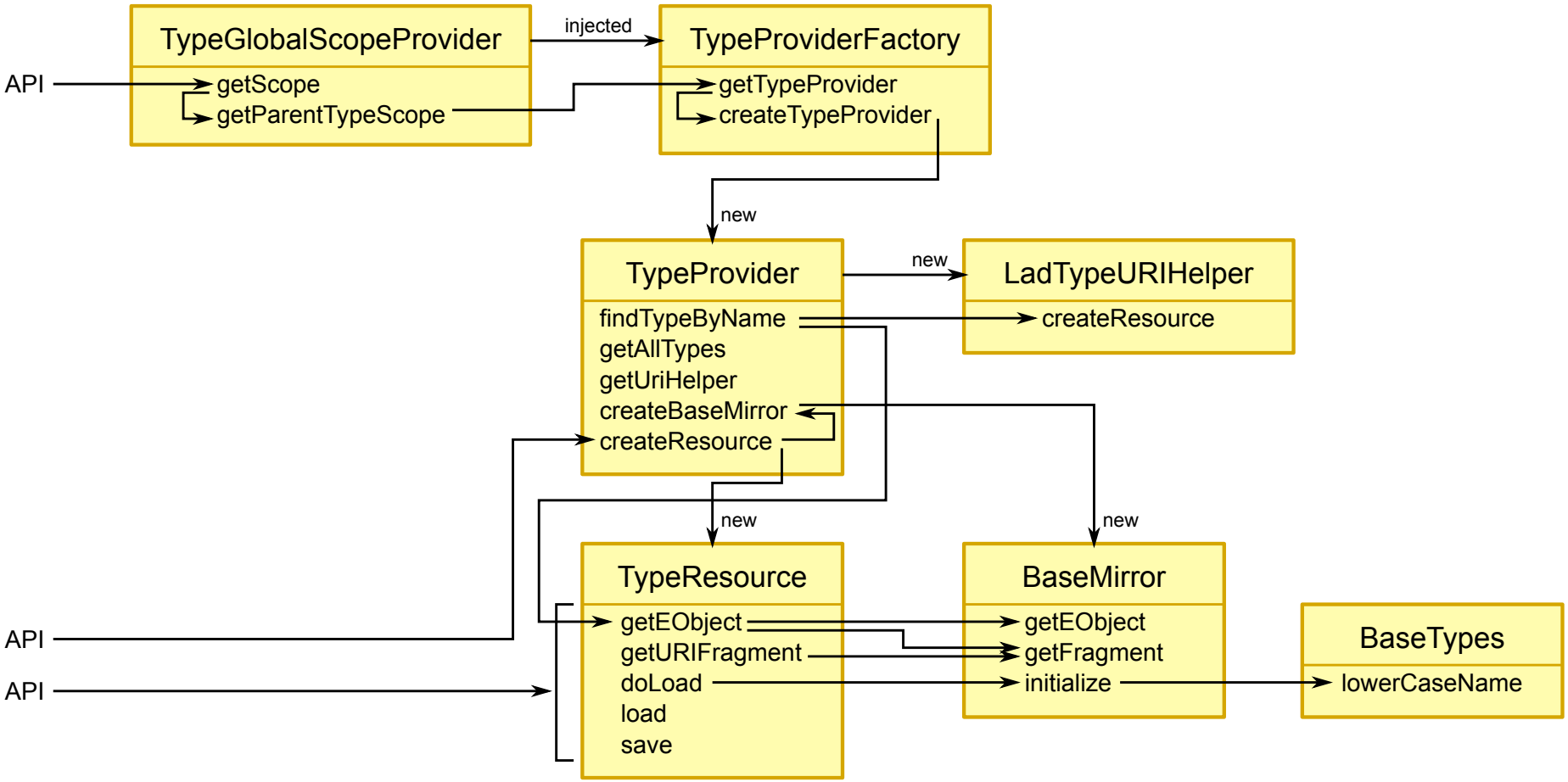
Embedding into Xtext



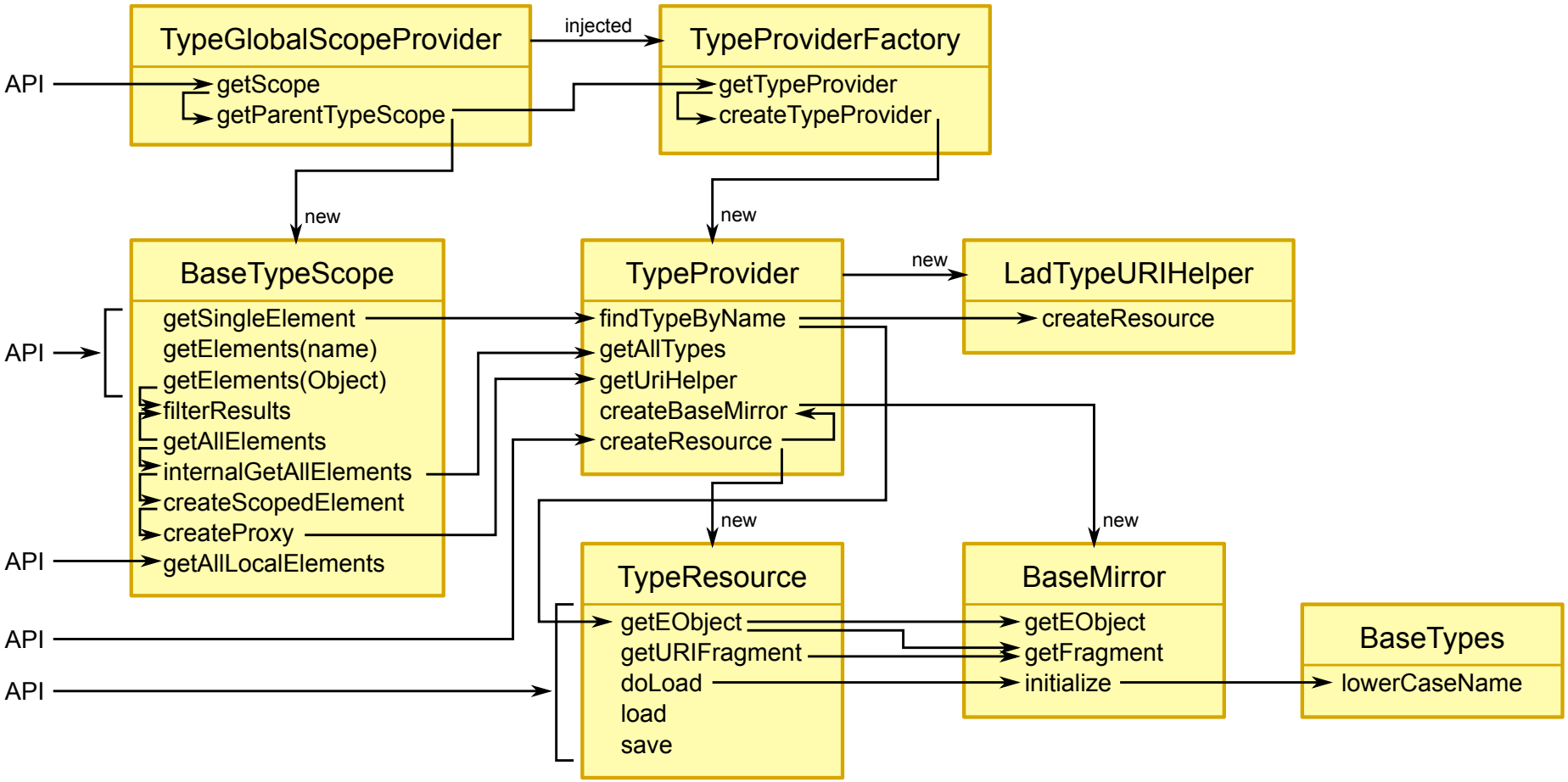
Embedding into Xtext



Embedding into Xtext



Embedding into Xtext



Types

```
Type returns type::Type: UserType ;
UserType returns type::UserType: ClassType | RecordType ;

TypeReference returns type::TypeReference:
{type::TypeReference} reference = [type::NamedType | ID] ('.' remainder=TypeReference )? |
{type::ArrayType} reference = [type::NamedType | ID] ( '[' sizes += INT ']' )+ ;
```

Property Declaration

```
PropertyDeclaration returns type::PropertyDeclaration:
    modifiers += Modifier type = TypeReference name = ID ;

FunctionDeclaration returns type::FunctionDeclaration:
    modifiers += Modifier type = TypeReference name = ID
    '(' (parameters += ParameterDeclaration (',' parameters += ParameterDeclaration)*)? ')'
    body = Body ;
```

Types

```
Type returns type::Type: UserType ;
UserType returns type::UserType: ClassType | RecordType ;

TypeReference returns type::TypeReference:
{type::TypeReference} reference = [type::NamedType | ID] ('.' remainder=TypeReference )? |
{type::ArrayType} reference = [type::NamedType | ID] ( '[' sizes += INT ']' )+ ;
```

Property Declaration

```
PropertyDeclaration returns type::PropertyDeclaration:
    modifiers += Modifier type = TypeReference name = ID ;

FunctionDeclaration returns type::FunctionDeclaration:
    modifiers += Modifier type = TypeReference name = ID
    '(' (parameters += ParameterDeclaration (',' parameters += ParameterDeclaration)*)? ')'
    body = Body ;
```

Types

```
Type returns type::Type: UserType ;
UserType returns type::UserType: ClassType | RecordType ;

TypeReference returns type::TypeReference:
{type::TypeReference} reference = [type::NamedType | ID] ('.' remainder=TypeReference )? |
{type::ArrayType} reference = [type::NamedType | ID] ( '[' sizes += INT ']' )+ ;
```

Property Declaration

```
PropertyDeclaration returns type::PropertyDeclaration:
    modifiers += Modifier type = TypeReference name = ID ;

FunctionDeclaration returns type::FunctionDeclaration:
    modifiers += Modifier type = TypeReference name = ID
    '(' (parameters += ParameterDeclaration (',' parameters += ParameterDeclaration)*)? ')'
    body = Body ;
```

Types

```
Type returns type::Type: UserType ;
UserType returns type::UserType: ClassType | RecordType ;

TypeReference returns type::TypeReference:
{type::TypeReference} reference = [type::NamedType | ID] ('.' remainder=TypeReference )? |
{type::ArrayType} reference = [type::NamedType | ID] ( '[' sizes += INT ']' )+ ;
```

Property Declaration

```
PropertyDeclaration returns type::PropertyDeclaration:
    modifiers += Modifier type = TypeReference name = ID ;

FunctionDeclaration returns type::FunctionDeclaration:
    modifiers += Modifier type = TypeReference name = ID
    '(' (parameters += ParameterDeclaration (',' parameters += ParameterDeclaration)*)? ')'
    body = Body ;
```

Base Types and Type Reference

```
/** Determine type of a literal: here a boolean value */
def dispatch Type getActualType(BooleanValue value) {
    return typeProvider.findTypeByName("boolean");
}

/** Recursive method to follow type references */
def dispatch Type getActualType(TypeReference ref) {
    return ref.remainder?.actualType?:ref.reference;
}
```

- Use of Xtend signature-based dispatcher

Base Types and Type Reference

```
/** Determine type of a literal: here a boolean value */
def dispatch Type getActualType(BooleanValue value) {
    return typeProvider.findTypeByName("boolean");
}

/** Recursive method to follow type references */
def dispatch Type getActualType(TypeReference ref) {
    return ref.remainder?.actualType?:ref.reference;
}
```

- Use of Xtend signature-based dispatcher
- Dispatch method returns type for boolean values

Base Types and Type Reference

```
/** Determine type of a literal: here a boolean value */
def dispatch Type getActualType(BooleanValue value) {
    return typeProvider.findTypeByName("boolean");
}

/** Recursive method to follow type references */
def dispatch Type getActualType(TypeReference ref) {
    return ref.remainder?.actualType?:ref.reference;
}
```

- Use of Xtend signature-based dispatcher
- Dispatch method returns type for boolean values
- Dispatch method resolves type reference

Declarations and Calls

```
/** Determine the type of a value, variable, or property declaration */
def dispatch Type getActualType(ValueDeclaration decl) {
    return decl?.typeReference?.actualType;
}

/** Determine the return type of a declared function */
def dispatch Type getActualType(FunctionDeclaration decl) {
    return decl.type.actualType;
}

/** Determines the type of a value, variable, or property reference */
def dispatch Type getActualType(ValueReference ref) {
    return ref?.reference?.actualType;
}

/** Determines the return type of a function call */
def dispatch Type getActualType(FunctionCall call) {
    return call.functionRef.actualType;
}
```

Meta-model for target language

- XML-schema to EMF converter[3]
- Existing meta-models e.g., JavaTypes [EEK+12]
- Standards or documentation e.g., IEC 61131-3 [IEC03]

[3] `http://yoxos.eclipsesource.com/places/node/org.eclipse.xsd.ecore.converter.feature.group`

Meta-model for target language

- XML-schema to EMF converter[3]
- Existing meta-models e.g., JavaTypes [EEK+12]
- Standards or documentation e.g., IEC 61131-3 [IEC03]

Semantics of target language

- Type structures
- Expression and statement

[3] `http://yoxos.eclipsesource.com/places/node/org.eclipse.xsd.ecore.converter.feature.group`

Example runtime concepts

- Memory management
- Instantiation
- Garbage Collection
- Synchronization

Type and Expression Mapping

Example runtime concepts

- Memory management
- Instantiation
- Garbage Collection
- Synchronization

Example type mapping

- Source: Datatype **interval**
  - Arbitrary integer range
  - Wraps on overflow/underflow
- Target: Datatype **signed int32**
  - Range checks
  - Integration with operators, e.g., +, ++

Goal Improve DSLs development and maintenance

Summary

Goal Improve DSLs development and maintenance

Methods

- Base type integration in Xtext
- Meta-models for type systems
- Grammar composition
- Transformation recipes

DSL design patterns

- Public accessible collection
- Discuss further typing concepts, e.g., let, generics
- Better language composition

Type systems and type transformation

- Integration of XSemantics
- Concept evaluation of DSL for type transformation

References I

[AF11] Itemis AG and Eclipse Foundation. Xtext - dsl devlopment framework. Website `http://www.eclipse.org/Xtext/`, 2011.

[AG11] Itemis AG. Xtend 2. Website `http://www.eclipse.org/Xtext/documentation/2_0_0/01-Xtend_Introduction.php`, 2011.

[BSVC12] Lorenzo Bettini, Dietmar Stoll, Markus Völter, and Serano Colameo. Approaches and Tools for Implementing Type Systems in Xtext. In *Software Language Engineering*, Lecture Notes in Computer Science. Springer, 2012. To Appear.

[Car04] Luca Cardelli. *The Computer Science and Engineering Handbook*, chapter 97 – Type Systems. CRC Press, 2004.

[EEK+12] Sven Efftinge, Moritz Eysholdt, Jan Köhnlein, Sebastian Zarnekow, Robert von Massow, Wilhelm Hasselbring, and Michael Hanus. Xbase: implementing domain-specific languages for java. In *Proceedings of the 11th International Conference on Generative Programming and Component Engineering*, GPCE '12, pages 112–121, New York, NY, USA, 2012. ACM.

References II

[GHH+12] Wolfgang Goerigk, Wilhelm Hasselbring, Gregor Hennings, Reiner Jung, Holger Neustock, Heiko Schaefer, Christian Schneider, Elferik Schultz, Thomas Stahl, Reinhard von Hanxleden, Steffen Weik, and Stefan Zeug. Entwurf einer domänenspezifischen sprache für elektronische stellwerke. In Stefan Jähnichen, Axel Küpper, and Sahin Albayrak, editors, *Software Engineering*, volume 198 of *LNI*, pages 119–130. GI, 2012.

[IEC03] Deutsche Kommission Elektrotechnik Elektronik Informationtechnik im DIN und VDE, Beuth Verlag, Berlin. *IEC EN 61131-3*, 2003-12 edition, 2003.

[MEG09] Verbundprojekt 5 im kompetenzverbund software systems engineering. Technical report, CAU, Institut für Informatik, 2009.

[Pie02] Benjamin C. Pierce. *Types and programming languages*. MIT Press, 2002.

[Pie04] Benjamin C. Pierce. *Advanced Topics in Types and Programming Languages*. The MIT Press, 2004.

[Vö11] Markus Völter. Xtext/ts - a typesystem framework for xtext. Website `http://www.infoq.com/articles/xtext_ts`, 2011.