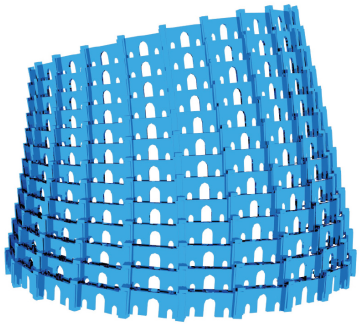
An ontological approach to technology documentation

Ralf Lämmel*
Software Languages Team
University of Koblenz-Landau, Germany
http://www.softlang.org/



softlang

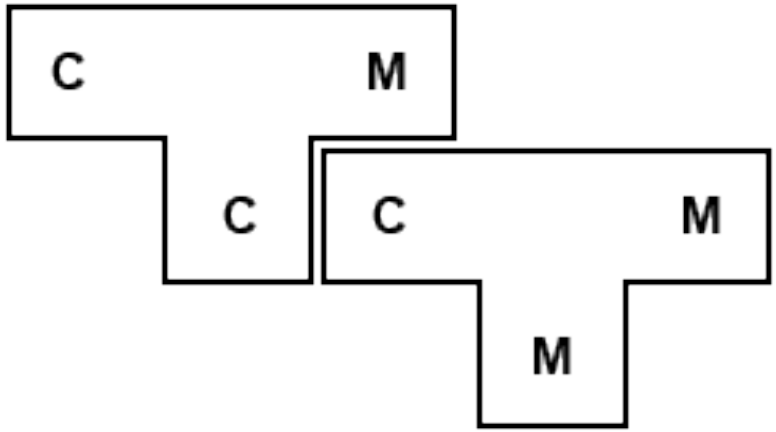
* Joint work with Johannes Härtel, Lukas Härtel, Marcel Heinz, and other members of the Software Languages Team.

Basic terminology

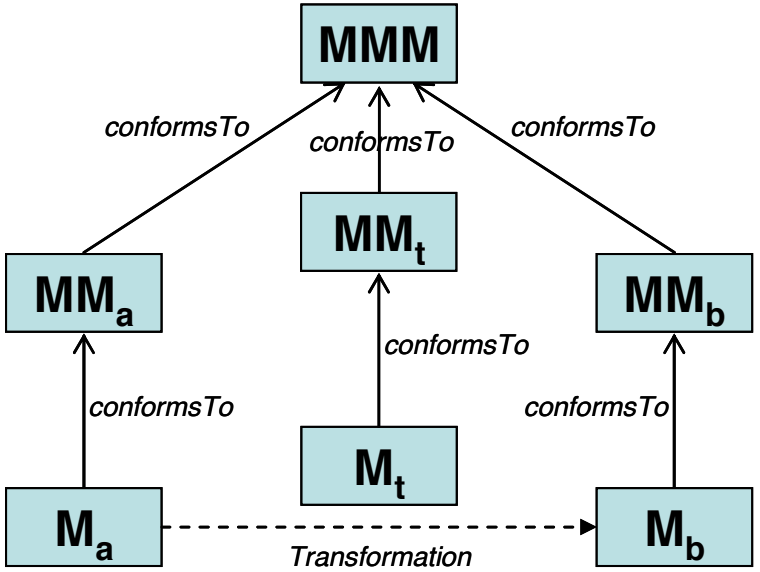
- Technology = Software technology, e.g.:
  - Web-application framework
  - O/R mapper
- Ontological documentation:
  - a form of **megamodeling**
  - also referred to as (model of) **linguistic architecture**

Ad-hoc megamodeling

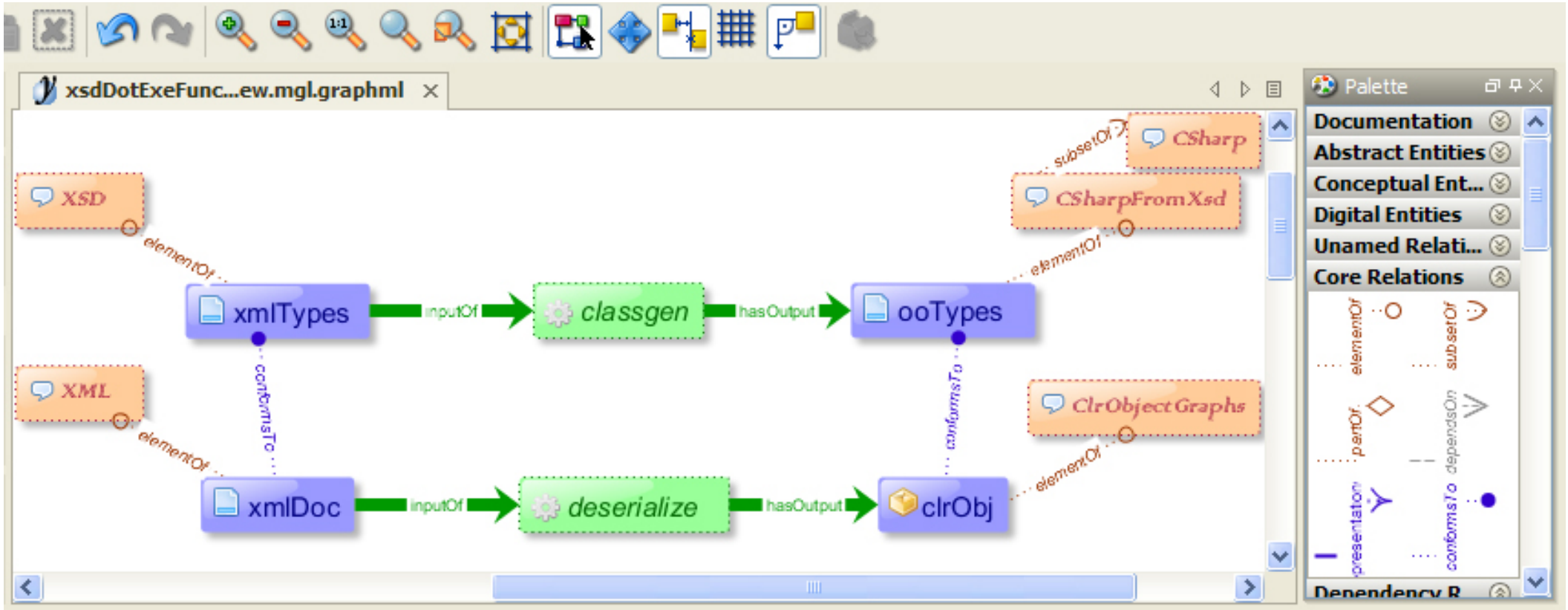
A tombstone diagram for bootstrapping a C compiler[8]

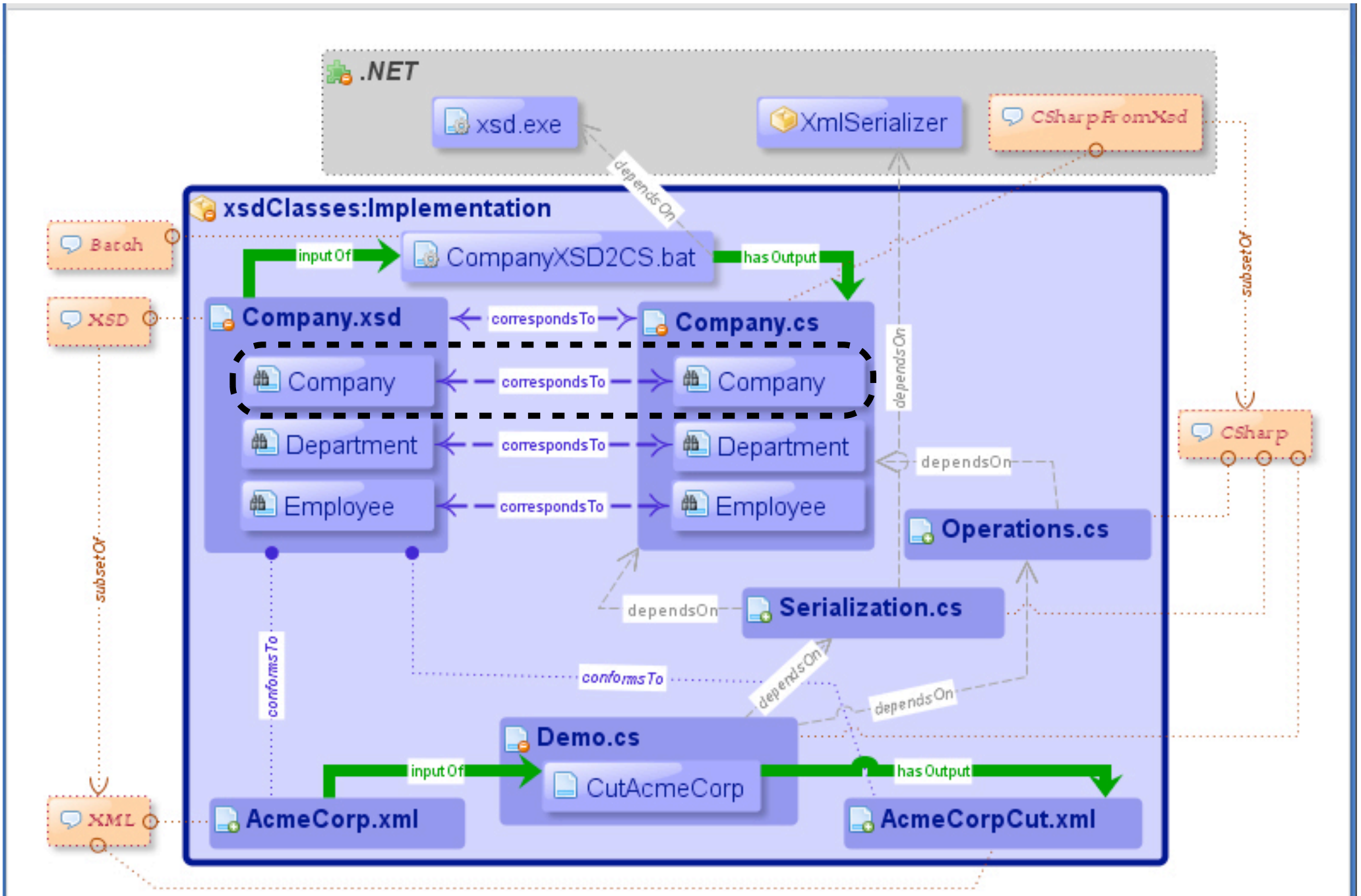


Mechanics of an ATL-based model transformation[9]



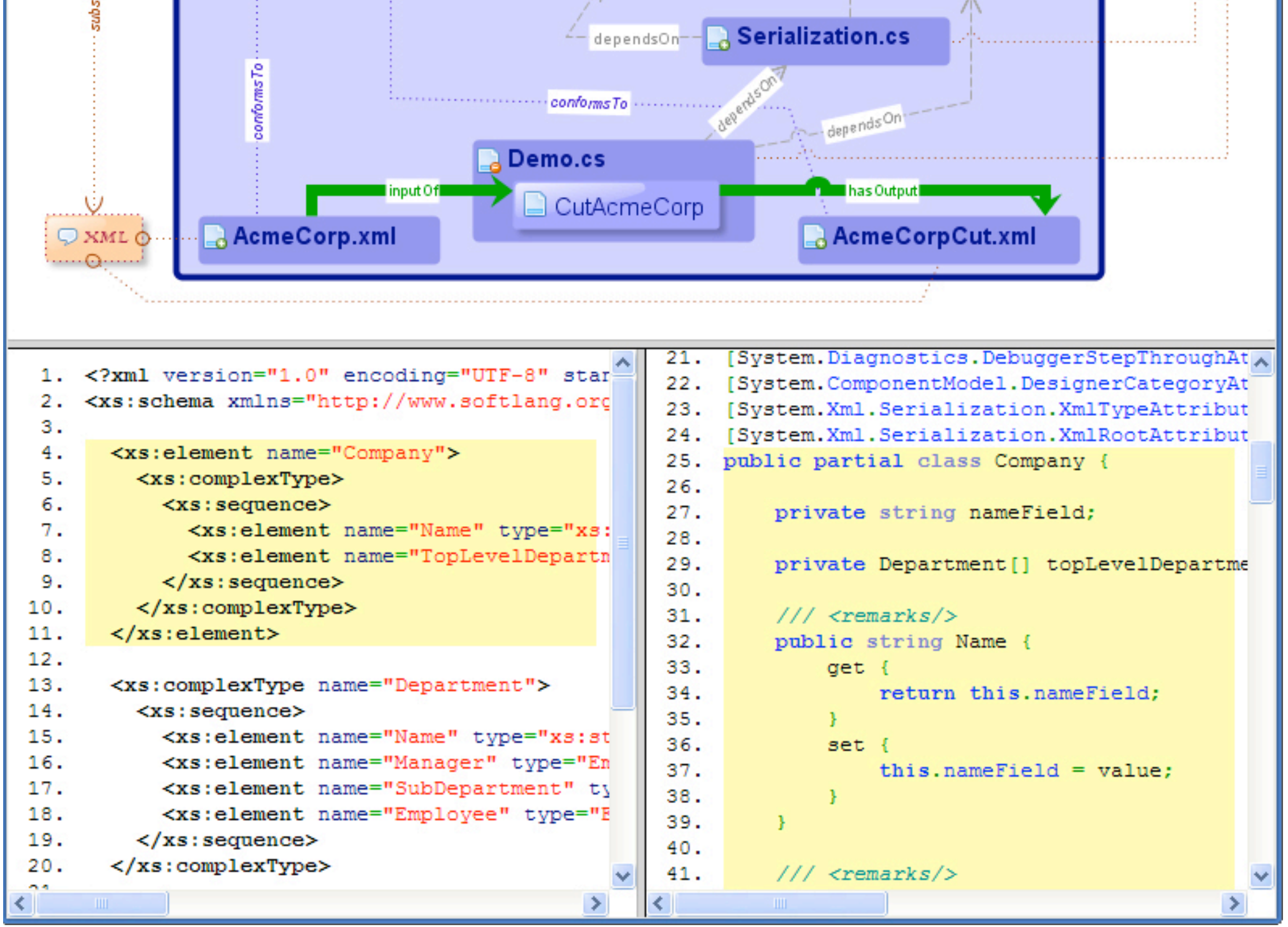
Linguistic architecture of xsd.exe






```
1.  <?xml version="1.0" encoding="UTF-8" star
2.  <xs:schema xmlns="http://www.softlang.org
3.
```

```
21.  [System.Diagnostics.DebuggerStepThroughAt
22.  [System.ComponentModel.DesignerCategoryAt
23.  [System.Xml.Serialization.XmlTypeAttribut
24.  [System.Xml.Serialization.XmlRootAttribut
```




```
1.  <?xml version="1.0" encoding="UTF-8" star
2.  <xs:schema xmlns="http://www.softlang.org
3.
4.    <xs:element name="Company">
5.      <xs:complexType>
6.        <xs:sequence>
7.          <xs:element name="Name" type="xs:
8.          <xs:element name="TopLevelDepartm
9.        </xs:sequence>
10.     </xs:complexType>
11.   </xs:element>
12.
13.   <xs:complexType name="Department">
14.     <xs:sequence>
15.       <xs:element name="Name" type="xs:st
16.       <xs:element name="Manager" type="Em
17.       <xs:element name="SubDepartment" ty
18.       <xs:element name="Employee" type="E
19.     </xs:sequence>
20.   </xs:complexType>
```

```
21.  [System.Diagnostics.DebuggerStepThroughAt
22.  [System.ComponentModel.DesignerCategoryAt
23.  [System.Xml.Serialization.XmlTypeAttribut
24.  [System.Xml.Serialization.XmlRootAttribut
25.  public partial class Company {
26.
27.      private string nameField;
28.
29.      private Department[] topLevelDepartme
30.
31.      /// <remarks/>
32.      public string Name {
33.          get {
34.              return this.nameField;
35.          }
36.          set {
37.              this.nameField = value;
38.          }
39.      }
40.
41.      /// <remarks/>
```

Overarching research questions

- What are problems with classic documentation?
- How to ontologically structure documentation?
- How to address related needs of developers?
- What is the underlying vocabulary and ontology?
- What sort of tool support is necessary or helpful?
- How to actually renarrate such documentation?

An ontological documentation

for using EMF+Xtext+ATL

for the purpose of DSML implementation

Discovery of entities and relationships

Id	Question	Relevant MegaL constructs
L1	Which languages can be identified?	Type *Language*
L2	Is one language contained in another?	Relationship *subsetOf*
A1	What artifacts participate in the scenario?	Type *Artifact*
A2	What is the language of each artifact?	Relationship *elementOf*
A3	Does an artifact conform to another artifact?	Relationship *conformsTo*
A4	Does an artifact define a language?	Relationship *defines*
F1	Is one artifact derived from another artifact?	Type *Function*
F2	What is domain and range of a function?	Function with domain & range
F3	How is a function applied?	Function application '$f(x) \mapsto y$'
F4	How is a function defined?	Relationship *defines*
R1	Are artifacts closely similar to each other?	Relationship *correspondsTo*
R2	Can a correspondence be structured?	Relationship *partOf*
R3	What causes a correspondence?	Function application '$f(x) \mapsto y$'
C1	Can the entity be described conceptually?	Type *Concept*
C2	Does the entity use the concept?	Relationship *uses*
C3	Does the entity help to use the concept	Relationship *facilitates*

Megamodel of the XML story

```
module XML import (Prelude) // Import basic vocabulary
XML : Language // Declare XML as a language entity
XSD : Language // and XSD (XML Schema), too
XSD subsetOf XML // Subset relationship on XSD and XML
xmlFile : Artifact // Declare artifact
xsdFiles : Artifact+ // Declare collection of artifacts
xmlFile elementOf XML // Assign language to artifact
xsdFiles elementOf XSD // Assign language to artifact
xmlFile conformsTo xsdFiles // XSD—based validation
```

Megamodel of the EMF story

```
module EMF import (Prelude)
Ecore : Language // As defined by metaMetaModel
Custom : Language // As defined by metaModel
metaModel : Artifact // A metamodel artifact
metaMetaModel : Artifact // The metametamodel
metaModel elementOf Ecore
metaMetaModel elementOf Ecore
metaModel conformsTo metaMetaModel
metaMetaModel conformsTo metaMetaModel
metaModel defines Custom
metaMetaModel defines Ecore
```

Megamodel of the ATL story

module ATL import (EMF) // *ATL depends on EMF*
transformation : Custom ⟶ Custom // *Function on DSL*
input : Artifact // *Input artifact*
output : Artifact // *Output artifact*
input elementOf Custom // *input (source) of transformation*
output elementOf Custom // *output (target) of transformation*
transformation(input) ⟼ output // *Function application*
ATL : Language // *The ATL language*
atlmodule : Artifact // *An ATL transformation module*
atlmodule elementOf ATL
atlmodule defines transformation // *Semantics of ATL module*

Megamodel of the Xtext story

module Xtext import (EMF) // *Xtext integrates with EMF*
Xtext : Language // *Xtext language*
grammar : Artifact // *An artifact for the grammar*
grammar elementOf Xtext // *An Xtext grammar*
EcoreWithoutOps : Language // *Relevant subset of Ecore*
EcoreWithoutOps subsetOf Ecore
metaModel elementOf EcoreWithoutOps // *Restriction of import*
metaModel correspondsTo grammar // *Correspondence*
generator : Xtext → EcoreWithoutOps // *Generator function*
generator(grammar) ↦ metaModel // *Generator application*
MWE2 : Language // *Language for generator configuration*
workflow : Artifact // *Workflow artifact*
workflow elementOf MWE2 // *Workflow is written in MWE2*
workflow defines generator // *Workflow defines generator function*

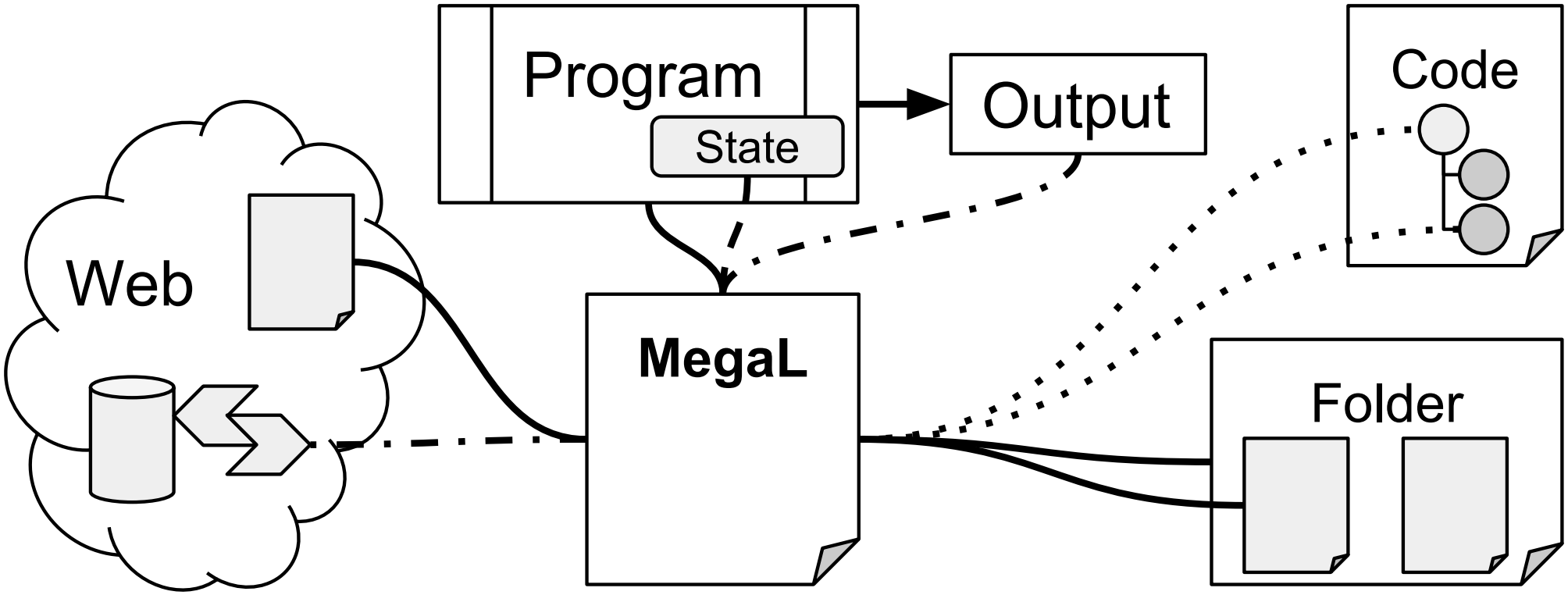
Megamodel of EMF Model API story

```
module EMFModelAPI import (
   EMF, // The EMF module is enhanced
   XML) // The XML module is needed for serialization
Java : Language // Java is a Language
EcoreJava : Language // A Java subset for EMF Model APIs
EcoreJava subsetOf Java // An EMF Model API is valid Java
EMFGenModel : Language // Language for the generator model
genModel : Artifact // Parameters of the generation
genModel elementOf EMFGenModel
genModel references metaModel // Referencing
EMFGenerator : EMFGenModel → EcoreJava
EMFGenerator(genModel) ↦ api // Application of generator
CustomObjects : Language // Object graphs for Custom
Serialization : CustomObjects → Custom
Deserialization : Custom → CustomObjects
XMI : Language // Format for default persistence for EMF
XMI subsetOf XML // XMI is a subset of XML
Custom subsetOf XMI // Custom uses default persistence
```

Deserialization : Custom → CustomObjects
XMI : Language *// Format for default persistence for EMF*
XMI subsetOf XML *// XMI is a subset of XML*
Custom subsetOf XMI *// Custom uses default persistence*
javaFiles : Artifact+ *// The modeled/defined API*
javaFiles elementOf EcoreJava
metaModel correspondsTo javaFiles *// Close resemblance*
javaFiles defines CustomObjects
javaFiles defines CustomSerialize
javaFiles defines CustomDeserialize
model : Artifact *// A serialized artifact*
model elementOf Custom *// ... of Custom language*
model conformsTo metaModel *// Conformance to metamodel*
objectGraph : Transient *// A runtime artifact*
objectGraph elementOf CustomObjects
objectGraph conformsTo javaFiles *// Conformance to Java classes*
CustomSerialize(objectGraph) ↦ model
CustomDeserialize(model) ↦ objectGraph

Interconnected linguistic architecture

Interconnection of model and system



URI-based resolution

`github://user/project/files/data.jar/content.xml/root/models/model#1`

- GitHub repo
- File
- Archive content
- File again
- XML-based (XPath-like) selection

```
// module EMF continued
metaMetaModel = 'eclipse:/org.eclipse.emf.ecore/model/Ecore.ecore'
```

MegaL

Semantic annotations

‘Identity’ links to Wikipedia etc.

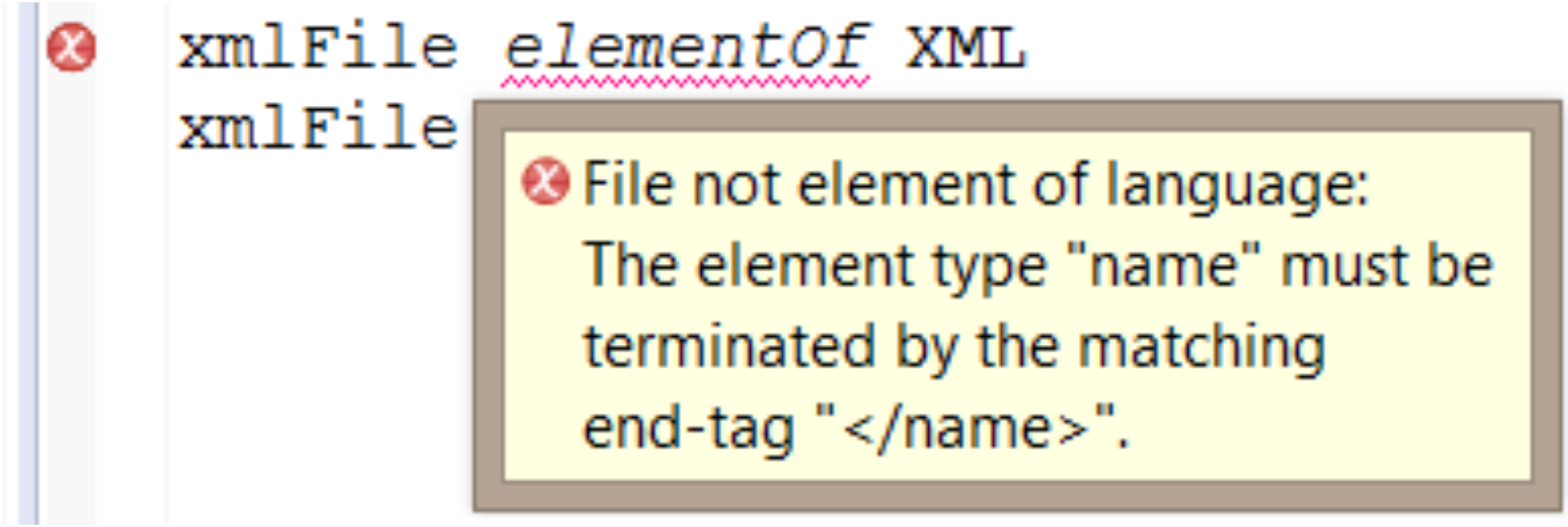
XML	http://dbpedia.org/page/XML
EMF	https://eclipse.org/modeling/emf/

```
// module XML continued
XML = 'http://dbpedia.org/page/XML'
```

MegaL

```
// module EMFModelAPI continued
Persistence : Concept
Persistence =
    'http://dbpedia.org/page/Persistence_(computer_science)'
CustomSerialize facilitates Persistence
CustomDeserialize facilitates Persistence
```

Pluggable analyses

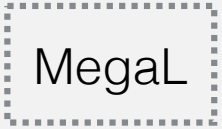


Plugin

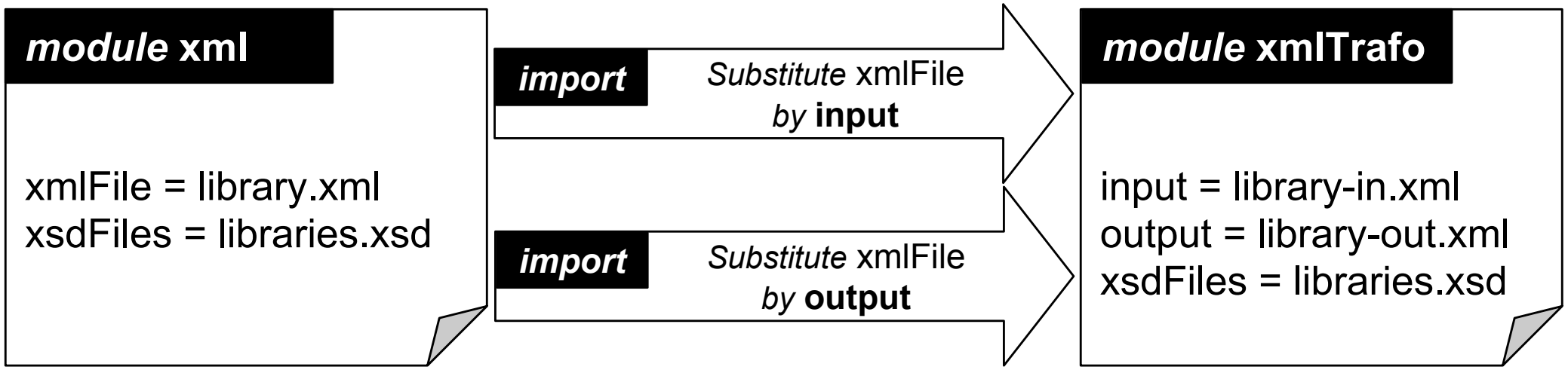
```
class XMLConformsToXSD extends MegaLEvaluator {
   // Returns an evaluation report on the model element
   protected Report<Void> evaluate(Relationship element) {
      // Use SAX for validation; translate exceptions to report
      …
   }
}
```

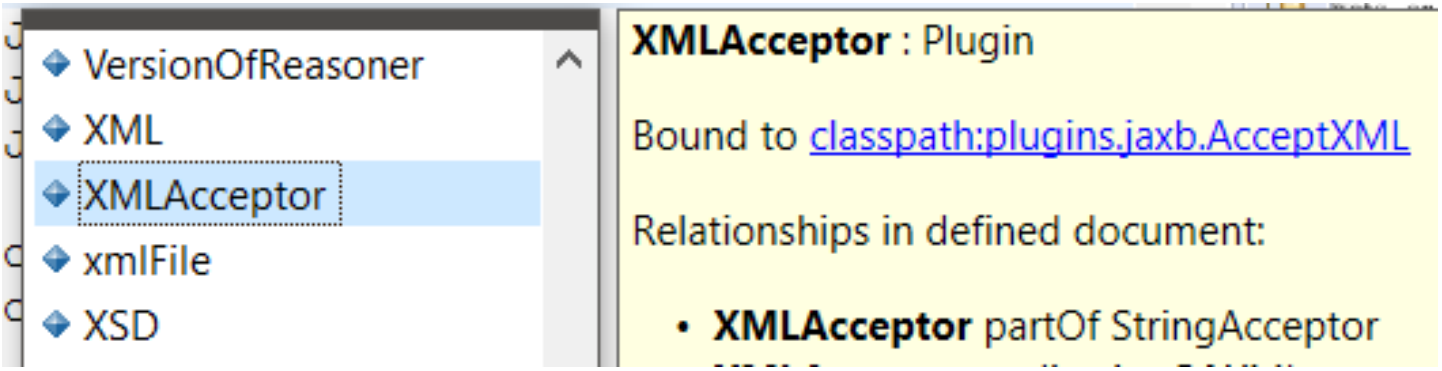
Plugging in

```
conformsTo < Artifact * Artifact // Relationship type per prelude
ConformsToEvaluator : Plugin // Root plugin for conformance
ConformsToEvaluator = "classpath:ConformsToEvaluator"
conformsTo evaluatedBy ConformsToEvaluator
XMLConformsToXSD : Plugin // XML/XSD conformance
XMLConformsToXSD = "classpath:XMLConformsToXSD"
XMLConformsToXSD partOf ConformsToEvaluator
SAX : Technology // Semantic annotation of plugin
SAX = 'http://dbpedia.org/page/Simple_API_for_XML'
XMLConformsToXSD uses SAX
```

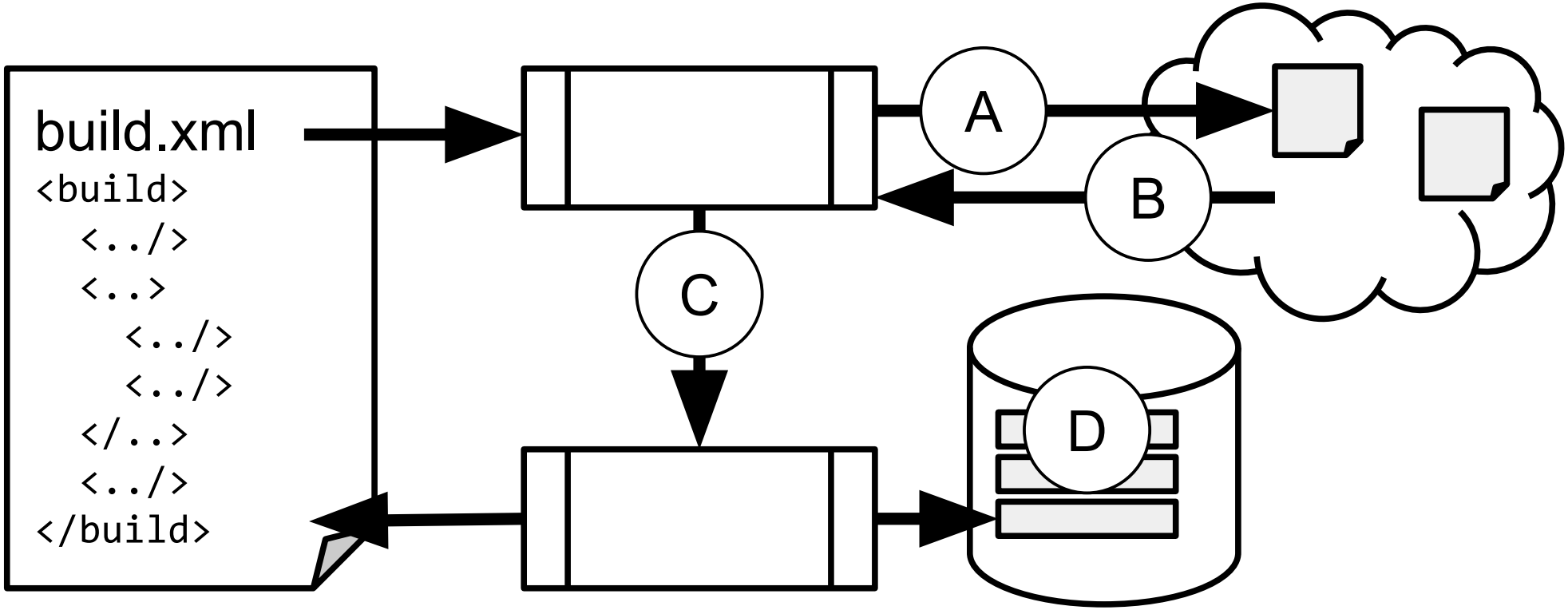


Modularized models





Transient artifacts



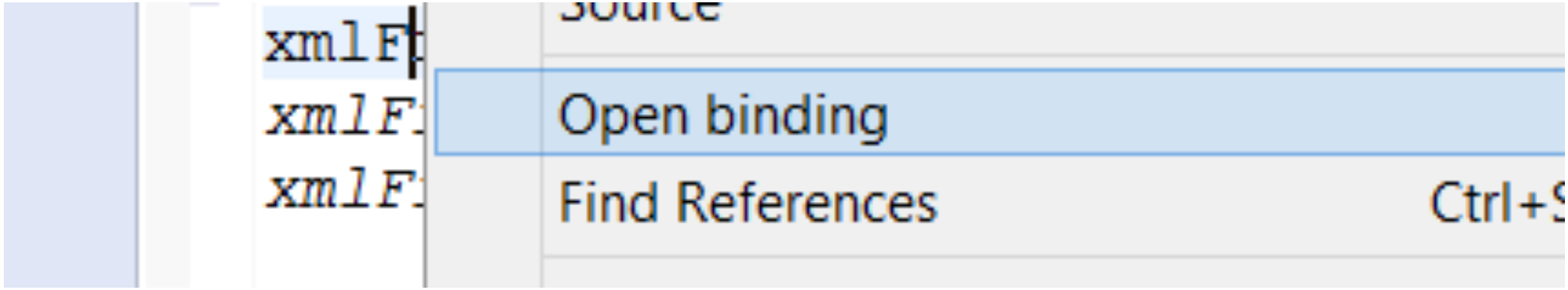
- A and B: web request and response
- C: piped program output
- D: transient data in memory or database

Model inference

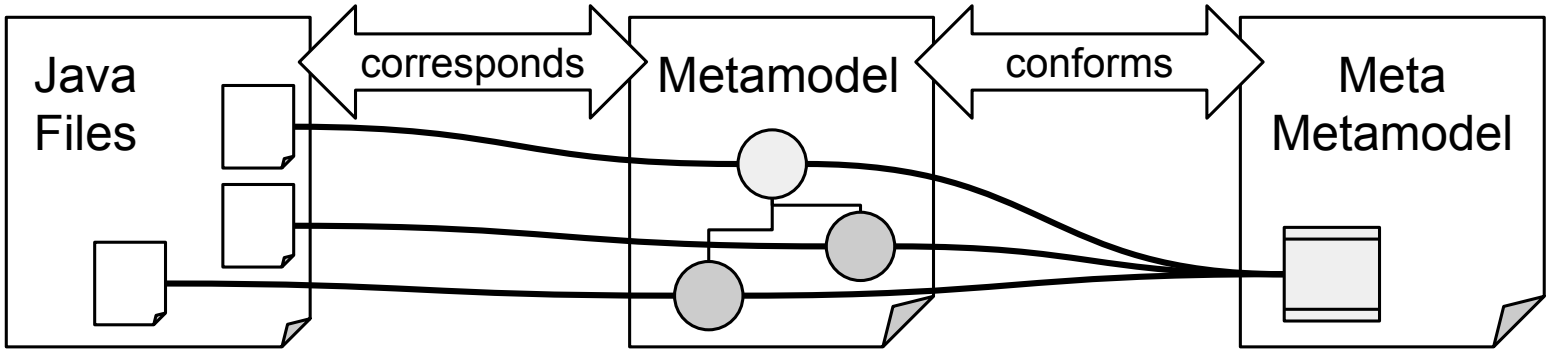
- Decomposition in parts
- Correspondence between parts
- Artifact bindings
- Subset relationship between languages

```
class EMFPartInferrer extends MegaLInferrer {
  // Returns an evaluation report and a model extension
  protected Report<Megamodel> infer(Entity element) { ... }
}
```

Explorable connections



Traceability links



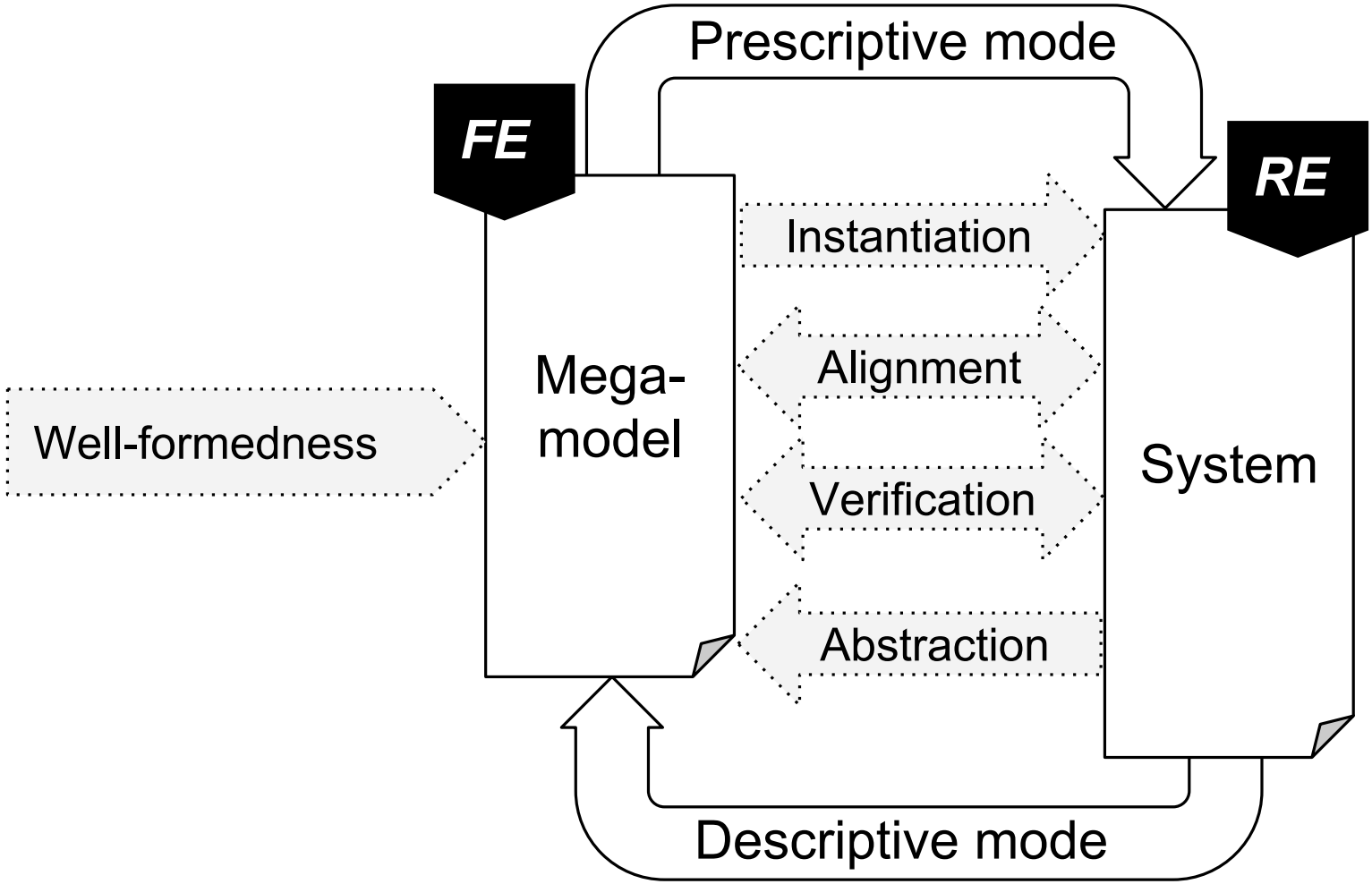
xsdFiles	javaFiles
/xs:schema/xs:complexType	org/softlang/company/xjc/Employee.java
/xs:schema/xs:element#0	org/softlang/company/xjc/Company.java
/xs:schema/xs:element#1	org/softlang/company/xjc/Department.java
xmlFile	objectGraph
company/department#0	org.softlang.company.xjc.Department@5fd1a6aa
employee#0	org.softlang.company.xjc.Employee@1a56a6c6
address:Utrecht	Utrecht
name:Erik	Erik
salary:12345	12345.0
employee#1	org.softlang.company.xjc.Employee@748e432b

Literature survey

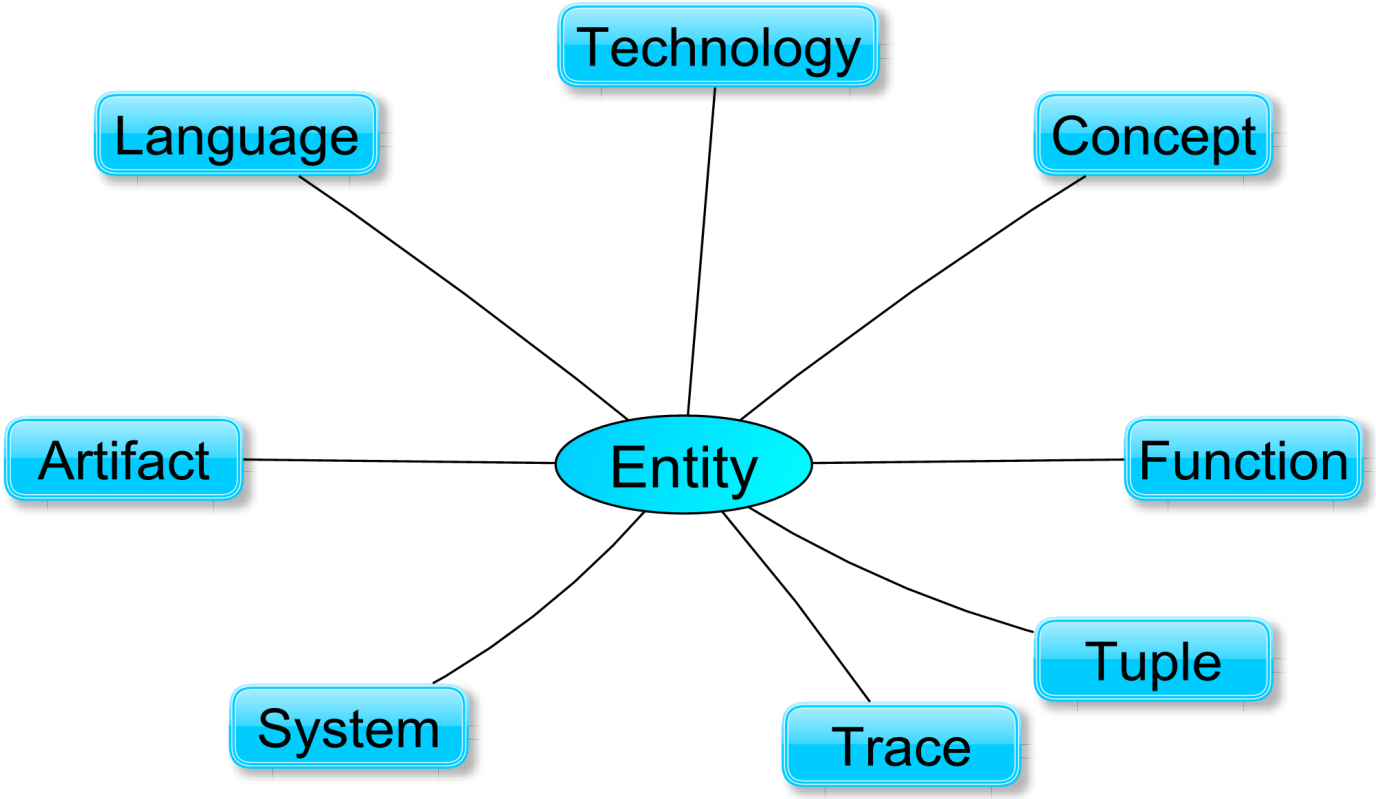
	3.8 Traceability links	3.1 Artifact binding	3.6 Model inference	3.3 Pluggable analyses	3.7 Explorable connections	3.4 Modularized models	3.2 Semantic annotations	3.5 Transient artifacts
[30]		·		●		●		·
[7]	●	·		·				
[25]	●	●	●	●		·		
[15]	·			·		·		
[11]	●	·			·			
[17]	●	·	●		·			
[12]	●	·	●	·				
[3]	●	·	●		·			
[4]	●	●	●	·	·			
([10])	●	●	·	·	●		●	
L_3		◯		◯			◯	
L_2			⊗	×	×	×		◯
L_1	⊗	×			◯	◯	×	×

- L_1-L_3 maturity levels
- **O** implementation
- × demonstration

Linguistic architecture in forward and reverse engineering



Axioms of linguistic architecture



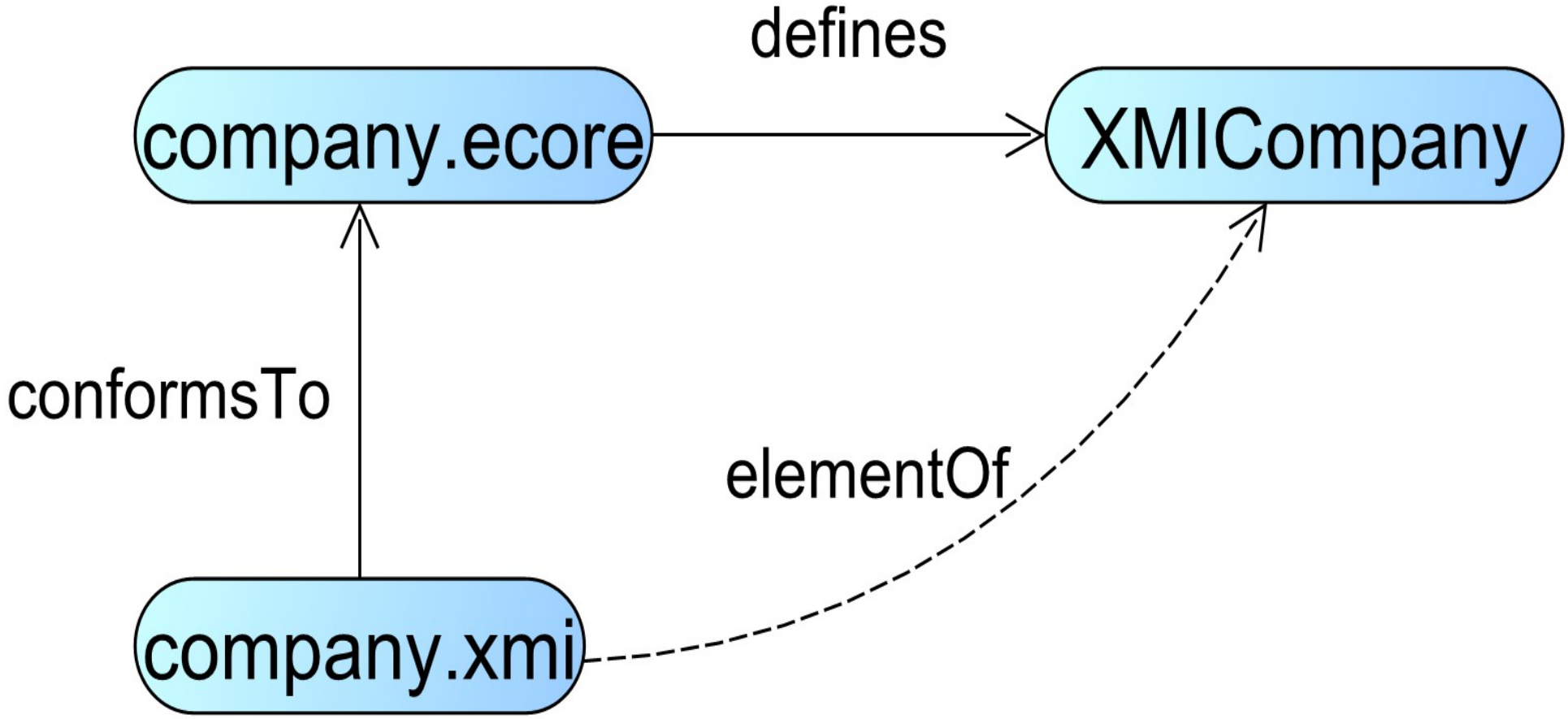
Entity types in megamodeling survey

Paper	Artifact	Function	Record	System	Technology	Language	Inf. resource	Fragment	Collection	Trace	Concept	Others
[1]	x	x	x				x					x
[2]	x	x	x		x					x		x
[3]	x			x	x						x	x
[4]					x	x	x				x	x
[5]	x						x	x		x		x
[6]	x		x									
[7]	x	x	x									
[8]	x									x		
[9]	x						x		x			
[10]	**x**	**x**	x		x	x		x	x			
[11]	x	x	x							x		
[12]				x								x
[13]	x	x								x		x
[14]	x	x										

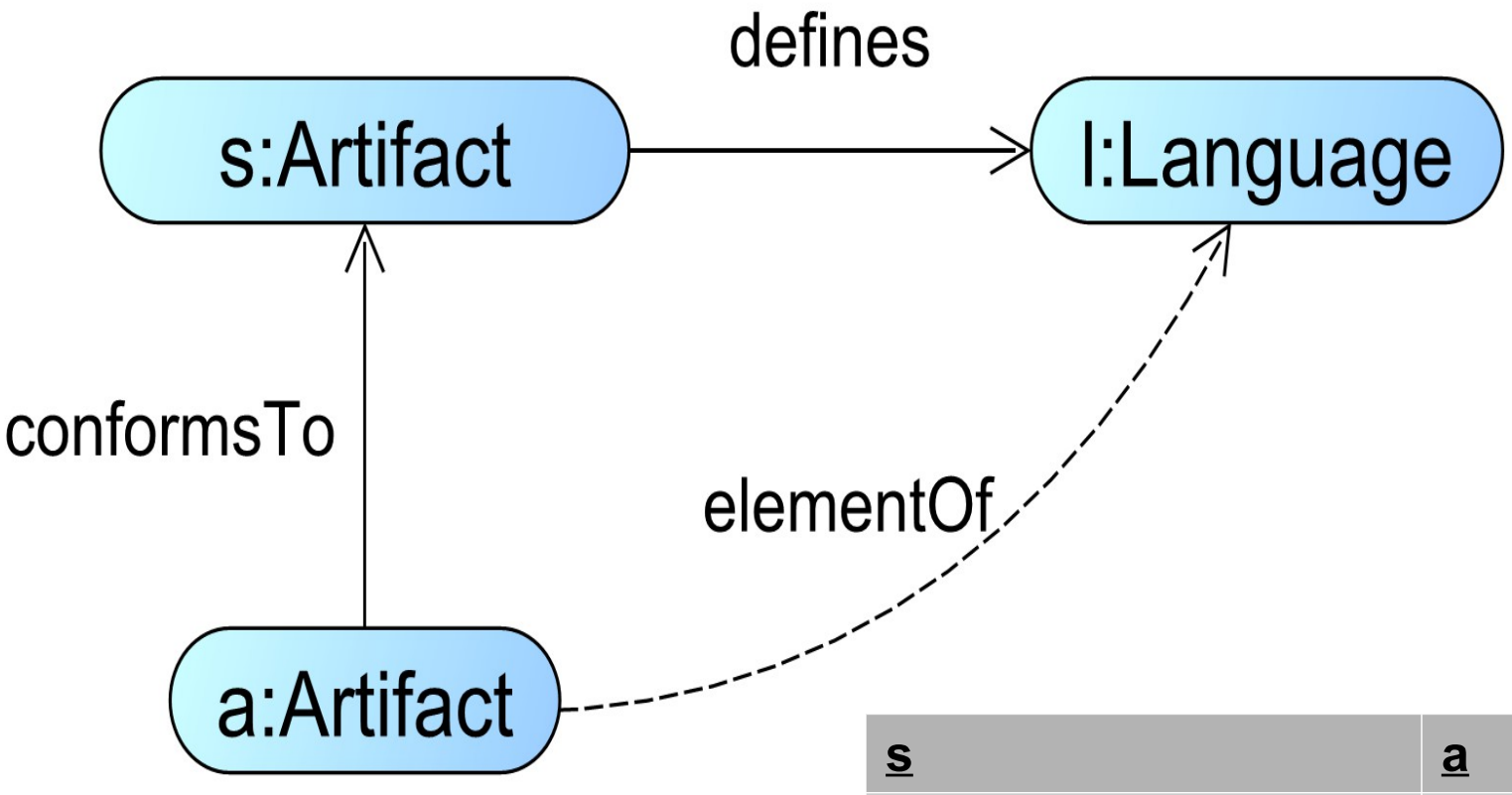
Relationship types in megamodeling survey

Paper	Conformance	Definition	Correspondence	Implementation	Usage	Membership	Typing	Dependency	Abstract rel.	Others
[1]					x					x
[2]	x									
[3]	x	x	x	x	x			x		x
[4]				x	x		x	x		x
[5]				x			x			x
[6]						x	x		x	
[7]										x
[8]									x	
[9]		x							x	
[10]	x	x	x	x		x	x	x	x	
[11]	x	x					x		x	
[12]		x								x
[13]							x			
[14]	x								x	

Understanding Membership

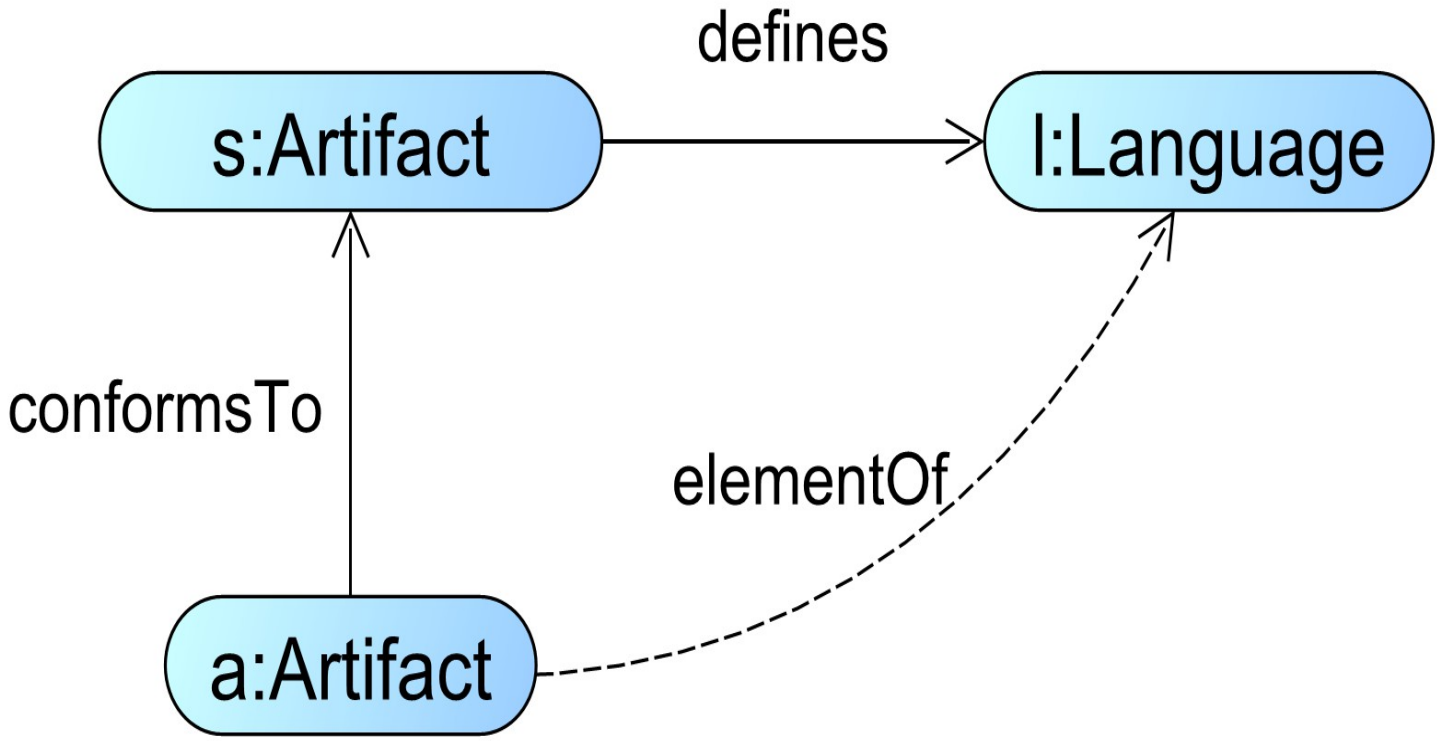


Understanding Membership



s	**a**
Grammar	Code
Schema	Instance
Metamodel	Model

Understanding Membership

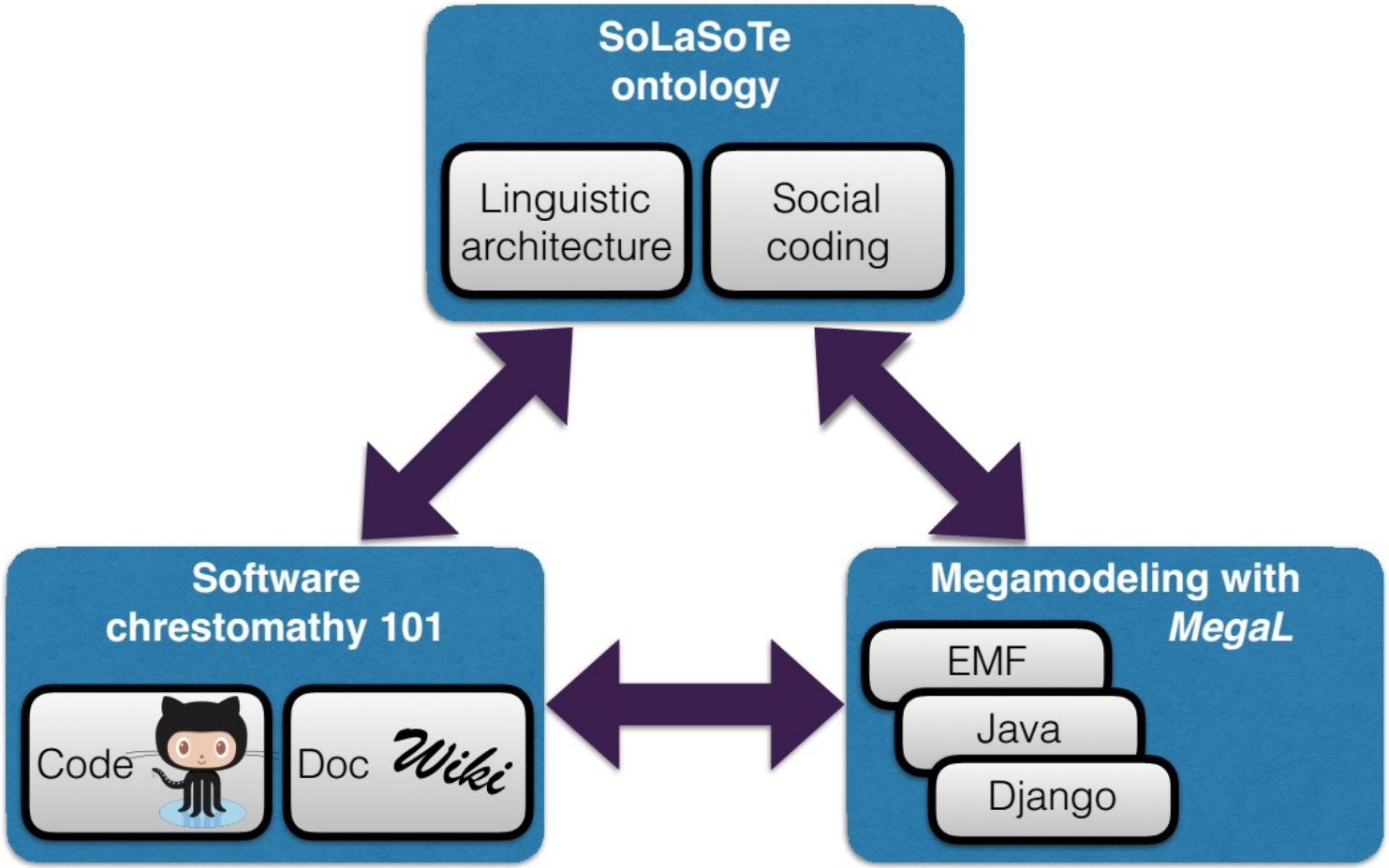


- $\mathsf{elementOf}(a, l) \Rightarrow \mathsf{Artifact}(a) \wedge \mathsf{Language}(l)\ldots$
- $\mathsf{elementOf}(a, l) \Leftarrow \exists s.\mathsf{defines}(s, l) \wedge \mathsf{conformsTo}(a, s).$

Understanding Membership

- $\mathsf{Specification}(a) \Rightarrow \mathsf{Artifact}(a)$.
- $\mathsf{Language}(l) \Rightarrow \exists s.\mathsf{Specification}(s) \wedge \mathsf{defines}(s, l)$ ...
- $\mathsf{defines}(a, e) \Rightarrow \mathsf{Artifact}(a) \wedge \mathsf{Entity}(e)$.
- $\mathsf{conformsTo}(a, s) \Rightarrow \mathsf{Artifact}(a) \wedge \mathsf{Artifact}(s)$.
- $\mathsf{conformsTo}(a, s) \Leftarrow (\forall p_a.\mathsf{partOf}(p_a, a) \wedge \exists p_s.\mathsf{partOf}(p_s, s)$
  $\wedge\ \mathsf{conformsTo}(p_a, p_s))$
  $\vee \exists t.\mathsf{defines}(s, t) \wedge \mathsf{elementOf}(a, t)$.

Ontology engineering



State of the union

Papers on megamodeling

- Modeling the Linguistic Architecture of Software Products: (.html)
- Interpretation of Linguistic Architecture: (.html)
- Interconnected Linguistic Architecture: (.html)
- Axioms of linguistic architecture: (.html)
- Coupled Software Transformations—-Revisited: (.html)
- Relationship Maintenance in Software Language Repositories: (.html)
- Language Support for Megamodel Renarration: (.html)

Megamodeling languages

- MegaL: (.html)
  - MegaL/Xtext: (GitHub)
  - MegaL/Text: (GitHub)
- LAL: (.html)
- Ueber: (.html)

http://www.softlang.org/mega

Thank you for your time and interest*!*