Trace Link Recovery using Static Program Analysis

B.Sc. Thesis Colloquium/Defense

Maximilian Meffert

18th January, 2018

University of Koblenz-Landau

General Information

Maximilian Meffert

Mat.-Nr.:	210 101 205
E-Mail:	maxmeffert@uni-koblenz.de
GitHub:	`https://github.com/maxmeffert`
Thesis-Repo.:	`https://github.com/maxmeffert/BScThesis`

Supervisors

Prof. Dr. Ralf Lämmel University of Koblenz-Landau, *Institute for Computer Science*
M.Sc. Johannes Härtel University of Koblenz-Landau, *Institute for Computer Science*

In a Nutshell

I implemented a system which

- recovers **trace links**
- with semantics of **Linguistic Architectures**
- using **Static Program Analysis** techniques

Motivation: Software as Cognitive Challenge i

View on the "Black Eye" galaxy provided by [6].

Modern Software Systems are:

- large
  (allover artifact count)
- heterogeneous
  (languages involved)

$\Rightarrow$ challenging for program comprehension tasks

Motivation: Software as Cognitive Challenge ii

```java
@XmlRootElement(name="employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee {

    @XmlAttribute
    private int id;

    @XmlAttribute
    private String name;

    @XmlAttribute
    private int age;

    @XmlAttribute
    private double salary;

    private Department department;

    private Department managedDepartment;

}
```

```xml
<xs:complexType name="employee">
 <xs:attribute name="id" type="xs:int"
       use="required"/>

 <xs:attribute name="name" type="xs:string"/>

 <xs:attribute name="age" type="xs:int"
       use="required"/>

 <xs:attribute name="salary" type="xs:double"
       use="required"/>

 <xs:sequence>

   <xs:element ref="department" minOccurs="0"/>

   <xs:element name="managedDepartment"
         type="department" minOccurs="0"/>

 </xs:sequence>
</xs:complexType>
```

```xml
<employee name="Max" age="26" salary="55000.0"/>
```

- Structural similarities among Java-, XSD- and XML-files can be observed.
- If you are new to a project, you want to know, what belongs together.

Traceability i

We apply *Traceability* as concept supporting program comprehension.

Definition (Trace)

(Noun) A specified triplet of elements comprising: *source artifact*, *target artifact* and a *trace link* associating the two *trace artifacts*. [2]

(Verb) The act of following a trace link. [2]

Definition (Traceability)

The potential for traces to be established and used. [2]

Linguistic Architectures i

Linguistic Architectures are ...

- ... the axiomatic study of software systems from software language perspective. [3] [4] [5] [1].
- ... used to provide semantics for trace links.

Linguistic Architectures ii

Axiom (partOf)

$$partOf(p, w) \Rightarrow Entity(p) \wedge Entity(w).$$
$$partOf(p, w) \Leftarrow p \text{ is a constituent part of } w.$$

Definition (porperPartOf)

$$\text{properPartOf}(p, w) \Rightarrow \text{Entity}(p) \wedge \text{Entity}(w).$$
$$\text{properPartOf}(x, y) \Leftarrow \text{partOf}(x, y) \wedge \neg\text{partOf}(y, x). \quad [7]\ [8]$$

Linguistic Architectures iii

Axiom (Fragment)

$$Fragment(f) \Rightarrow Artifact(a) \wedge \neg(File(f) \vee Folder(f)).$$
$$Fragment(f) \Rightarrow \exists a.Artifact(a) \wedge properPartOf(f, a).$$

Definition (fragmentOf)

$$\text{fragmentOf}(f, x) \Rightarrow \text{Fragment}(f) \wedge \text{Artifact}(x).$$
$$\text{fragmentOf}(f, x) \Leftarrow \text{Fragment}(f) \wedge \text{Artifact}(x) \wedge \text{properPartOf}(f, x).$$

Linguistic Architectures iv

Axiom (correspondsTo)

Two artifacts representing the same data or information.

$$correspondsTo(x, y) \Rightarrow Artifact(x) \wedge Artifact(y).$$

$$\begin{aligned} correspondsTo(x, y) \Leftarrow\ & (\forall px.properPartOf(px, x) \Rightarrow \exists py.properPartOf(py, y) \wedge correspondsTo(px, py)) \\ & \wedge (\forall py.properPartOf(py, y) \Rightarrow \exists px.properPartOf(px, x) \wedge correspondsTo(py, px)) \\ & \vee (\nexists p.properPartOf(p, x) \vee properPartOf(p, y)) \wedge sameAs(x, y). \end{aligned}$$

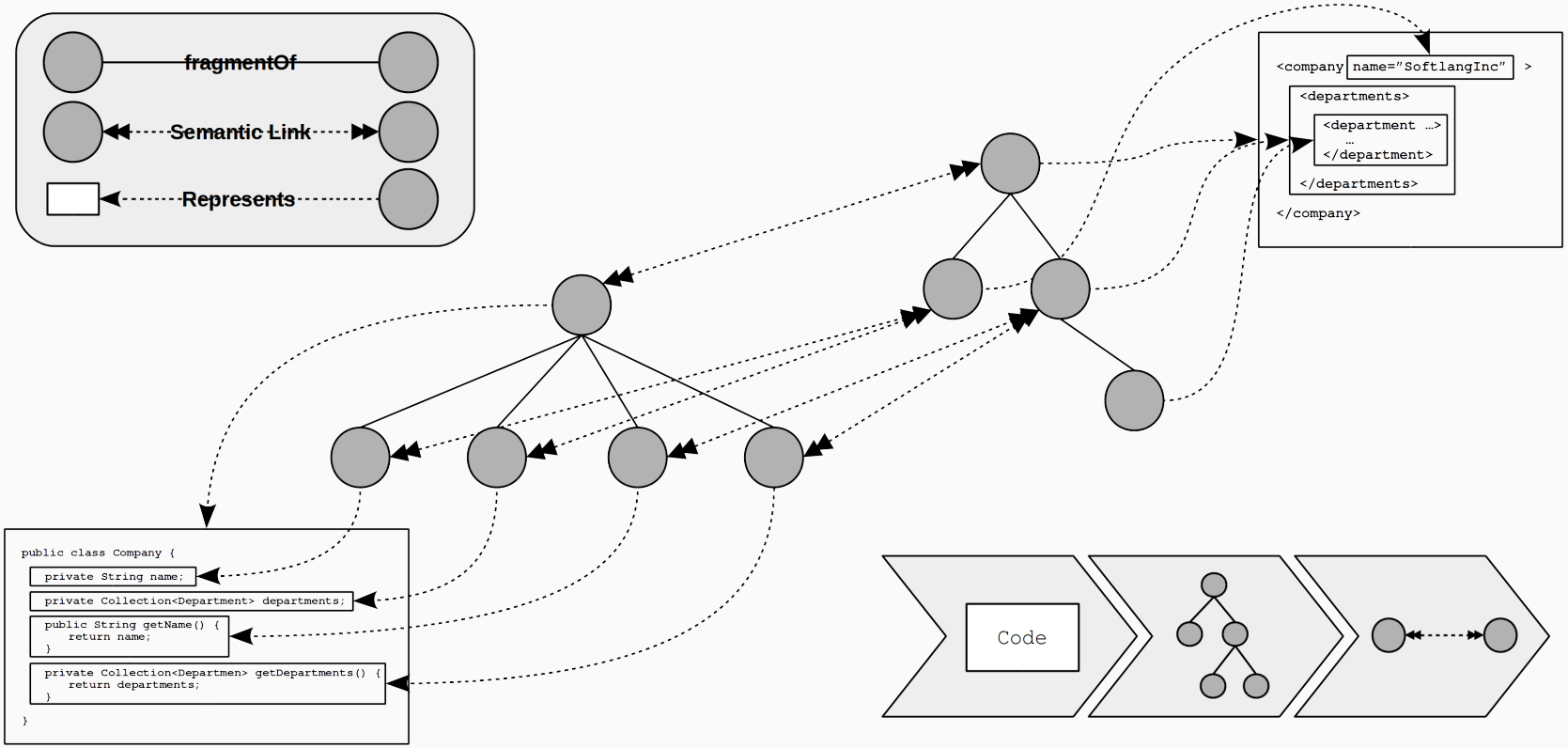
Axiom (conformsTo)

Two artifacts where one defines the other.

$$conformsTo(a, d) \Rightarrow Artifact(a) \wedge Artifact(d).$$

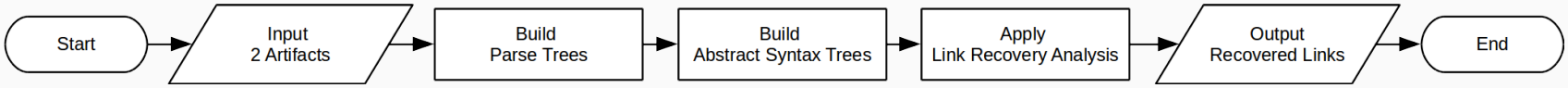
$$\begin{aligned} conformsTo(a, d) \Leftarrow\ & (\forall pa.properPartOf(pa, a) \wedge \exists pd.properPartOf(pd, d) \wedge conformsTo(pa, pd)) \\ & \vee \exists l.defines(d, l) \wedge elementOf(a, l). \end{aligned}$$

Trace Link Recovery Approach i



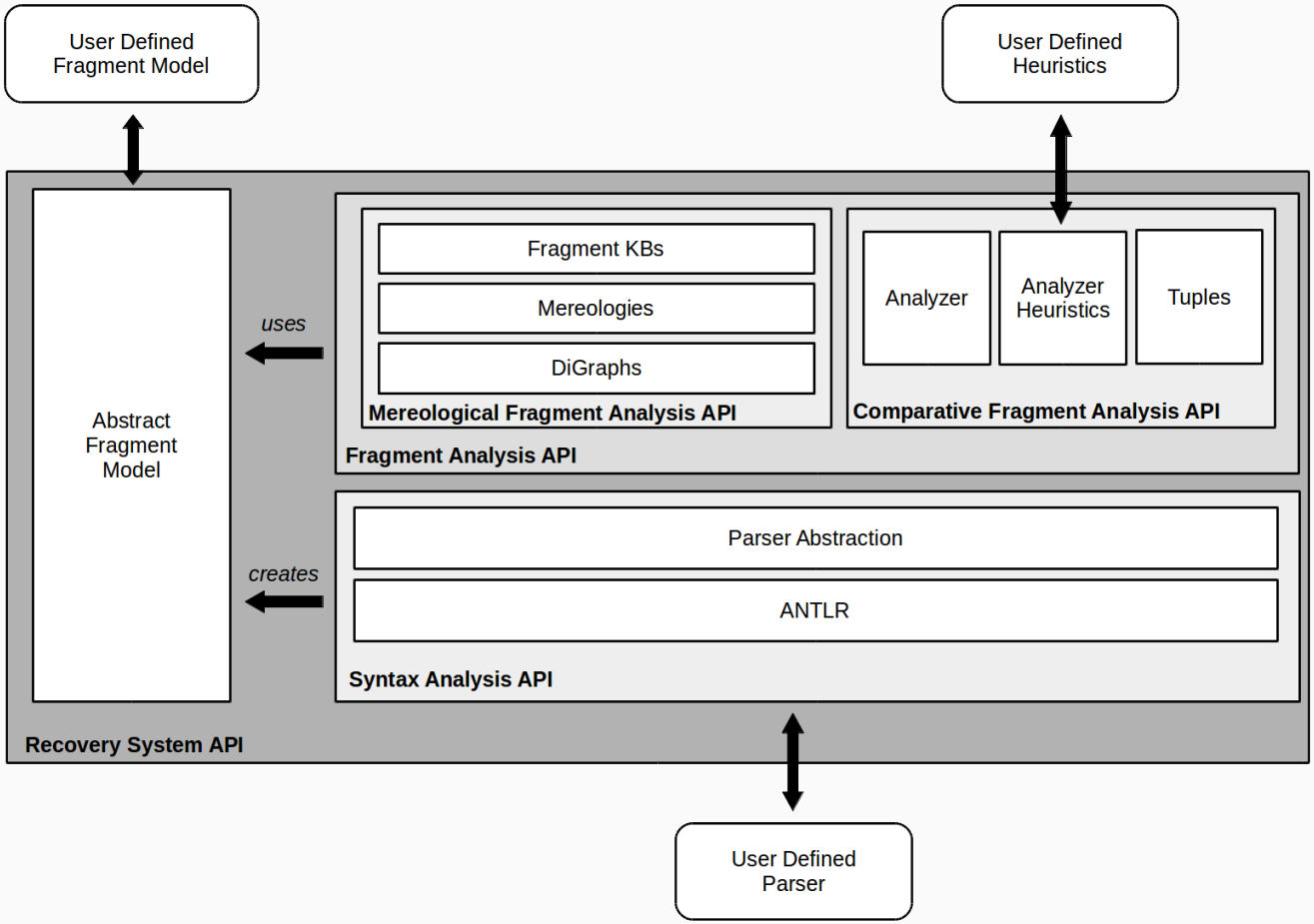
Trace Link Recovery Approach ii

Trace Link Recovery Process



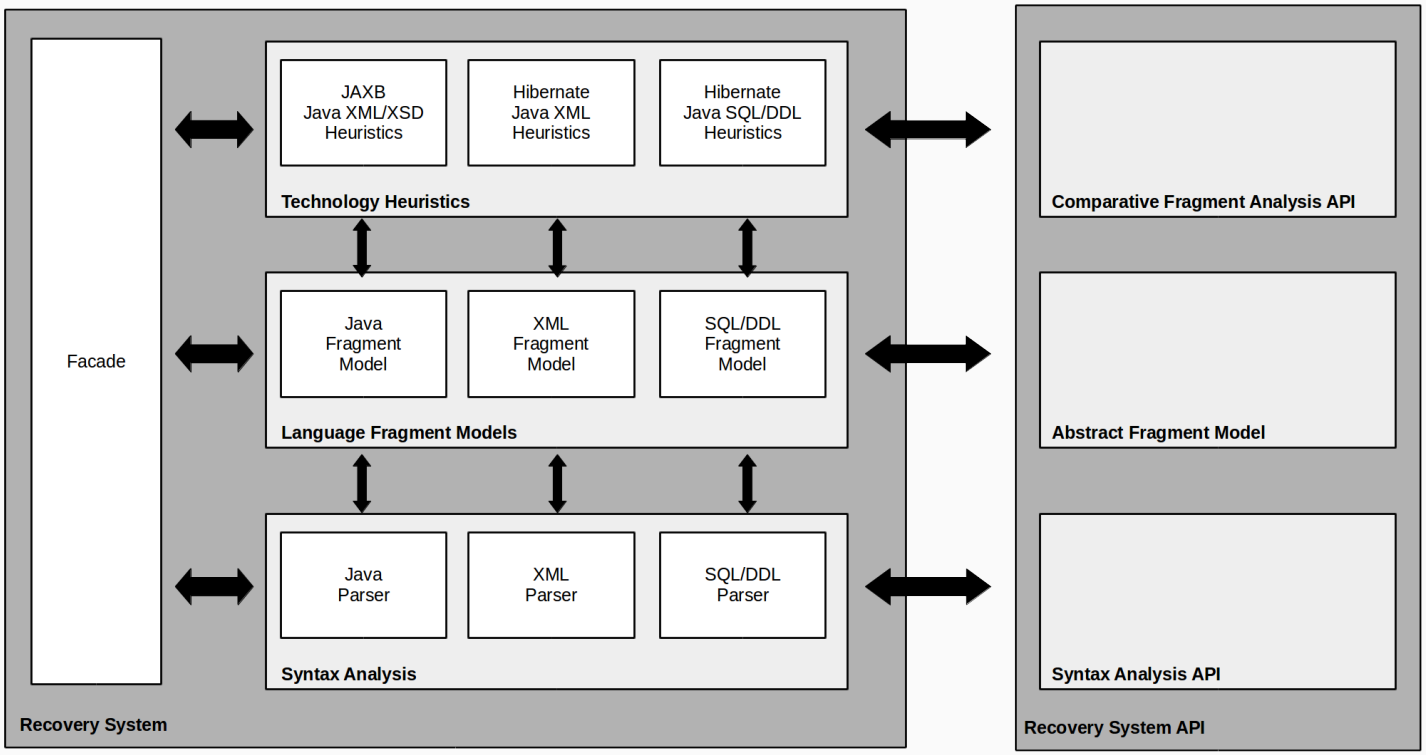
1. Create two Parse Trees from both inputs respectively.
2. Both Parse Trees are transformed to Abstract Syntax Trees supporting analysis of the represented contents.
3. Comparative Analysis of both Abstract Syntax Trees: while traversing through both trees we check whether each pair of nodes is a trace link.

Trace Link Recovery System i



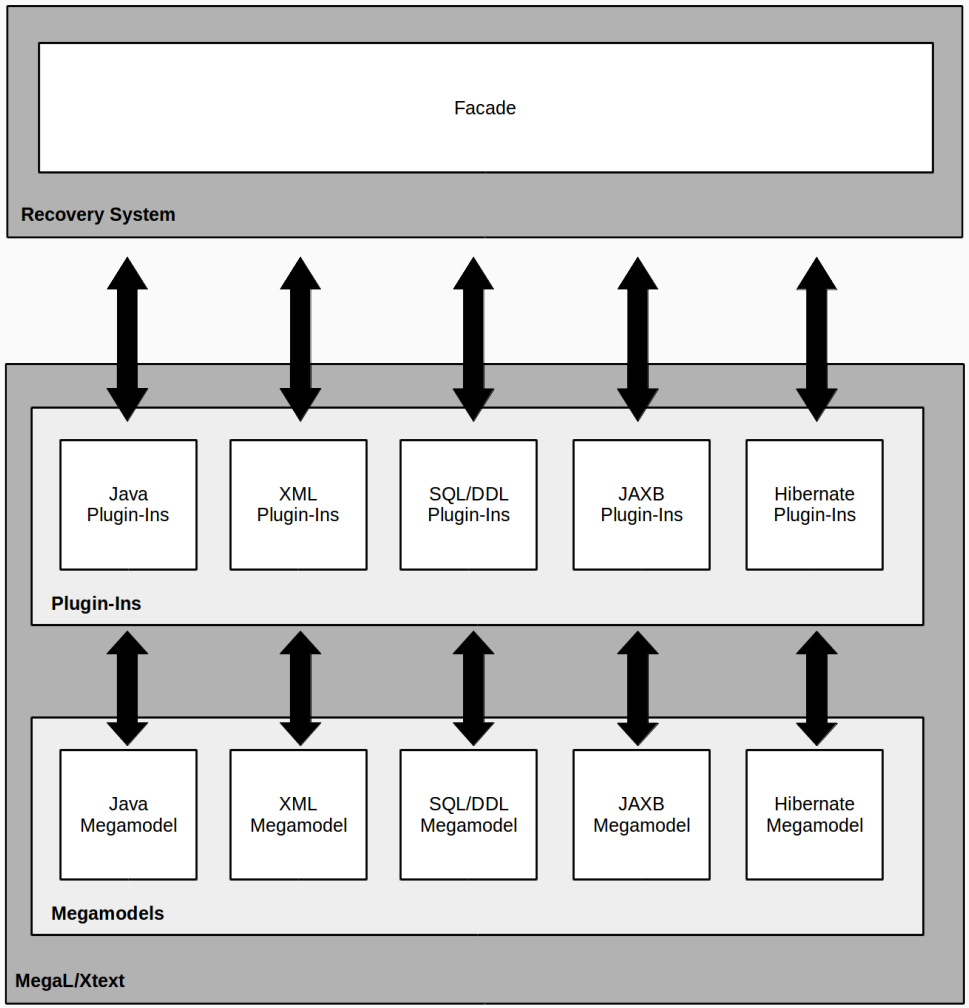
Recovery System API

Trace Link Recovery System ii



Recovery System

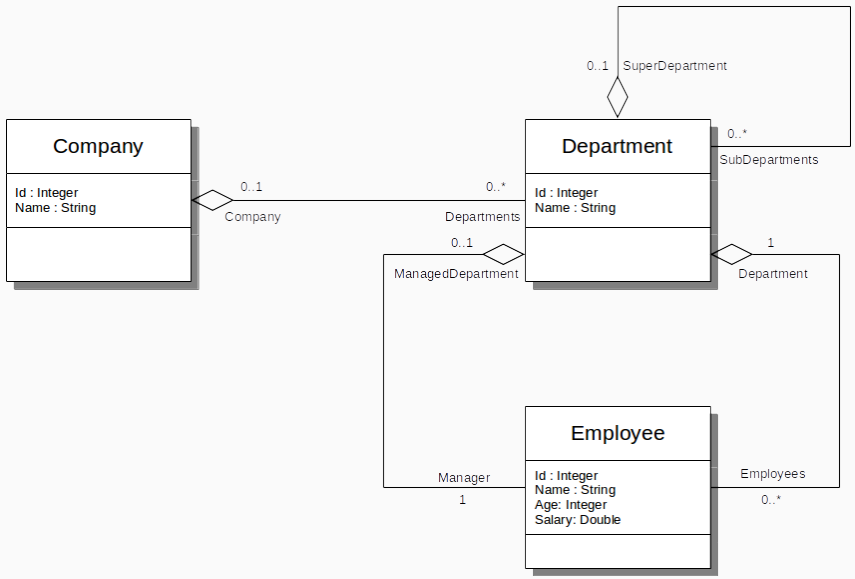
Trace Link Recovery System iii



Integration into MegaL/Xtext

Mini Case Study i

- Evaluates whether the Recovery Systems preserves semantics of Linguistic Architectures.
- Uses a JAXB and Hibernate implementation of the 101companies HRMS model as corpus.



https://101wiki.softlang.org/101:@system

Mini Case Study ii

Setup (1)

```
companyJavaFile : File
companyJavaFile = '/input/Company.java'

companyHbmFile : File
companyHbmFile = '/input/Company.hbm.xml'

departmentJavaFile : File
departmentJavaFile = '/input/Department.java'

departmentHbmFile : File
departmentHbmFile = '/input/Department.hbm.xml'

employeeJavaFile : File
employeeJavaFile = '/input/Employee.java'

employeeHbmFile : File
employeeHbmFile = '/input/Employee.hbm.xml'

companiesXmlFile: File
companiesXmlFile = '/input/companies.xml'

companiesXsdFile: File
companiesXsdFile = '/input/companies.xsd'

comaniesSqlFile : File
comaniesSqlFile = '/input/companies.ddl.sql'
```

Setup (2)

```
companyJavaFile correspondsTo companiesXsdFile
companyJavaFile correspondsTo comaniesSqlFile
companyJavaFile correspondsTo companyHbmFile

departmentJavaFile correspondsTo companiesXsdFile
departmentJavaFile correspondsTo comaniesSqlFile
departmentJavaFile correspondsTo departmentHbmFile

employeeJavaFile correspondsTo companiesXsdFile
employeeJavaFile correspondsTo comaniesSqlFile
employeeJavaFile correspondsTo employeeHbmFile

companiesXmlFile conformsTo companiesXsdFile
```

Absolute Metrics

$$\#\text{Artifact} := |\{x : \text{Artifact}(x)\}|$$

$$\#\text{File} := |\{x : \text{File}(x)\}|$$

$$\#\text{Fragment} := |\{x : \text{Fragment}(x)\}|$$

$$\#\text{partOf} := |\{(x, y) : \text{partOf}(x, y)\}|$$

$$\#\text{fragmentOf} := |\{(x, y) : \text{fragmentOf}(x, y)\}|$$

$$\#\text{correspondsTo} := |\{(x, y) : \text{correspondsTo}(x, y)\}|$$

$$\#\text{conformsTo} := |\{(x, y) : \text{conformsTo}(x, y)\}|$$

Relative Metrics

$$\#\text{partOf}(x) := |\{y : \text{partOf}(y, x)\}|$$

$$\#\text{fragmentOf}(x) := |\{y : \text{fragmentOf}(y, x)\}|$$

$$\#\text{correspondsTo}(y) := |\{x : \text{correspondsTo}(x, y') \wedge \text{fragmentOf}(y', y)\}|$$

$$\#\text{conformsTo}(y) := |\{x : \text{conformsTo}(x, y') \wedge \text{fragmentOf}(y', y)\}|$$

$$\#LHS(R) := |\{x : R(x, y)\}|$$

$$\#RHS(R) := |\{y : R(x, y)\}|$$

Mini Case Study vi

#Artifact	#File	#Fragment
483	9	474

#partOf	#fragmentOf	#correspondsTo	#conformsTo
1847	1837	106	81

- #Artifact = #File + #Fragment
- #partOf = #fragmentOf + 10 (due to MegaL/Xtextplug-in configuration)

Mini Case Study vii

x	#partOf(x)	#fragmentOf(x)
companyJavaFile	9	9
companyHbmFile	51	51
departmentJavaFile	20	20
departmentHbmFile	112	112
employeeJavaFile	15	15
employeeHbmFile	76	76
companiesXmlFile	89	89
companiesXsdFile	85	85
comaniesSqlFile	17	17
Sum:	474	474

- $\forall x.\#\text{partOf}(x) = \#\text{fragmentOf}(x)$
- $\#\text{Fragment} = \sum_{x} \#\text{fragmentOf}(x)$

$\Rightarrow$ Indicates, each fragment is indeed a part.

Mini Case Study viii

x	#correspondsTo(x)	#conformsTo(x)
companyJavaFile	12	0
companyHbmFile	4	0
departmentJavaFile	15	0
departmentHbmFile	5	0
employeeJavaFile	17	0
employeeHbmFile	5	0
companiesXmlFile	0	0
companiesXsdFile	16	67
comaniesSqlFile	11	0

- $x \neq \text{companiesXsdFile} \Rightarrow \#\text{conformsTo}(x) = 0$
- $\#\text{correspondsTo}(\text{companiesXmlFile}) = 0$

$\Rightarrow$ Indicates no correspondence between model- and instance-level entities. Vice versa, conformance only occurs between model- and instance-level entities.

Mini Case Study ix

R	$\#LHS(R)$	$\#RHS(R)$
partOf	484	121
fragmentOf	474	117
correspondsTo	68	68
conformsTo	68	19

- $\#LHS(\text{correspondsTo}) = \#RHS(\text{correspondsTo})$

$\Rightarrow$ Indicates correspondence is a bijection as postulated.

References i

[1] Favre, J., Lämmel, R., Varanovich, A.: Modeling the Linguistic Architecture of Software Products. In: MoDELS. Lecture Notes in Computer Science, vol. 7590, pp. 151–167. Springer (2012)

[2] Gotel, O., Cleland-Huang, J., Hayes, J.H., Zisman, A., Egyed, A., Grünbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J.I., Mäder, P.: Traceability Fundamentals. In: Software and Systems Traceability, pp. 3–22. Springer (2012)

[3] Heinz, M., Lämmel, R., Varanovich, A.: Axioms of Linguistic Architecture. In: MODELSWARD. pp. 478–486. SciTePress (2017)

[4] Lämmel, R.: Coupled software transformations revisited. In: SLE. pp. 239–252. ACM (2016)

References ii

[5] Lämmel, R., Varanovich, A.: Interpretation of Linguistic Architecture. In: ECMFA. Lecture Notes in Computer Science, vol. 8569, pp. 67–82. Springer (2014)

[6] NASA, STScI: Dust Band Around the Nucleus of "Black Eye Galaxy" M64 (2004), `http://hubblesite.org/image/1447/news_release/2004-04`, retrieved 29th December, 2017

[7] Varzi, A.C.: Parts, Wholes, and Part-Whole Relations: The Prospects of Mereotopology. Data Knowl. Eng. 20(3), 259–286 (1996)

References iii

[8] Varzi, A.C.: Mereology. In: Zalta, E.N. (ed.) The Stanford encyclopedia of philosophy. Winter 2016 ed. edn. (2016), https://plato.stanford.edu/archives/win2016/entries/mereology/