See the 101companies wiki
http://101companies.uni-koblenz.de

Understanding Programming Technologies by Analogy, Examples, and Abstraction

Software Languages Team
University of Koblenz-Landau

Follow @meganalysis

SATTOSE community = experts in programming technology – think of queries & transformations.

Why such a presentation on programming technologies at SATTOSE?

We have a problem.

XPATH TXL Sesame JPA
Jena Rose JDBC EMF.gen XText
jDOM JAXB Jersey RDF(S) UTF8 Jena ODM
XSD MOF JeanBeans Teneo
UML BNF VLDB
Stratego
xerces GWT SLE2010 Json
sax
OCL Ralf
RDFS OWL saxon OWL
Ecore
Rest
MySQL XMI ORACLE RDF Jean OMG
JMI EMF JMF
XSD ArgoUML
xalan ODBC SparQL XMLSpy Yacc RDFa
LALR Prolog
XSLT JAXP Protegé
SBVR DOM Java ATOM ER
CFG
SQL DDL Dragan SQL
Antlr QVT
TENEO XLST
sed Awk DTD Saxon TCS
grep ASCII XSD
Hibernate XQuery

Thanks to Jean-Marie Favre for this excellent slide*!*

Today’s Issues

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

Sesame XPATH TXL JPA MF.gen XText jDOM JAXB Jersey RDF(S) UTF8 Jena ODM MOF XSD JeanBeans Teneo UML VLDB BNF xerces GWT SLE2010 Json Ralf RDFS saxon OWL OWL Ecore MySQL XMI ORACLE RDF Jean OMG JMI EMF JMF XSD ArgoUML RDFa SparQL XMLSpy yacc LALR Prolog XSLT Protegé SBVR DOM Java ATOM ER CFG SQL DDL Dragan SQL Antlr QVT TENEO XLST Awk DTD sed Saxon TCS grep ASCII XSD Hibernate XQuery

Why would you study computer science,
if your ultimate destiny is
to **get lost in space and technology**?

Today's Issues

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

Our Approach

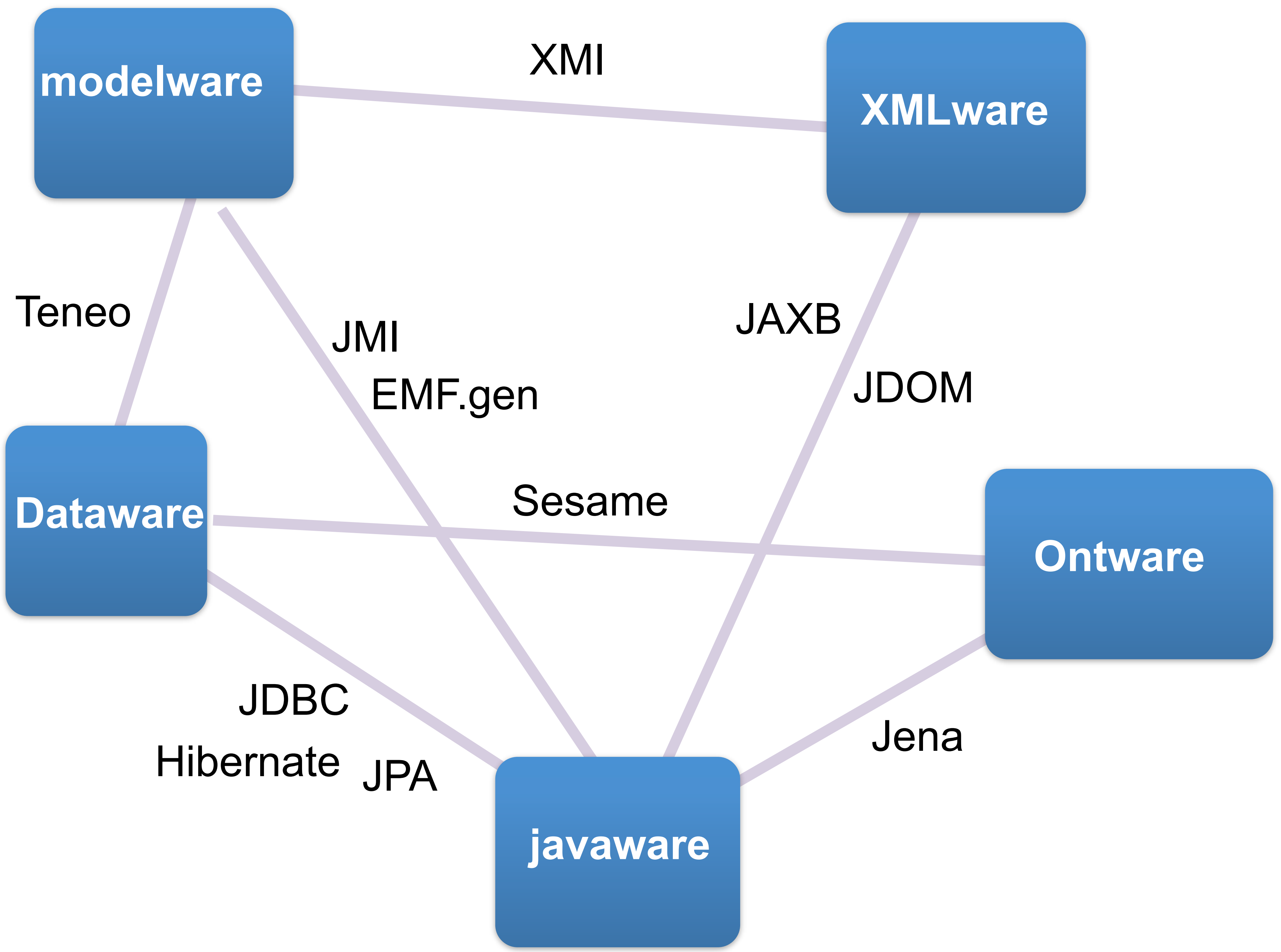
- Analogy
- Examples
- Abstraction

Sesame XPATH TXL JPA EMF.gen XText JDBC jDOM JAXB Jersey RDF(S) UTF8 Jena ODM MOF XSD UML JeanBeans Teneo VLDB BNF Stratego xerces GWT SLE2010 Json Ralf OCL RDFS saxon OWL OWL Ecore Rest MySQL XMI ORACLE RDF Jean OMG JMI JMF XSD ArgoUML xalan ODBC SparQL XMLSpy Yacc RDFa XSLT JAXP Protege ATOM ER CFG SBVR DOM Java SQL DDL Dragan SQL QVT Antlr TENEO XLST Awk DTD TCS sed Saxon grep ASCII XSD Hibernate XQuery

Analogy, examples, abstraction

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

Analogy in space travel



Analogy, **examples**, abstraction

Total salaries

Cut salaries

Store companies

Navigate companies

```
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
        }
```

•••

Functionality on companies

Total salaries in XQuery

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

Cut salaries in SQL DML

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

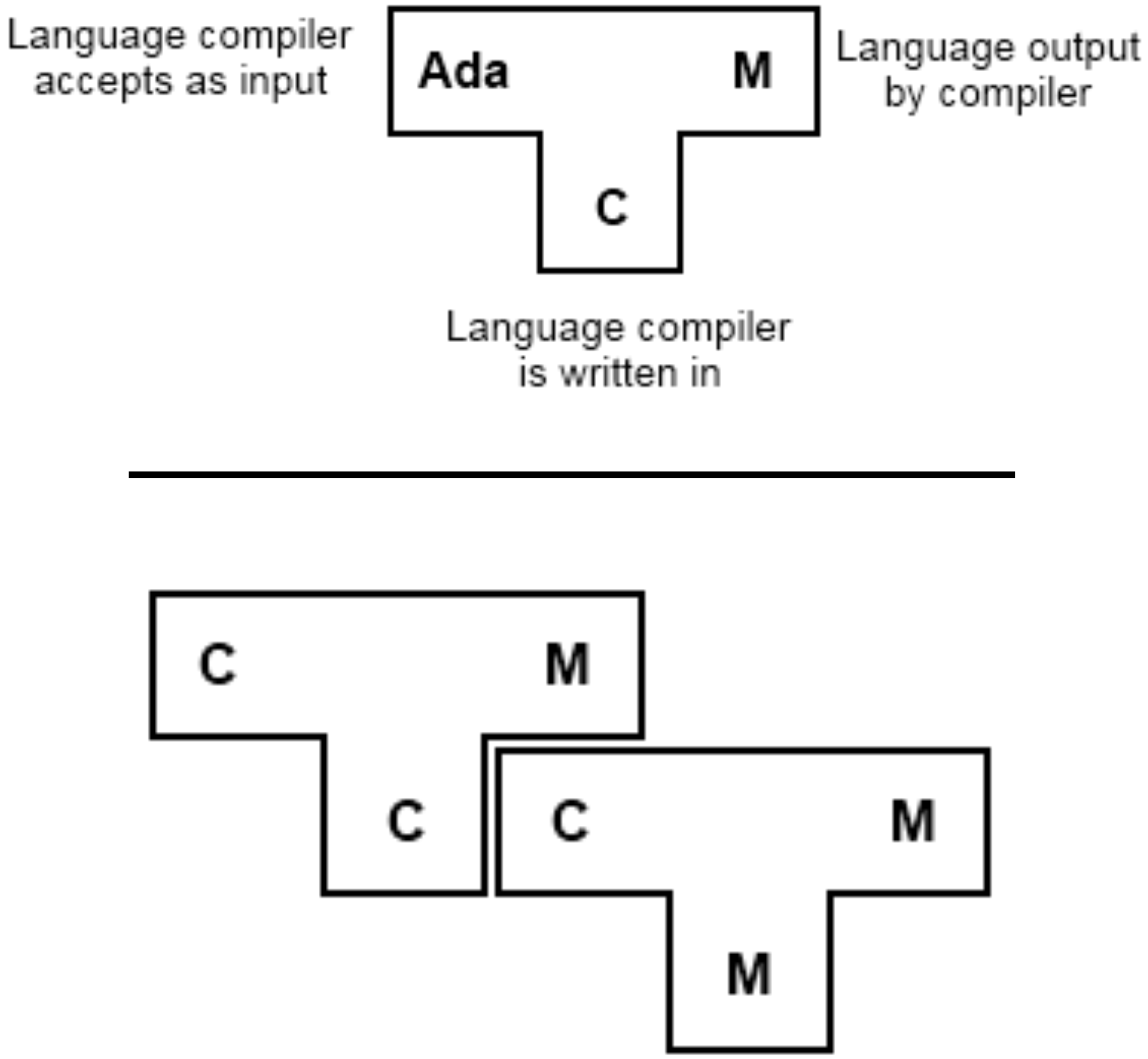
Variation points for examples

- 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

Analogy, examples, **abstraction**

- What's the **essence** of technology *xyz*?
- What's the ontology of programming technologies?

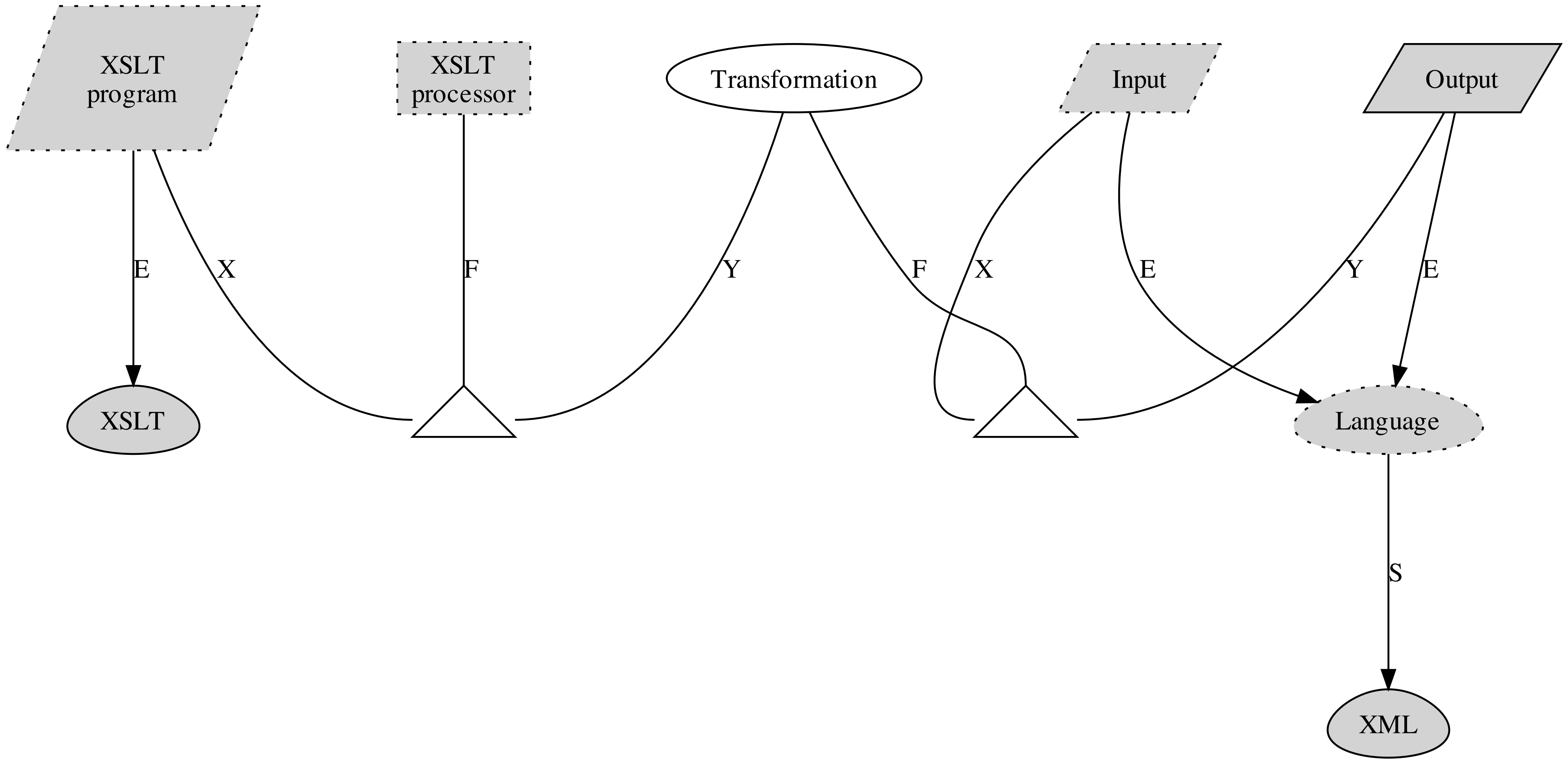
Remember *Tombstone diagrams?*


Language compiler accepts as input
Ada
M
Language output by compiler
C
Language compiler is written in
C
M
C
C
M
M

"Used for describing complicated processes for bootstrapping, porting, and self-compiling of compilers, interpreters, and macro-processors."

http://en.wikipedia.org/wiki/T-diagram

Abstraction with megamodels



An XSLT transformation

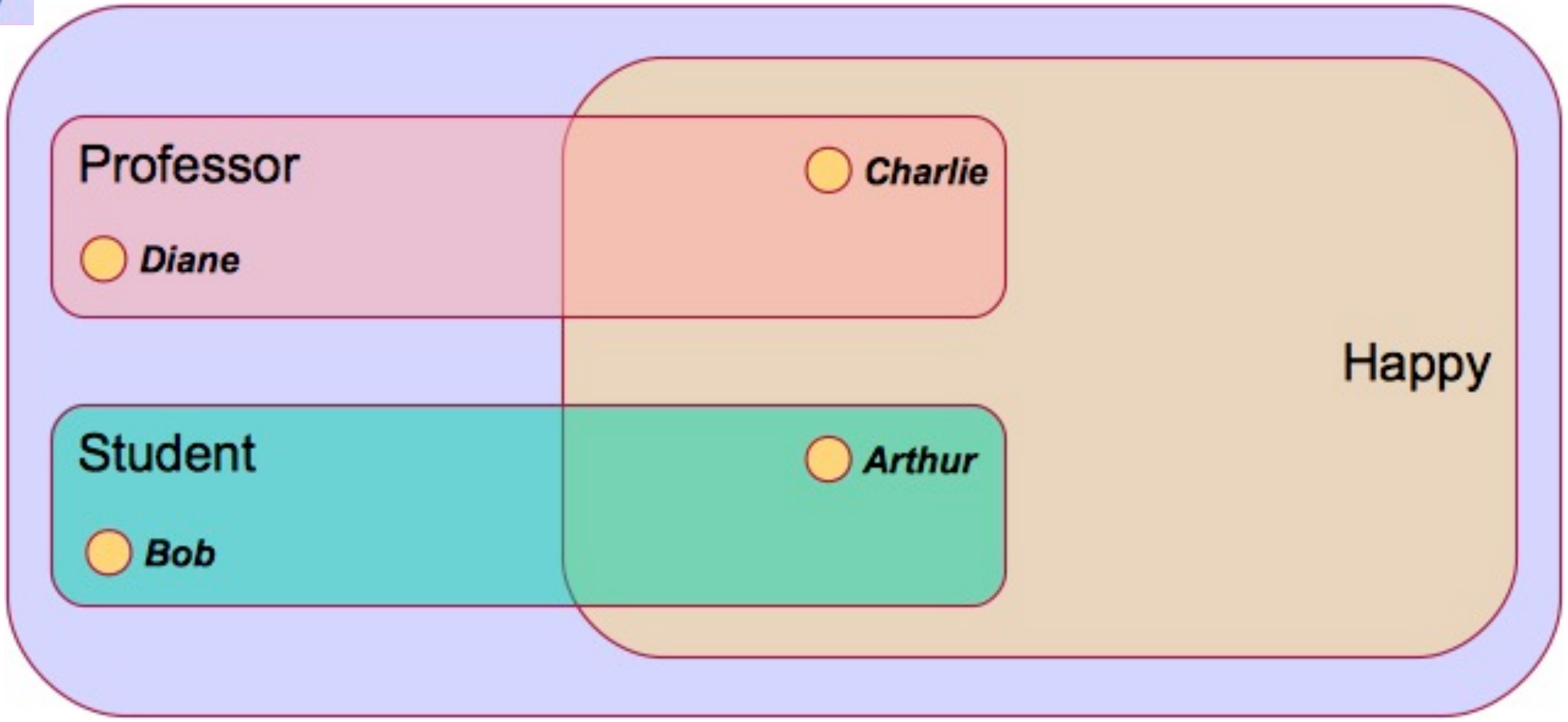
What's the ontology we need?

```
Class(a:Person partial)
Class(a:Academic partial a:Person)
Class(a:Happy partial a:Person)
Class(a:Lecturer partial  a:Academic)
Class(a:Professor partial  a:Academic)
Class(a:Student partial  a:Person)

ObjectProperty(a:hasFriend)
ObjectProperty(a:isFriendOf
 inverseOf(a:hasFriend))

DisjointClasses(a:Student a:Academic)
```

```
Individual(a:arthur type(a:Student) type(a:Happy))
Individual(a:bob type(a:Student) type(complementOf(a:Happy)))
Individual(a:charlie type(a:Professor) type(a:Happy))
Individual(a:diane type(a:Professor) type(complementOf(a:Happy)))
```



http://owl.man.ac.uk/tutorial/

Abstraction with an ontology

Capability	a principle capability in programming to address non-functional requirements
– **Access control**	the capability to control access to data and resources within programs
– **Distribution**	the capability to distribute programs (objects) over computers in a network
– **Indexing**	the capability for access to keyed and ordered records
– **Interaction**	the capability of interactions between the user and the system
– **Logging**	the capability of logging certain events along program execution
– **Mapping**	the capability of bridging technical spaces
– – *O/R mapping*	the capability of bridging the technical spaces objectware and tupleware
– – *O/X mapping*	the capability of bridging the technical spaces objectware and XMLware
– – *R/X mapping*	the capability of bridging the technical spaces tupleware and XMLware
– **Parallelism**	the capability to execute a program in parallel
– **Parsing**	the capability of analyzing software artifacts in terms of their concrete syntax
– **Persistence**	the capability to maintain program data beyond the runtime of the program
– **Serialization**	the capability of converting program data into a format for storage or transmission
– **Streaming**	the capability for processing data in a stream as opposed to in-memory

101companies system

“Specification”

The 101companies system (in the sequence: just the "system") is a conceived system in the application domain of **human resources**. The present specification is meant to be informal and liberal; it should facilitate different implementations of the system with different programming technologies and techniques, and with different feature sets. The system is concerned with **companies, departments, managers, and employees, and it supports functionality for totaling salaries, cutting salaries**, computing other data, and checking data in some ways. The system may also be subject to additional capabilities---similar to non-functional requirements, e.g.: serialization, persistence, or logging. The following feature model breaks down all required or optional features of the system.

Feature model

101feature	features of the 101companies system
– **101basics**	basic features of the 101companies system
– – *101feature:Company*	a data model for companies
– – *101feature:Cut*	cut all salaries in half
– – *101feature:Total*	total all salaries in a company
– **101capabilities**	capability-related features of the 101companies system
– – *101feature:Interaction*	interaction with companies though a user interface
– – *101feature:Logging*	logging for mutations of companies
– – *101feature:Persistence*	persistence for companies
– – *101feature:Serialization*	serialization for companies
– **101extras**	extra features of the 101companies system
– – *101feature:Depth*	determine depth of department nesting
– – *101feature:Mentoring*	associate employees with mentors
– – *101feature:Precedence*	check that salaries increase with rank in hierarchy

101feature:Company

What's a company?

- A *company* is structured as follows:
  - There is a *name*.
  - There is any number of (possibly nested) *departments*.
- Each department is structured as follows.
  - There is a *name*.
  - There is any number of *employees*.
  - There is a *manager* as a special employee.
  - There is any number of (possibly nested) *sub-departments*.
- Employees are characterized by *name*, *salary*, and possibly other properties.
- The idea is that each employee can serve only in one position in the company.

Functionality on companies

Total salaries in XQuery

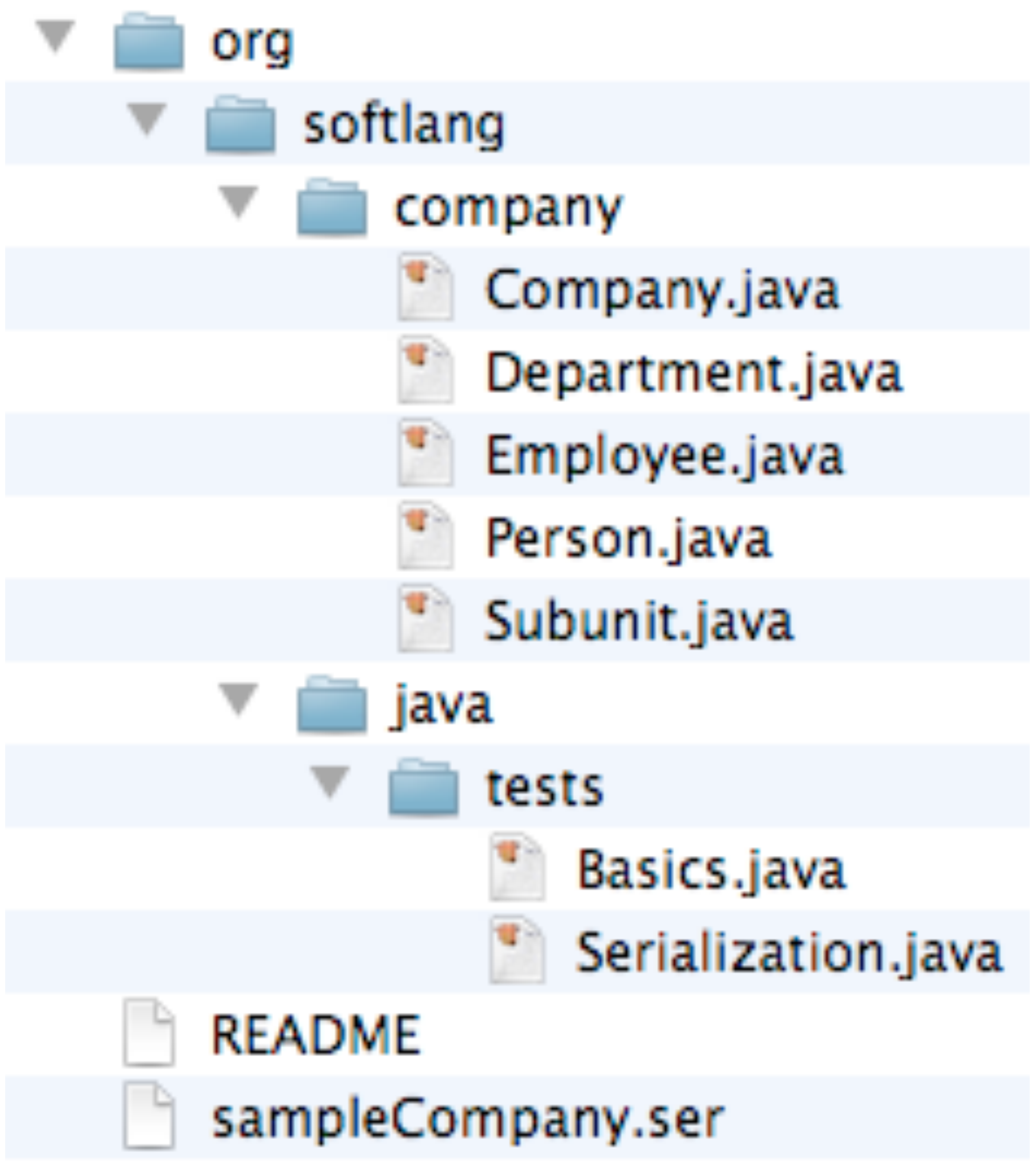
```
<result>
  {sum(//salary)}
</result>
```

Cut salaries in SQL DML

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

Implementation *java*

Implementation *java*



```
public class Company implements Serializable {

    private static final long serialVersionUID = ...;
    private String name;
    private List<Department> depts;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public List<Department> getDepts() { return depts; }
}
```

```
public class Department implements Serializable {
    private static final long serialVersionUID = ...;
    private String name;
    private Employee manager;
    private List<Department> subdepts;
    private List<Employee> employees;
    public Department() {
        subdepts = new LinkedList<Department>();
        employees = new LinkedList<Employee>();
    }
    public String getName() { return name; }
    public void setName(String n) { name = n; }
    public Employee getManager() { return manager; }
    public void setManager(Employee m) { manager = m; }
    public List<Department> getSubdepts() { return subdepts; }
    public List<Employee> getEmployees() { return employees; }
}
```

```
public class Employee implements Serializable {
    private static final long serialVersionUID = ...;
    private String name;
    private String address;
    private double salary;
    public String getName() { return name; }
    public void setName(String n) { name = n; }
    public String getAddress() { return address; }
    public void setAddress(String address) { address = a; }
    public double getSalary() { return salary; }
    public void setSalary(double salary) { salary = s; }
}
```

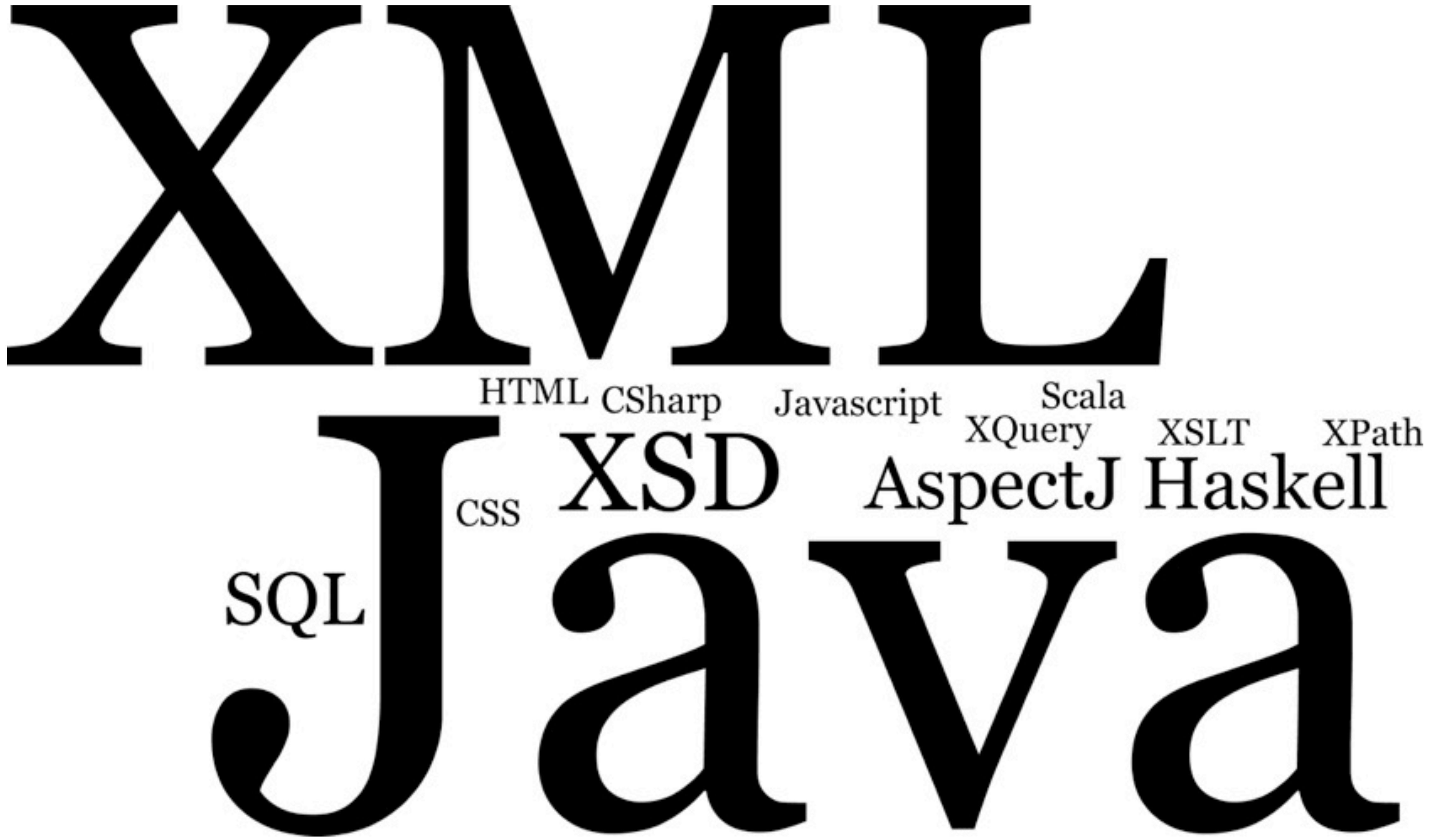
Keywords “java”

- POJO
- Containers
- Composite pattern
- Subtyping
- Virtual methods
- Object serialization
- Marker interface pattern

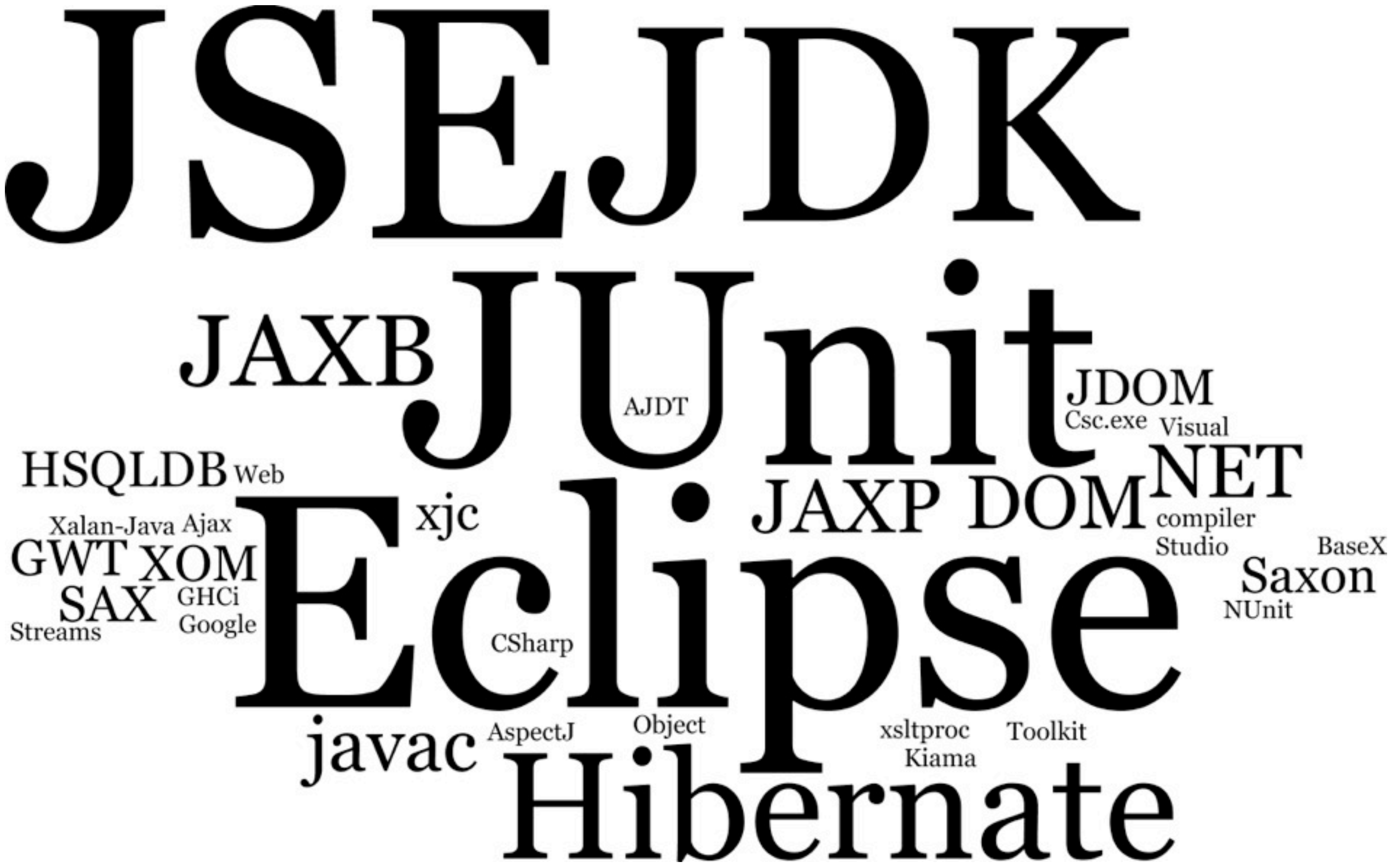
See the 101companies wiki
http://101companies.uni-koblenz.de

Follow @meganalysis

Languages cited by implementations



Technologies cited by implementations



Over to
Andrei Varanovich
and
Thomas Schmorleiz