

Presentation at
9th IFIP WG 2.11 Meeting, Waterloo, Dec 2010

http://resource-aware.org/do/view/WG211/M9Schedule*

These slides are *potentially* available here:

<http://userpages.uni-koblenz.de/~laemmel/101203-waterloo.pdf>

Modeling Programming Technologies

Ralf Lämmel (Uni of Koblenz-Landau)

joint work with

Jean-Marie Favre (OneTree Technologies, Luxembourg)

Dragan Gasevic (Athabasca University, Canada).

Special thanks to Jean-Marie for several cool slides that he contributed to our joint SLE/GTTSE 2010 tutorial.

We have a problem.

A word cloud of various technologies and standards is arranged around the central text. The words are in different sizes and shades of blue. Some of the visible words include: XPath, TXL, Sesame, JPA, Jena, Rose, JDBC, EMF.gen, XText, jDOM, JAXB, Jersey, RDF(S), UTF8, ODM, XSD, MOF, VLDB, JeanBeans, Teneo, UML, BNF, Stratego, xerces, GWT, SLE2010, Json, OCL, sax, OWL, Ecore, Ralf, RDFS, Rest, ORACLE, RDF, Jean, OMG, MySQL, XMI, JMI, EMF, JMF, XSD, ArgoUML, xalan, ODBC, SparQL, XMLSpy, Yacc, RDFa, LALR, Prolog, XSLT, JAXP, SBVR, DOM, Java, ATOM, ER, CFG, SQL, DDL, Dragan, SQL, Antlr, QVT, TENEIO, XLST, sed, Awk, DTD, TCS, grep, Hibernate, ASCII, XQuery, and Saxon.

Today's Issues

- **Silos of knowledge**
- **Combining technologies**
- **Complexity of technologies**
- **Entering a new space**
- **Teaching technologies?**

A course on Programming Technologies

— [The Expression Problem

— [The Visitor Design Pattern

— [Parsing

— [XML Processing

— [XML Validation

— [XML Data Binding

— [Database Access

— [O/R Mapping

— [Model View Controller

— [More Design Patterns

— [Reflection

— [Aspect-Oriented Programming

— [Functional OO Programming

— [Combinator Libraries

— [Generic Programming

— [Programming with Threads

— [Distributed Programming

— [WebService Programming

Today's Issues

- **Silos of knowledge**
- **Combining technologies**
- **Complexity of technologies**
- **Entering a new space**
- **Teaching technologies?**

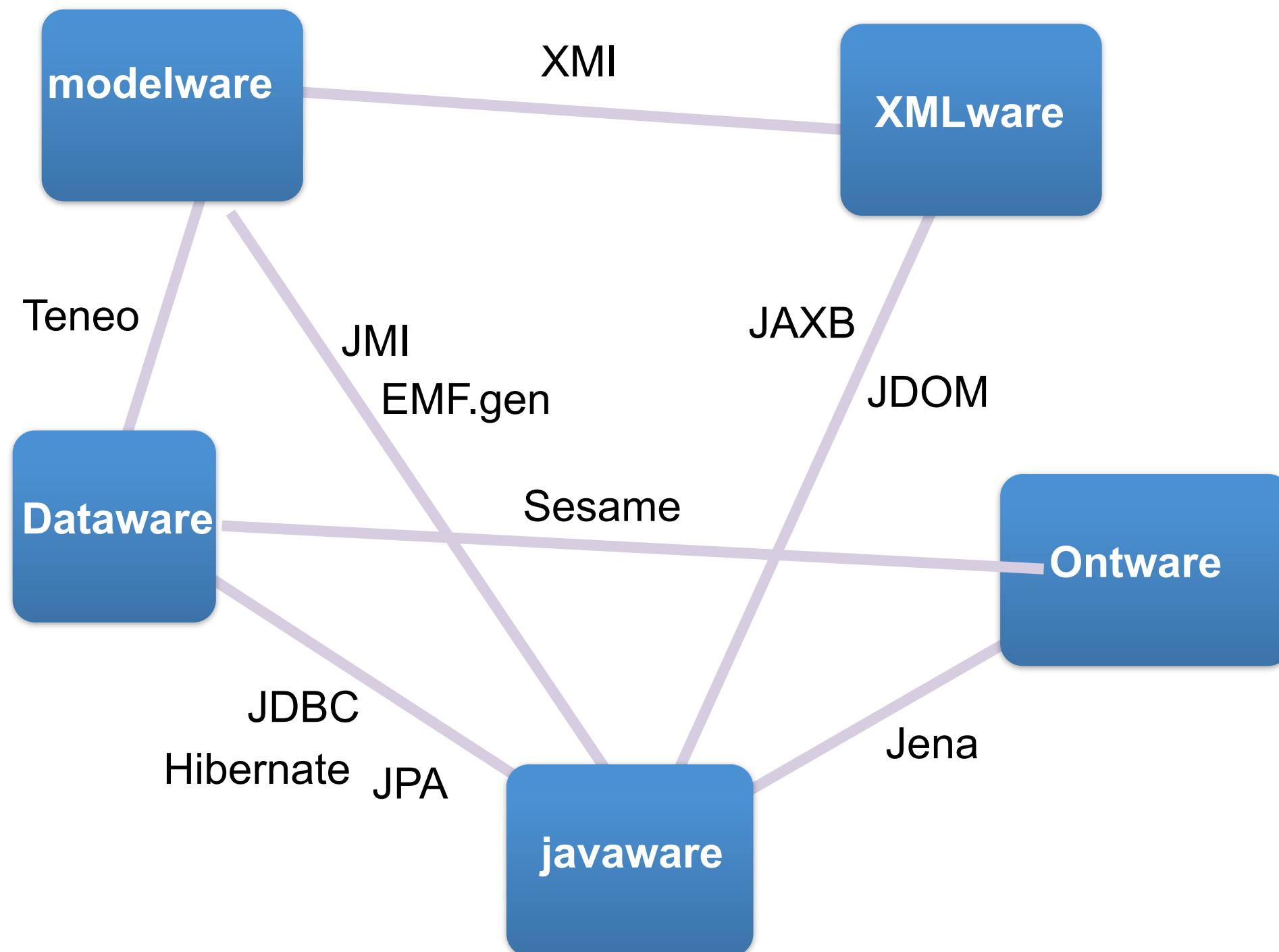
Our Approach

- **Analogy**
- **Examples**
- **Abstraction**

Working by analogy

	<i>Modelware</i>	<i>XMLware</i>	<i>Ontoware</i>	<i>Dataware</i>	<i>Grammarware</i>
<i>Meta language</i>	MOF	XSD	RDFS	SQL.DDL	EBNF
<i>Navigation</i>		XPath			
<i>Query</i>	OCL	XQuery	SPARQL	SQL	
<i>Transfo.</i>	QVT	XSLT			TXL ASF
<i>Toolkit</i>	ArgoUML Rose	XMLSpy VS-XML	Protégé Topbeard	MySQL Oracle	MetaEnv.
<i>Conferences</i>	MoDELS ECMDA	XML VLDB	ICSW ESWC	VLDB SIGMOD	CC POPL

From one space to another...



**We need a catalog of
concrete examples.**

101companies

Summary

<http://sourceforge.net/apps/mediawiki/developers/index.php?title=101companies> 

- *101companies* is a software corpus for company modeling and processing.
- Many different models and scenario implementations are exercised.
- The diversity feeds into a major megamodeling effort.

What's a company?

- A *company* is a nested structure of *departments* with *employees* as leafs.
- Employees are characterized by *name*, *salary*, and possibly other properties.
- Companies and departments have names, too.
- Each department has a *manager*.
- Employees may be associated with *mentees*.

Implementations may differ with regard to the level of detail.

A sample company

This company is named *meganalysis*.

For what it matters, *meganalysis* is into megamodeling (as opposed to selling ice cream).

We only capture some basic structural facets of *meganalysis* below.

```
company "meganalysis" {
  department "Research" {
    manager "Craig" {
      address "Redmond"
      salary 123456
    }
    employee "Erik" {
      address "Utrecht"
      salary 12345
    }
    employee "Ralf" {
      address "Koblenz"
      salary 1234
    }
  }
  department "Development" {
    manager "Ray" {
      address "Redmond"
      salary 234567
    }
    department "Dev1" {
      manager "Klaus" {
        address "Boston"
        salary 23456
      }
      department "Dev1.1" {
```

...

Company scenarios

- *total*: Total all salaries in a company.
- *cut*: Cut all salaries in half.
- *depth*: Determine depth of department nesting.
- *containment*: Check that tree topology holds for the instance.
- *precedence*: Check that salaries increase with rank in hierarchy.

Implementations do not need to cover all scenarios.

Implementations

All implementations are *labeled* for consistency of reference.

The implementations are listed in alphabetical order.

- *alpha*: a simple POJO object model for companies with methods for some of the scenarios.
- *antlr*: an ANTLR-based acceptor for a human-readable notation for companies.
- *antlr2*: a variation on *antlr* that actually constructs ASTs over some generic object model.
- *antlr3*: a variation on *antlr2* that constructs ASTs according to an object model for companies.
- *atl*: Ecore/ATL-based model transformations for some of the company scenarios.
- *atl2*: a variation on *atl* that uses a slightly different metamodel (with proper subtyping).
- *atl3*: a variation on *atl2* that uses KM3 instead of Ecore; this option is potentially obscure.
- *dom*: in-memory XML processing in Java with the DOM API.
- *emf*: EMF/Java-based model queries and transformations.
- *gwt*: C/S (Browser/Server) architecture for a WebApp based on GWT (Google Web Toolkit).
- *haskell*: model companies and implement company scenarios in Haskell 98 + SYB (using GHC).
- *hibernate*: maintain companies in RDBMS and access them through hibernate's O/R mapping.
- *hibernate2*: variation on *hibernate* to illustrate a different O/R mapping.
- *jaxb*: represent companies in XML and access them through JAXB's XML data binding.
- *jaxb2*: variation on *jaxb* to illustrate a different X/O mapping.
- *jdbc*: Relational database programming in Java with JDBC.
- *jdbc2*: A sophistication of *jdbc* to approach to O/R mapping in a homegrown manner.
- *jena*: in-memory RDF processing in Java with the Jena API (RDF part).
- *jena2*: a further use of *jena* which leverages Jena's query engine / ARQ implementation of SPARQL.
- *library*: a collection of third-party libraries that are leveraged by the implementations.
- *prolog*: model companies and implement company scenarios through logic programming in SWI-Prolog.
- *sax*: push-based XML processing in Java with the SAX API.
- *sql*: SQL DML-based implementation of some of the company scenarios.
- *swing*: a simple GUI for navigating companies and performing scenarios based on Swing/AWT.
- *xpath*: in-memory XML processing in Java with the XPath embedding into DOM.
- *xslt*: XSLT-based XML processing.
- *xquery*: XQuery-based XML processing.

Implementations

All implementations are *labeled* for consistency of reference.

The implementations are listed in alphabetical order.

- *alpha*: a simple POJO object model for companies with methods for some of the scenarios.
- *antlr*: an ANTLR-based acceptor for a human-readable notation for companies.
- *antlr2*: a variation on *antlr* that actually constructs ASTs over some generic object model.
- *antlr3*: a variation on *antlr2* that constructs ASTs according to an object model for companies.
- *atl*: Ecore/ATL-based model transformations for some of the company scenarios.
- *atl2*: a variation on *atl* that uses a slightly different metamodel (with proper subtyping).
- *atl3*: a variation on *atl[2]* that uses KM3 instead of Ecore; this option is potentially obscure.
- *dom*: in-memory XML processing in Java with the DOM API.
- *emf*: EMF/Java-based model queries and transformations.
- *gwt*: C/S (Browser/Server) architecture for a WebApp based on GWT (Google Web Toolkit).

Code snippets

The *total* scenario in XQuery:

```
<result>
    {sum(//salary)}
</result>
```

The *cut* scenario in SQL DML:

```
UPDATE employee SET salary = salary / 2;
```

The *total* scenario with Jena's RDF API:

```
public static double total(CompanyModel c) {
    double total = 0;
    StmtIterator i =
        c.getModel().listStatements(
            new SimpleSelector(
                null, c.SALARY, (RDFNode) null));
    while (i.hasNext()) {
        Statement s = i.next();
        total += s.getDouble();
    }
    return total;
}
```

A few variation points

X vs. O vs. R vs. λ etc.

Static typing vs. dynamic typing

Textual vs. abstract vs. visual syntax

GPPL vs. DSL vs. embedding vs. API

Instance- vs. operation-based mapping

Type checking vs. inference vs. reasoning

Code first vs. schema first vs. mapping only

In-memory processing vs. push vs. pull parsing

Pure vs. impure transformations (or in between)

Code vs. generative vs. model-driven vs. mapping

DEMO

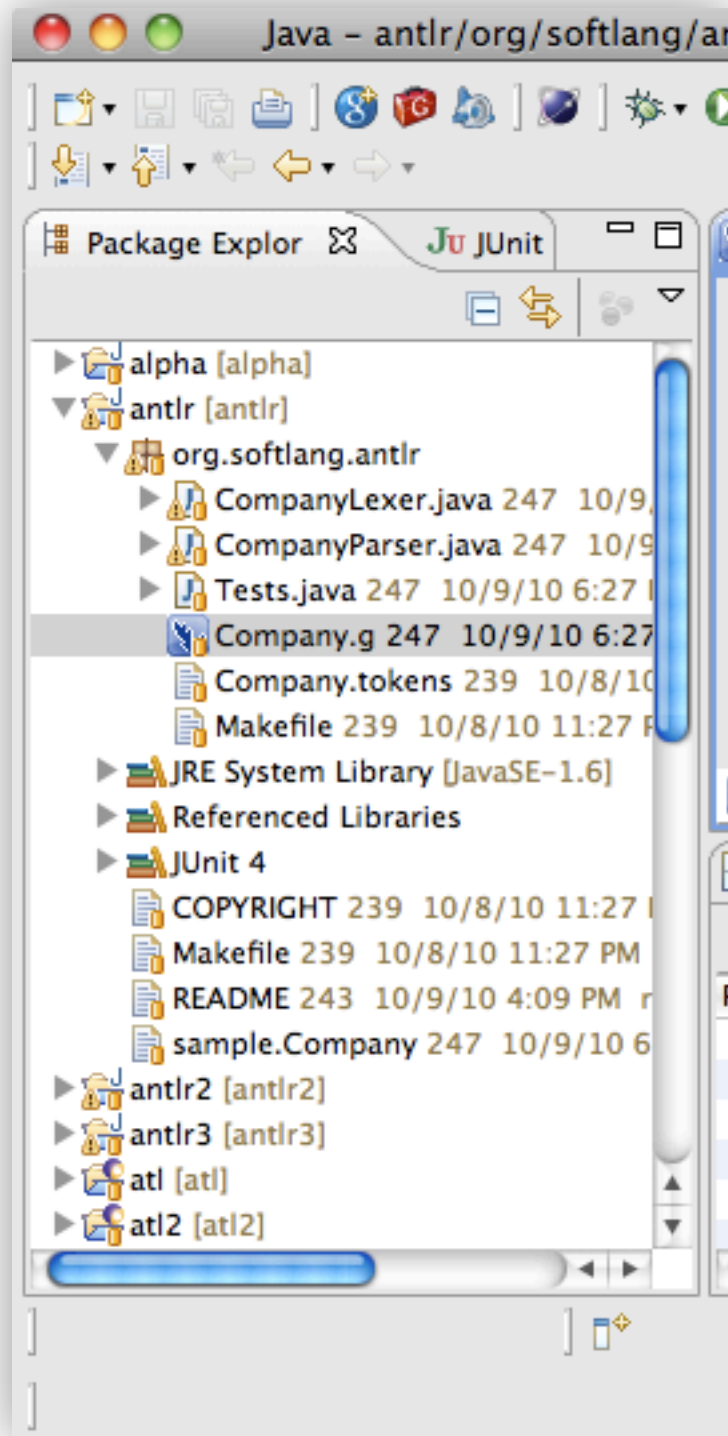
Your contribution is
appreciated.

<http://sourceforge.net/apps/mediawiki/developers/index.php?title=101companies>

We need modeling.

We need **megamodeling**.

Empirical megamodeling



(ANTLR)

Company.g

company : ...

accept (or parse)

sample.Company

company "meganalysis" { ...

Meganalysis
company

A megamodel

Specific megamodeling

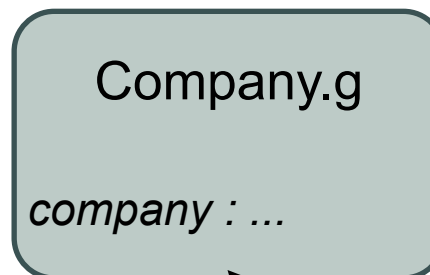
$w \in L(G_{\text{Company}})$

...

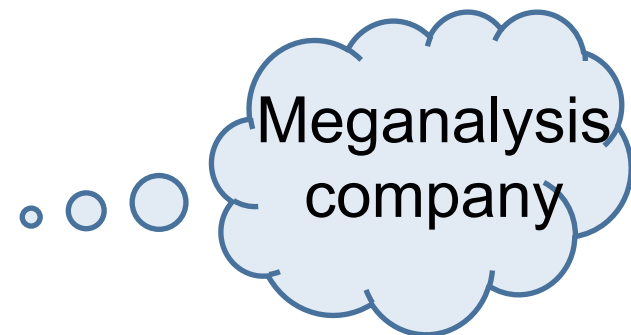
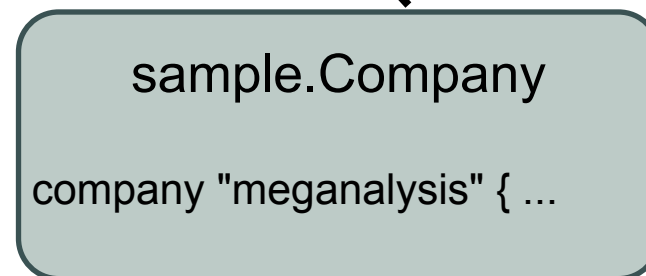
A megamodel



(ANTLR)



accept (or parse)



A megamodel

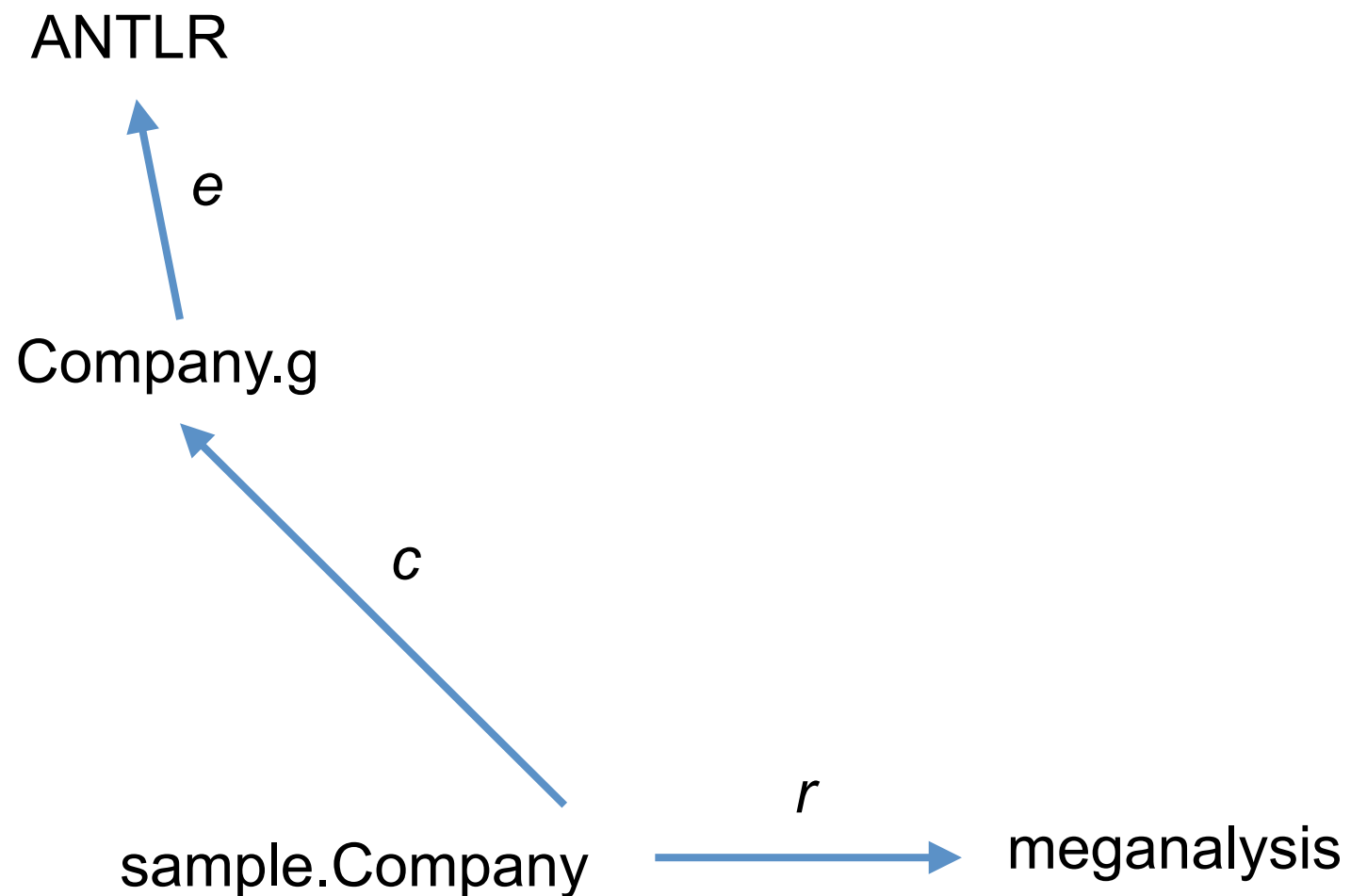
**We need to be
more precise and more abstract.**

Precise megamodeling

r ... “represents”

c ... “conforms to”

e ... “element of”



The megamodel
metamodel

a megamodel

Relationships (all binary)

- [e “element of” (in set-theoretic sense)
- [s “subset of” (in set-theoretic sense)
- [c “conforms to” (in the sense of generative semantics)
- [r “represents” (in information-theoretic+structural sense)
- [i “input for” (... a transformation, essentially a function)
- [o “output for” (... a transformation, essentially a function)

Entities

- [Languages

- in a set-theoretic sense

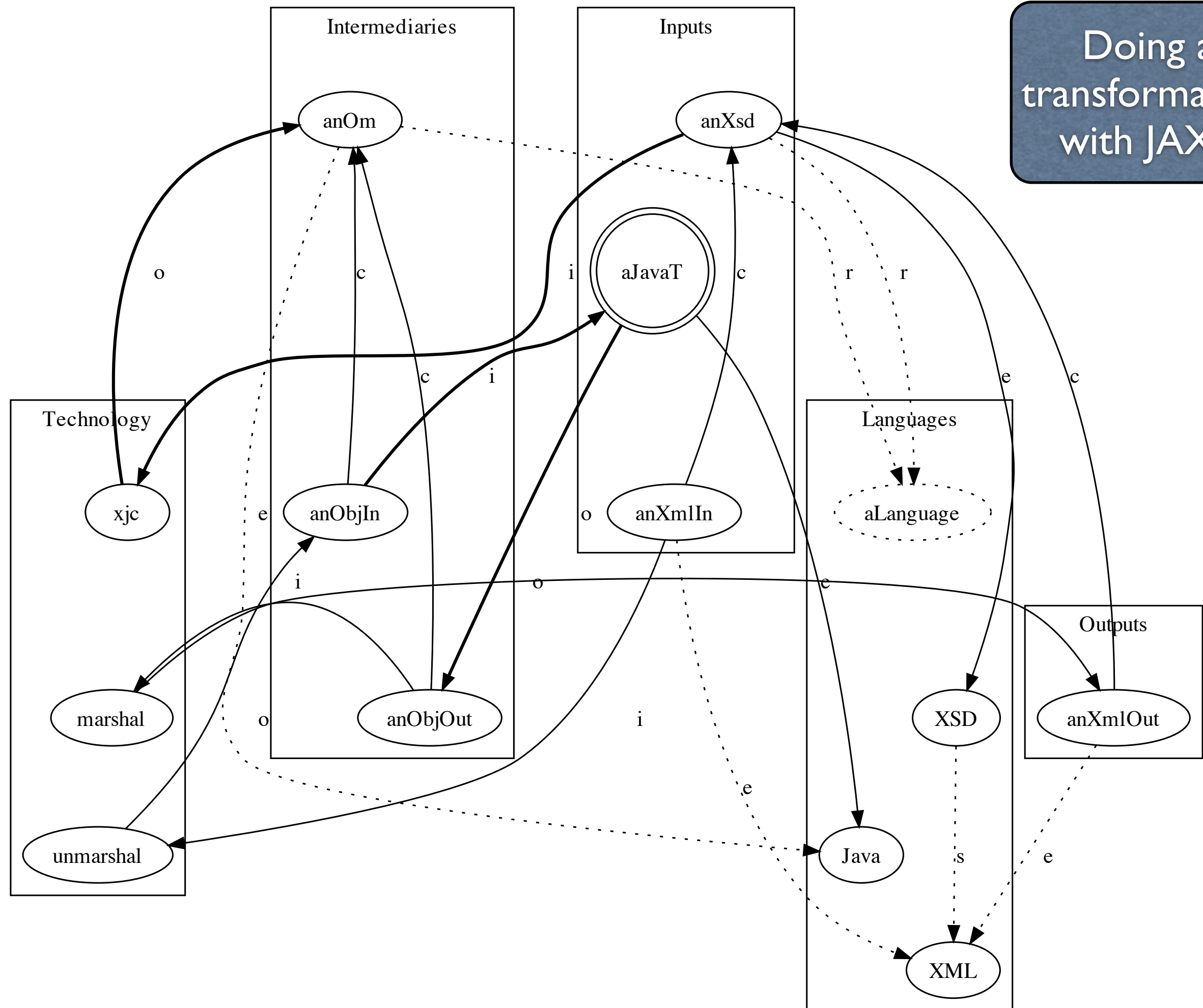
- in an intentional sense

- [Elements of languages

- [Functions (for transformations, for example)

- [Interpretable entities (as functions)

Doing a transformation with JAXB



<http://sourceforge.net/apps/mediawiki/developers/index.php?title=101companies>

Your contribution is
appreciated.

Thanks!
Questions or comments?

I am currently not @notquiteabba.
Rather I am @notreallyabba (and occasionally @ralfileaks).