XMLware

14 June 2012
CWI, NL

Specware

Modelware

Javaware

Understanding Technological Spaces

Objectware

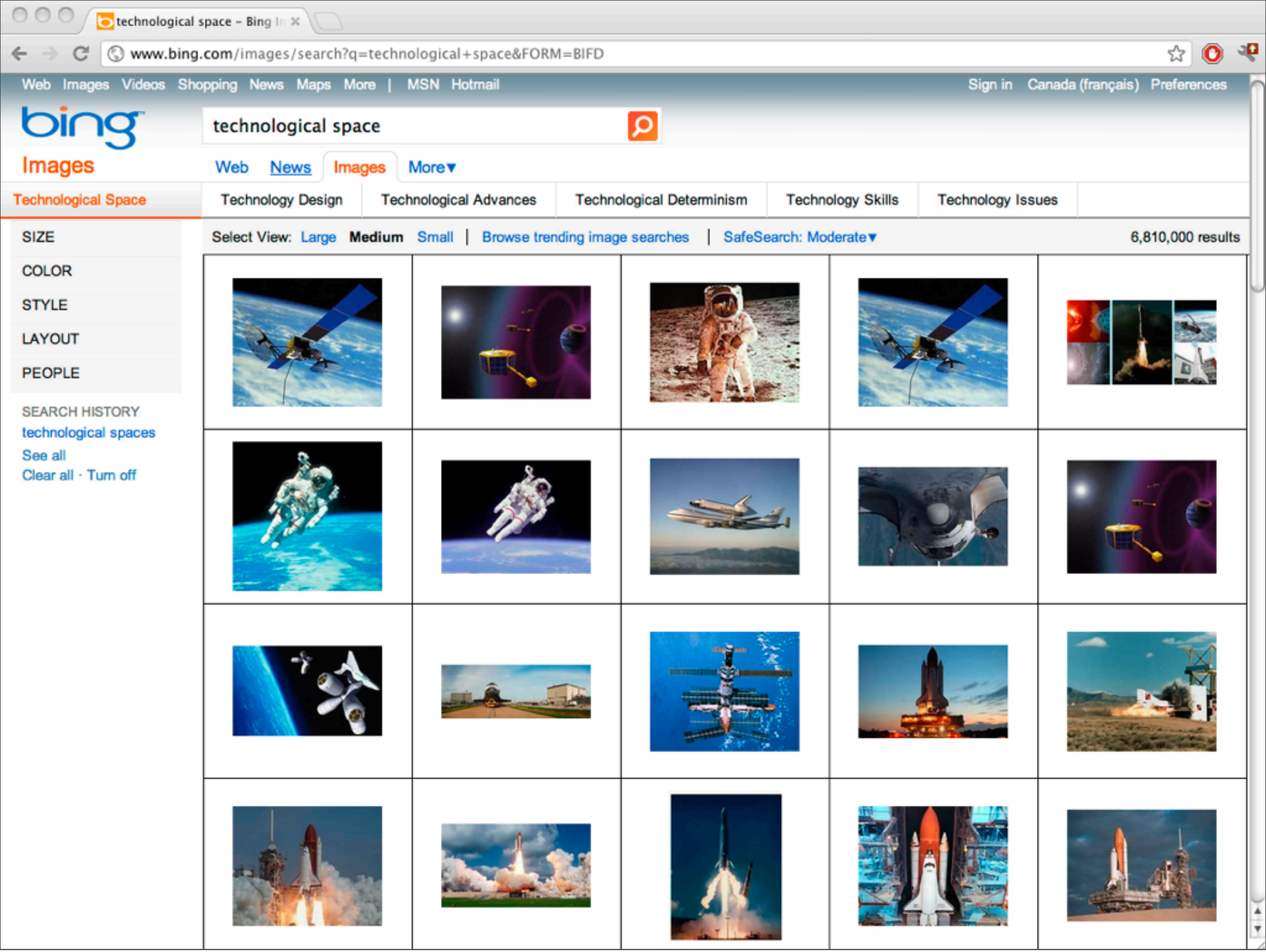
CSVware

Ralf Lämmel
Software Languages Team
University of Koblenz-Landau

Grammarware

Lambdaware

Binging/Googling “Technological spaces”



technological space - Bing Images

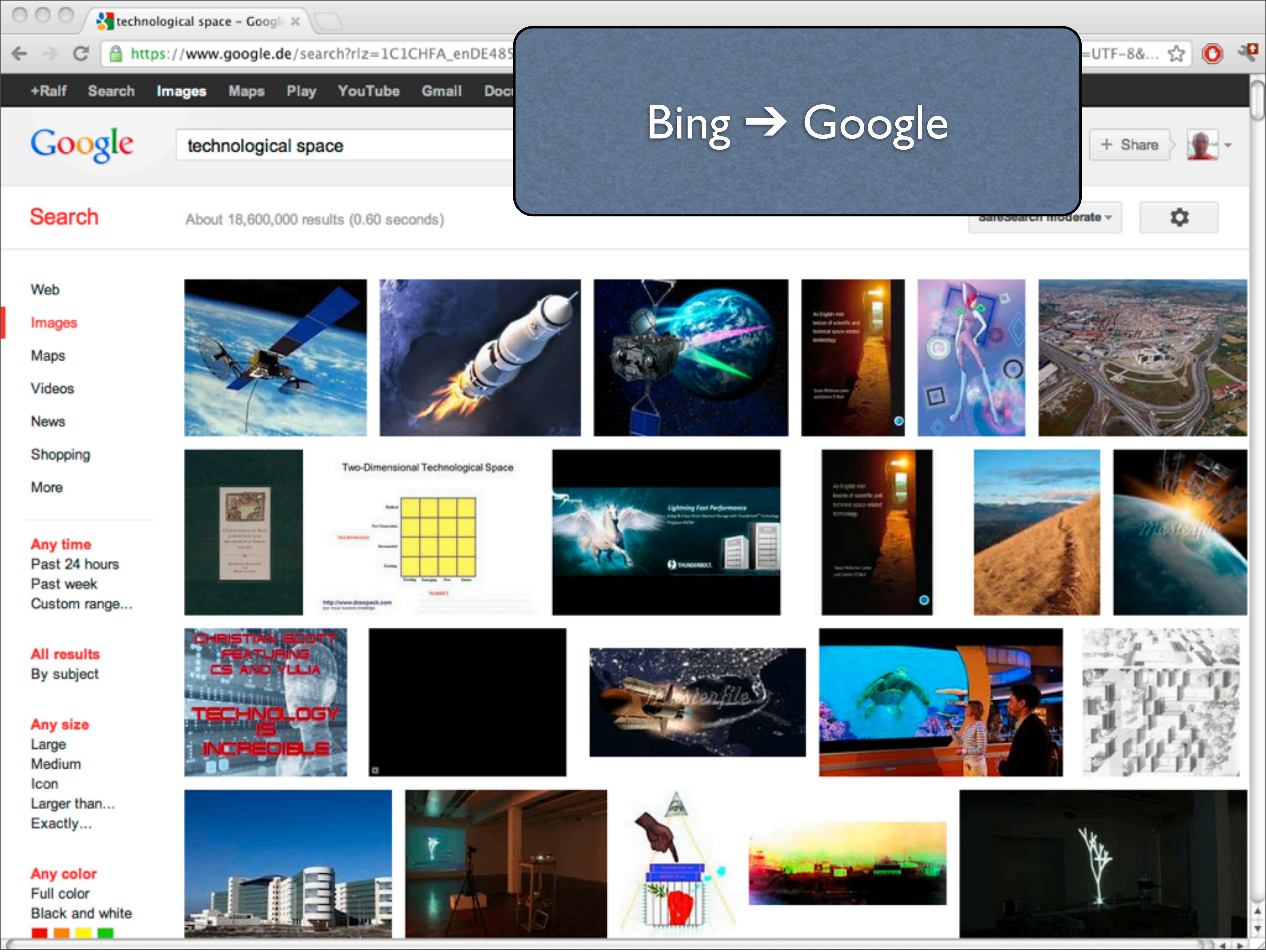
www.bing.com/images/search?q=technological+space&FORM=BIFD

technological space

Select View: Large **Medium** Small | Browse trending image searches | SafeSearch: Moderate▾

6,810,000 results

Bing → Google

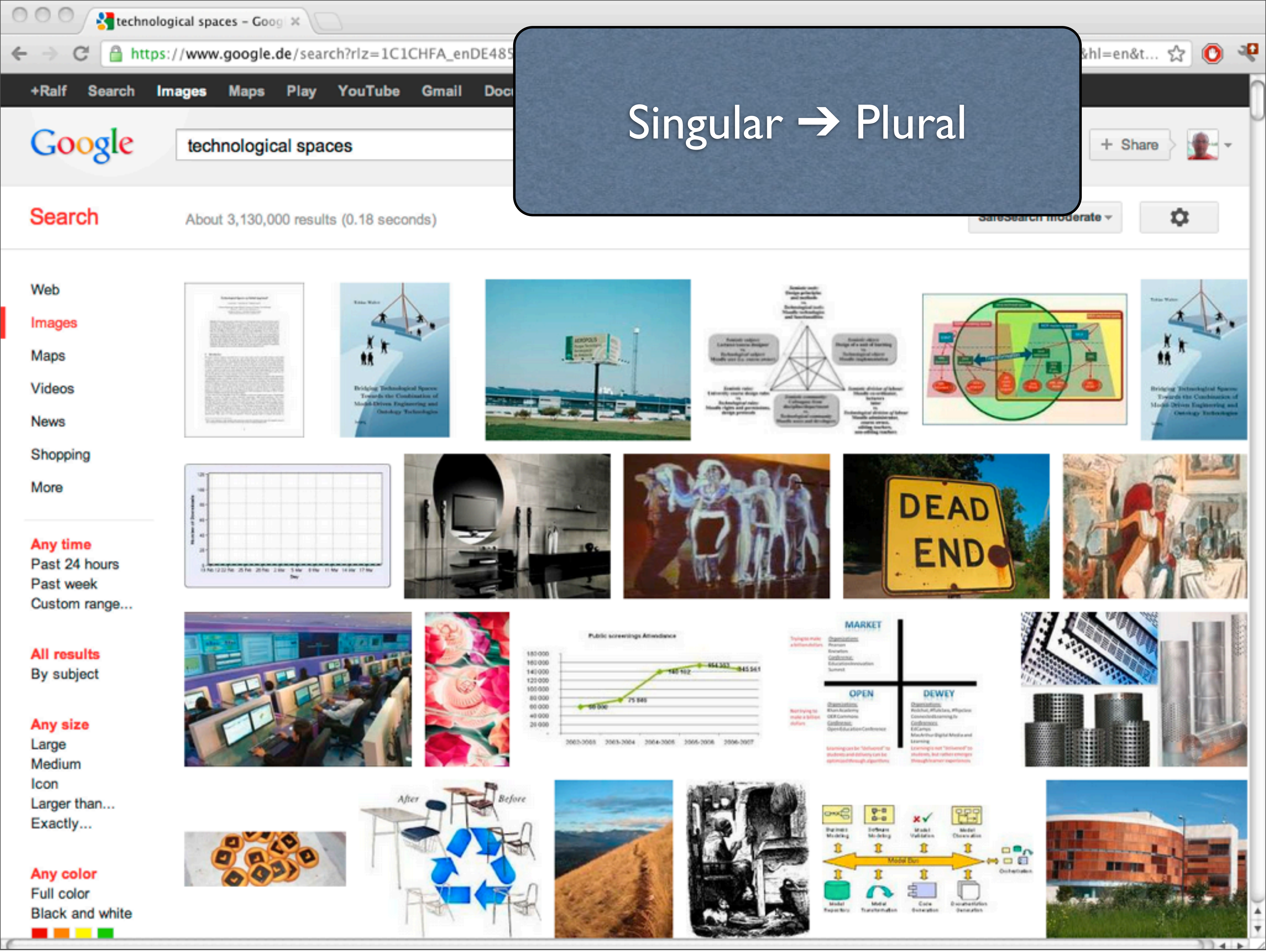


https://www.google.de/search?rlz=1C1CHFA_enDE485 …=UTF-8&…

technological space

About 18,600,000 results (0.60 seconds)

Singular → Plural

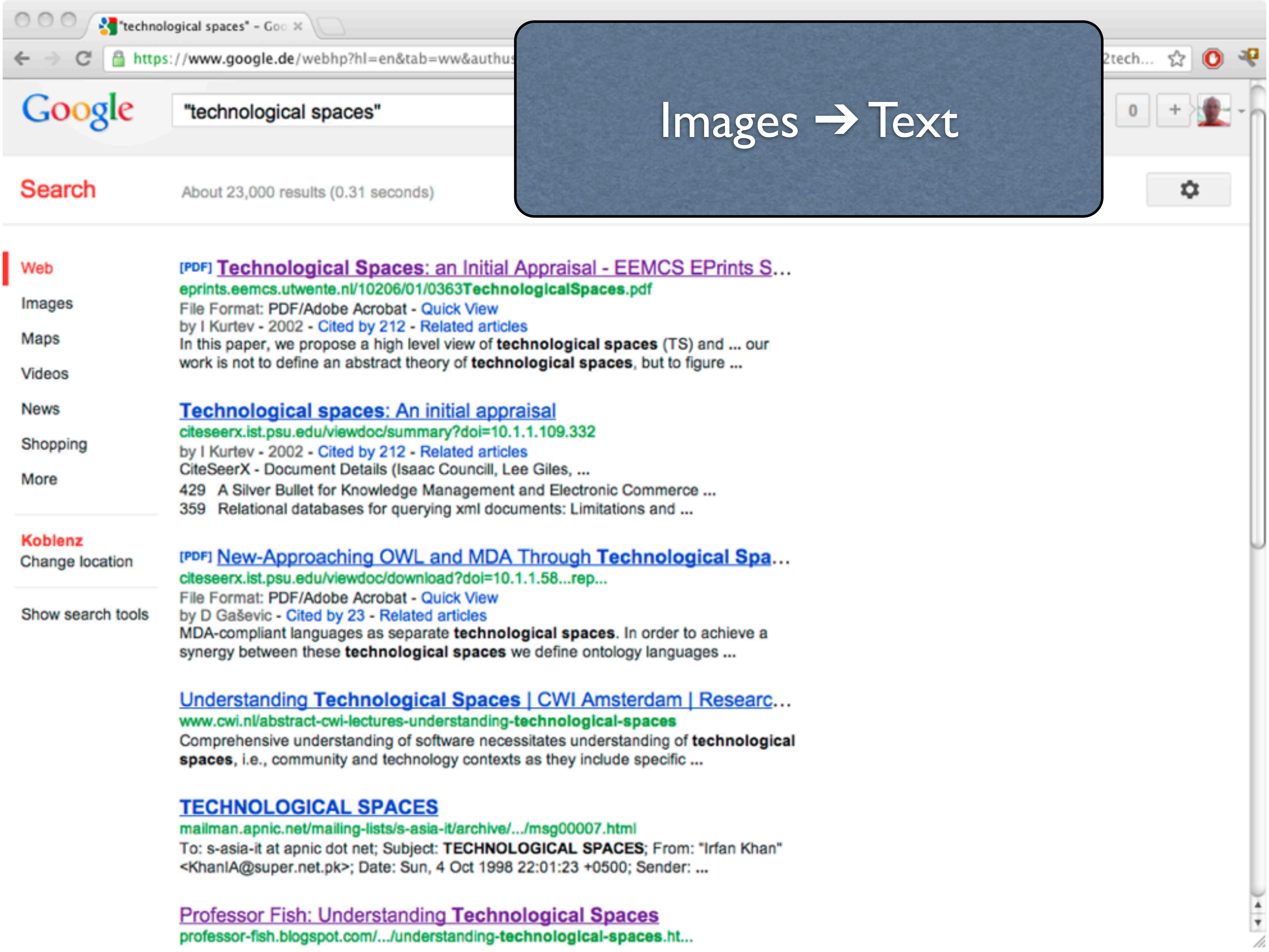


https://www.google.de/search?rlz=1C1CHFA_enDE485…&hl=en&t…

technological spaces

About 3,130,000 results (0.18 seconds)

Images → Text


"technological spaces" - Goo ×
https://www.google.de/webhp?hl=en&tab=ww&authus
2tech...
Google
"technological spaces"
0
+

Search

About 23,000 results (0.31 seconds)

[PDF] Technological Spaces: an Initial Appraisal - EEMCS EPrints S...
eprints.eemcs.utwente.nl/10206/01/0363TechnologicalSpaces.pdf
File Format: PDF/Adobe Acrobat - Quick View
by I Kurtev - 2002 - Cited by 212 - Related articles
In this paper, we propose a high level view of **technological spaces** (TS) and ... our work is not to define an abstract theory of **technological spaces**, but to figure ...

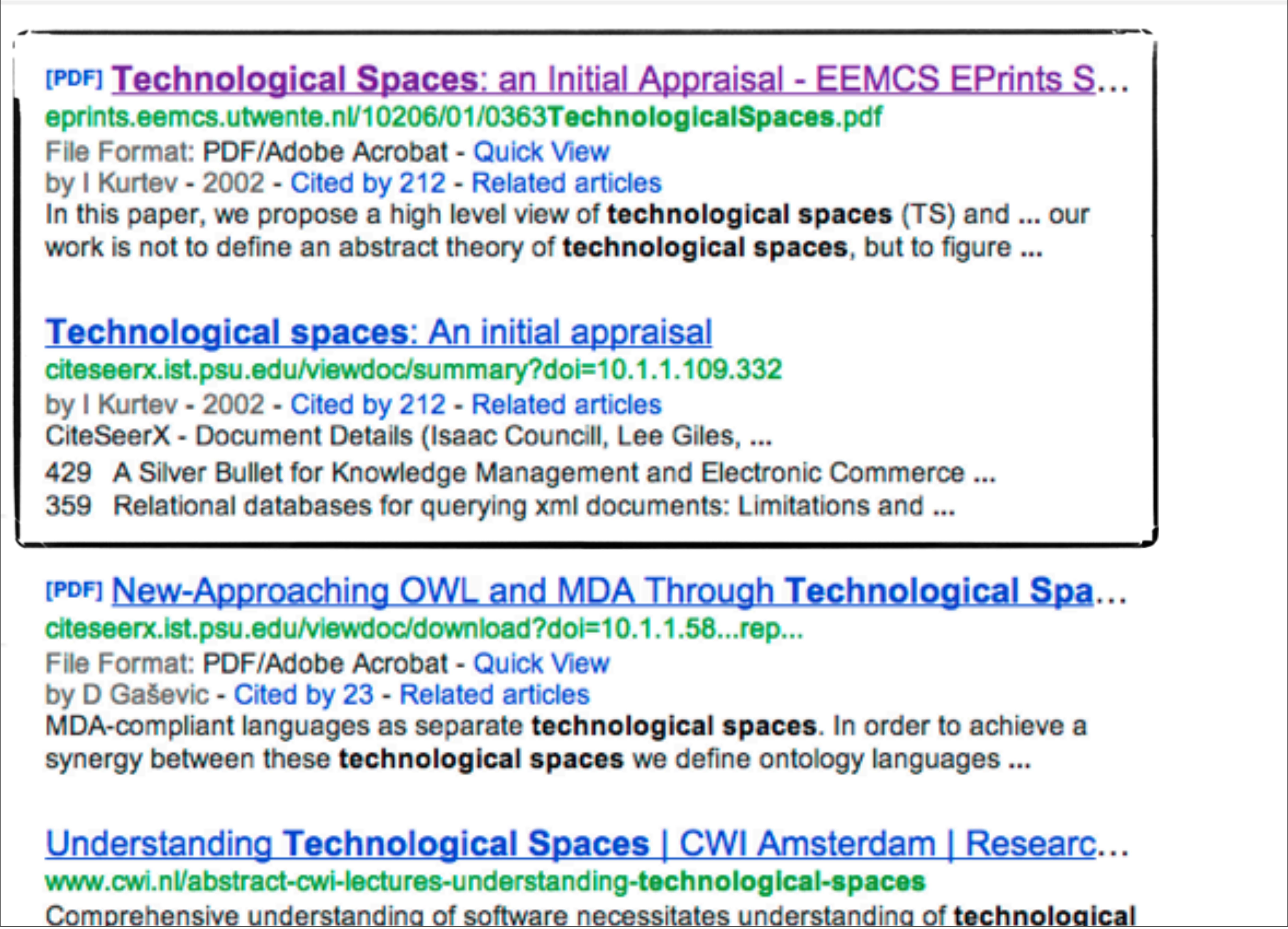
Technological spaces: An initial appraisal
citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.332
by I Kurtev - 2002 - Cited by 212 - Related articles
CiteSeerX - Document Details (Isaac Councill, Lee Giles, ...
429 A Silver Bullet for Knowledge Management and Electronic Commerce ...
359 Relational databases for querying xml documents: Limitations and ...

[PDF] New-Approaching OWL and MDA Through **Technological Spa**...
citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58...rep...
File Format: PDF/Adobe Acrobat - Quick View
by D Gaševic - Cited by 23 - Related articles
MDA-compliant languages as separate **technological spaces**. In order to achieve a synergy between these **technological spaces** we define ontology languages ...

Understanding **Technological Spaces** | CWI Amsterdam | Researc...
www.cwi.nl/abstract-cwi-lectures-understanding-**technological-spaces**
Comprehensive understanding of software necessitates understanding of **technological spaces**, i.e., community and technology contexts as they include specific ...

TECHNOLOGICAL SPACES
mailman.apnic.net/mailing-lists/s-asia-it/archive/.../msg00007.html
To: s-asia-it at apnic dot net; Subject: **TECHNOLOGICAL SPACES**; From: "Irfan Khan" <KhanIA@super.net.pk>; Date: Sun, 4 Oct 1998 22:01:23 +0500; Sender: ...

Professor Fish: Understanding **Technological Spaces**
professor-fish.blogspot.com/.../understanding-**technological-spaces**.ht...

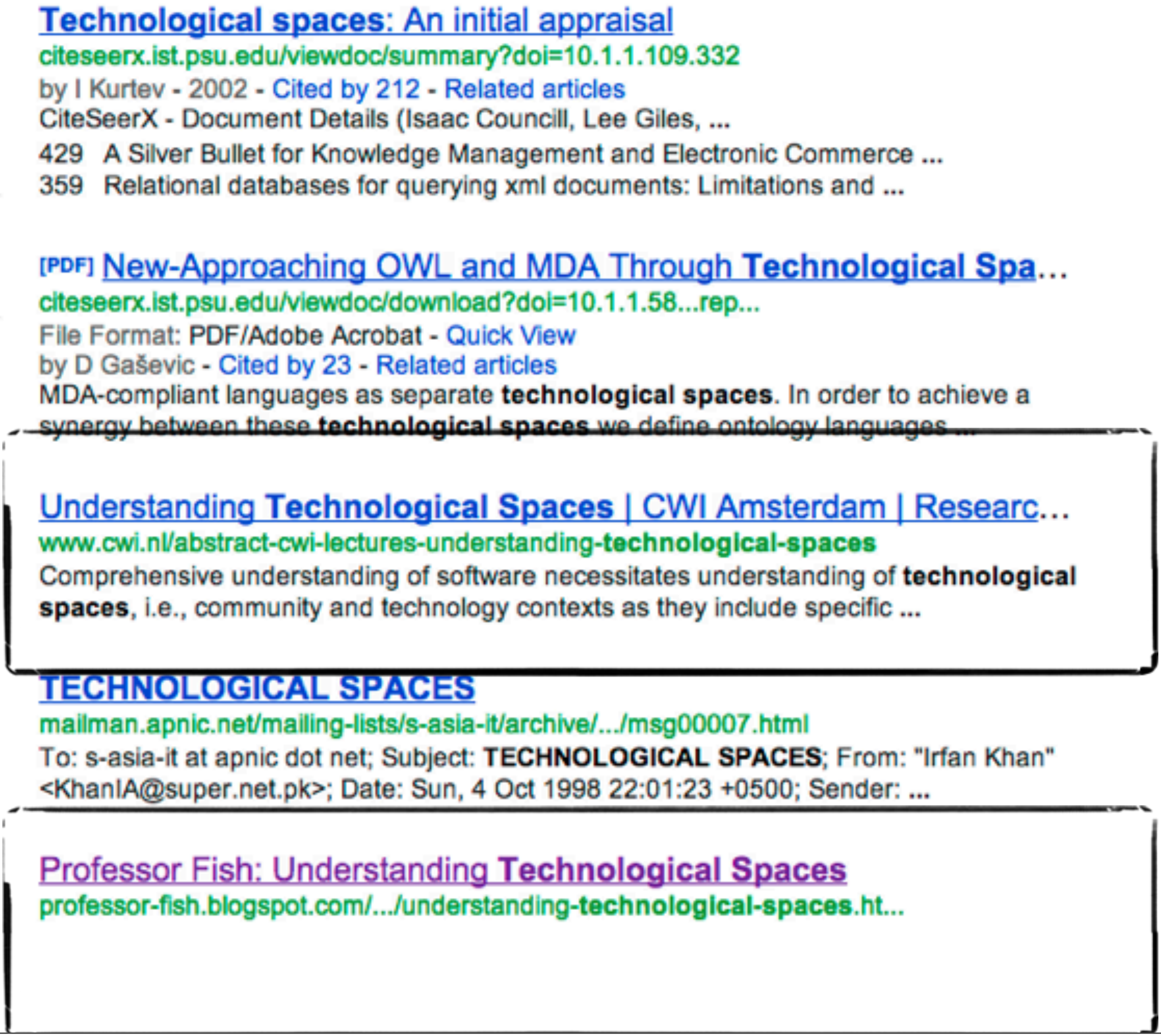


[PDF] Technological Spaces: an Initial Appraisal - EEMCS EPrints S…
eprints.eemcs.utwente.nl/10206/01/0363**TechnologicalSpaces**.pdf
File Format: PDF/Adobe Acrobat - Quick View
by I Kurtev - 2002 - Cited by 212 - Related articles
In this paper, we propose a high level view of **technological spaces** (TS) and ... our work is not to define an abstract theory of **technological spaces**, but to figure ...

Technological spaces: An initial appraisal
citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.332
by I Kurtev - 2002 - Cited by 212 - Related articles
CiteSeerX - Document Details (Isaac Councill, Lee Giles, ...
429 A Silver Bullet for Knowledge Management and Electronic Commerce ...
359 Relational databases for querying xml documents: Limitations and ...

[PDF] New-Approaching OWL and MDA Through **Technological Spa**…
citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58...rep...
File Format: PDF/Adobe Acrobat - Quick View
by D Gaševic - Cited by 23 - Related articles
MDA-compliant languages as separate **technological spaces**. In order to achieve a synergy between these **technological spaces** we define ontology languages ...

Understanding **Technological Spaces** | CWI Amsterdam | Researc…
www.cwi.nl/abstract-cwi-lectures-understanding-**technological-spaces**
Comprehensive understanding of software necessitates understanding of **technological**



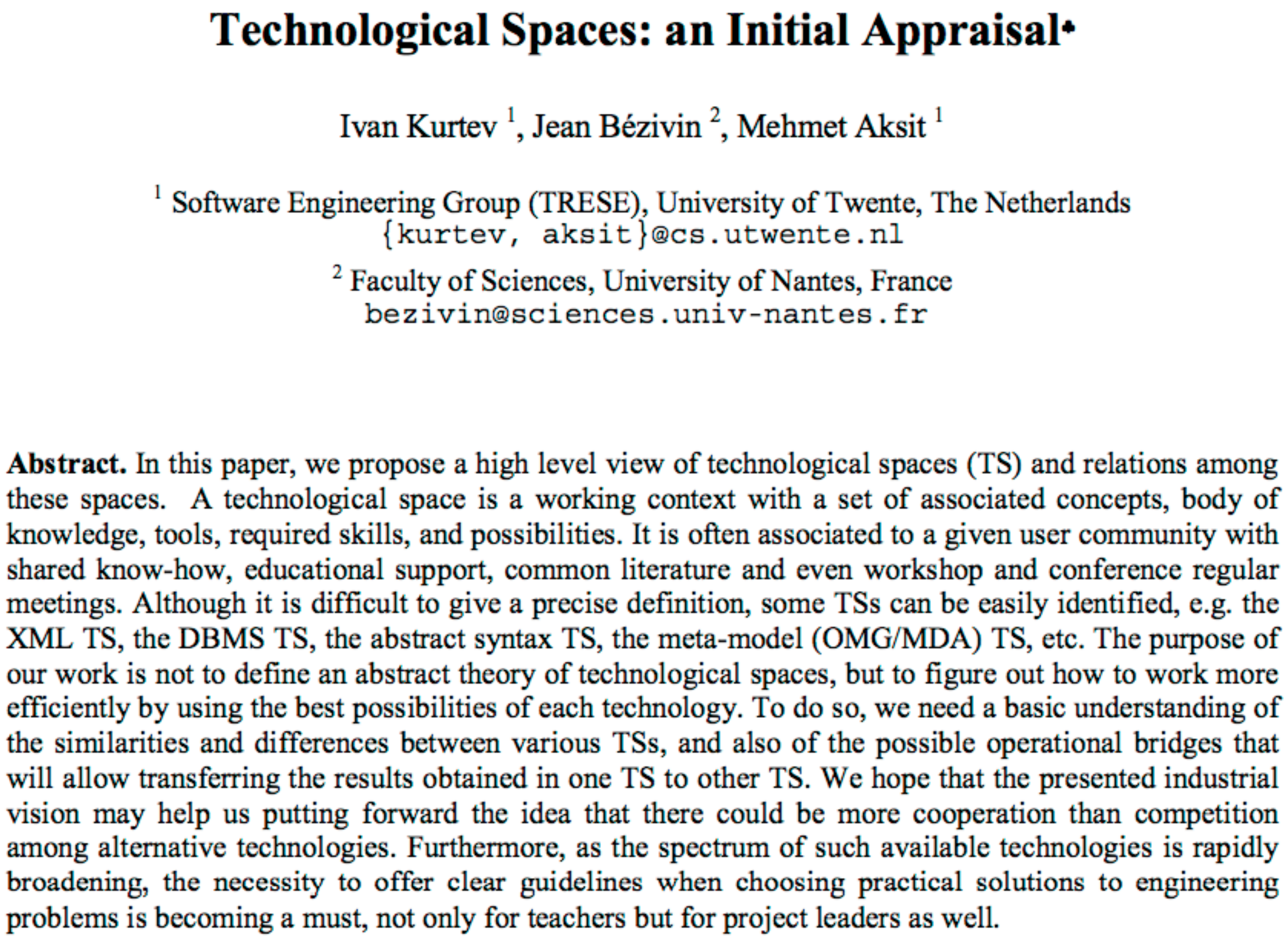
Technological spaces: An initial appraisal
citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.332
by I Kurtev - 2002 - Cited by 212 - Related articles
CiteSeerX - Document Details (Isaac Councill, Lee Giles, ...
429 A Silver Bullet for Knowledge Management and Electronic Commerce ...
359 Relational databases for querying xml documents: Limitations and ...

[PDF] New-Approaching OWL and MDA Through **Technological Spa**...
citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58...rep...
File Format: PDF/Adobe Acrobat - Quick View
by D Gaševic - Cited by 23 - Related articles
MDA-compliant languages as separate **technological spaces**. In order to achieve a synergy between these **technological spaces** we define ontology languages ...

Understanding **Technological Spaces** | CWI Amsterdam | Researc...
www.cwi.nl/abstract-cwi-lectures-understanding-**technological-spaces**
Comprehensive understanding of software necessitates understanding of **technological spaces**, i.e., community and technology contexts as they include specific ...

TECHNOLOGICAL SPACES
mailman.apnic.net/mailing-lists/s-asia-it/archive/.../msg00007.html
To: s-asia-it at apnic dot net; Subject: **TECHNOLOGICAL SPACES**; From: "Irfan Khan" <KhanIA@super.net.pk>; Date: Sun, 4 Oct 1998 22:01:23 +0500; Sender: ...

Professor Fish: Understanding **Technological Spaces**
professor-fish.blogspot.com/.../understanding-**technological-spaces**.ht...



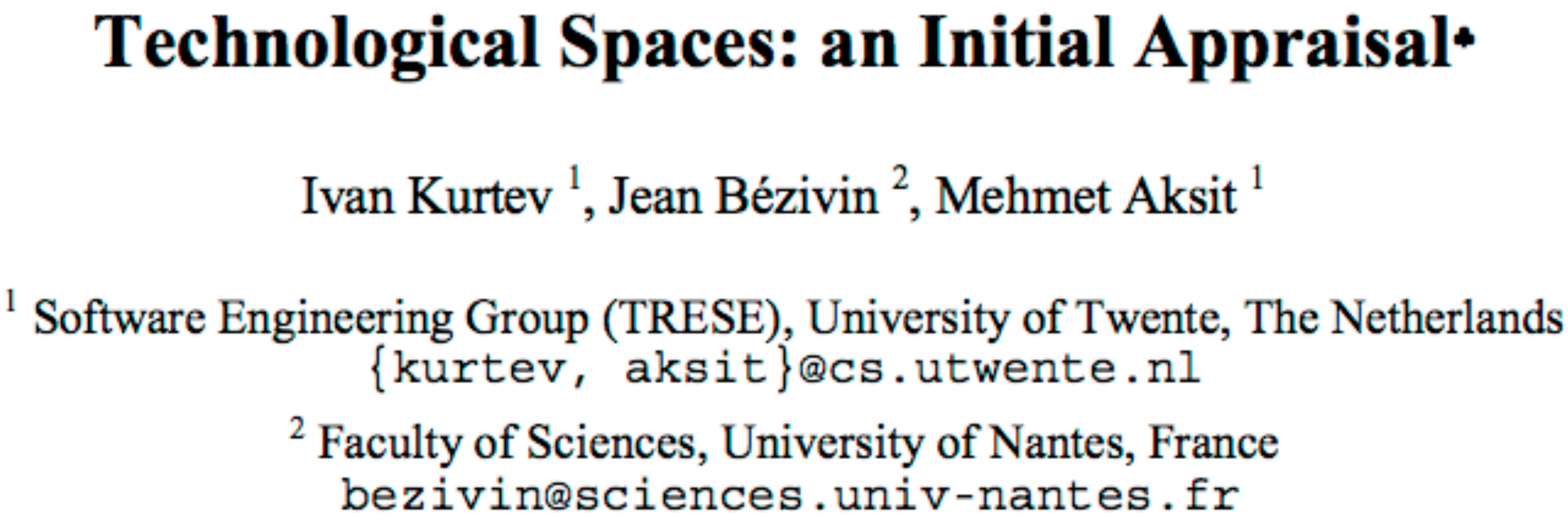
Technological Spaces: an Initial Appraisal*

Ivan Kurtev [1], Jean Bézivin [2], Mehmet Aksit [1]

[1] Software Engineering Group (TRESE), University of Twente, The Netherlands
{kurtev, aksit}@cs.utwente.nl

[2] Faculty of Sciences, University of Nantes, France
bezivin@sciences.univ-nantes.fr

Abstract. In this paper, we propose a high level view of technological spaces (TS) and relations among these spaces. A technological space is a working context with a set of associated concepts, body of knowledge, tools, required skills, and possibilities. It is often associated to a given user community with shared know-how, educational support, common literature and even workshop and conference regular meetings. Although it is difficult to give a precise definition, some TSs can be easily identified, e.g. the XML TS, the DBMS TS, the abstract syntax TS, the meta-model (OMG/MDA) TS, etc. The purpose of our work is not to define an abstract theory of technological spaces, but to figure out how to work more efficiently by using the best possibilities of each technology. To do so, we need a basic understanding of the similarities and differences between various TSs, and also of the possible operational bridges that will allow transferring the results obtained in one TS to other TS. We hope that the presented industrial vision may help us putting forward the idea that there could be more cooperation than competition among alternative technologies. Furthermore, as the spectrum of such available technologies is rapidly broadening, the necessity to offer clear guidelines when choosing practical solutions to engineering problems is becoming a must, not only for teachers but for project leaders as well.



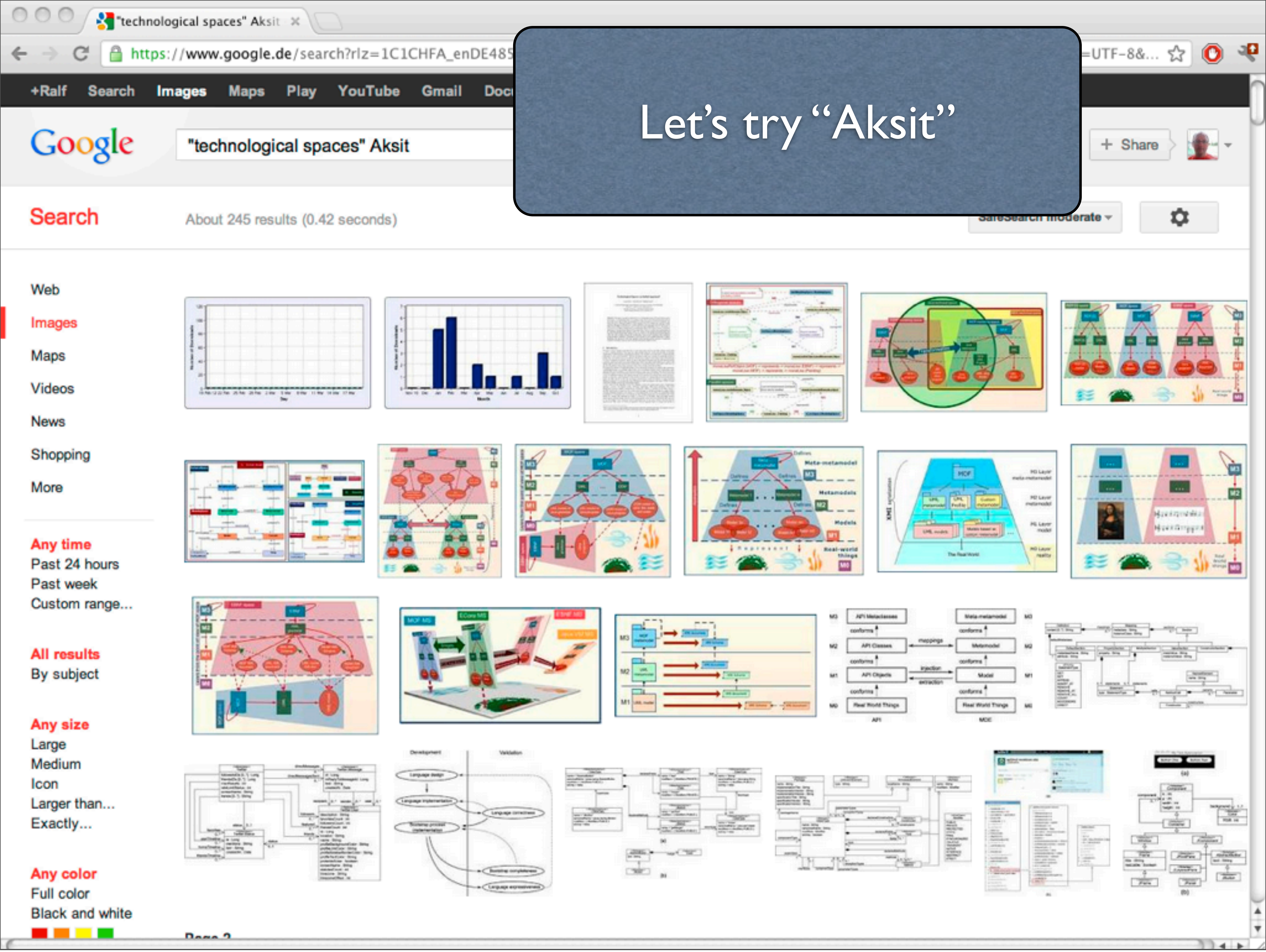
Technological Spaces: an Initial Appraisal•

Ivan Kurtev [1], Jean Bézivin [2], Mehmet Aksit [1]

[1] Software Engineering Group (TRESE), University of Twente, The Netherlands
{kurtev, aksit}@cs.utwente.nl

[2] Faculty of Sciences, University of Nantes, France
bezivin@sciences.univ-nantes.fr

A technological space is **a working context with a set of associated concepts, body of knowledge,tools, required skills, and possibilities**. It is often associated to a given user community with shared know-how, educational support, common literature and even workshop and conference meetings. It is at the same time a zone of established expertise and ongoing research and a repository for abstract and concrete resources.



Let’s try “Aksit”

"technological spaces" Aksit

https://www.google.de/search?rlz=1C1CHFA_enDE485 …=UTF-8&…

+Ralf Search Images Maps Play YouTube Gmail Doc…

"technological spaces" Aksit

+ Share

Search

About 245 results (0.42 seconds)

SafeSearch moderate

Web
Images
Maps
Videos
News
Shopping
More

Any time
Past 24 hours
Past week
Custom range...

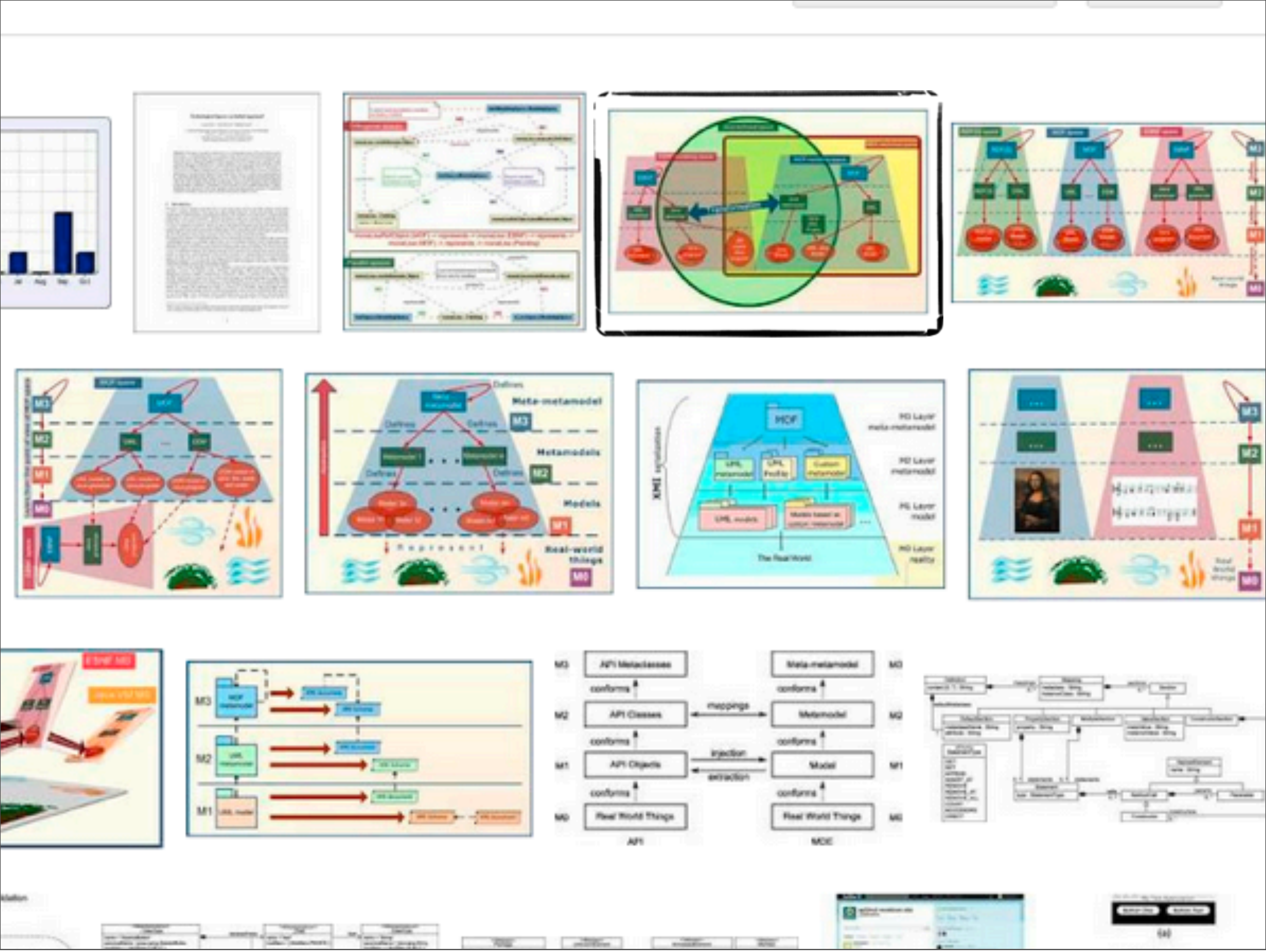
All results
By subject

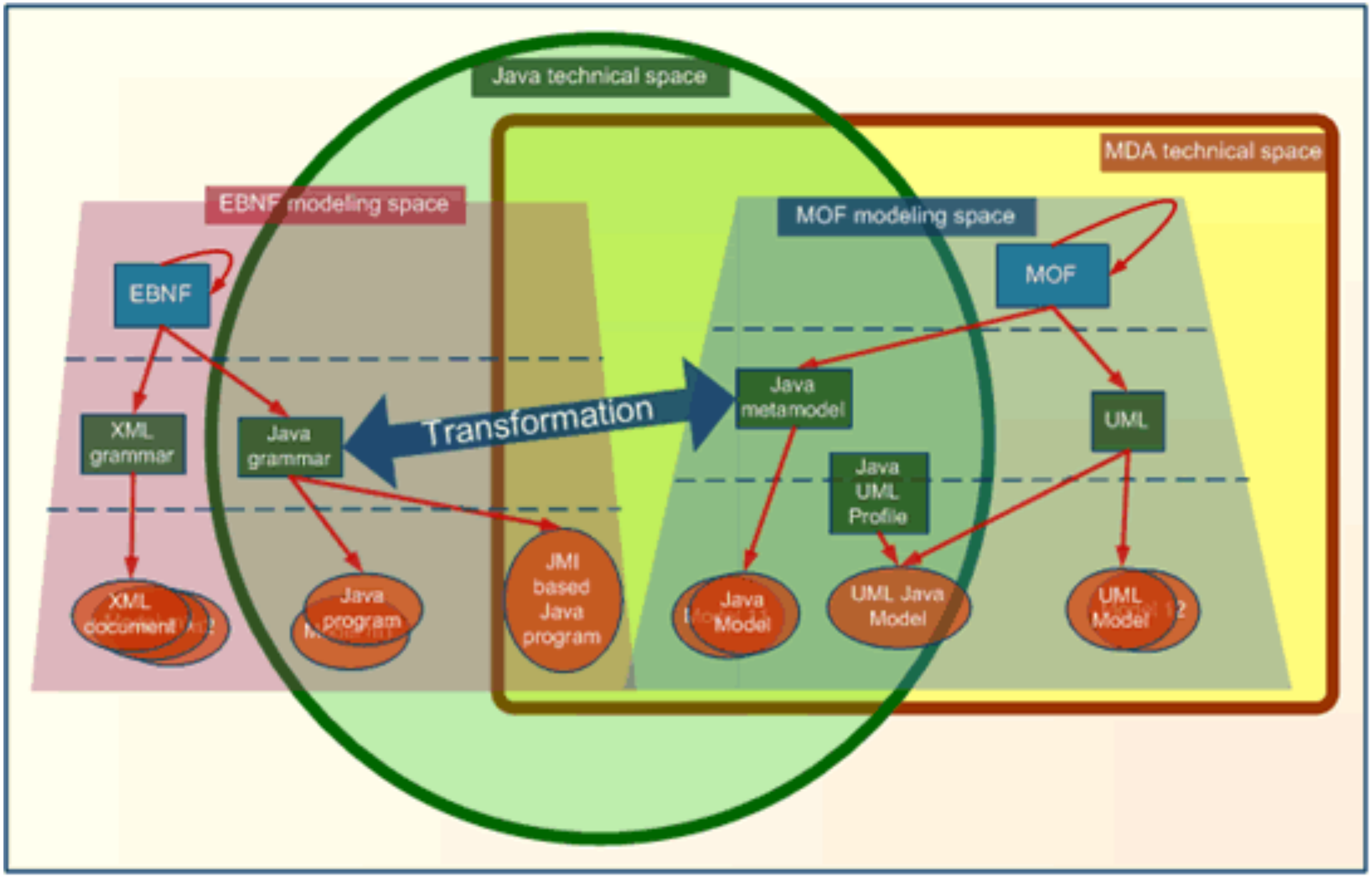
Any size
Large
Medium
Icon
Larger than...
Exactly...

Any color
Full color
Black and white

Page 2

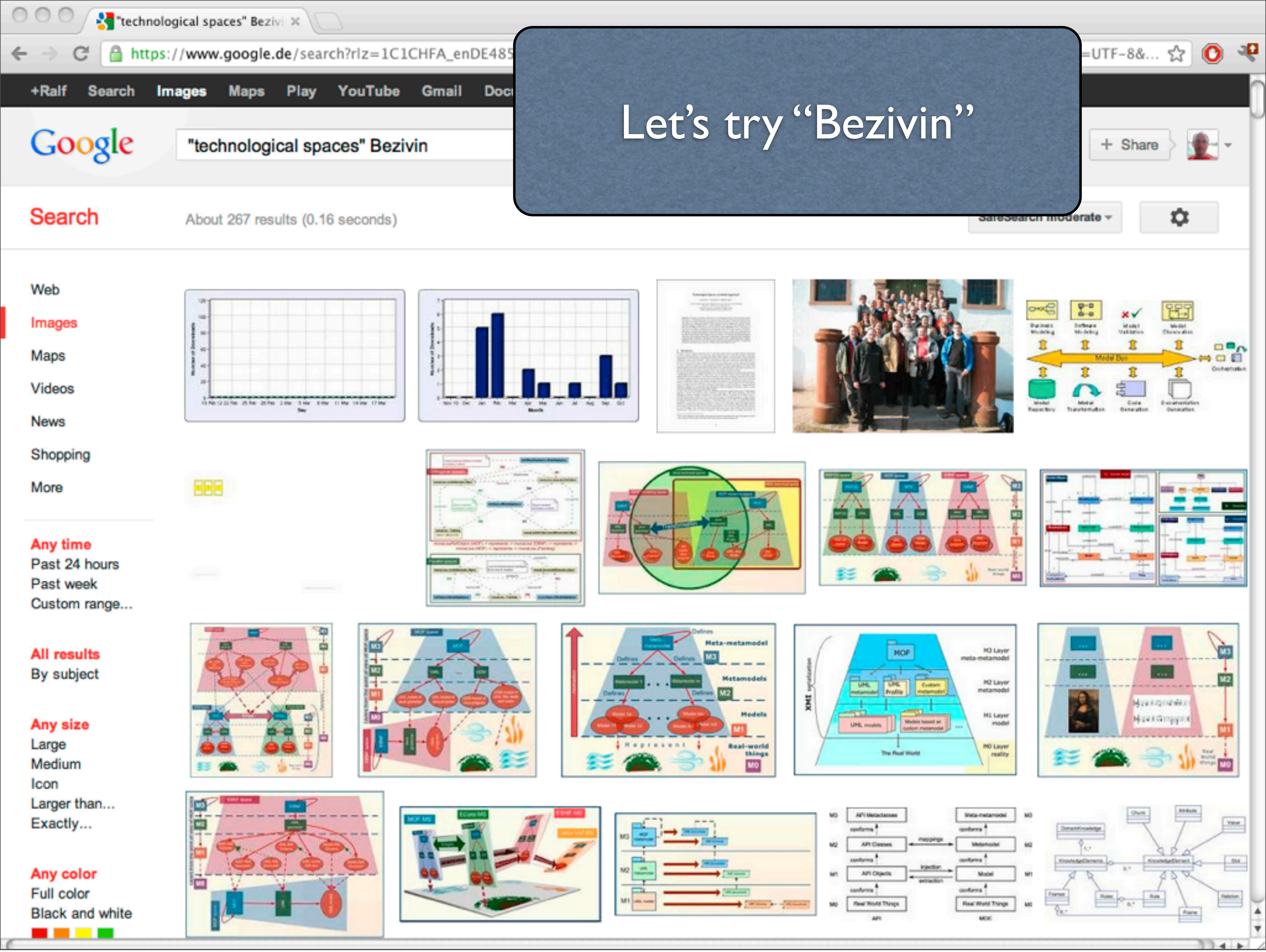
Wednesday, June 20, 12


Meta-metamodel
Metamodels
Models
Real-world things
Defines
Represent
M3
M2
M1
M0
MOF
M3 Layer meta-metamodel
M2 Layer metamodel
M1 Layer model
M0 Layer reality
The Real World
XMI
API Metaclasses
API Classes
API Objects
Real World Things
Meta-metamodel
Metamodel
Model
conforms
mappings
injection
extraction
API
MDE



Technical space comprises one or more modeling spaces

Djuric, D., Gaševic, D., Devedžic, V.,: "The Tao of Modeling Spaces", in *Journal of Object Technology*, vol. 5. no. 8, Novmeber-December 2006, pp. 125-147.http://www.jot.fm/issues/issue_2006_11/article4

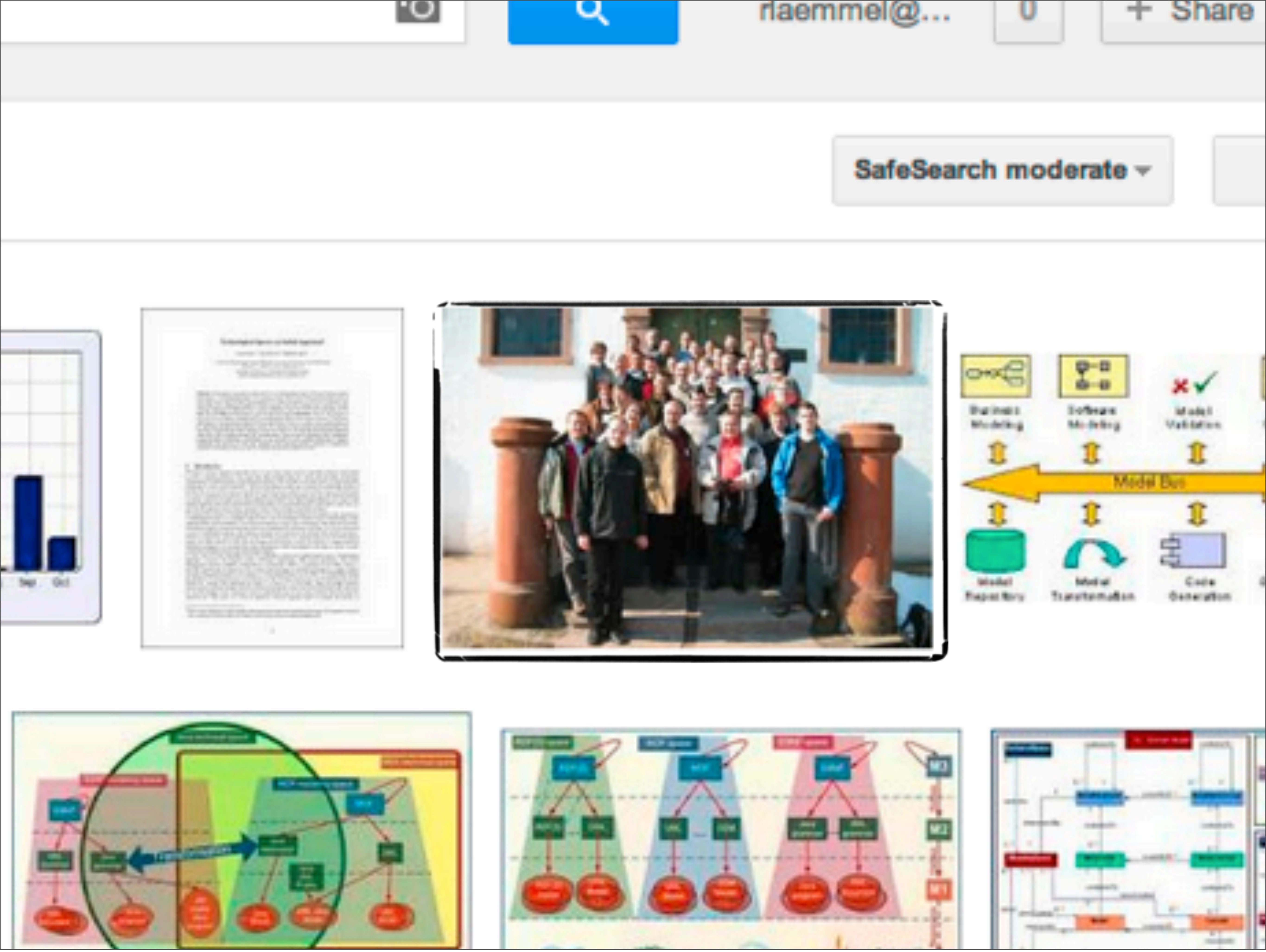


"technological spaces" Bezivin

About 267 results (0.16 seconds)

Let's try "Bezivin"

Wednesday, June 20, 12



rlaemmel@...

0

+ Share

SafeSearch moderate

http://www.dagstuhl.de/04101

29.02.04 — 05.03.04, Seminar 04101

Language Engineering for
Model-Driven Software Development

Organizers *J. Bézivin and R. Heckel*

http://www.dagstuhl.de/04101

29.02.04 — 05.03.04, Seminar 04101

Language Engineering for

Model-Driven Software Development

Organizers *J. Bézivin and R. Heckel*

http://www.dagstuhl.de/04101

29.02.04 — 05.03.04, Seminar 04101

Language Engineering for

Model-Driven Software Development

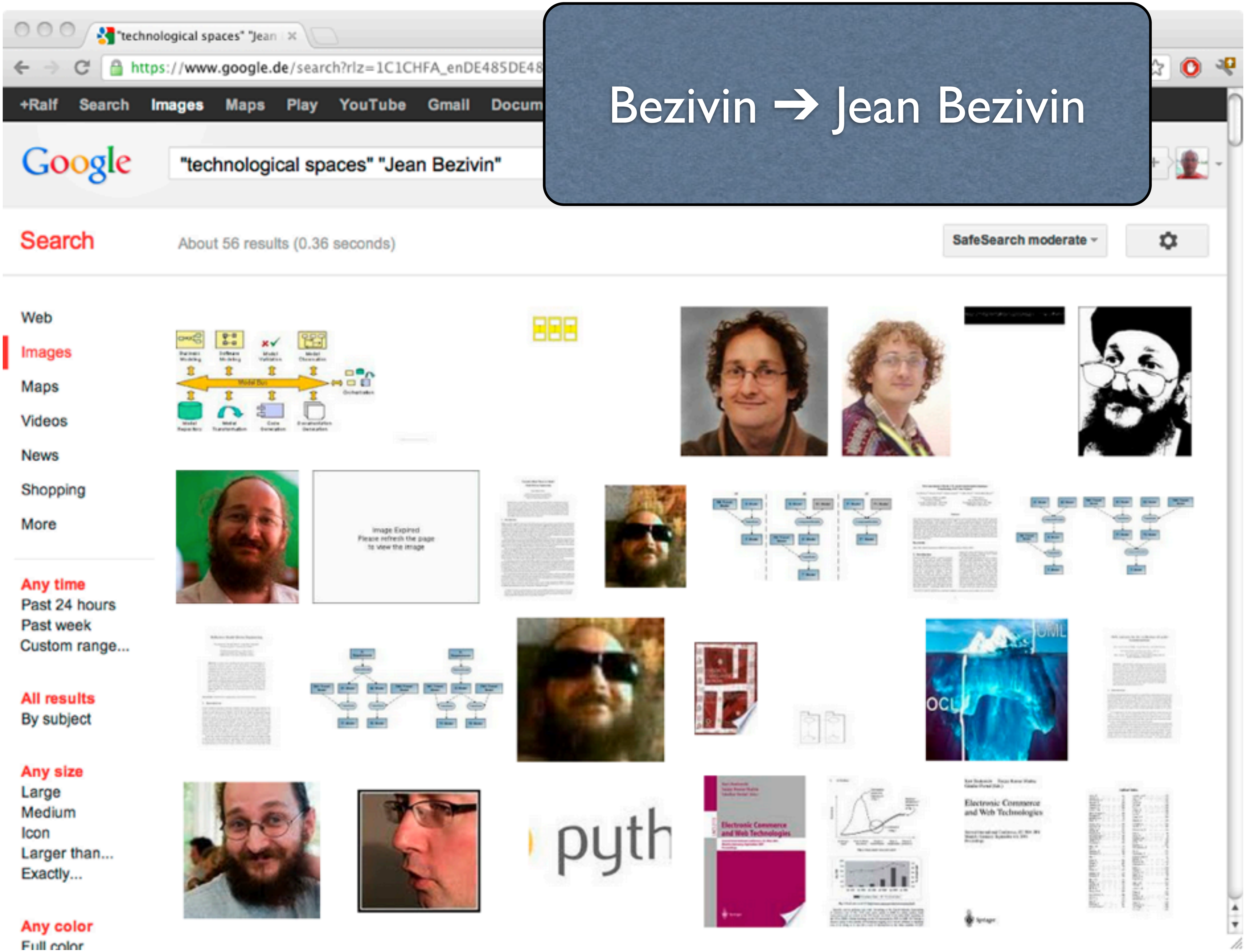
Organizers *J. Bézivin and R. Heckel*

http://www.dagstuhl.de/04101
29.02.04 — 05.03.04, Seminar 04101
Language Engineering for
Model-Driven Software Development
Organizers *J. Bézivin and R. Heckel*

<u>http://www.dagstuhl.de/04101</u>
29.02.04 — 05.03.04, Seminar 04101
Language Engineering for
Model-Driven Software Development
Organizers *J. Bézivin and R. Heckel*

Where you there?

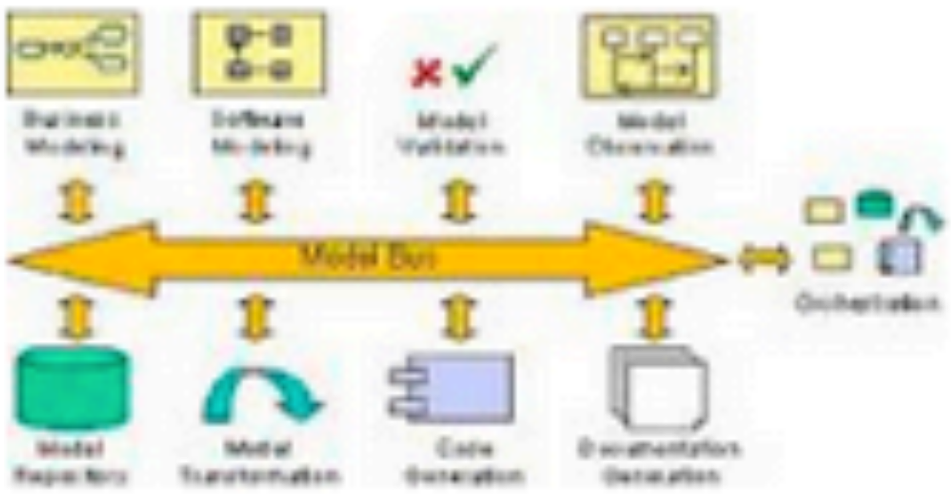
http://www.dagstuhl.de/04101
29.02.04 — 05.03.04, Seminar 04101
Language Engineering for
Model-Driven Software Development
Organizers *J. Bézivin and R. Heckel*



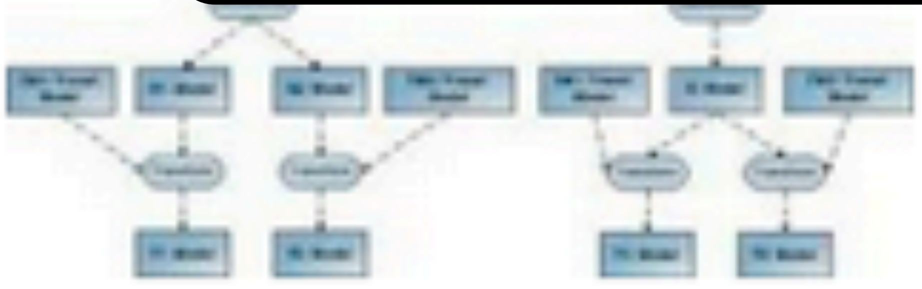
"technological spaces" "Jean Bezivin"

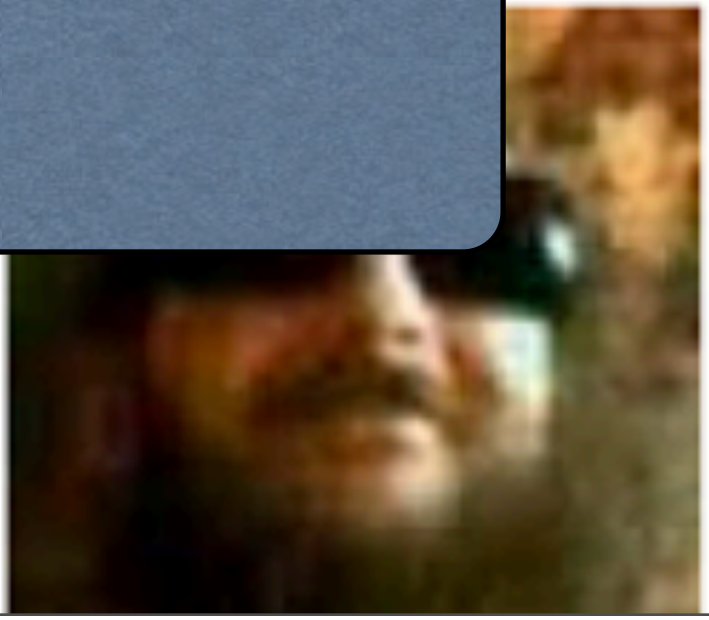
About 56 results (0.36 seconds)

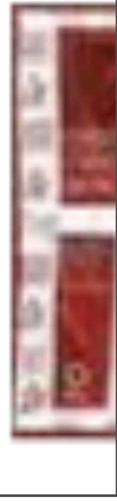
Bezivin → Jean Bezivin

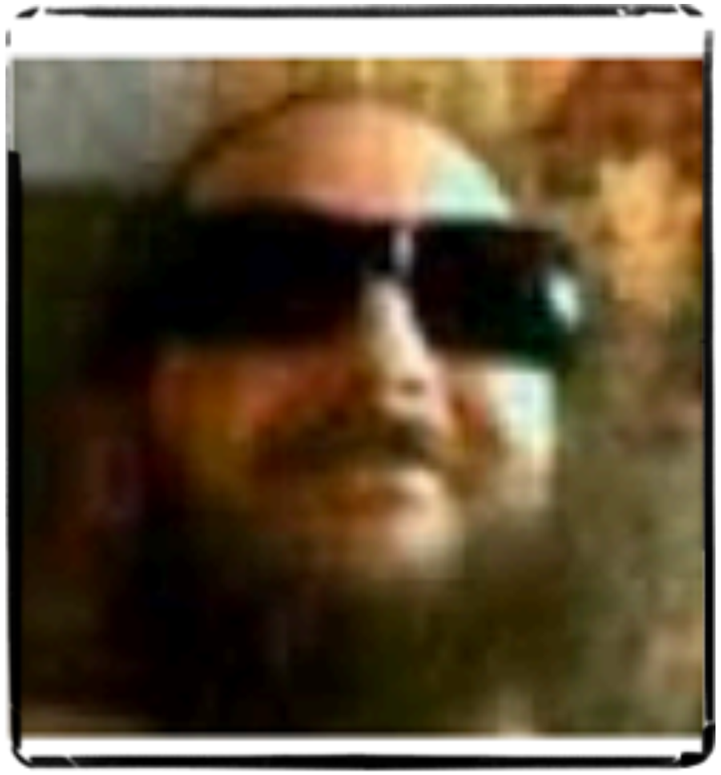


This is not Jean Bezivin.

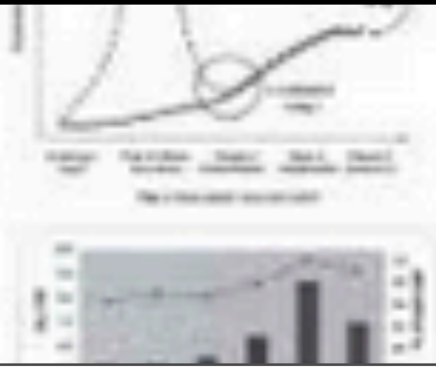


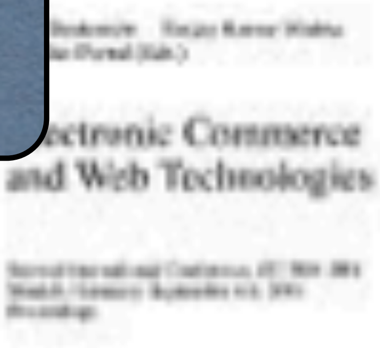


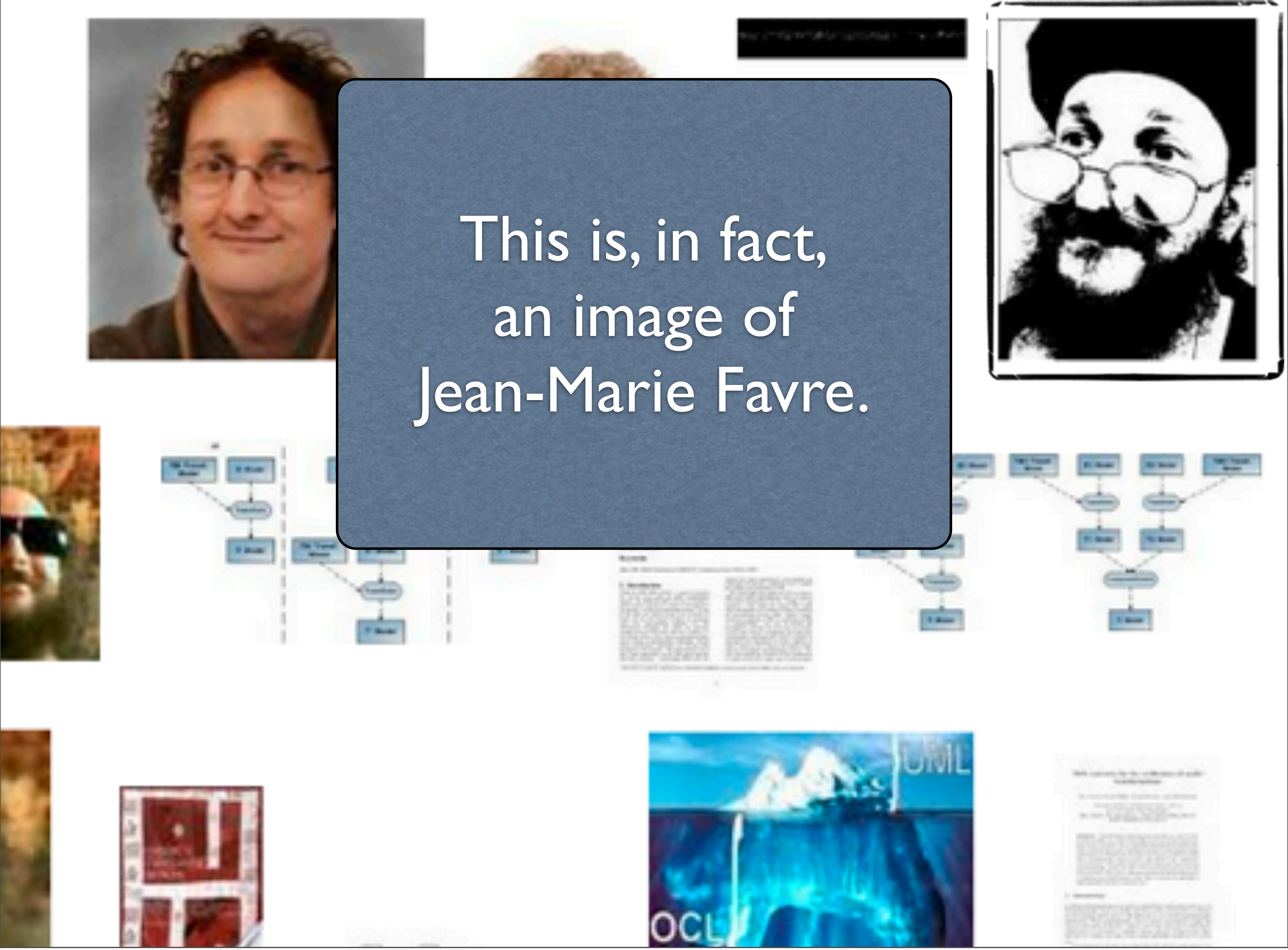




This is
Jean-Marie Favre.



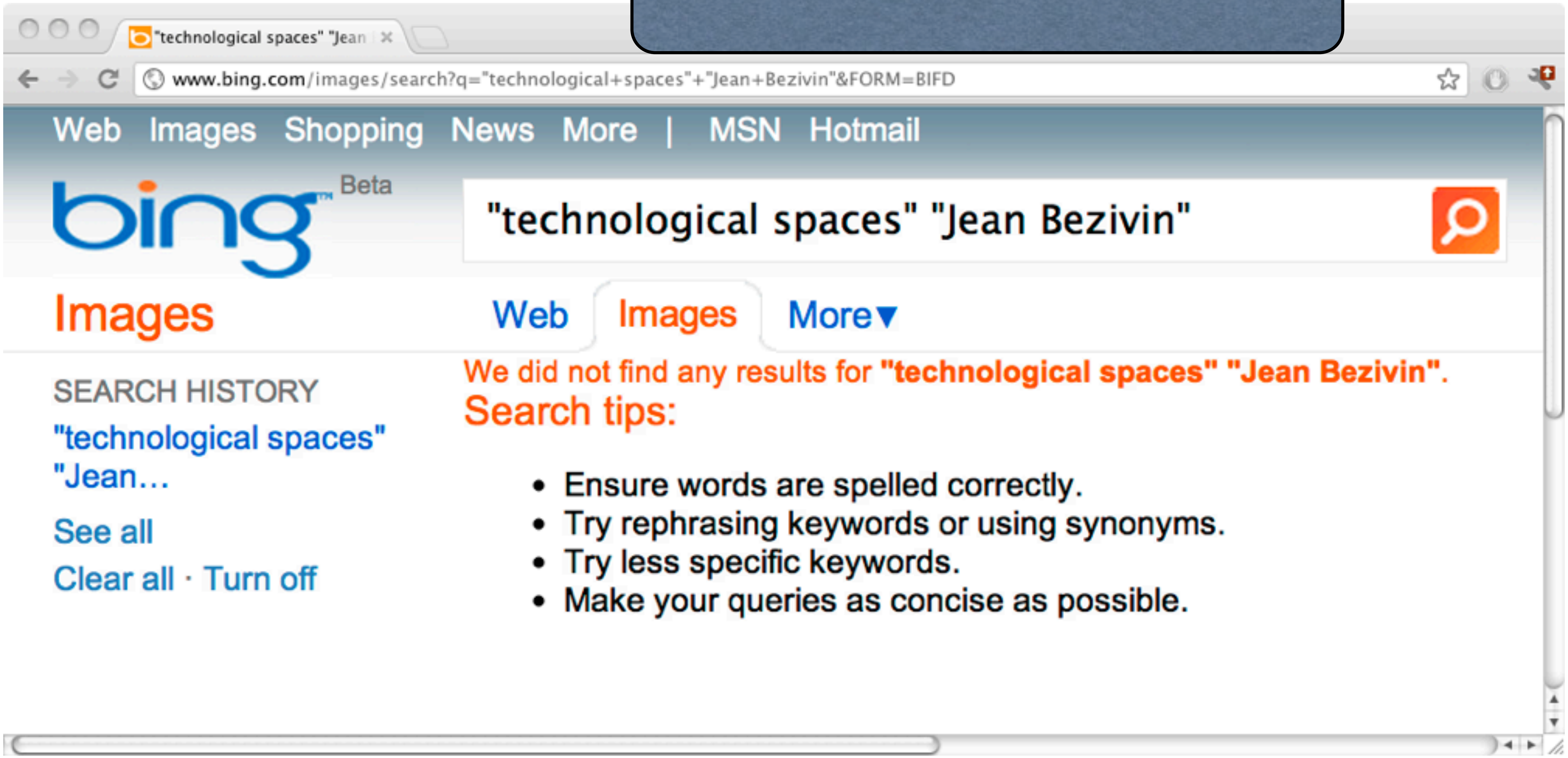




This is, in fact,
an image of
Jean-Marie Favre.

Google → Bing

Google → Bing



"technological spaces" "Jean
www.bing.com/images/search?q="technological+spaces"+"Jean+Bezivin"&FORM=BIFD
Web Images Shopping News More | MSN Hotmail
bing Beta
"technological spaces" "Jean Bezivin"
Images
Web Images More▼
SEARCH HISTORY
"technological spaces" "Jean...
See all
Clear all · Turn off

We did not find any results for **"technological spaces" "Jean Bezivin"**.

Search tips:

- Ensure words are spelled correctly.
- Try rephrasing keywords or using synonyms.
- Try less specific keywords.
- Make your queries as concise as possible.

Technological space
= Technology and community context in software engineering

- Objectware
- Modelware
- Grammarware
- XMLware
- Ontoware
- Tableware
- ...

More technological spaces ...

More technological spaces ...

Data driven journalism

Specware

CSVware

TOOLS 2007-2012 †

Javaware

???

Lambdaware

Rubyware

101companies to the rescue

101companies: a community project on software technologies and software languages
by *Jean-Marie Favre, Ralf Lämmel, Thomas Schmorleiz, Andrei Varanovich*.
In Proceedings of TOOLS 2012. http://softlang.uni-koblenz.de/101companies/inauguration/

We have a problem!

Kind regards from Jean-Marie Favre

We have a problem!

Too much software technologies.

Too much software languages.

Too little time.

XPATH TXL Sesame JPA

Jena Rose JDBC EMF.gen XText

jDOM JAXB Jersey RDF(S) UTF8 Jena ODM

XSD MOF

UML BNF VLDB JeanBeans Teneo

Stratego

xerces GWT SLE2010 Json

OCL sax Ralf

RDFS OWL saxon OWL

Ecore

Rest

MySQL XMI ORACLE RDF Jean OMG

JMI EMF JMF XSD ArgoUML

xalan ODBC SparQL XMLSpy Yacc RDFa

XSLT LALR Prolog

JAXP Protegé

CFG SBVR DOM Java ATOM ER

SQL DDL Dragan SQL

Antlr QVT

TENEO XLST

Awk DTD TCS

sed Saxon

grep ASCII XSD

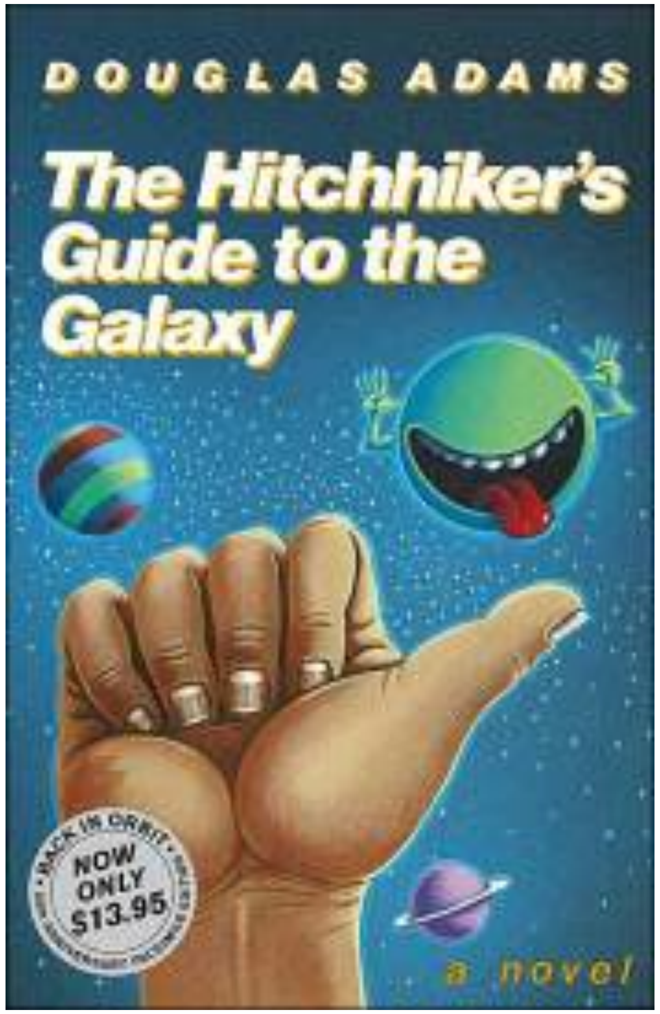
Hibernate XQuery

What’s 101companies?

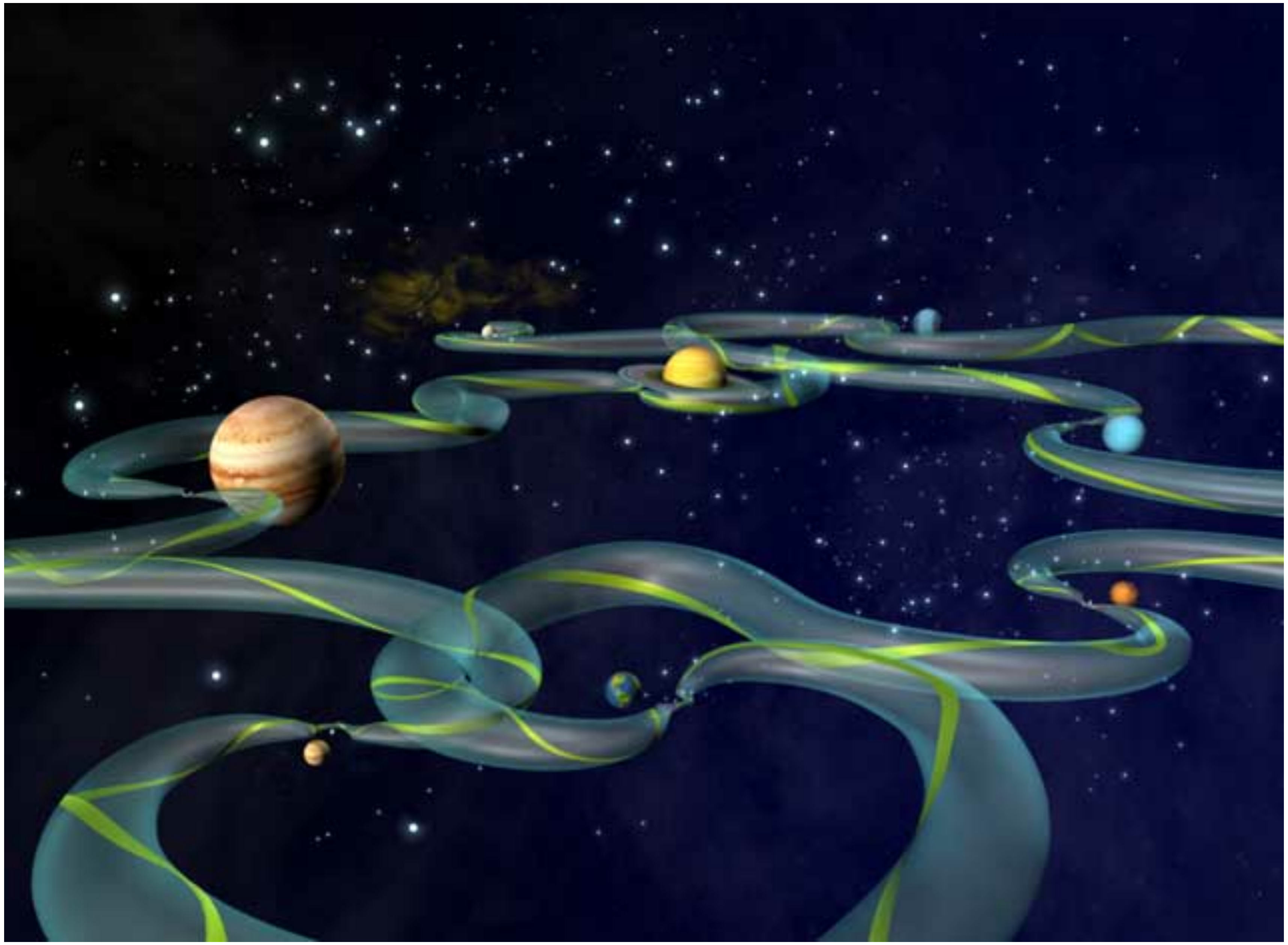
What’s 101companies?

It is ...

The Hitchhiker's Guide to the ***Software*** Galaxy



Software developers need to be space travelers!



http://www.nasa.gov/images/content/63114main_highway_med.jpg

Why is space travel a burden?

- Computational models
- Type systems
- Culture
- Conventions, terminology
- Graphs vs. trees vs. ...
- ...
- Accidental complexity

What’s 101companies?

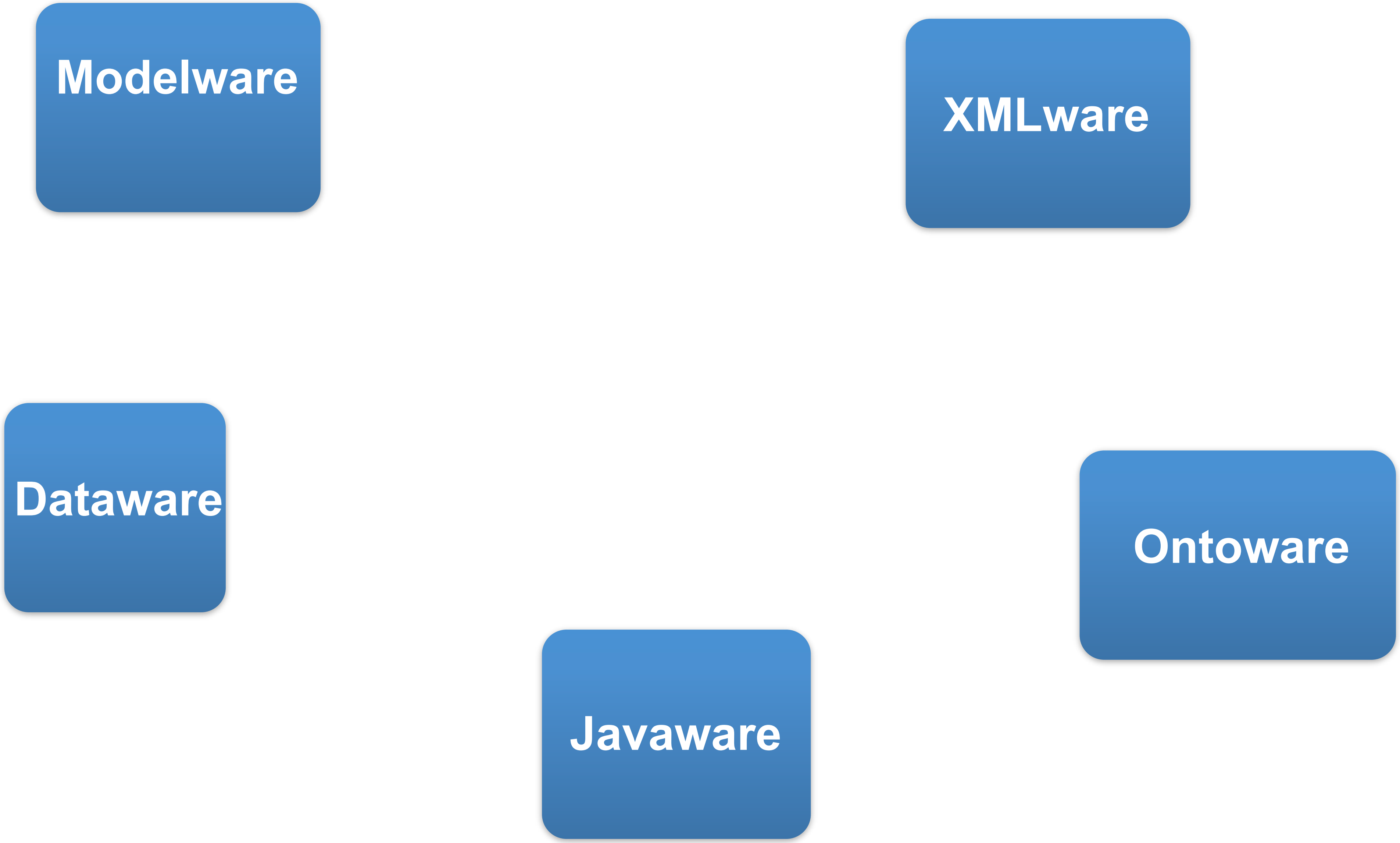
http://upload.wikimedia.org/wikipedia/en/thumb/6/66/E_t_the_extra_terrestrial_ver3.jpg/220px-E_t_the_extra_terrestrial_ver3.jpg

What’s 101companies?

It is a knowledge resource for technological space travel.

http://upload.wikimedia.org/wikipedia/en/thumb/6/66/E_t_the_extra_terrestrial_ver3.jpg/220px-E_t_the_extra_terrestrial_ver3.jpg

What's 101companies?



Modelware

XMLware

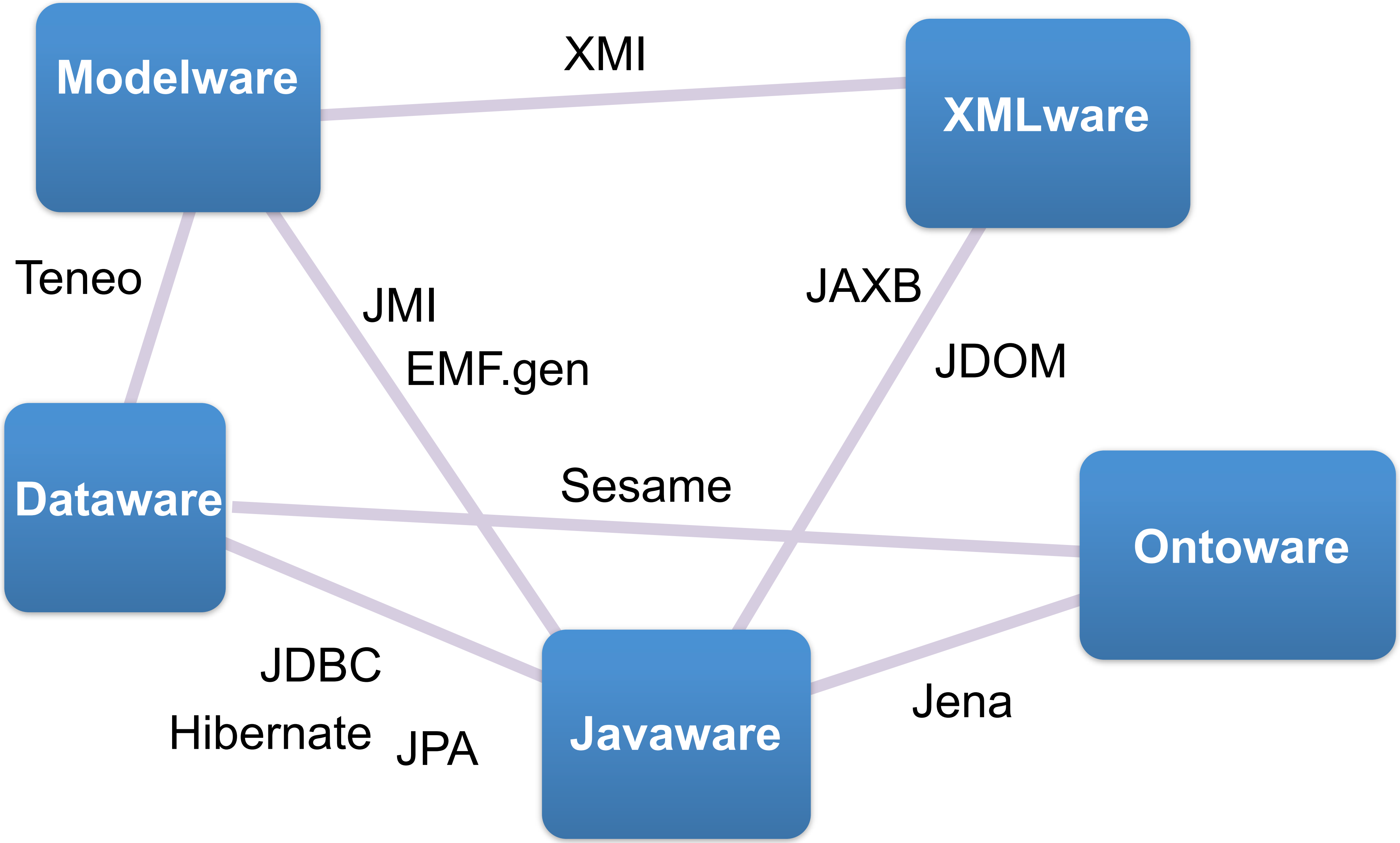
Dataware

Ontoware

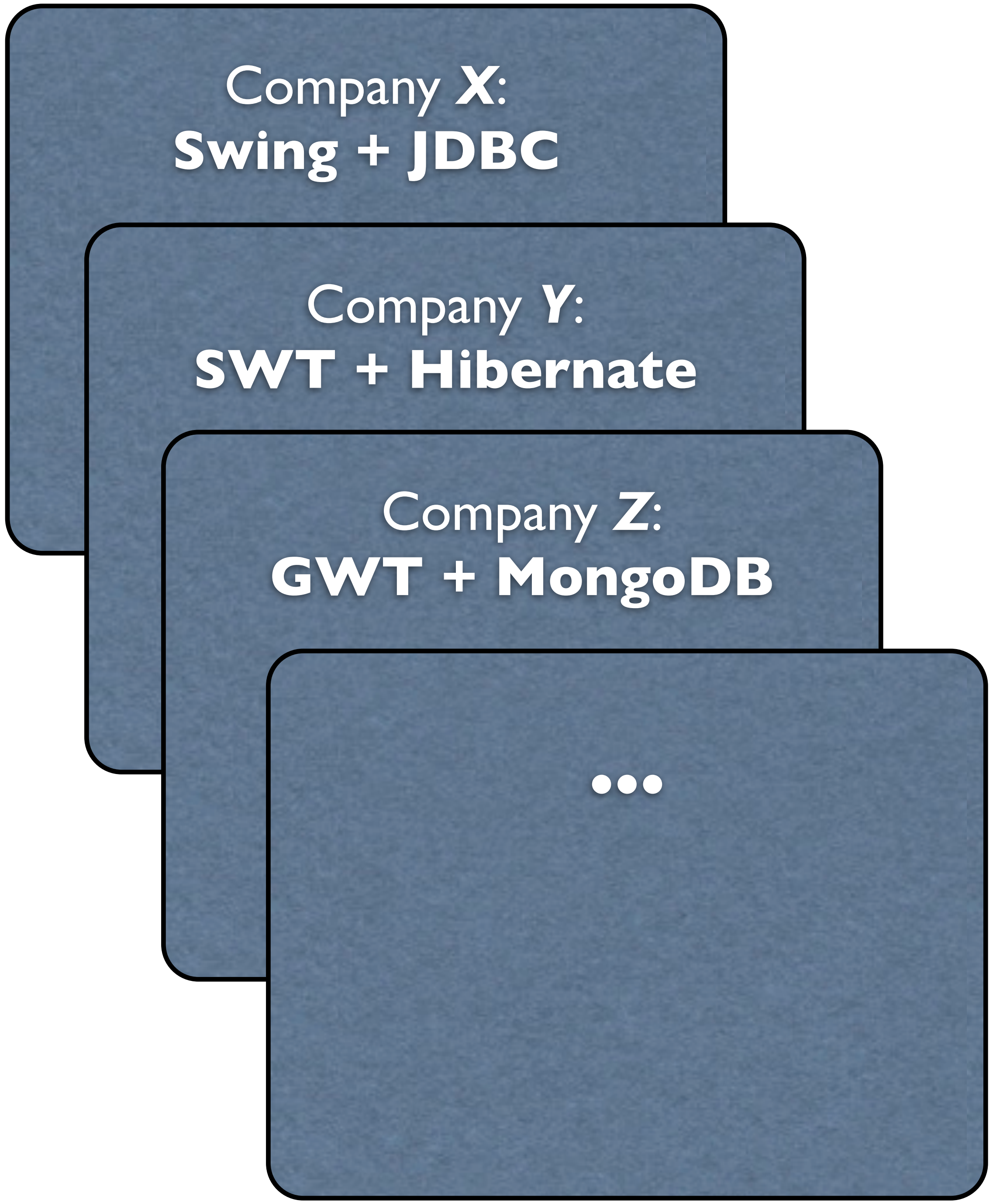
Javaware

What's 101companies?

It is a knowledge resource for technological space travel.



What’s 101companies?



Company *X*:
Swing + JDBC

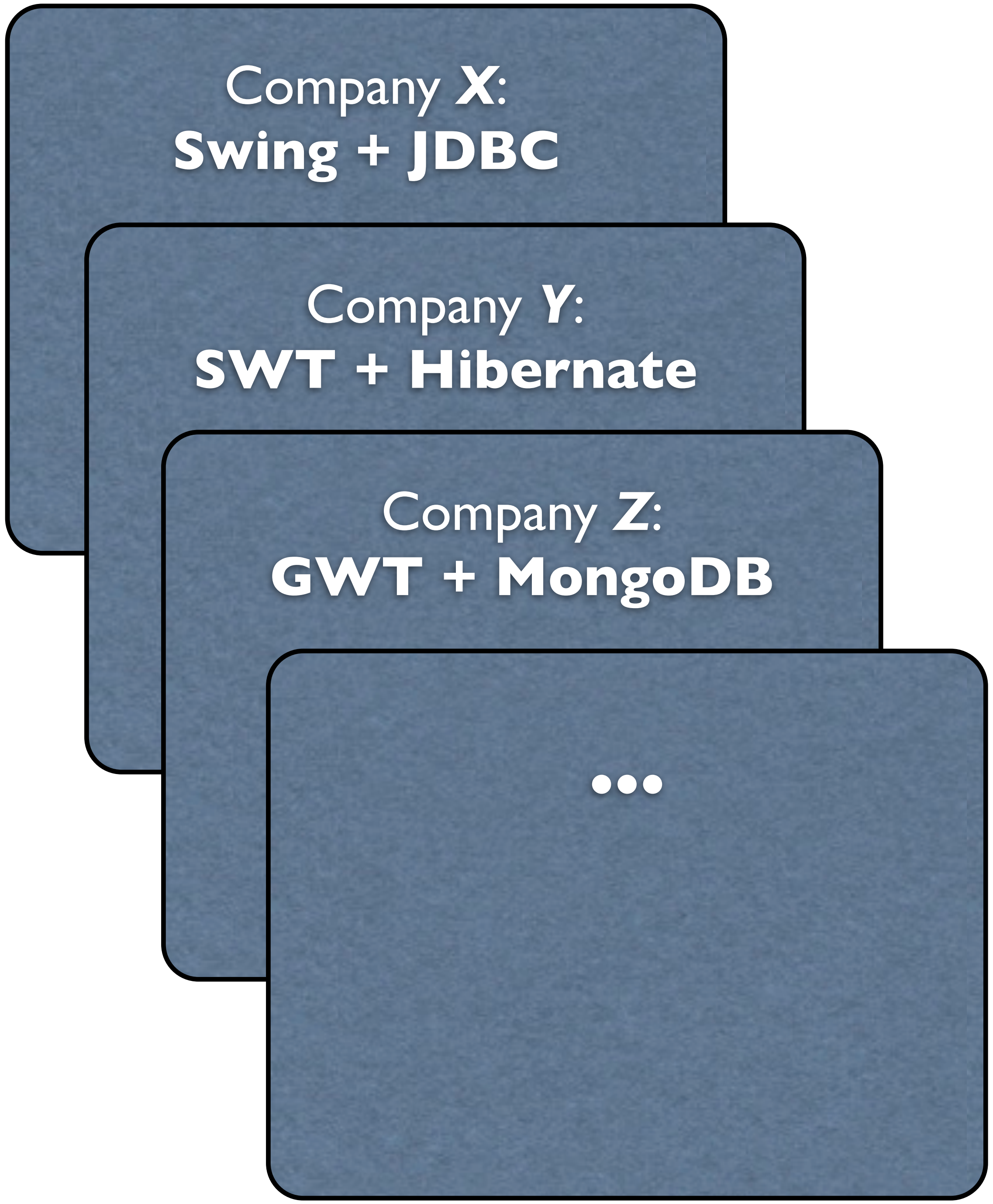
Company *Y*:
SWT + Hibernate

Company *Z*:
GWT + MongoDB

...

A **community project** aiming at a **knowledge base** about software **technologies and languages** based on implementations of a human-resources management system.

Why is it called “101companies”?



Company ***X***:
Swing + JDBC

Company ***Y***:
SWT + Hibernate

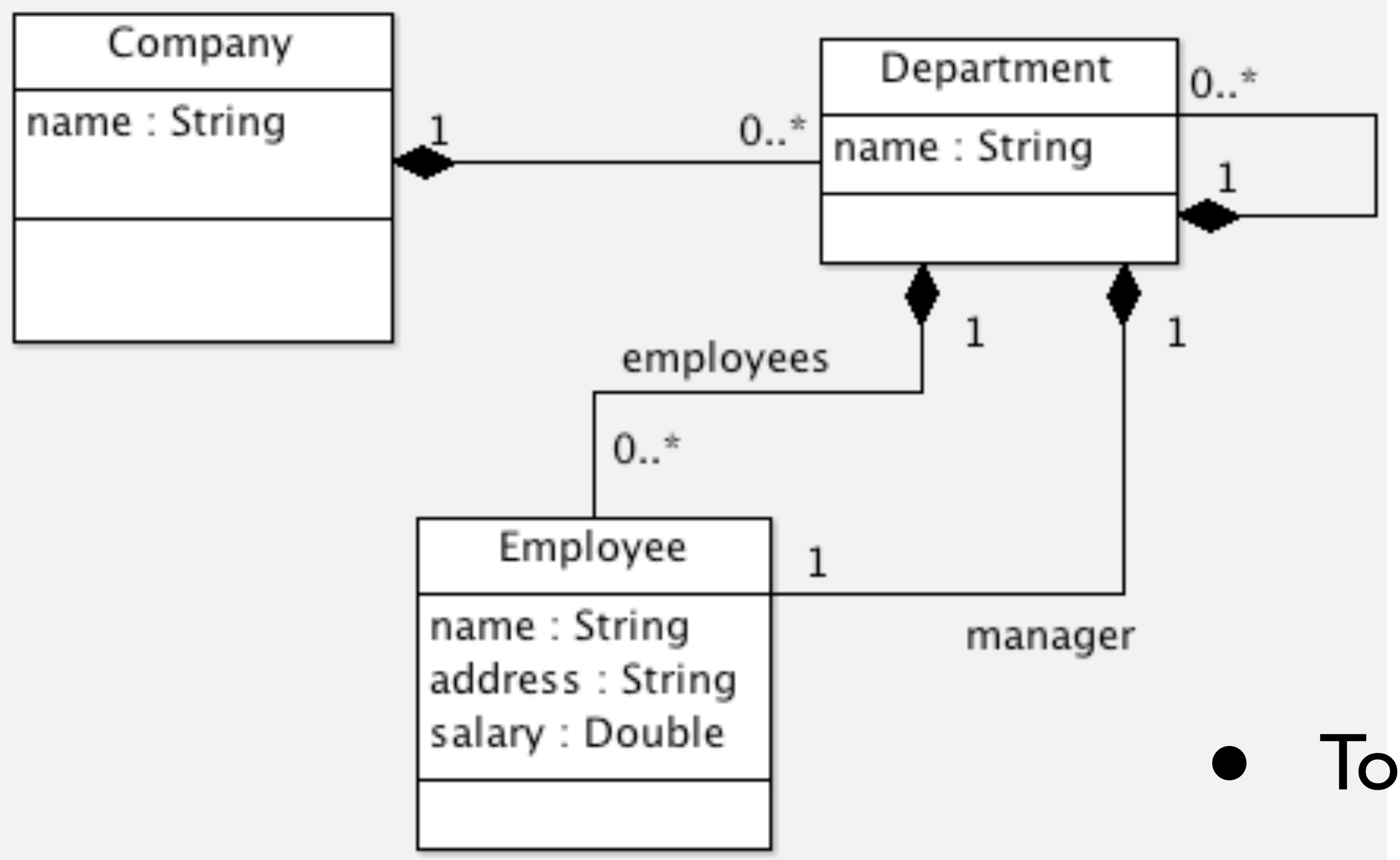
Company ***Z***:
GWT + MongoDB

…

101 ways of building a HRMS.

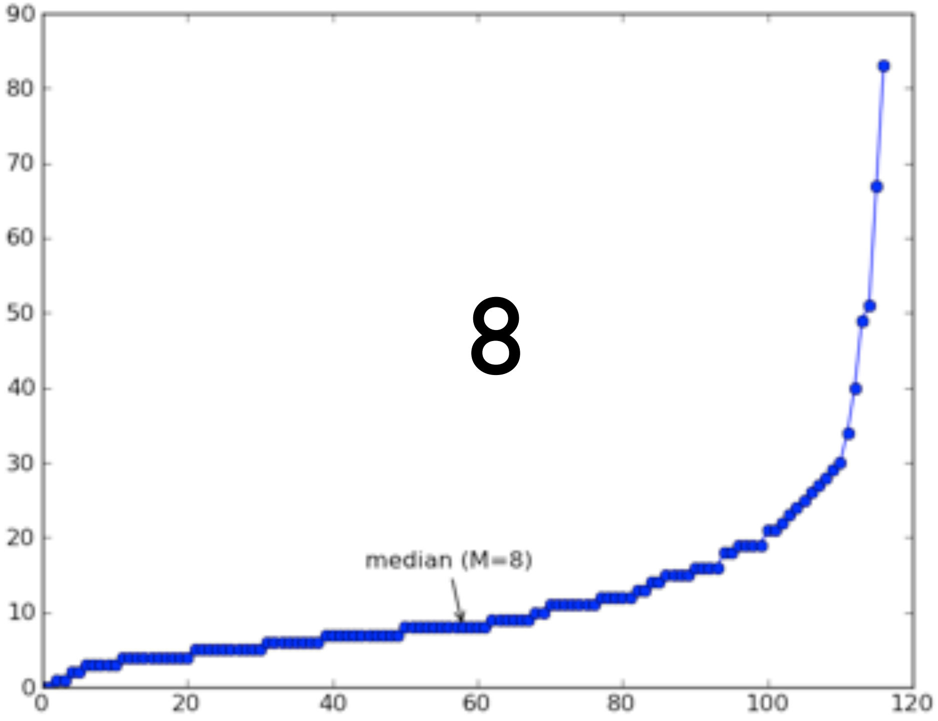
Building a HRMS for 101 companies.

A Human Resources Management System

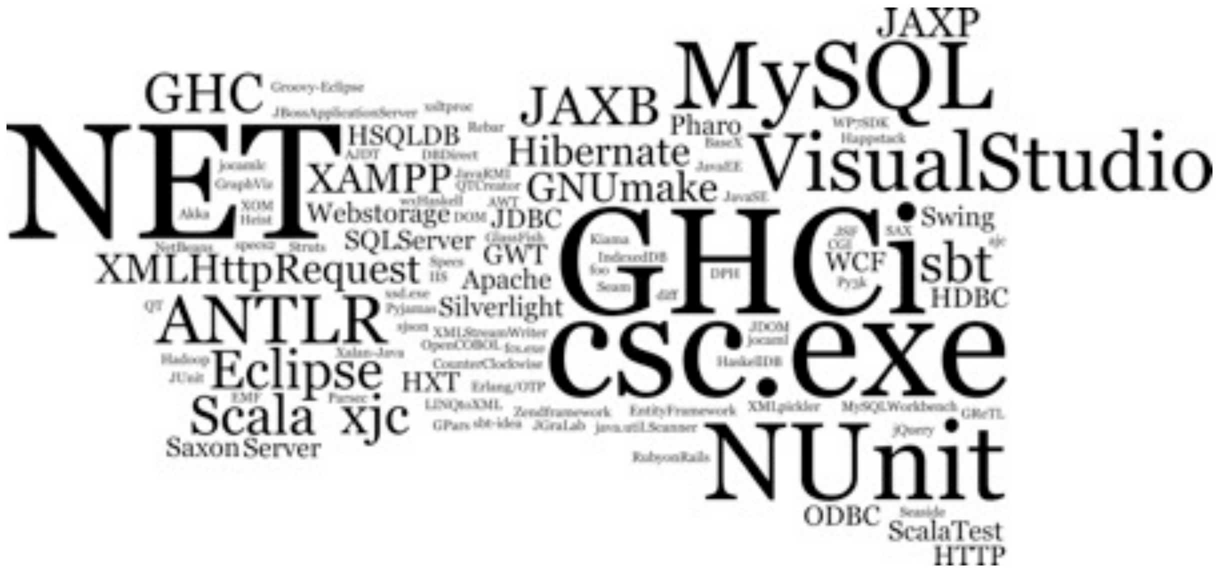


- Total salaries
- ~~Increase salaries~~
- Cut salaries
- Edit employee data
- Import / export company data

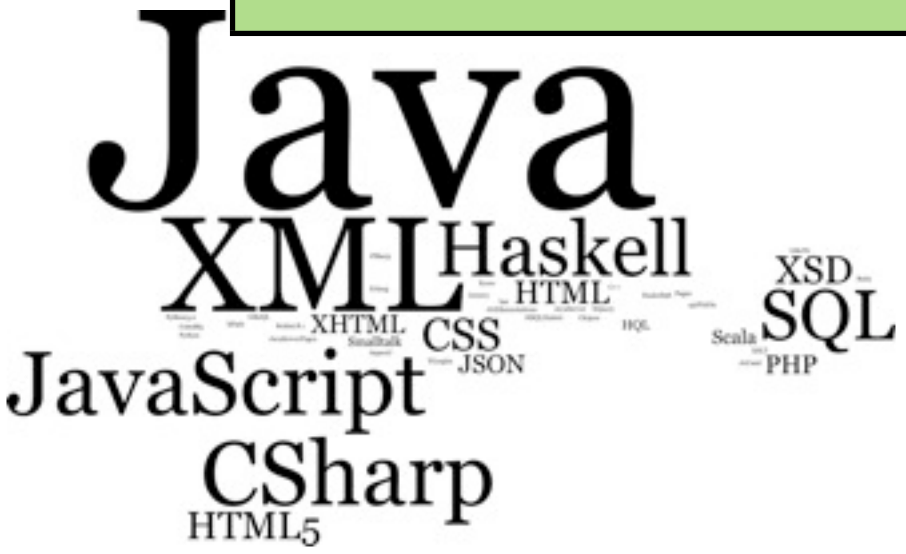
The 101companies **Repository**

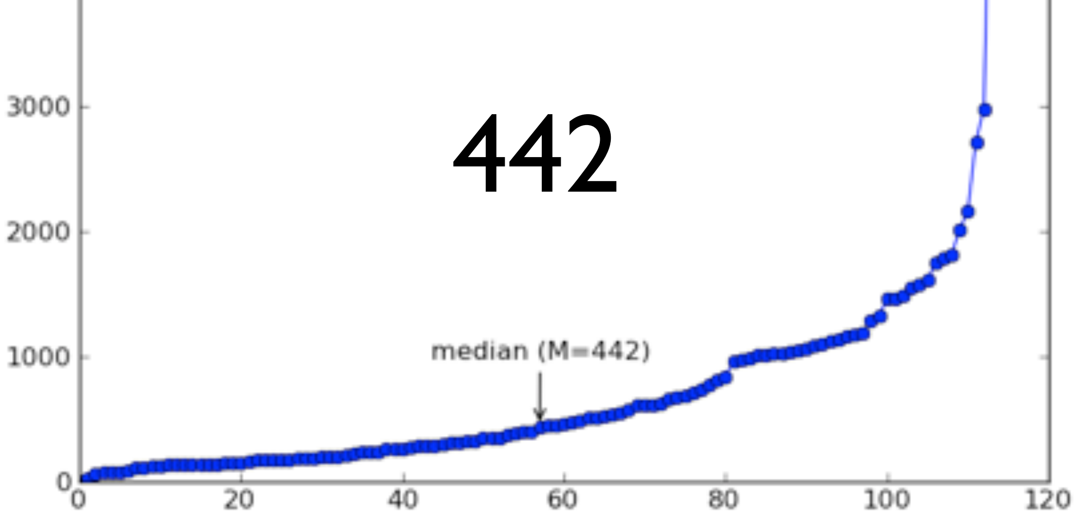


8



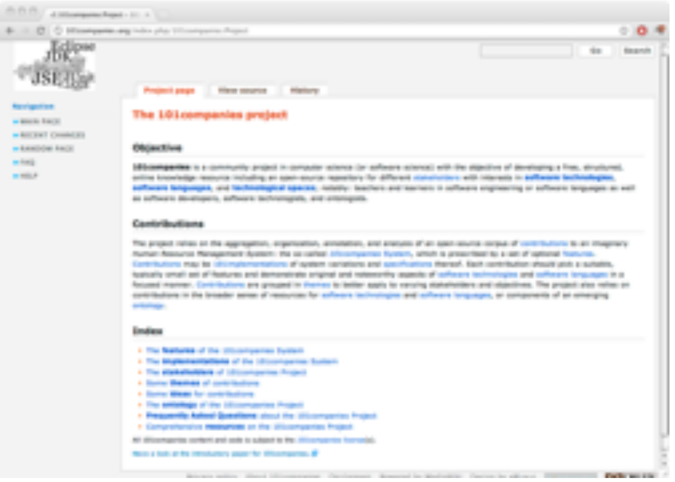
#Files per implementation	Technologies
Languages	LOC per implementation





442

The 101companies **Wiki**



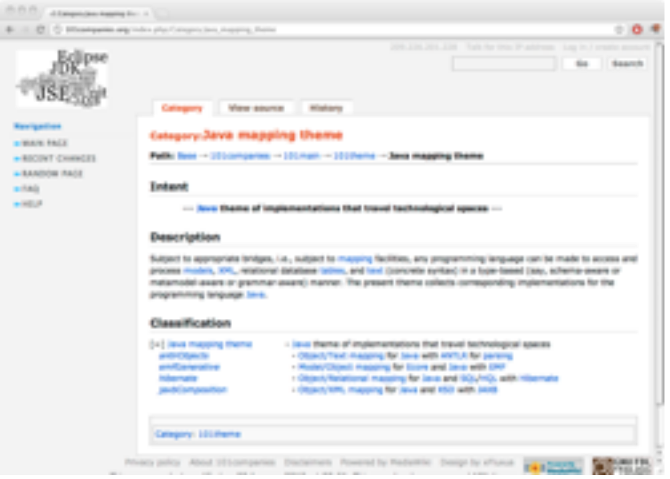
The 101companies project

Objective

Contributions

Index

Classification



Category:Java mapping theme

Intent

Description

Classification

101implementation:jaxbComposition

Path: Base → 101companies → 101main → 101theme → Java mapping theme → 101implementation:jaxbComposition

Github: jaxbComposition

This is an implementation in the 101companies software corpus.

Contents [hide]
1 Headline
2 Languages
3 Technologies
4 Features
5 Motivation
6 Illustration
7 Architecture
8 Usage
9 Contributors
10 Citations

Headline

— Object/XML mapping for Java and XSD with JAXB —

Languages

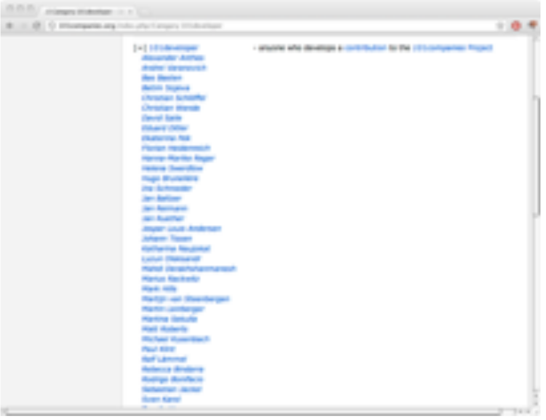
Technologies

Features

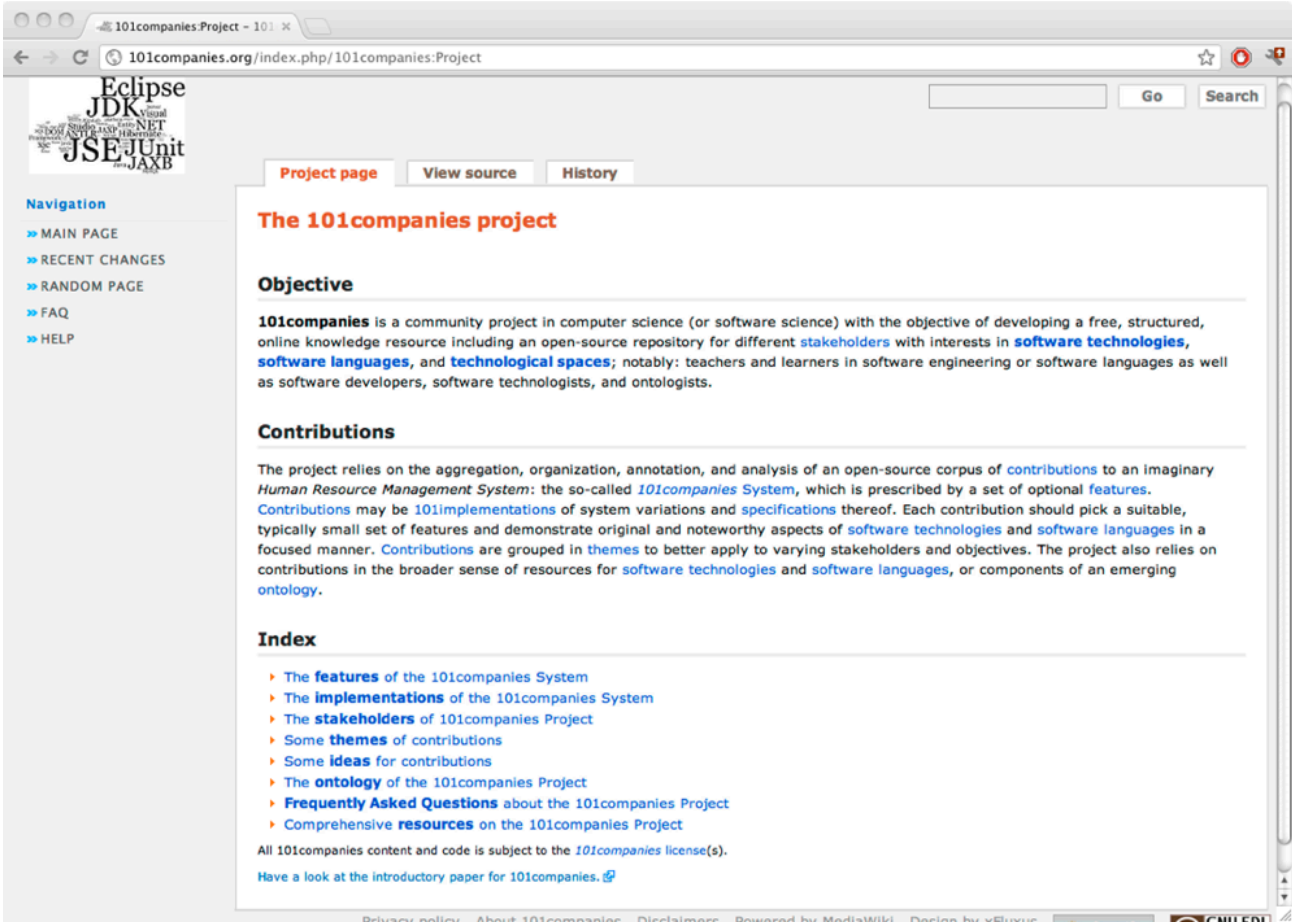
Motivation

Illustration

Architecture



© 2012, 101companies



The 101companies project

Objective

101companies is a community project in computer science (or software science) with the objective of developing a free, structured, online knowledge resource including an open-source repository for different stakeholders with interests in **software technologies**, **software languages**, and **technological spaces**; notably: teachers and learners in software engineering or software languages as well as software developers, software technologists, and ontologists.

Contributions

The project relies on the aggregation, organization, annotation, and analysis of an open-source corpus of contributions to an imaginary *Human Resource Management System*: the so-called *101companies* System, which is prescribed by a set of optional features. Contributions may be 101implementations of system variations and specifications thereof. Each contribution should pick a suitable, typically small set of features and demonstrate original and noteworthy aspects of software technologies and software languages in a focused manner. Contributions are grouped in themes to better apply to varying stakeholders and objectives. The project also relies on contributions in the broader sense of resources for software technologies and software languages, or components of an emerging ontology.

Index

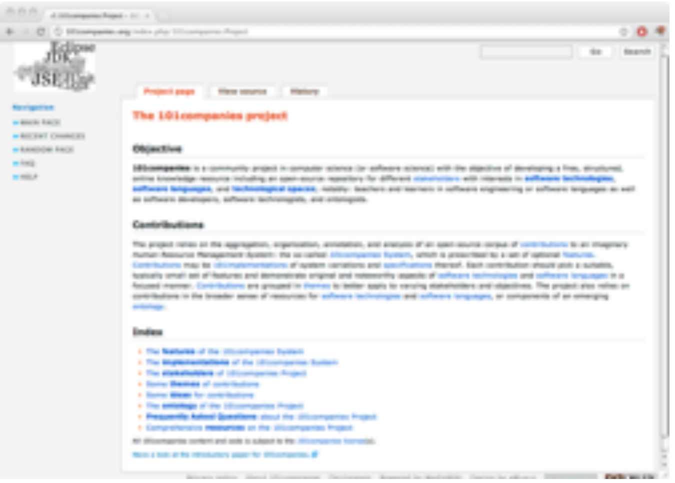
- The **features** of the 101companies System
- The **implementations** of the 101companies System
- The **stakeholders** of 101companies Project
- Some **themes** of contributions
- Some **ideas** for contributions
- The **ontology** of the 101companies Project
- **Frequently Asked Questions** about the 101companies Project
- Comprehensive **resources** on the 101companies Project

All 101companies content and code is subject to the *101companies* license(s).

Have a look at the introductory paper for 101companies.

Privacy policy About 101companies Disclaimers Powered by MediaWiki Design by vFluxus

The 101companies **Wiki**



Project page | View source | History

The 101companies project

Objective

101companies is a community project in computer science (or software science) with the objective of developing a free, structured, online knowledge resource including an open-source repository for different stakeholders with interests in software technologies, software languages, and technological spaces; notably: teachers and learners in software engineering or software languages as well as software developers, software technologists, and ontologists.

Contributions

The project relies on the aggregation, organization, annotation, and analysis of an open-source corpus of contributions to an imaginary Human Resource Management System: the so-called 101companies System, which is prescribed by a set of optional features. Contributions may be 101implementations of system variations and specifications thereof. Each contribution should pick a suitable, typically small set of features and demonstrate original and noteworthy aspects of software technologies and software languages in a focused manner. Contributions are grouped in themes to better apply to varying stakeholders and objectives. The project also relies on contributions in the broader sense of resources for software technologies and software languages, or components of an emerging ontology.

Index

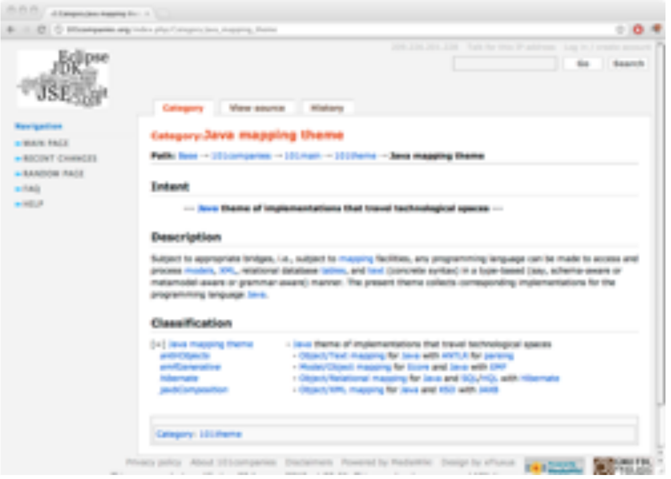
- The features of the 101companies System
- The implementations of the 101companies System
- The stakeholders of 101companies Project
- Some themes of contributions
- Some ideas for contributions
- The ontology of the 101companies Project
- Frequently Asked Questions about the 101companies Project
- Comprehensive resources on the 101companies Project

All 101companies content and code is subject to the 101companies license(s).

Have a look at the introductory paper for 101companies.

Classification

[−] 101feature	- All features of the 101companies System
[−] Behavioral 101feature	
Data export	- An export operation for company data
Data import	- An import operation for company data
Logging	- Logging of salary changes
Structure-driven query	- A query for the depth of department nesting
Type-driven query	- A query for totaling the salaries of all employees
Type-driven transformation	- A transformation for a salary cut
Visualization	- The visualization of company data
[+] Meta 101feature	
[−] Quality 101feature	
[+] 101design	- Design qualities of the 101companies System
[−] 101execution	- Execution qualities of the 101companies System
Access control	- Access control for company data
Data parallelism	- Data parallelism for operations on company data
Distribution	- Distribution of company data and operations
Persistence	- Persistence for company data
Reliability	- Reliability of the system services
Serialization	- Serialization for company data
Task parallelism	- Task parallelism for operations on company data
[−] Structural 101feature	
Global invariant	- A constraint on salaries within the company hierarchy
Graph structure	- An association between mentees and mentors
Many-to-many	- A friend relationship between employees
Tree structure	- Tree-like structure of companies and departments
Type hierarchy	- A common base type for departments and employees
[+] UI 101feature	



Category | View source | History

Category:Java mapping theme

Path: Base → 101companies → 101main → 101theme → Java mapping theme

Intent

— Java theme of implementations that travel technological spaces —

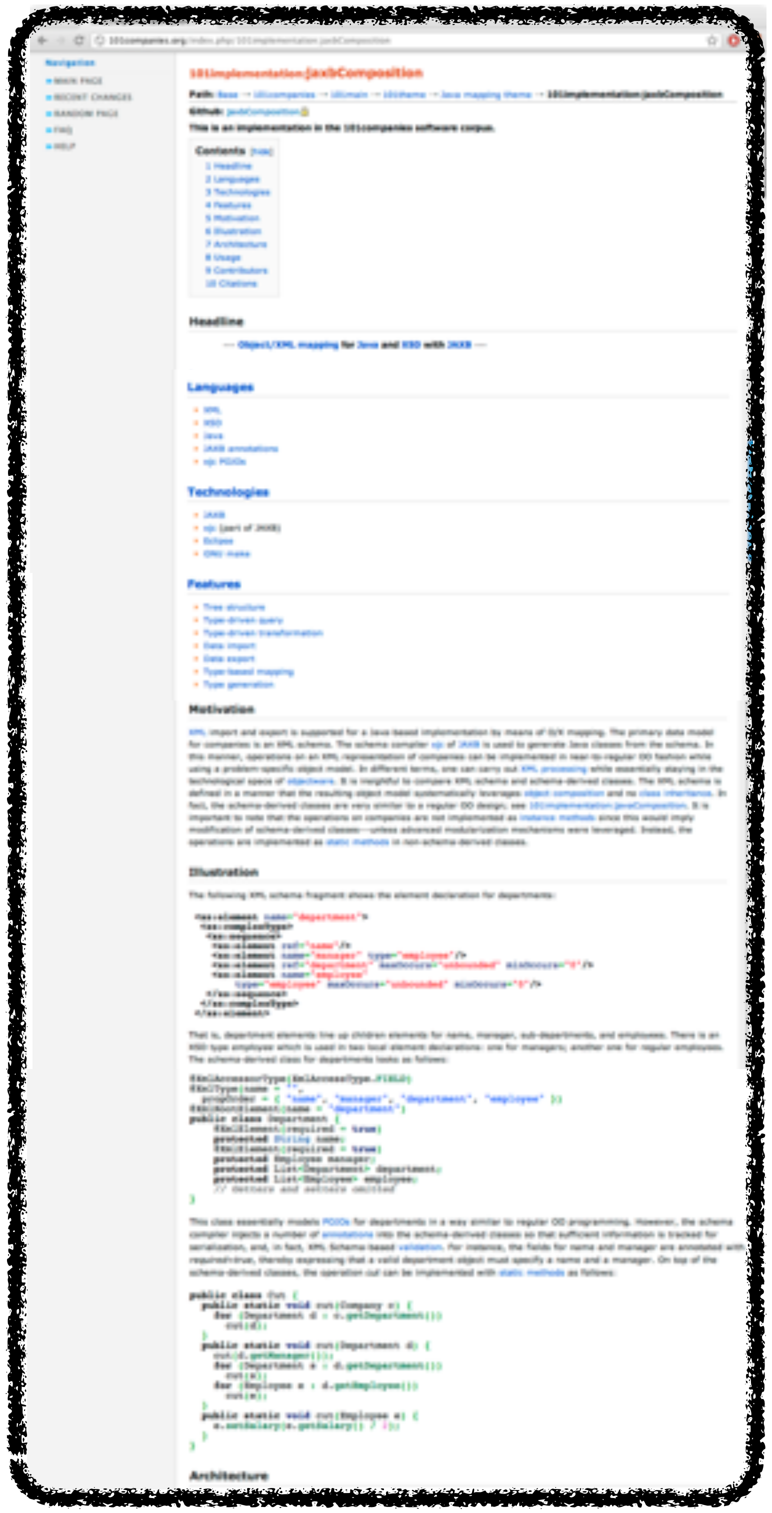
Description

Subject to appropriate bridges, i.e., subject to mapping facilities, any programming language can be made to access and process models, XML, relational database tables, and text (concrete syntax) in a type-based (say, schema-aware or metamodel-aware or grammar-aware) manner. The present theme collects corresponding implementations for the programming language Java.

Classification

[+] Java mapping theme	- Java theme of implementations that travel technological spaces
antlrObjects	- Object/Text mapping for Java with ANTLR for parsing
emfGenerative	- Model/Object mapping for Ecore and Java with EMF
hibernate	- Object/Relational mapping for Java and SQL/HQL with Hibernate
jaxbComposition	- Object/XML mapping for Java and XSD with JAXB

Category: 101theme



101implementation:jaxbComposition

Path: Base → 101companies → 101main → 101theme → Java mapping theme → 101implementation:jaxbComposition

Github: jaxbComposition

This is an implementation in the 101companies software corpus.

Contents [hide]
1 Headline
2 Languages
3 Technologies
4 Features
5 Motivation
6 Illustration
7 Architecture
8 Usage
9 Contributors
10 Citations

Headline

— Object/XML mapping for Java and XSD with JAXB —

Languages

- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs

Technologies

- JAXB
- xjc (part of JAXB)
- Eclipse
- GNU make

Features

- Tree structure
- Type-driven query
- Type-driven transformation
- Data import
- Data export
- Type-based mapping
- Type generation

Motivation

XML import and export is supported for a Java-based implementation by means of O/X mapping. The primary data model for companies is an XML schema. The schema compiler xjc of JAXB is used to generate Java classes from the schema. In this manner, operations on an XML representation of companies can be implemented in near-to-regular OO fashion while using a problem-specific object model. In different terms, one can carry out XML processing while essentially staying in the technological space of objectware. It is insightful to compare XML schema and schema-derived classes. The XML schema is defined in a manner that the resulting object model systematically leverages object composition and no class inheritance. In fact, the schema-derived classes are very similar to a regular OO design; see 101implementation:javaComposition. It is important to note that the operations on companies are not implemented as instance methods since this would imply modification of schema-derived classes—unless advanced modularization mechanisms were leveraged. Instead, the operations are implemented as static methods in non-schema-derived classes.

Illustration

The following XML schema fragment shows the element declaration for departments:

```
<xs:element name="department">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element name="manager" type="employee"/>
      <xs:element ref="department" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element name="employee"
          type="employee" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

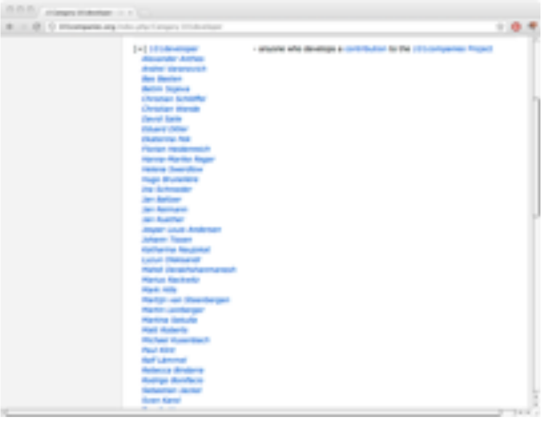
That is, department elements line up children elements for name, manager, sub-departments, and employees. There is an XSD type employee which is used in two local element declarations: one for managers; another one for regular employees. The schema-derived class for departments looks as follows:

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "",
  propOrder = { "name", "manager", "department", "employee" })
@XmlRootElement(name = "department")
public class Department {
    @XmlElement(required = true)
    protected String name;
    @XmlElement(required = true)
    protected Employee manager;
    protected List<Department> department;
    protected List<Employee> employee;
    // getters and setters omitted
}
```

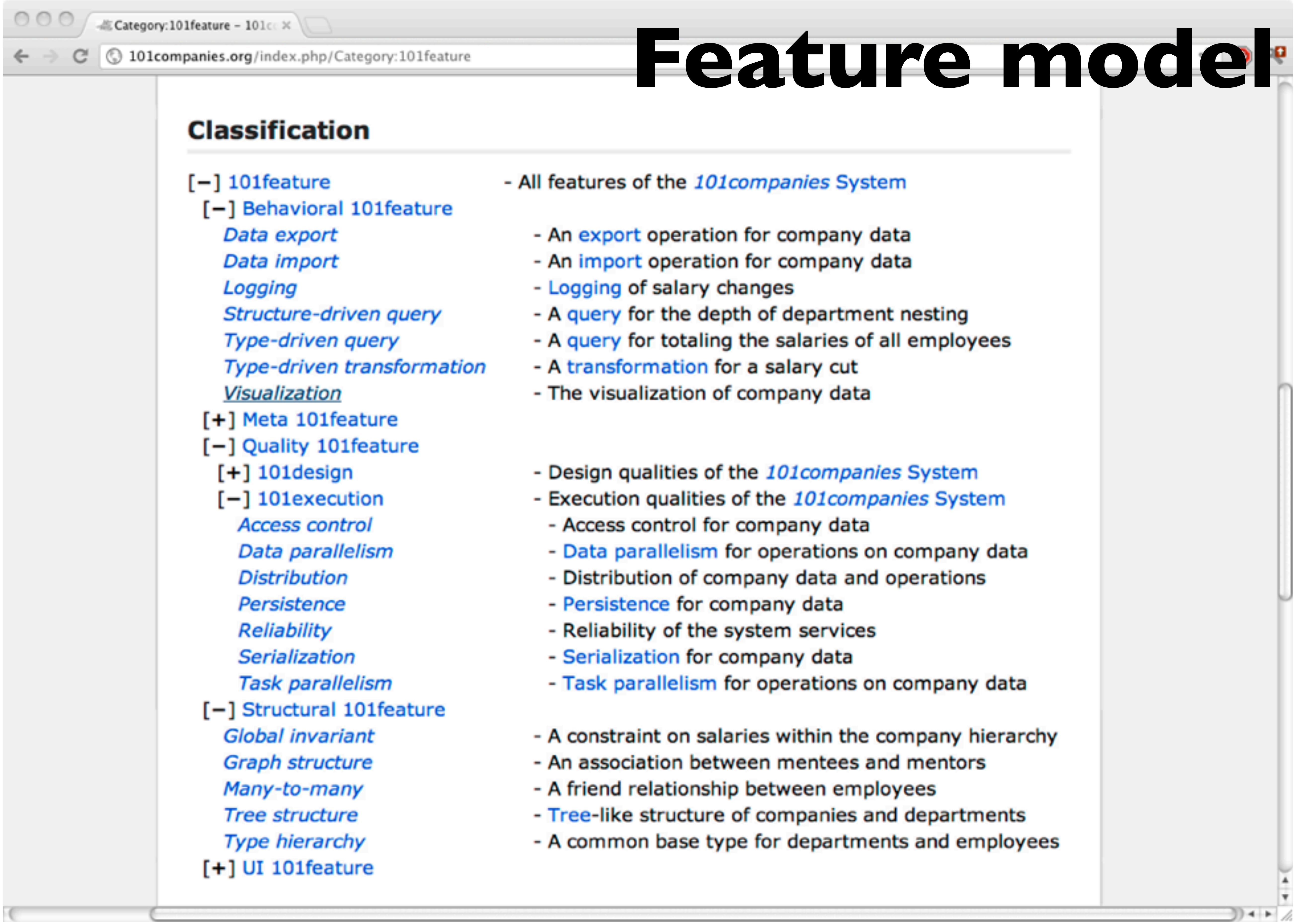
[illegible]

```
public class Cut {
  public static void cut(Company c) {
    for (Department d : c.getDepartment())
      cut(d);
  }
  public static void cut(Department d) {
    cut(d.getManager());
    for (Department s : d.getDepartment())
      cut(s);
    for (Employee e : d.getEmployee())
      cut(e);
  }
  public static void cut(Employee e) {
    e.setSalary(e.getSalary() / 2);
  }
}
```

Architecture

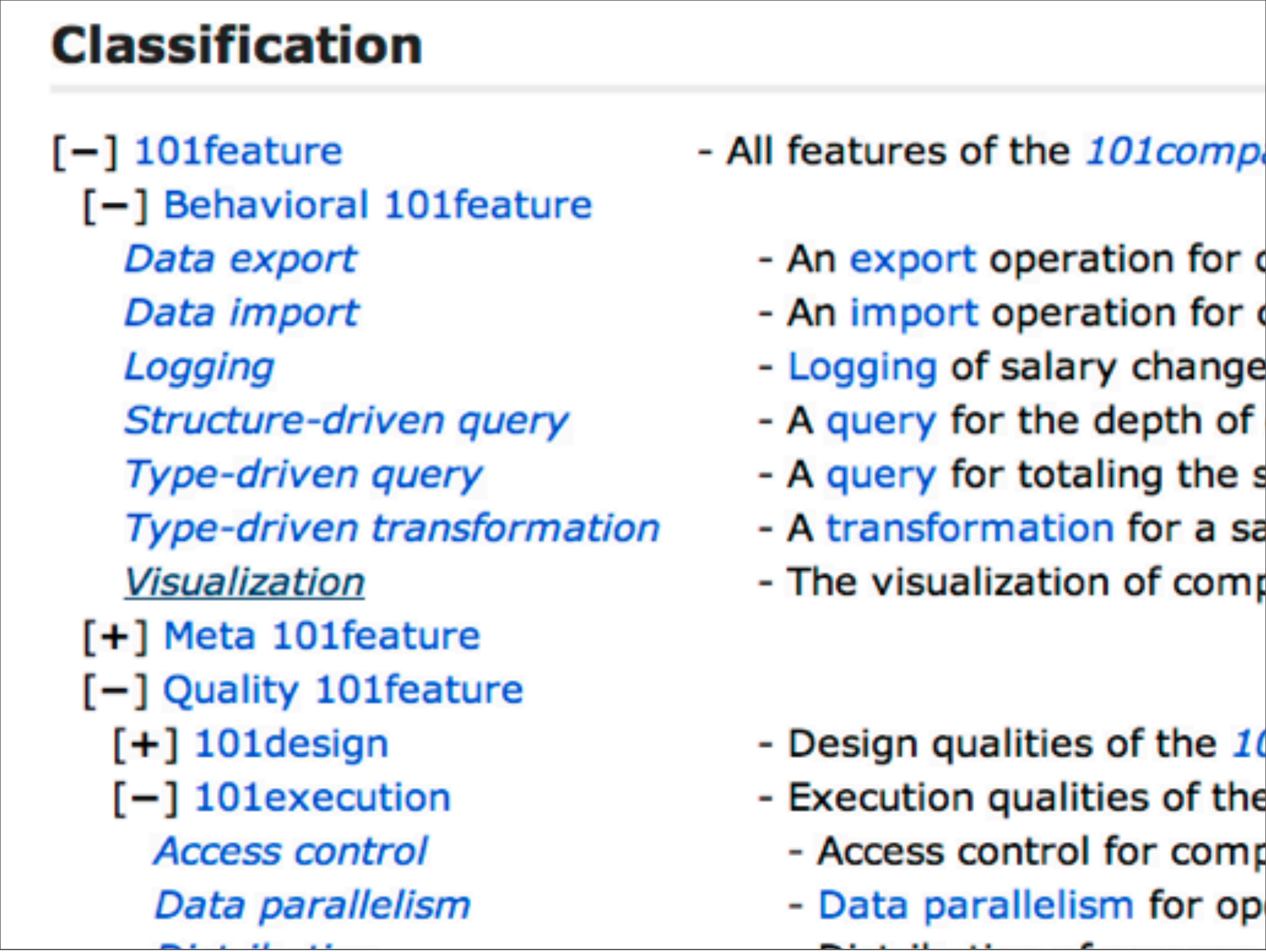


Feature model


Category:101feature – 101c
101companies.org/index.php/Category:101feature

Classification

[–] 101feature	- All features of the *101companies* System
[–] Behavioral 101feature	
Data export	- An export operation for company data
Data import	- An import operation for company data
Logging	- Logging of salary changes
Structure-driven query	- A query for the depth of department nesting
Type-driven query	- A query for totaling the salaries of all employees
Type-driven transformation	- A transformation for a salary cut
Visualization	- The visualization of company data
[+] Meta 101feature	
[–] Quality 101feature	
[+] 101design	- Design qualities of the *101companies* System
[–] 101execution	- Execution qualities of the *101companies* System
Access control	- Access control for company data
Data parallelism	- Data parallelism for operations on company data
Distribution	- Distribution of company data and operations
Persistence	- Persistence for company data
Reliability	- Reliability of the system services
Serialization	- Serialization for company data
Task parallelism	- Task parallelism for operations on company data
[–] Structural 101feature	
Global invariant	- A constraint on salaries within the company hierarchy
Graph structure	- An association between mentees and mentors
Many-to-many	- A friend relationship between employees
Tree structure	- Tree-like structure of companies and departments
Type hierarchy	- A common base type for departments and employees
[+] UI 101feature	



Classification

[−] 101feature - All features of the *101comp*

[−] Behavioral 101feature

- *Data export* - An export operation for
- *Data import* - An import operation for
- *Logging* - Logging of salary change
- *Structure-driven query* - A query for the depth of
- *Type-driven query* - A query for totaling the s
- *Type-driven transformation* - A transformation for a sa
- *Visualization* - The visualization of com

[+] Meta 101feature

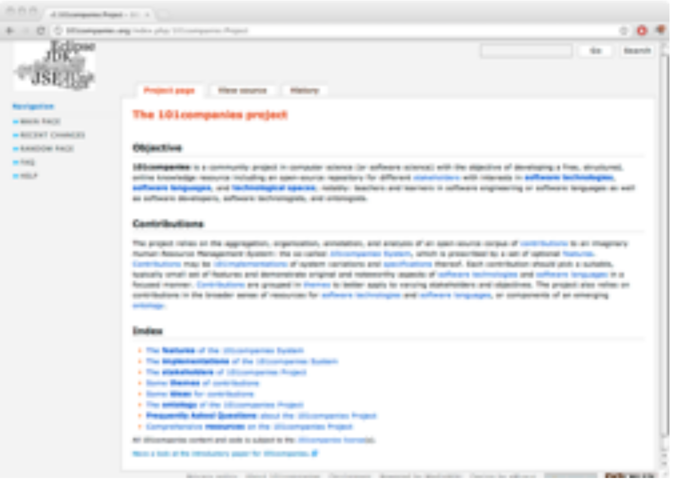
[−] Quality 101feature

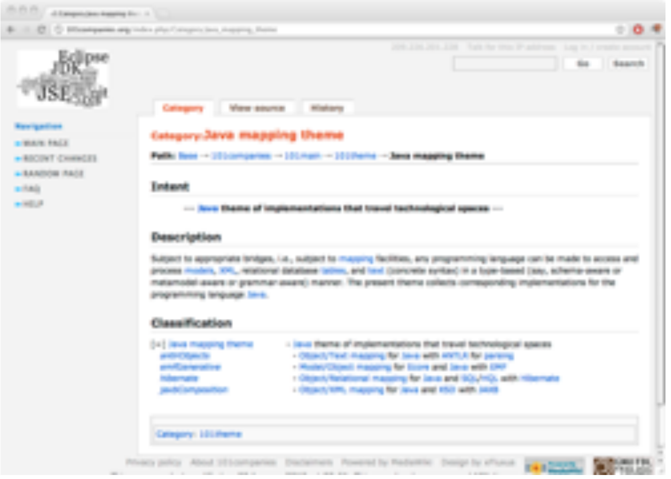
[+] 101design - Design qualities of the *1*

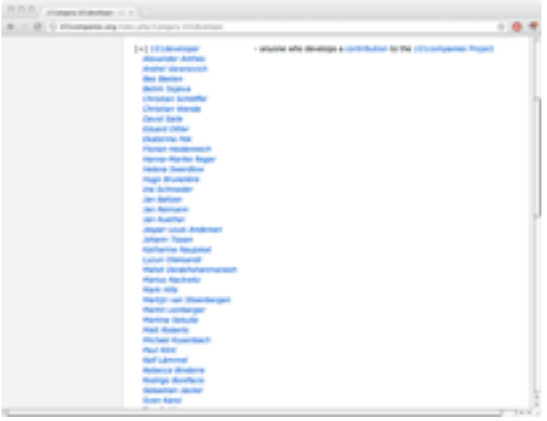
[−] 101execution - Execution qualities of the

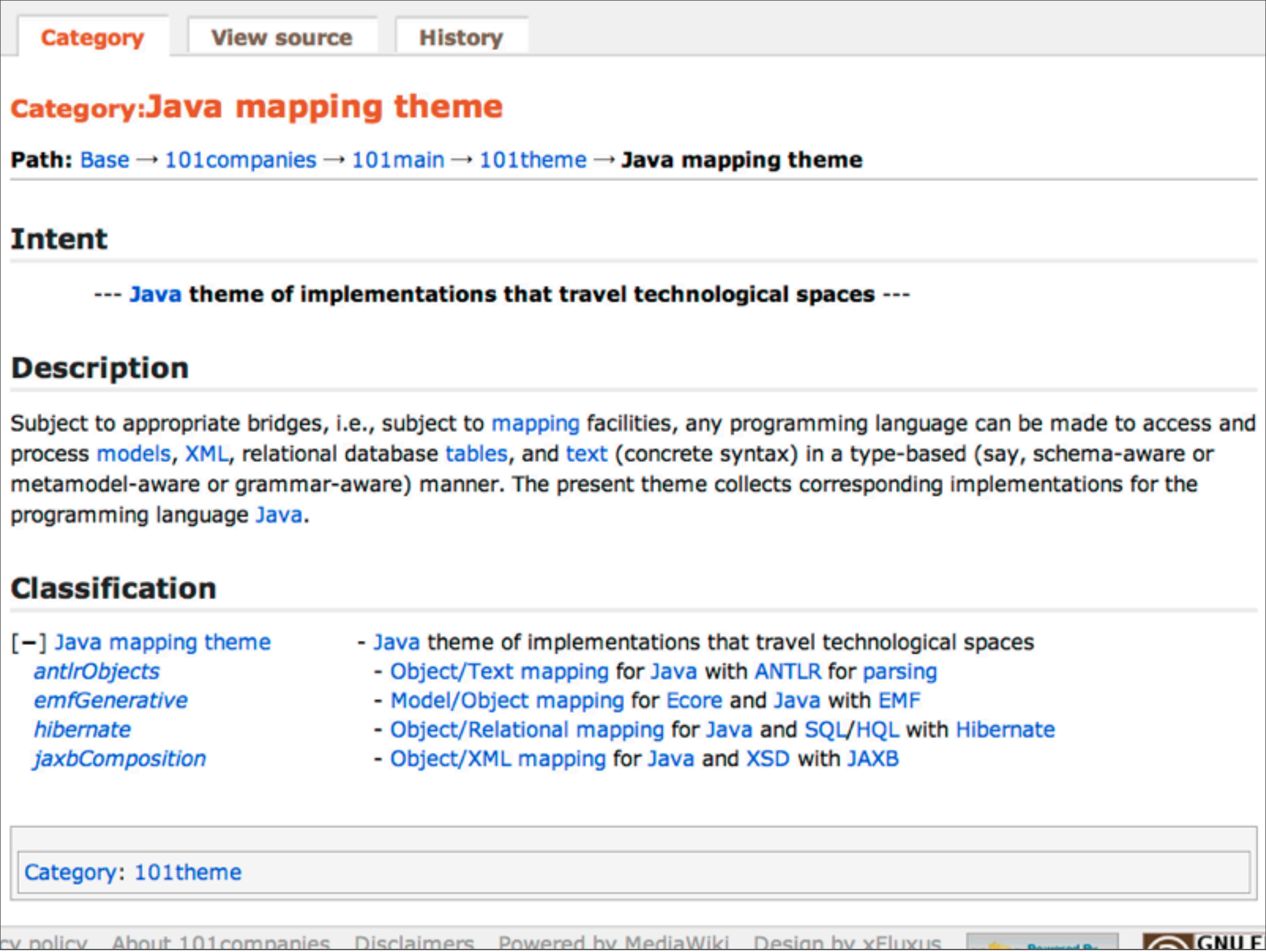
- *Access control* - Access control for com
- *Data parallelism* - Data parallelism for op

The 101companies **Wiki**









Category:Java mapping theme

Path: Base → 101companies → 101main → 101theme → **Java mapping theme**

Intent

--- Java theme of implementations that travel technological spaces ---

Description

Subject to appropriate bridges, i.e., subject to mapping facilities, any programming language can be made to access and process models, XML, relational database tables, and text (concrete syntax) in a type-based (say, schema-aware or metamodel-aware or grammar-aware) manner. The present theme collects corresponding implementations for the programming language Java.

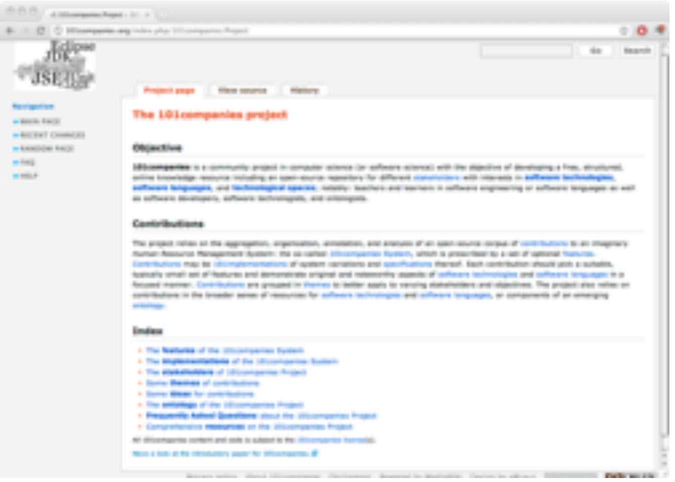
Classification

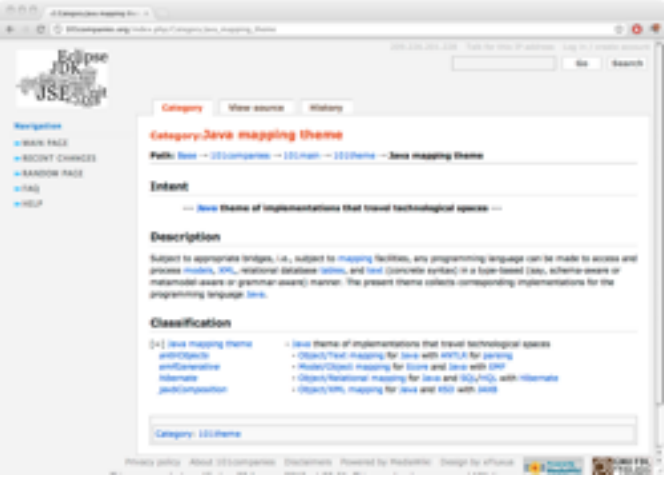
[−] Java mapping theme	- Java theme of implementations that travel technological spaces
antlrObjects	- Object/Text mapping for Java with ANTLR for parsing
emfGenerative	- Model/Object mapping for Ecore and Java with EMF
hibernate	- Object/Relational mapping for Java and SQL/HQL with Hibernate
jaxbComposition	- Object/XML mapping for Java and XSD with JAXB

Category: 101theme

cy policy About 101companies Disclaimers Powered by MediaWiki Design by xFluxus
Powered By
GNU F

The 101companies **Wiki**





101implementation:jaxbComposition

This is an implementation in the 101companies software corpus.

Contents

1 Headline
2 Languages
3 Technologies
4 Features
5 Motivation
6 Illustration
7 Architecture
8 Usage
9 Contributors
10 Citations

Headline

— Object/XML mapping for Java and XSD with JAXB —

Languages

- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs

Technologies

- JAXB
- xjc (part of JAXB)
- Eclipse
- GNU make

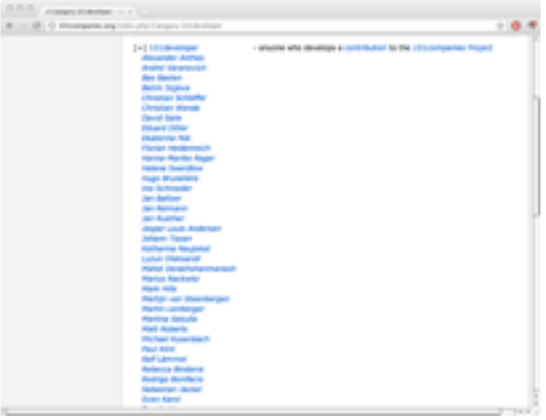
Features

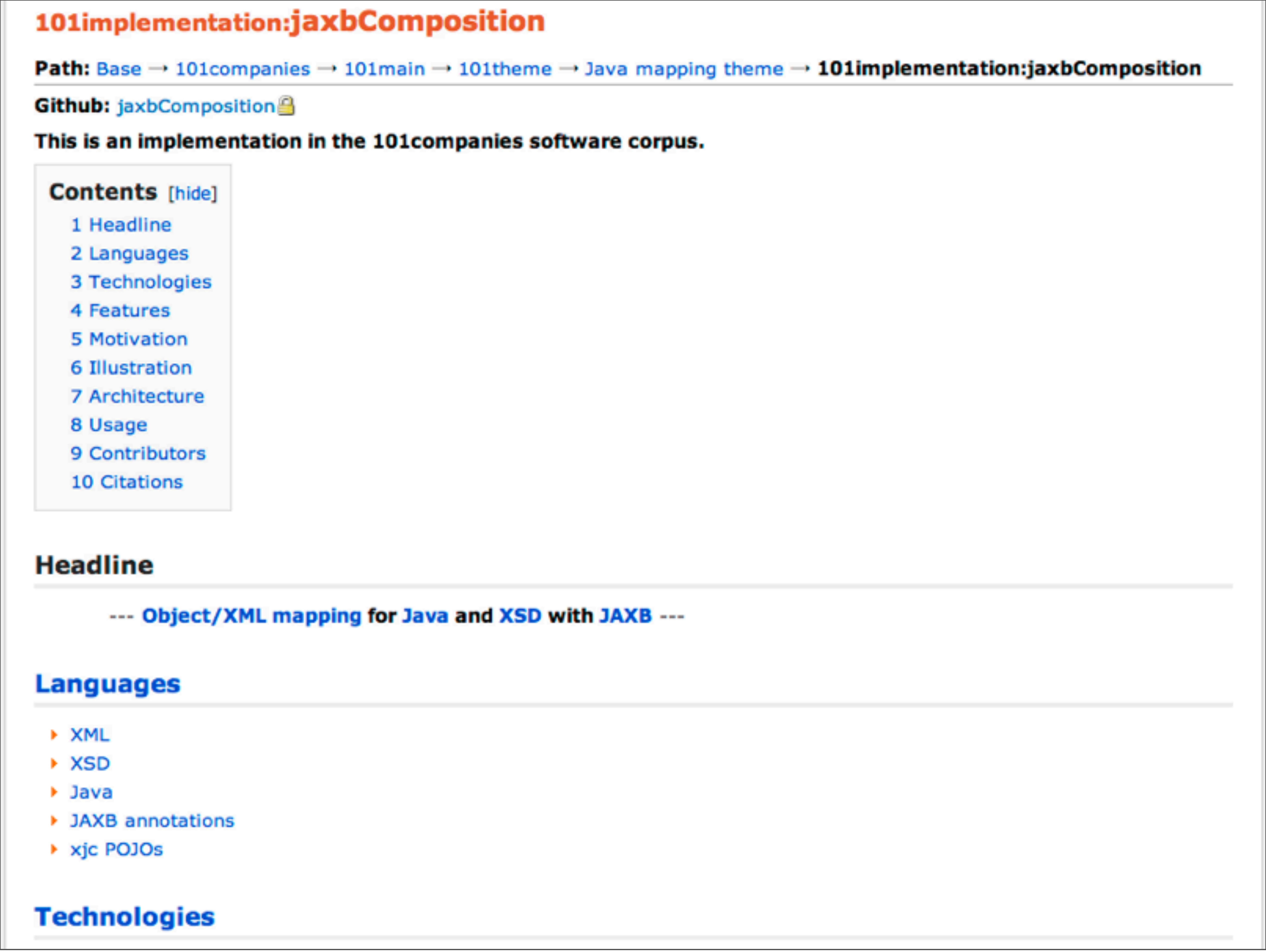
- Tree structure
- Type-driven query
- Type-driven transformation
- Data import
- Data export
- Type-based mapping
- Type generation

Motivation

Illustration

Architecture





101implementation:jaxbComposition

Path: Base → 101companies → 101main → 101theme → Java mapping theme → **101implementation:jaxbComposition**

Github: jaxbComposition

This is an implementation in the 101companies software corpus.

Contents [hide]

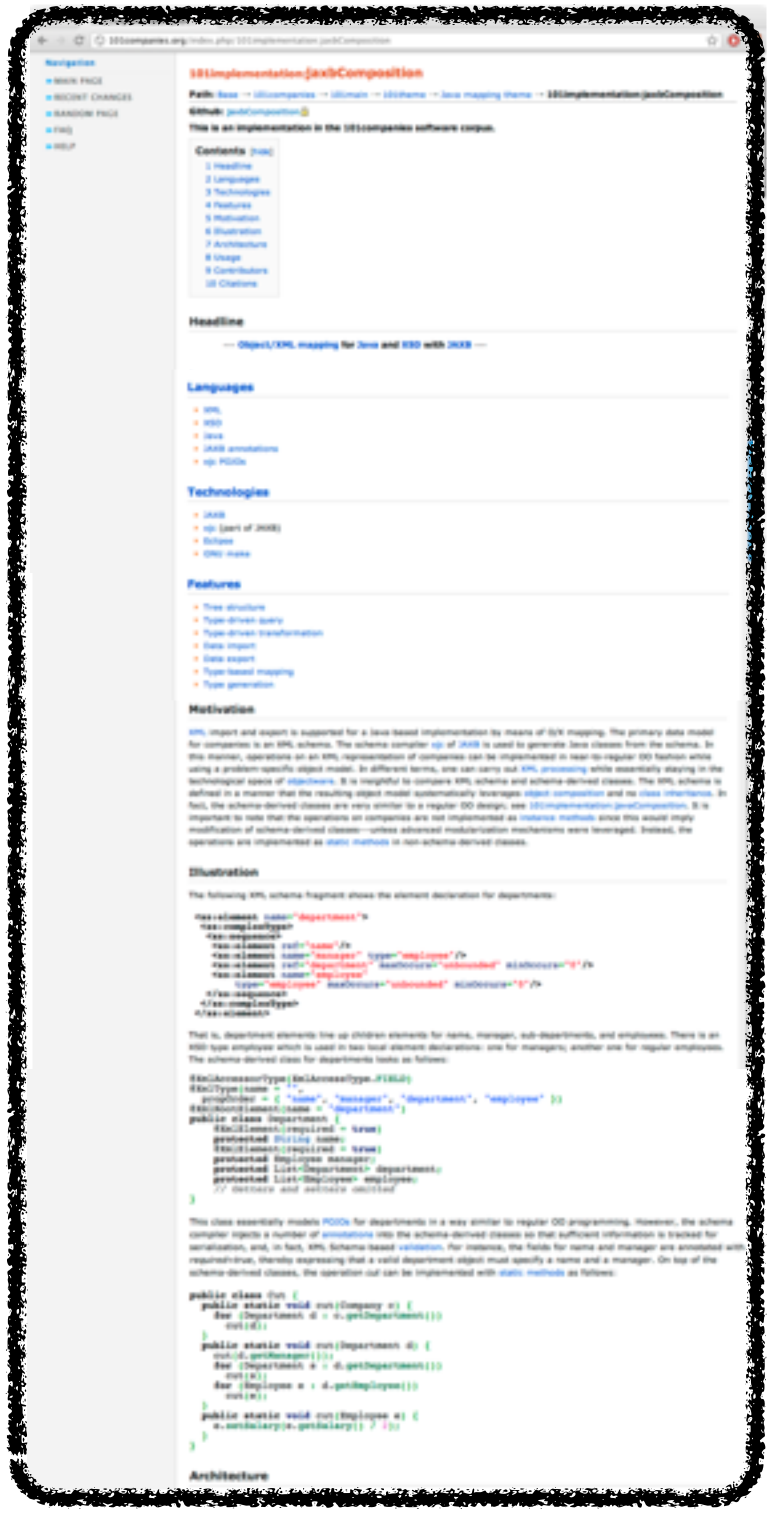
Headline

--- Object/XML mapping for Java and XSD with JAXB ---

Languages

- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs

Technologies



101companies.org/index.php/101implementation:jaxbComposition
Navigation
MAIN PAGE
RECENT CHANGES
RANDOM PAGE
FAQ
HELP

101implementation:jaxbComposition

Path: Base → 101companies → 101main → 101theme → Java mapping theme → **101implementation:jaxbComposition**

Github: jaxbComposition

This is an implementation in the 101companies software corpus.

Contents [hide]

Headline

— **Object/XML mapping for Java and XSD with JAXB** —

Languages

- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs

Technologies

- JAXB
- xjc (part of JAXB)
- Eclipse
- GNU make

Features

- Tree structure
- Type-driven query
- Type-driven transformation
- Data import
- Data export
- Type-based mapping
- Type generation

Motivation

XML import and export is supported for a Java-based implementation by means of O/X mapping. The primary data model for companies is an XML schema. The schema compiler xjc of JAXB is used to generate Java classes from the schema. In this manner, operations on an XML representation of companies can be implemented in near-to-regular OO fashion while using a problem-specific object model. In different terms, one can carry out XML processing while essentially staying in the technological space of objectware. It is insightful to compare XML schema and schema-derived classes. The XML schema is defined in a manner that the resulting object model systematically leverages object composition and no class inheritance. In fact, the schema-derived classes are very similar to a regular OO design; see 101implementation:javaComposition. It is important to note that the operations on companies are not implemented as instance methods since this would imply modification of schema-derived classes—unless advanced modularization mechanisms were leveraged. Instead, the operations are implemented as static methods in non-schema-derived classes.

Illustration

The following XML schema fragment shows the element declaration for departments:

```
<xs:element name="department">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="name"/>
   <xs:element name="manager" type="employee"/>
   <xs:element ref="department" maxOccurs="unbounded" minOccurs="0"/>
   <xs:element name="employee"
       type="employee" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

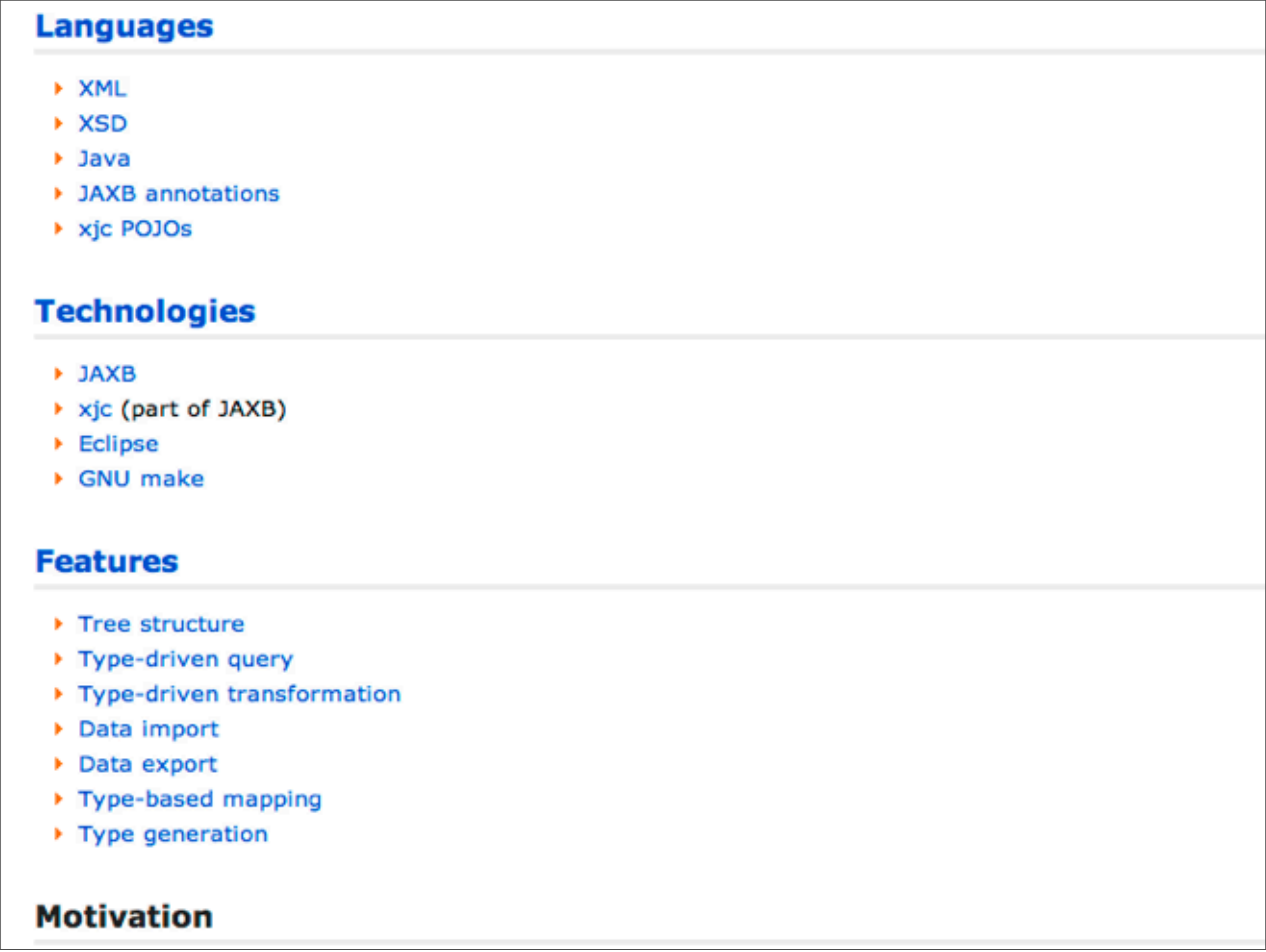
That is, department elements line up children elements for name, manager, sub-departments, and employees. There is an XSD type employee which is used in two local element declarations: one for managers; another one for regular employees. The schema-derived class for departments looks as follows:

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "",
  propOrder = { "name", "manager", "department", "employee" })
@XmlRootElement(name = "department")
public class Department {
    @XmlElement(required = true)
    protected String name;
    @XmlElement(required = true)
    protected Employee manager;
    protected List<Department> department;
    protected List<Employee> employee;
    // Getters and setters omitted
}
```

This class essentially models POJOs for departments in a way similar to regular OO programming. However, the schema compiler injects a number of annotations into the schema-derived classes so that sufficient information is tracked for serialization, and, in fact, XML Schema-based validation. For instance, the fields for name and manager are annotated with required=true, thereby expressing that a valid department object must specify a name and a manager. On top of the schema-derived classes, the operation cut can be implemented with static methods as follows:

```
public class Cut {
  public static void cut(Company c) {
    for (Department d : c.getDepartment())
      cut(d);
  }
  public static void cut(Department d) {
    cut(d.getManager());
    for (Department s : d.getDepartment())
      cut(s);
    for (Employee e : d.getEmployee())
      cut(e);
  }
  public static void cut(Employee e) {
    e.setSalary(e.getSalary() / 2);
  }
}
```

Architecture



Languages

- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs

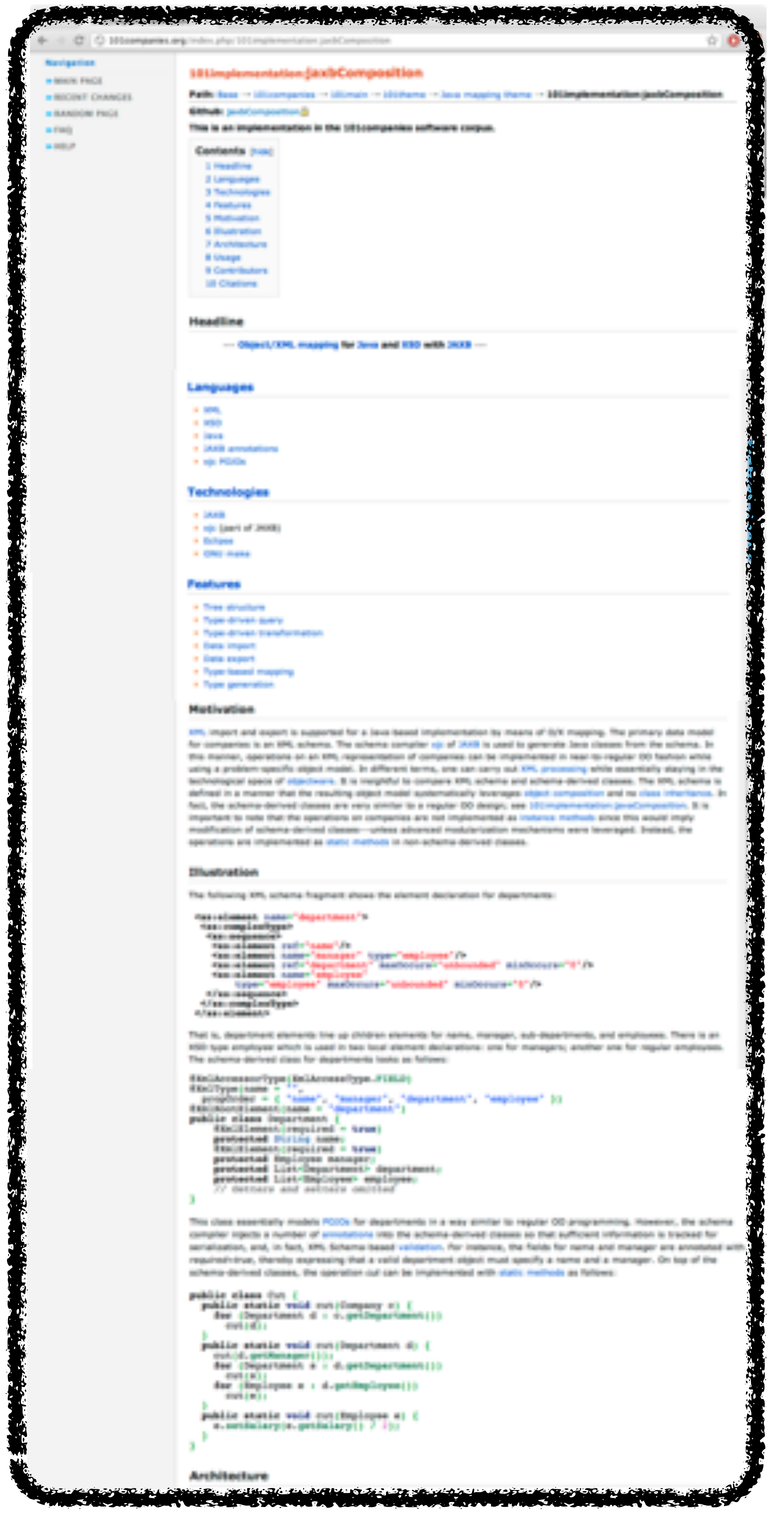
Technologies

- JAXB
- xjc (part of JAXB)
- Eclipse
- GNU make

Features

- Tree structure
- Type-driven query
- Type-driven transformation
- Data import
- Data export
- Type-based mapping
- Type generation

Motivation



101companies.org/index.php/101implementation:jaxbComposition
Navigation
MAIN PAGE
RECENT CHANGES
RANDOM PAGE
FAQ
HELP

101implementation:jaxbComposition

Path: Base → 101companies → 101main → 101theme → Java mapping theme → 101implementation:jaxbComposition

Github: jaxbComposition

This is an implementation in the 101companies software corpus.

Headline

— **Object/XML mapping for Java and XSD with JAXB** —

Languages

- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs

Technologies

- JAXB
- xjc (part of JAXB)
- Eclipse
- GNU make

Features

- Tree structure
- Type-driven query
- Type-driven transformation
- Data import
- Data export
- Type-based mapping
- Type generation

Motivation

XML import and export is supported for a Java-based implementation by means of O/X mapping. The primary data model for companies is an XML schema. The schema compiler xjc of JAXB is used to generate Java classes from the schema. In this manner, operations on an XML representation of companies can be implemented in near-to-regular OO fashion while using a problem-specific object model. In different terms, one can carry out XML processing while essentially staying in the technological space of objectware. It is insightful to compare XML schema and schema-derived classes. The XML schema is defined in a manner that the resulting object model systematically leverages object composition and no class inheritance. In fact, the schema-derived classes are very similar to a regular OO design; see 101implementation:javaComposition. It is important to note that the operations on companies are not implemented as instance methods since this would imply modification of schema-derived classes—unless advanced modularization mechanisms were leveraged. Instead, the operations are implemented as static methods in non-schema-derived classes.

Illustration

The following XML schema fragment shows the element declaration for departments:

```
<xs:element name="department">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="name"/>
   <xs:element name="manager" type="employee"/>
   <xs:element ref="department" maxOccurs="unbounded" minOccurs="0"/>
   <xs:element name="employee"
       type="employee" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

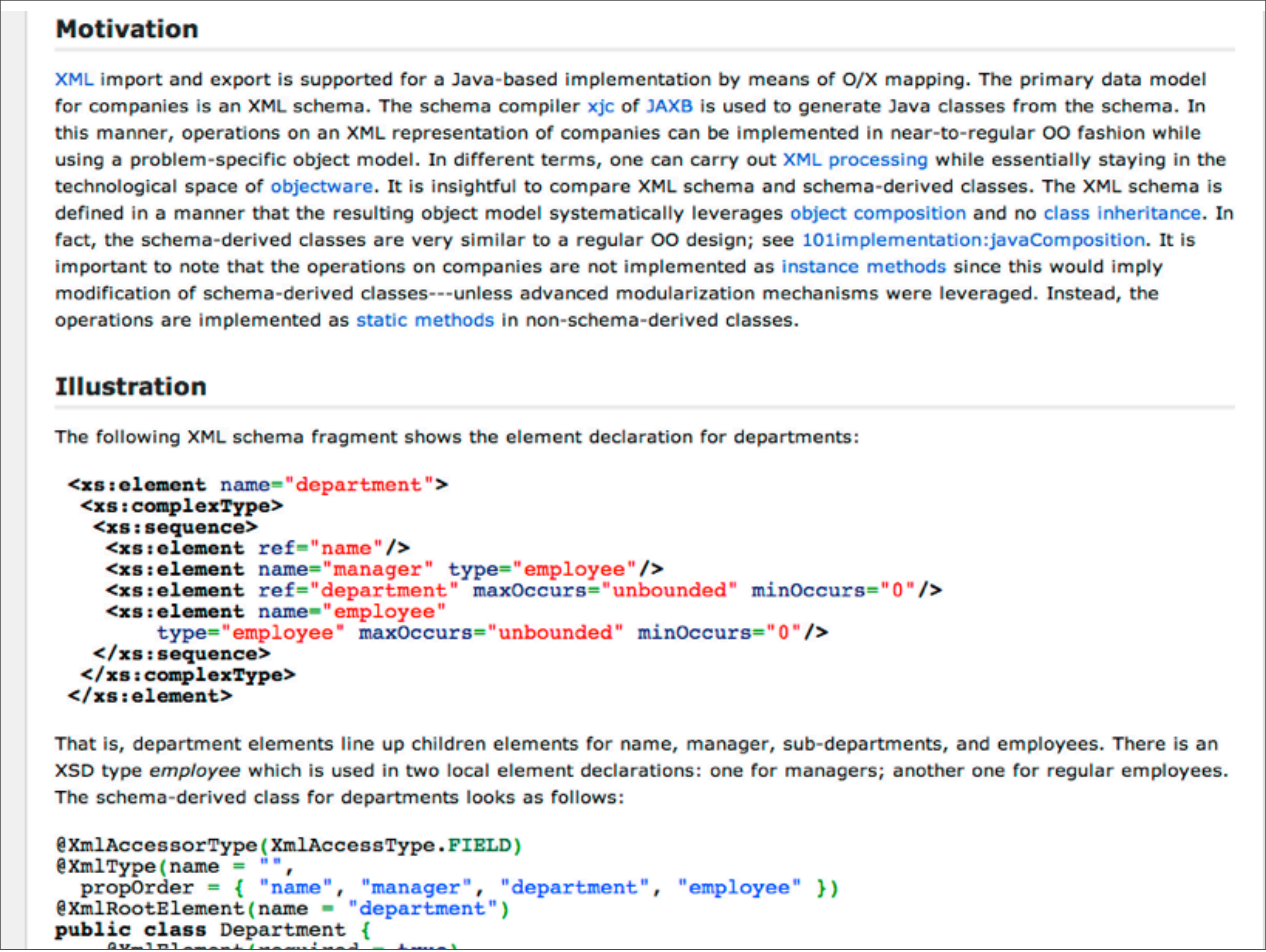
That is, department elements line up children elements for name, manager, sub-departments, and employees. There is an XSD type employee which is used in two local element declarations: one for managers; another one for regular employees. The schema-derived class for departments looks as follows:

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "",
  propOrder = { "name", "manager", "department", "employee" })
@XmlRootElement(name = "department")
public class Department {
    @XmlElement(required = true)
    protected String name;
    @XmlElement(required = true)
    protected Employee manager;
    protected List<Department> department;
    protected List<Employee> employee;
    // getters and setters omitted
}
```

This class essentially models POJOs for departments in a way similar to regular OO programming. However, the schema compiler injects a number of annotations into the schema-derived classes so that sufficient information is tracked for serialization, and, in fact, XML Schema-based validation. For instance, the fields for name and manager are annotated with required=true, thereby expressing that a valid department object must specify a name and a manager. On top of the schema-derived classes, the operation cut can be implemented with static methods as follows:

```
public class Cut {
  public static void cut(Company c) {
    for (Department d : c.getDepartment())
      cut(d);
  }
  public static void cut(Department d) {
    cut(d.getManager());
    for (Department s : d.getDepartment())
      cut(s);
    for (Employee e : d.getEmployee())
      cut(e);
  }
  public static void cut(Employee e) {
    e.setSalary(e.getSalary() / 2);
  }
}
```

Architecture



Motivation

XML import and export is supported for a Java-based implementation by means of O/X mapping. The primary data model for companies is an XML schema. The schema compiler xjc of JAXB is used to generate Java classes from the schema. In this manner, operations on an XML representation of companies can be implemented in near-to-regular OO fashion while using a problem-specific object model. In different terms, one can carry out XML processing while essentially staying in the technological space of objectware. It is insightful to compare XML schema and schema-derived classes. The XML schema is defined in a manner that the resulting object model systematically leverages object composition and no class inheritance. In fact, the schema-derived classes are very similar to a regular OO design; see 101implementation:javaComposition. It is important to note that the operations on companies are not implemented as instance methods since this would imply modification of schema-derived classes---unless advanced modularization mechanisms were leveraged. Instead, the operations are implemented as static methods in non-schema-derived classes.

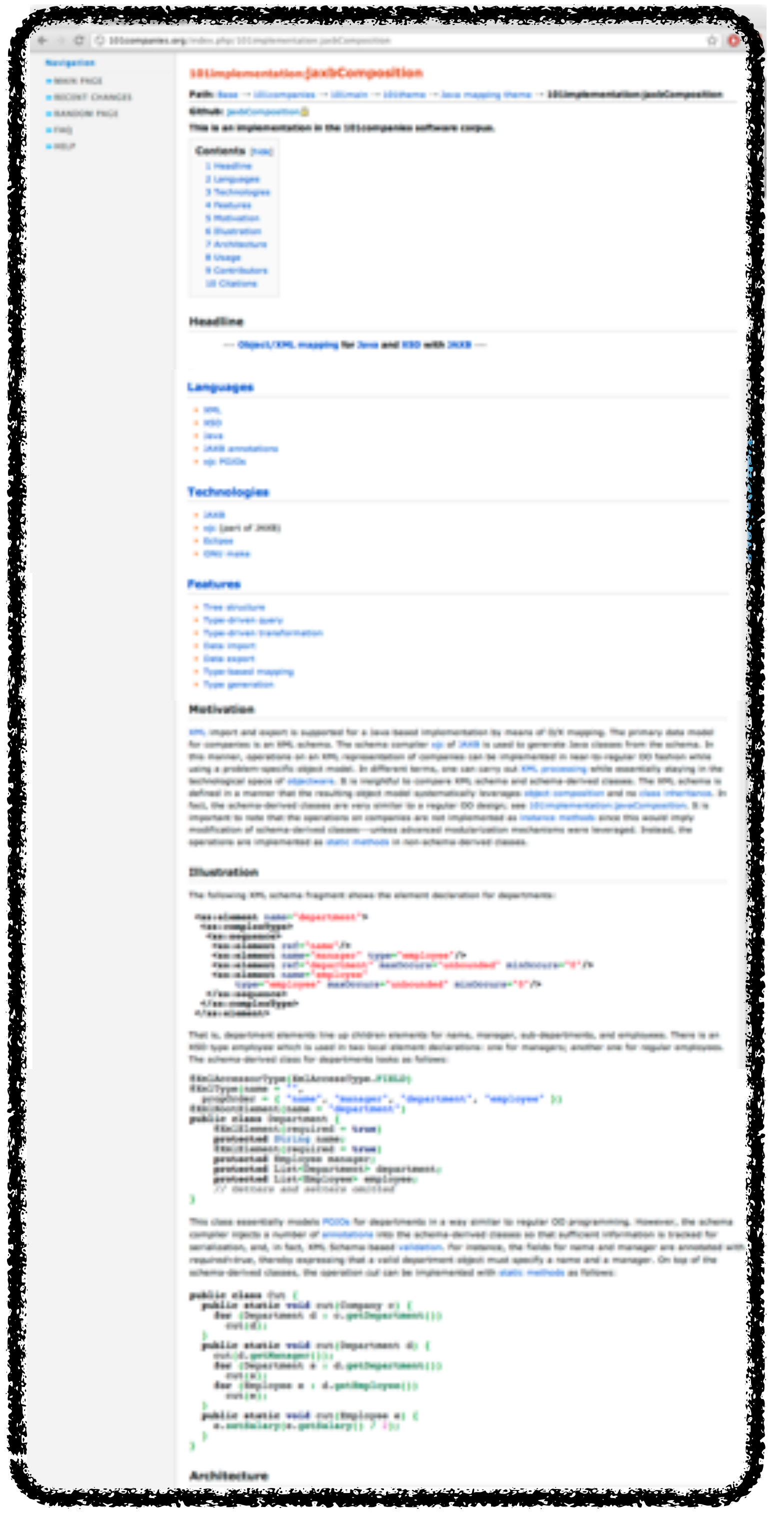
Illustration

The following XML schema fragment shows the element declaration for departments:

```
<xs:element name="department">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="name"/>
   <xs:element name="manager" type="employee"/>
   <xs:element ref="department" maxOccurs="unbounded" minOccurs="0"/>
   <xs:element name="employee"
       type="employee" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

That is, department elements line up children elements for name, manager, sub-departments, and employees. There is an XSD type *employee* which is used in two local element declarations: one for managers; another one for regular employees. The schema-derived class for departments looks as follows:

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "",
  propOrder = { "name", "manager", "department", "employee" })
@XmlRootElement(name = "department")
public class Department {
    @XmlElement(required = true)
```



101companies.org/index.php/101implementation:jaxbComposition
Navigation
MAIN PAGE
RECENT CHANGES
RANDOM PAGE
FAQ
HELP

101implementation:jaxbComposition

Path: Base → 101companies → 101main → 101theme → Java mapping theme → 101implementation:jaxbComposition

Github: jaxbComposition

This is an implementation in the 101companies software corpus.

Contents [hide]

Headline

— Object/XML mapping for Java and XSD with JAXB —

Languages

- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs

Technologies

- JAXB
- xjc (part of JAXB)
- Eclipse
- GNU make

Features

- Tree structure
- Type-driven query
- Type-driven transformation
- Data import
- Data export
- Type-based mapping
- Type generation

Motivation

XML import and export is supported for a Java-based implementation by means of O/X mapping. The primary data model for companies is an XML schema. The schema compiler xjc of JAXB is used to generate Java classes from the schema. In this manner, operations on an XML representation of companies can be implemented in near-to-regular OO fashion while using a problem-specific object model. In different terms, one can carry out XML processing while essentially staying in the technological space of objectware. It is insightful to compare XML schema and schema-derived classes. The XML schema is defined in a manner that the resulting object model systematically leverages object composition and no class inheritance. In fact, the schema-derived classes are very similar to a regular OO design; see 101implementation:javaComposition. It is important to note that the operations on companies are not implemented as instance methods since this would imply modification of schema-derived classes—unless advanced modularization mechanisms were leveraged. Instead, the operations are implemented as static methods in non-schema-derived classes.

Illustration

The following XML schema fragment shows the element declaration for departments:

```
<xs:element name="department">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="name"/>
   <xs:element name="manager" type="employee"/>
   <xs:element ref="department" maxOccurs="unbounded" minOccurs="0"/>
   <xs:element name="employee"
       type="employee" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

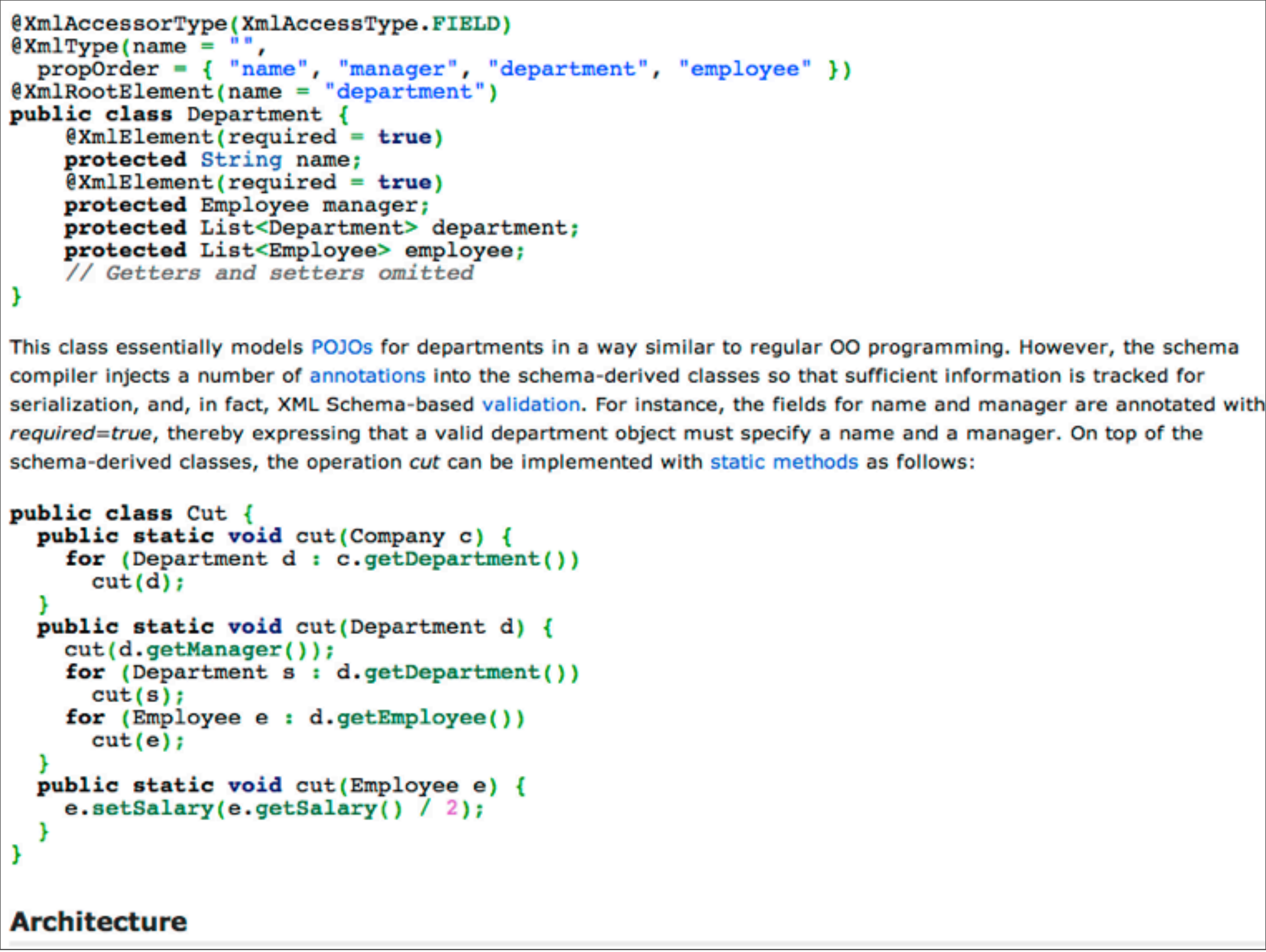
That is, department elements line up children elements for name, manager, sub-departments, and employees. There is an XSD type employee which is used in two local element declarations: one for managers; another one for regular employees. The schema-derived class for departments looks as follows:

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "",
  propOrder = { "name", "manager", "department", "employee" })
@XmlRootElement(name = "department")
public class Department {
    @XmlElement(required = true)
    protected String name;
    @XmlElement(required = true)
    protected Employee manager;
    protected List<Department> department;
    protected List<Employee> employee;
    // getters and setters omitted
}
```

This class essentially models POJOs for departments in a way similar to regular OO programming. However, the schema compiler injects a number of annotations into the schema-derived classes so that sufficient information is tracked for serialization, and, in fact, XML Schema-based validation. For instance, the fields for name and manager are annotated with required=true, thereby expressing that a valid department object must specify a name and a manager. On top of the schema-derived classes, the operation cut can be implemented with static methods as follows:

```
public class Cut {
  public static void cut(Company c) {
    for (Department d : c.getDepartment())
      cut(d);
  }
  public static void cut(Department d) {
    cut(d.getManager());
    for (Department s : d.getDepartment())
      cut(s);
    for (Employee e : d.getEmployee())
      cut(e);
  }
  public static void cut(Employee e) {
    e.setSalary(e.getSalary() / 2);
  }
}
```

Architecture



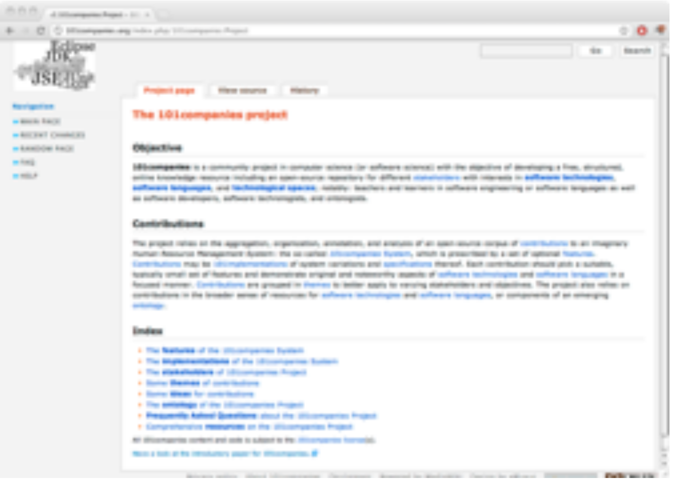
```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "",
  propOrder = { "name", "manager", "department", "employee" })
@XmlRootElement(name = "department")
public class Department {
    @XmlElement(required = true)
    protected String name;
    @XmlElement(required = true)
    protected Employee manager;
    protected List<Department> department;
    protected List<Employee> employee;
    // Getters and setters omitted
}
```

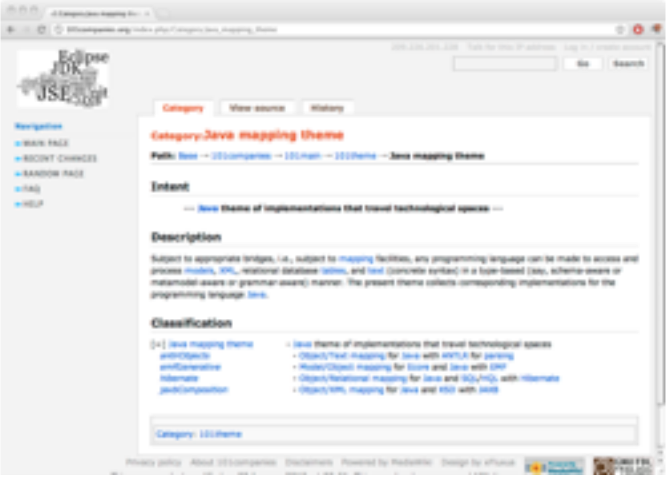
This class essentially models POJOs for departments in a way similar to regular OO programming. However, the schema compiler injects a number of annotations into the schema-derived classes so that sufficient information is tracked for serialization, and, in fact, XML Schema-based validation. For instance, the fields for name and manager are annotated with *required=true*, thereby expressing that a valid department object must specify a name and a manager. On top of the schema-derived classes, the operation *cut* can be implemented with static methods as follows:

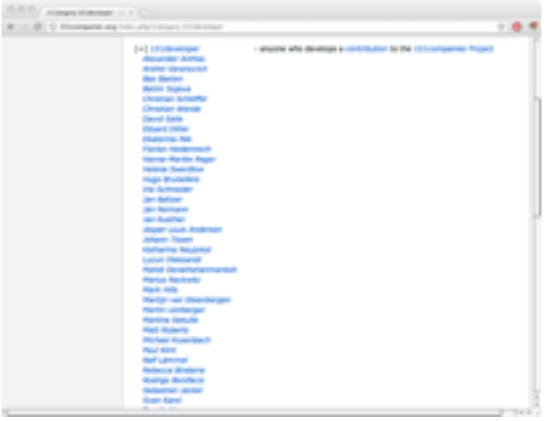
```
public class Cut {
  public static void cut(Company c) {
    for (Department d : c.getDepartment())
      cut(d);
  }
  public static void cut(Department d) {
    cut(d.getManager());
    for (Department s : d.getDepartment())
      cut(s);
    for (Employee e : d.getEmployee())
      cut(e);
  }
  public static void cut(Employee e) {
    e.setSalary(e.getSalary() / 2);
  }
}
```

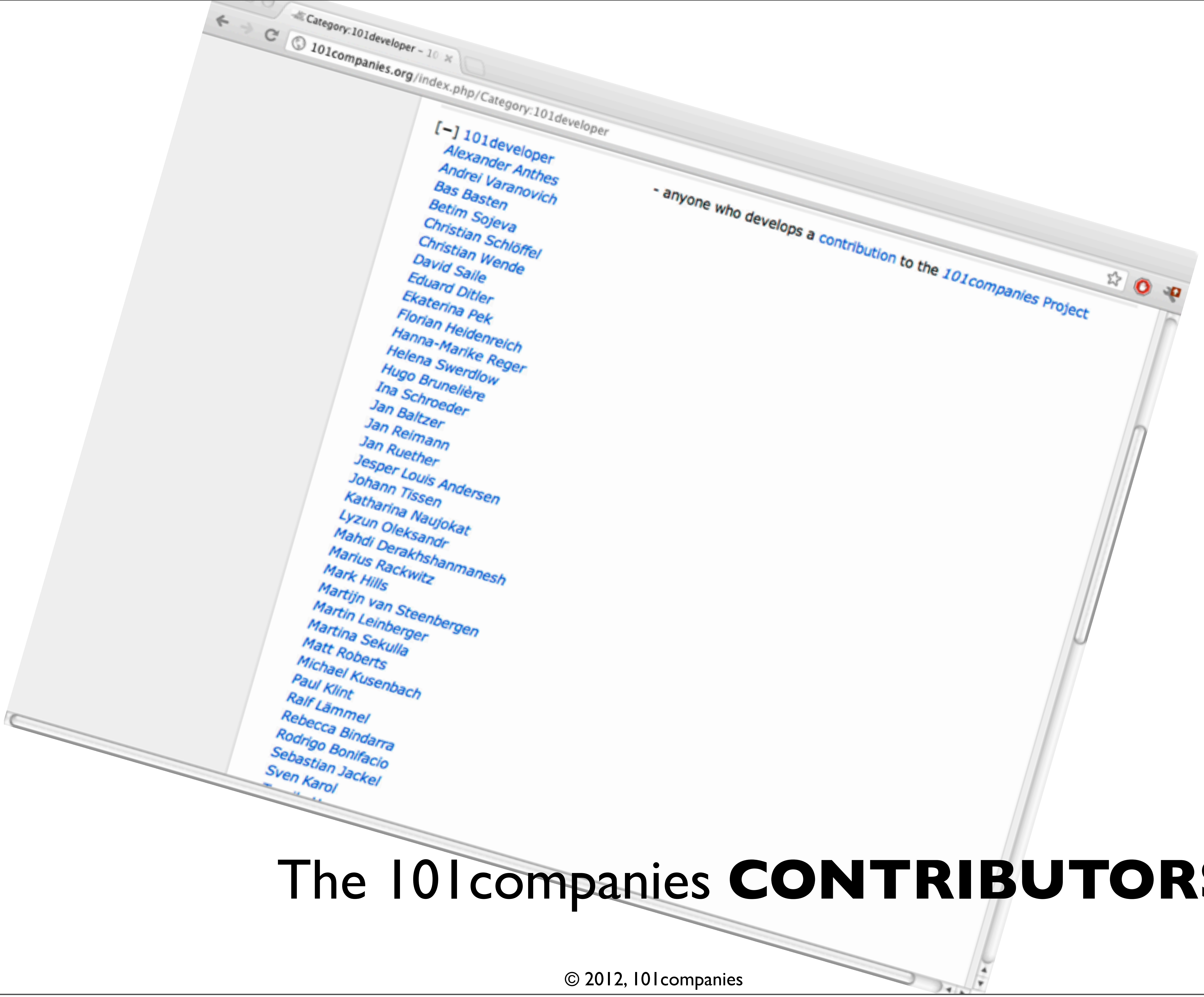
Architecture

The 101companies **Wiki**









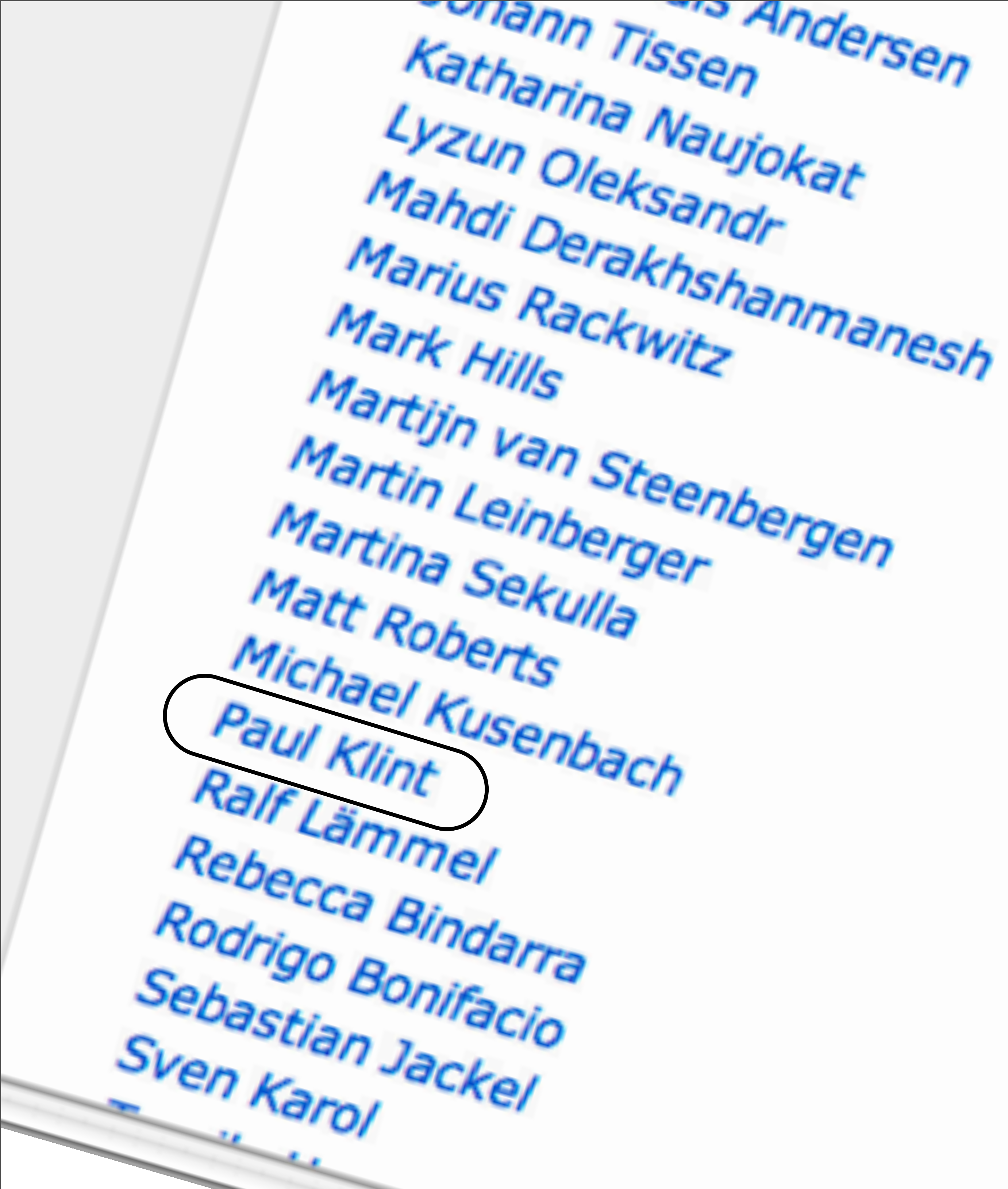
Category:101developer - 10
101companies.org/index.php/Category:101developer

[−] 101developer

- anyone who develops a contribution to the 101companies Project

Alexander Anthes
Andrei Varanovich
Bas Basten
Betim Sojeva
Christian Schlöffel
Christian Wende
David Saile
Eduard Ditler
Ekaterina Pek
Florian Heidenreich
Hanna-Marike Reger
Helena Swerdlow
Hugo Brunelière
Ina Schroeder
Jan Baltzer
Jan Reimann
Jan Ruether
Jesper Louis Andersen
Johann Tissen
Katharina Naujokat
Lyzun Oleksandr
Mahdi Derakhshanmanesh
Marius Rackwitz
Mark Hills
Martijn van Steenbergen
Martin Leinberger
Martina Sekulla
Matt Roberts
Michael Kusenbach
Paul Klint
Ralf Lämmel
Rebecca Bindarra
Rodrigo Bonifacio
Sebastian Jackel
Sven Karol

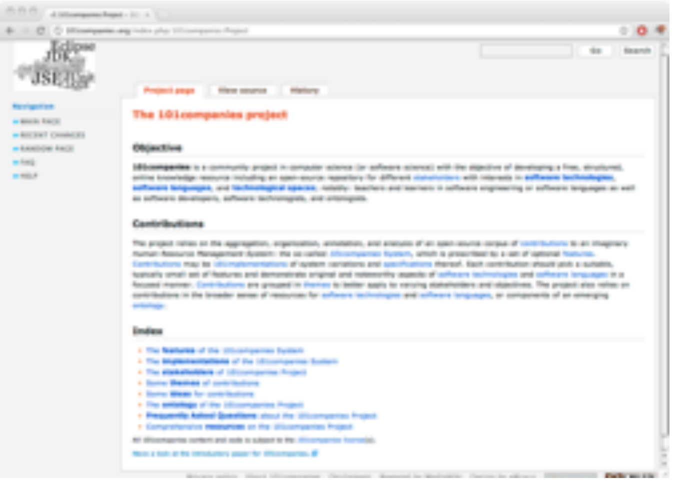
The 101companies **CONTRIBUTORS**



s Andersen
ohann Tissen
Katharina Naujokat
Lyzun Oleksandr
Mahdi Derakhshanmanesh
Marius Rackwitz
Mark Hills
Martijn van Steenbergen
Martin Leinberger
Martina Sekulla
Matt Roberts
Michael Kusenbach
Paul Klint
Ralf Lämmel
Rebecca Bindarra
Rodrigo Bonifacio
Sebastian Jackel
Sven Karol

The 101companies **CONTRIBUTORS**

The 101companies **Wiki**



Project page | View source | History

Navigation
- MAIN PAGE
- RECENT CHANGES
- RANDOM PAGE
- FAQ
- HELP

The 101companies project

Objective

101companies is a community project in computer science (or software science) with the objective of developing a free, structured, online knowledge resource including an open-source repository for different stakeholders with interests in **software technologies**, **software languages**, and **technological spaces**; notably: teachers and learners in software engineering or software languages as well as software developers, software technologists, and ontologists.

Contributions

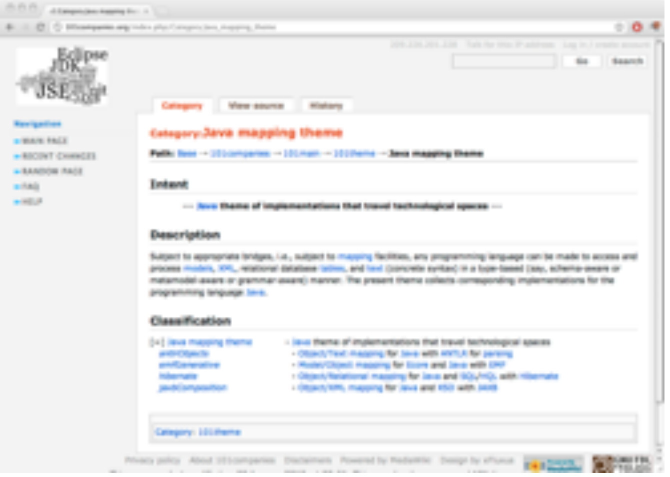
[illegible]

Index

- The **features** of the 101companies System
- The **implementations** of the 101companies System
- The **stakeholders** of 101companies Project
- Some **themes** of contributions
- Some **ideas** for contributions
- The **ontology** of the 101companies Project
- **Frequently Asked Questions** about the 101companies Project
- Comprehensive **resources** on the 101companies Project

Classification

[−] 101feature - All features of the 101companies System
[−] Behavioral 101feature
Data export - An export operation for company data
Data import - An import operation for company data
Logging - Logging of salary changes
Structure-driven query - A query for the depth of department nesting
Type-driven query - A query for totaling the salaries of all employees
Type-driven transformation - A transformation for a salary cut
Visualization - The visualization of company data
[+] Meta 101feature
[−] Quality 101feature
[+] 101design - Design qualities of the 101companies System
[−] 101execution - Execution qualities of the 101companies System
Access control - Access control for company data
Data parallelism - Data parallelism for operations on company data
Distribution - Distribution of company data and operations
Persistence - Persistence for company data
Reliability - Reliability of the system services
Serialization - Serialization for company data
Task parallelism - Task parallelism for operations on company data
[−] Structural 101feature
Global invariant - A constraint on salaries within the company hierarchy
Graph structure - An association between mentees and mentors
Many-to-many - A friend relationship between employees
Tree structure - Tree-like structure of companies and departments
Type hierarchy - A common base type for departments and employees
[+] UI 101feature



Category | View source | History

Category:Java mapping theme

Path: Base → 101companies → 101main → 101theme → **Java mapping theme**

Intent

— Java **theme of implementations that travel technological spaces** —

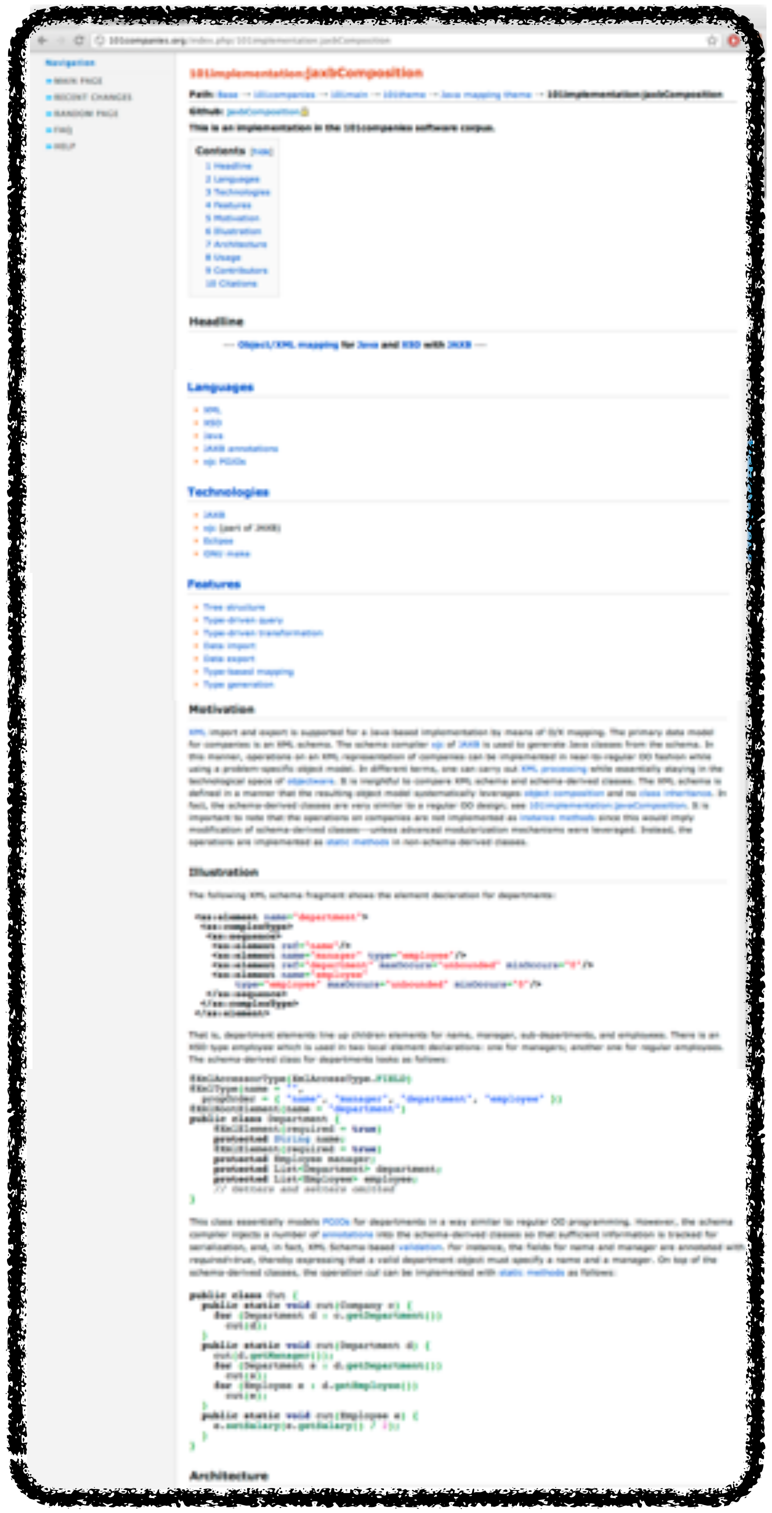
Description

Subject to appropriate bridges, i.e., subject to mapping facilities, any programming language can be made to access and process models, XML, relational database tables, and text (concrete syntax) in a type-based (say, schema-aware or metamodel-aware or grammar-aware) manner. The present theme collects corresponding implementations for the programming language Java.

Classification

[−] Java mapping theme - Java theme of implementations that travel technological spaces
antlrObjects - Object/Text mapping for Java with ANTLR for parsing
emfGenerative - Model/Object mapping for Ecore and Java with EMF
hibernate - Object/Relational mapping for Java and SQL/HQL with Hibernate
jaxbComposition - Object/XML mapping for Java and XSD with JAXB

Category: 101theme



Navigation
- MAIN PAGE
- RECENT CHANGES
- RANDOM PAGE
- FAQ
- HELP

101implementation:jaxbComposition

Path: Base → 101companies → 101main → 101theme → Java mapping theme → **101implementation:jaxbComposition**

Github: jaxbComposition

This is an implementation in the 101companies software corpus.

Contents [hide]
1 Headline
2 Languages
3 Technologies
4 Features
5 Motivation
6 Illustration
7 Architecture
8 Usage
9 Contributors
10 Citations

Headline

— **Object/XML mapping for Java and XSD with JAXB** —

Languages

- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs

Technologies

- JAXB
- xjc (part of JAXB)
- Eclipse
- GNU make

Features

- Tree structure
- Type-driven query
- Type-driven transformation
- Data import
- Data export
- Type-based mapping
- Type generation

Motivation

[illegible]

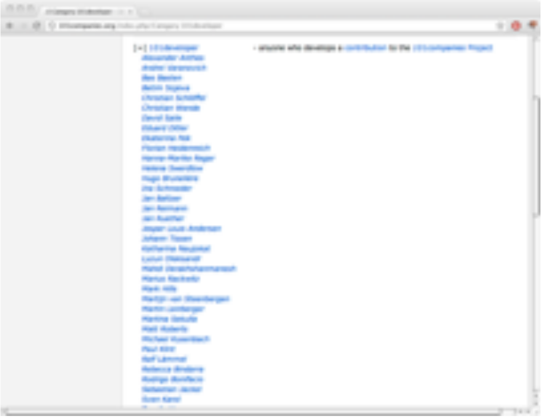
Illustration

The following XML schema fragment shows the element declaration for departments:

```
[illegible]
```

[illegible]

Architecture



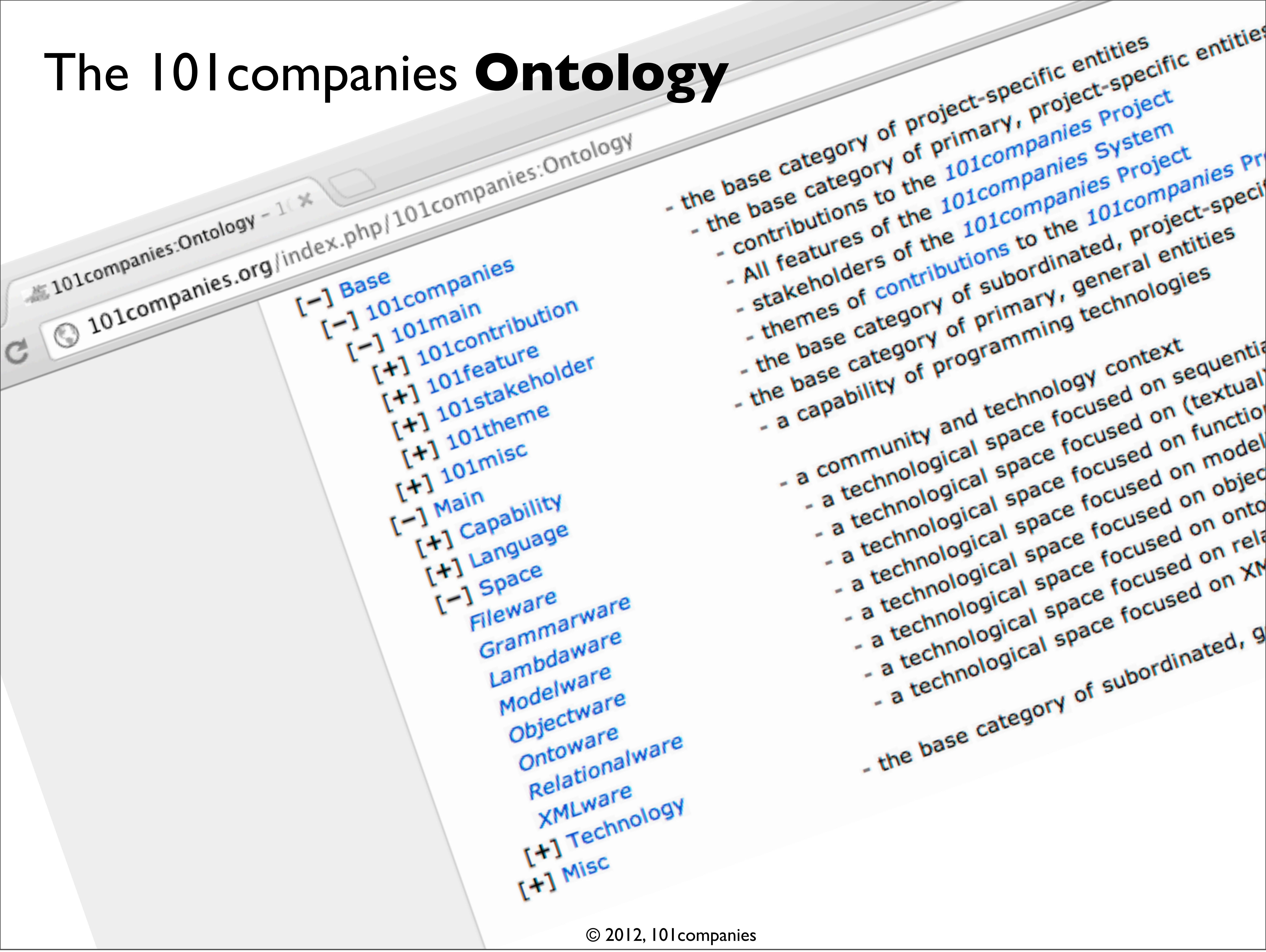
[−] 101developer - anyone who develops a contribution to the 101companies Project

[illegible]

[−] Base - the base category of project-specific entities
[−] 101companies - the base category of primary, project-specific entities
[−] 101main
[+] 101contribution - contributions to the 101companies Project
[+] 101feature - All features of the 101companies System
[+] 101stakeholder - stakeholders of the 101companies Project
[+] 101theme - themes of contributions to the 101companies Project
[+] 101misc - the base category of subordinated, project-specific entities
[−] Main - the base category of primary, general entities
[+] Capability - a capability of programming technologies
[+] Language
[−] Space - a community and technology context
Fileware - a technological space focused on sequential and indexed files
Grammarware - a technological space focused on (textual) language processing
Lambdaware - a technological space focused on functions and functional programming
Modelware - a technological space focused on modeling and model-driven engineering
Objectware - a technological space focused on objects and OO programming
Ontoware - a technological space focused on ontologies and knowledge engineering
Relationalware - a technological space focused on relational databases
XMLware - a technological space focused on XML representation and XML processing
[+] Technology
[+] Misc - the base category of subordinated, general entities

© 2012, 101companies

The 101companies **Ontology**



101companies.org/index.php/101companies:Ontology

[−] Base
  [−] 101companies
    [−] 101main
      [−] 101contribution
        [+] 101feature
        [+] 101stakeholder
        [+] 101theme
        [+] 101misc
      [+] Main
    [−] Capability
      [+] Language
      [+] Space
      [−] Fileware
        Grammarware
        Lambdaware
        Modelware
        Objectware
        Ontoware
        Relationalware
        XMLware
      [+] Technology
    [+] Misc

- the base category of project-specific entities
- the base category of primary, project-specific entities
- contributions to the 101companies Project
- All features of the 101companies System
- stakeholders of the 101companies Project
- themes of contributions to the 101companies Pr
- the base category of subordinated, project-speci
- the base category of primary, general entities
- a capability of programming technologies

- a community and technology context
- a technological space focused on sequentia
- a technological space focused on (textual
- a technological space focused on functio
- a technological space focused on model
- a technological space focused on objec
- a technological space focused on onto
- a technological space focused on rela
- a technological space focused on XM

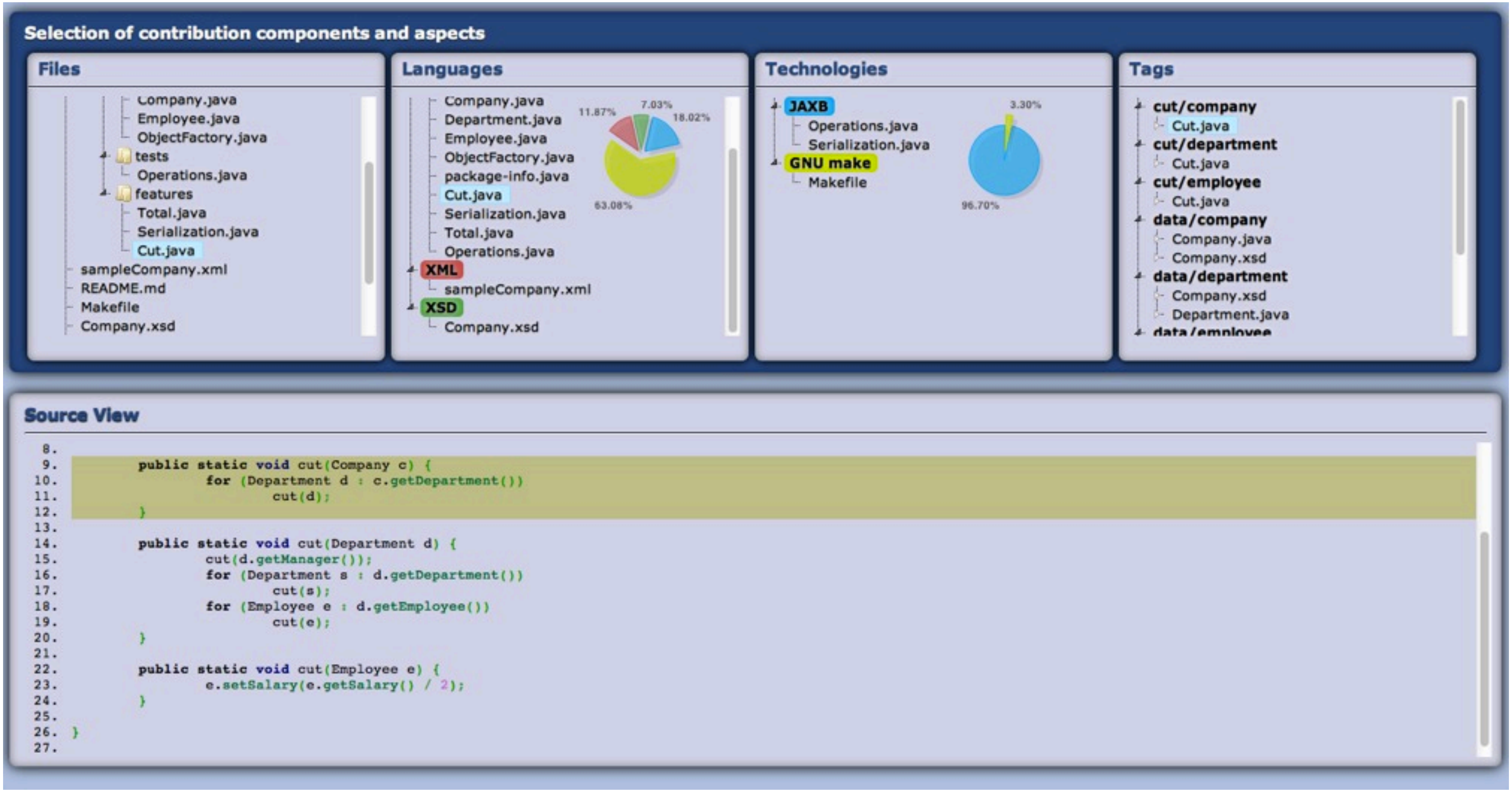
- the base category of subordinated, g

© 2012, 101companies

The 101companies **Ontology**

- 101companies contribution *X* ...
  - ... uses languages *L*
  - ... uses technology *T*
  - ... implements features *F*
  - ... demonstrates concepts *C*
- 101companies developer *D* ...
  - ... has skills regarding language *L*
  - ... has skills regarding technology *T*
- ...

The 101companies **Explorer**



Selection of contribution components and aspects

Files

Company.java
Employee.java
ObjectFactory.java
tests
Operations.java
features
Total.java
Serialization.java
Cut.java
sampleCompany.xml
README.md
Makefile
Company.xsd

Languages

Company.java
Department.java
Employee.java
ObjectFactory.java
package-info.java
Cut.java
Serialization.java
Total.java
Operations.java
XML
sampleCompany.xml
XSD
Company.xsd

Technologies

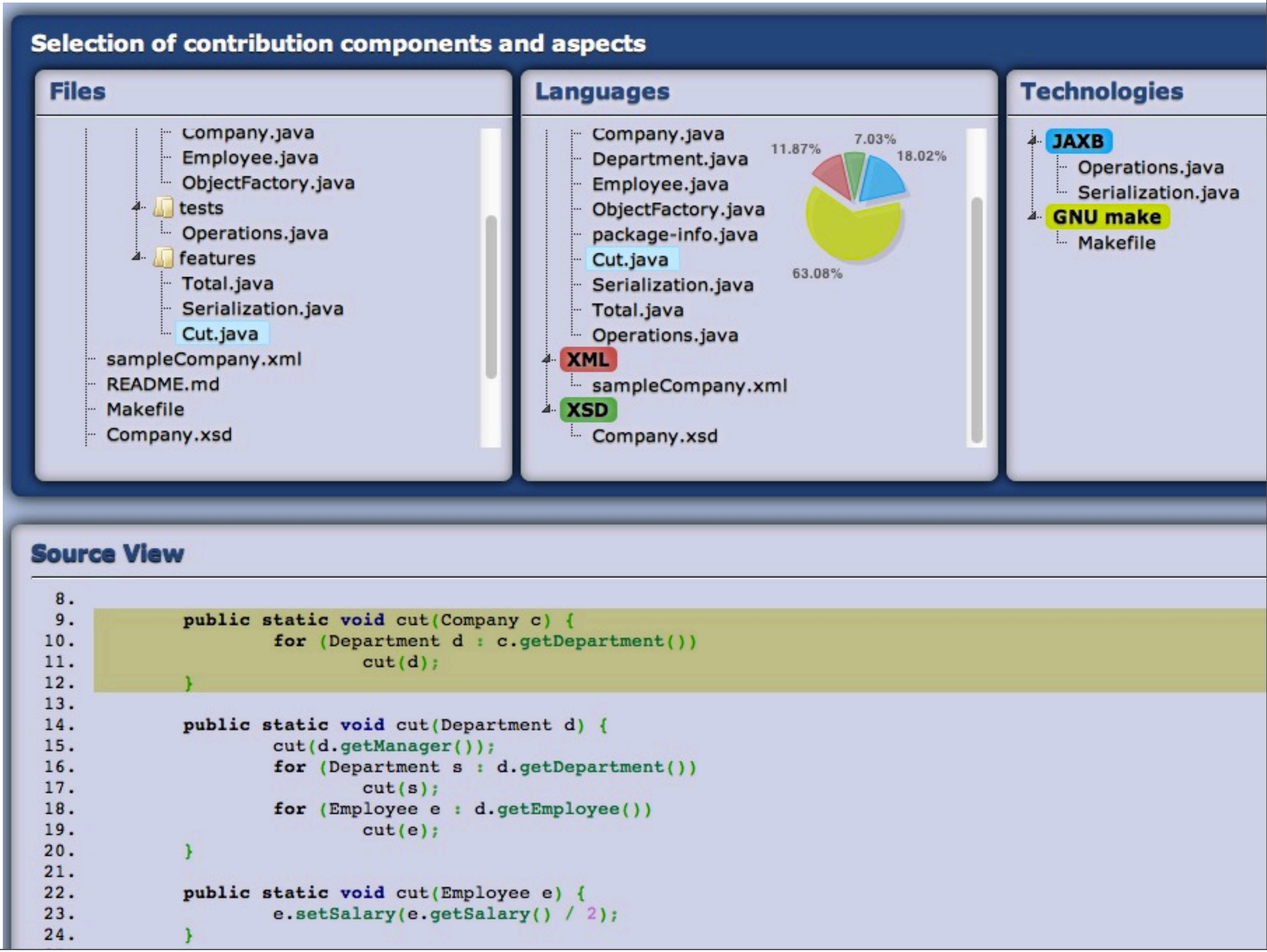
JAXB
Operations.java
Serialization.java
GNU make
Makefile

Tags

cut/company
Cut.java
cut/department
Cut.java
cut/employee
Cut.java
data/company
Company.java
Company.xsd
data/department
Company.xsd
Department.java
data/employee

Source View

```

	public static void cut(Company c) {
		for (Department d : c.getDepartment())
			cut(d);
	}

	public static void cut(Department d) {
		cut(d.getManager());
		for (Department s : d.getDepartment())
			cut(s);
		for (Employee e : d.getEmployee())
			cut(e);
	}

	public static void cut(Employee e) {
		e.setSalary(e.getSalary() / 2);
	}

}

```



Selection of contribution components and aspects

Files

- Company.java
- Employee.java
- ObjectFactory.java
- tests
  - Operations.java
- features
  - Total.java
  - Serialization.java
  - Cut.java
- sampleCompany.xml
- README.md
- Makefile
- Company.xsd

Languages

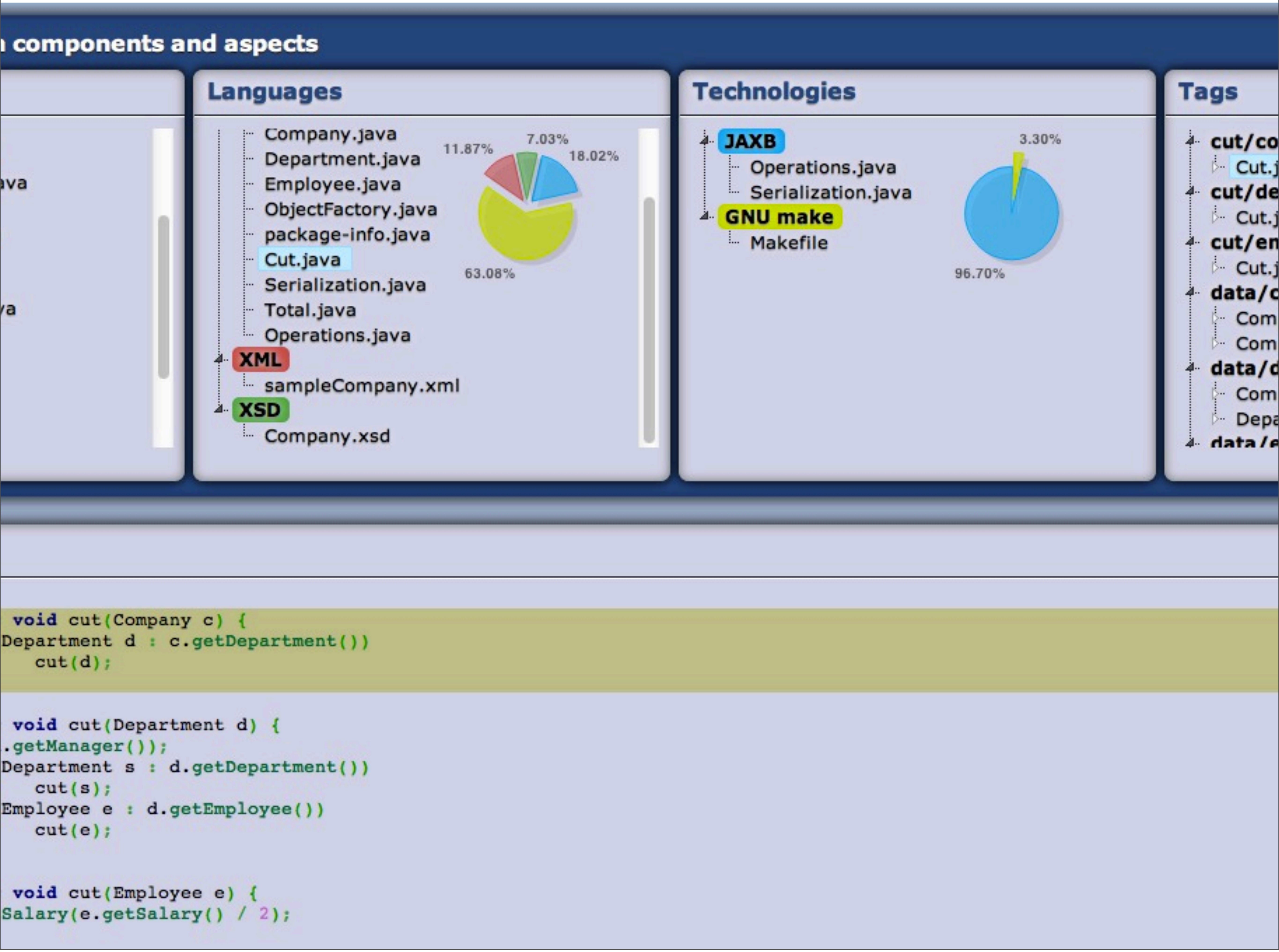
- Company.java
- Department.java
- Employee.java
- ObjectFactory.java
- package-info.java
- Cut.java
- Serialization.java
- Total.java
- Operations.java
- XML
  - sampleCompany.xml
- XSD
  - Company.xsd

Technologies

- JAXB
  - Operations.java
  - Serialization.java
- GNU make
  - Makefile

Source View

```

        public static void cut(Company c) {
                for (Department d : c.getDepartment())
                        cut(d);
        }

        public static void cut(Department d) {
                cut(d.getManager());
                for (Department s : d.getDepartment())
                        cut(s);
                for (Employee e : d.getEmployee())
                        cut(e);
        }

        public static void cut(Employee e) {
                e.setSalary(e.getSalary() / 2);
        }
```



components and aspects

ava

a

Languages

- Company.java
- Department.java
- Employee.java
- ObjectFactory.java
- package-info.java
- Cut.java
- Serialization.java
- Total.java
- Operations.java

XML
- sampleCompany.xml

XSD
- Company.xsd

Technologies

JAXB
- Operations.java
- Serialization.java

GNU make
- Makefile

Tags

cut/co
- Cut.j

cut/de
- Cut.j

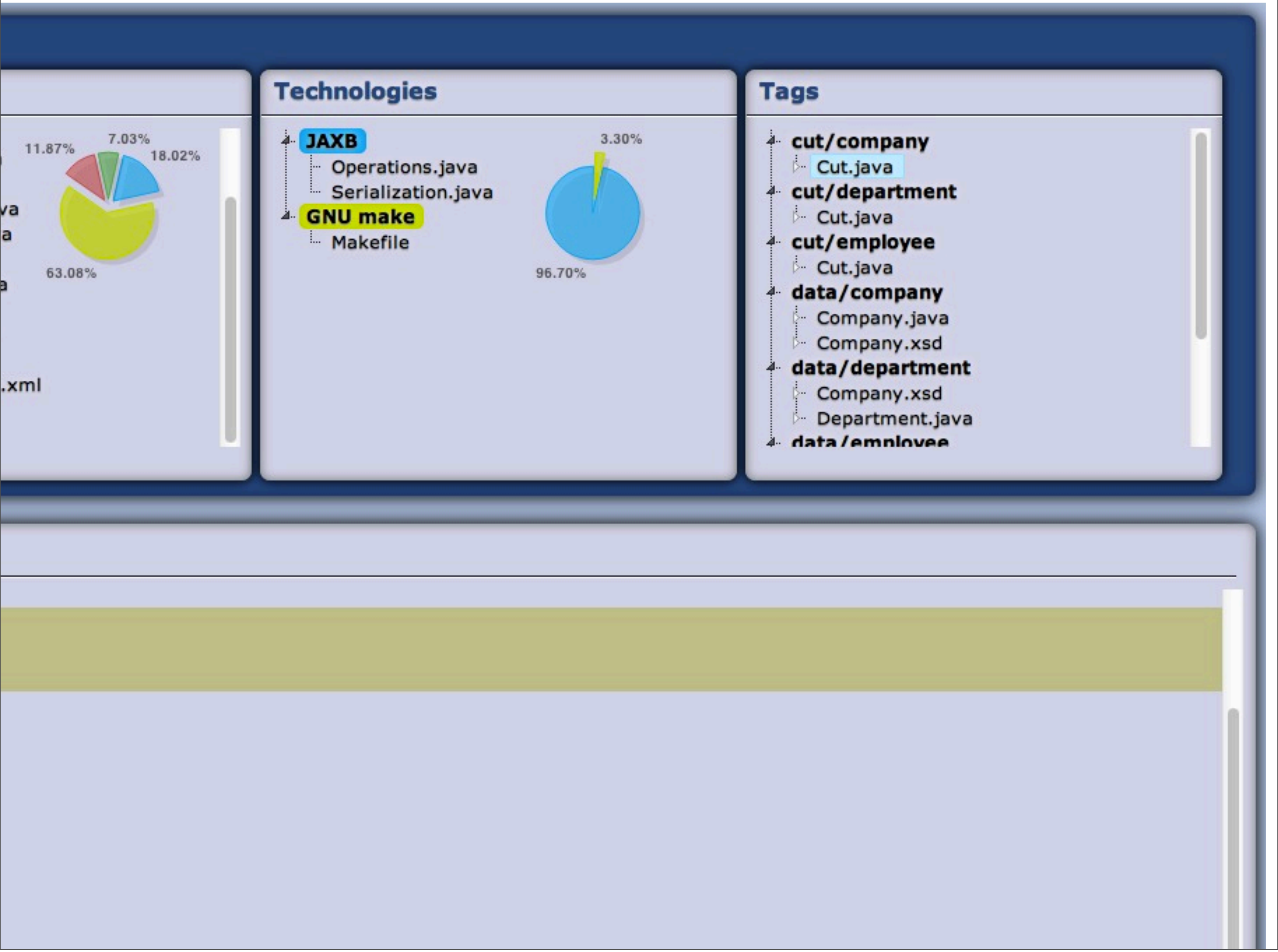
cut/en
- Cut.j

data/c
- Com
- Com

data/d
- Com
- Depa

data/e

```
 void cut(Company c) {
Department d : c.getDepartment())
   cut(d);


 void cut(Department d) {
.getManager());
Department s : d.getDepartment())
   cut(s);
Employee e : d.getEmployee())
   cut(e);


 void cut(Employee e) {
Salary(e.getSalary() / 2);
```

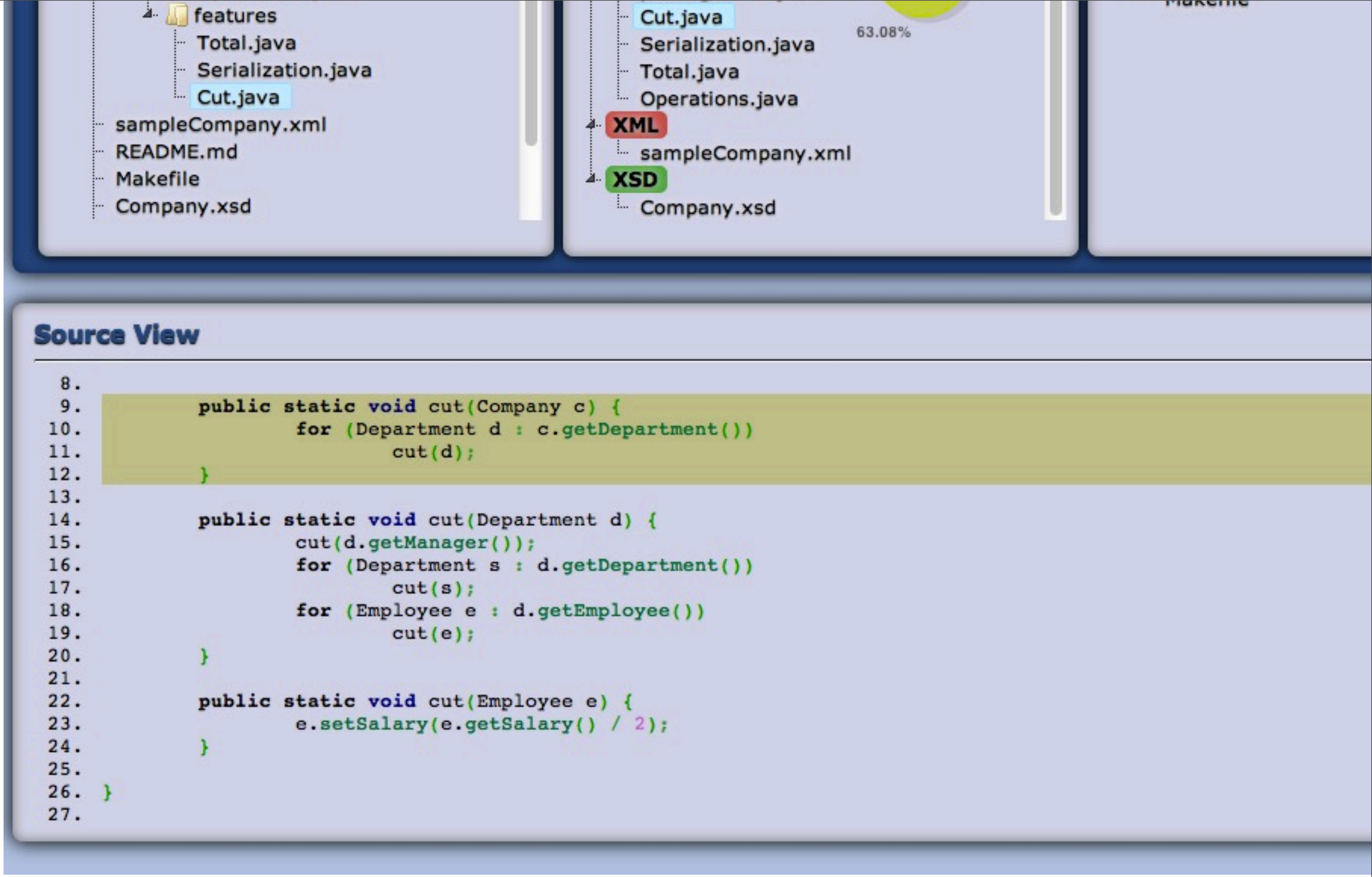


Technologies

- JAXB
  - Operations.java
  - Serialization.java
- GNU make
  - Makefile

Tags

- cut/company
  - Cut.java
- cut/department
  - Cut.java
- cut/employee
  - Cut.java
- data/company
  - Company.java
  - Company.xsd
- data/department
  - Company.xsd
  - Department.java
- data/employee



```

    public static void cut(Company c) {
        for (Department d : c.getDepartment())
            cut(d);
    }

    public static void cut(Department d) {
        cut(d.getManager());
        for (Department s : d.getDepartment())
            cut(s);
        for (Employee e : d.getEmployee())
            cut(e);
    }

    public static void cut(Employee e) {
        e.setSalary(e.getSalary() / 2);
    }

}

```

What's in for research?

- ***Megamodeling*** for software technologies
- ***Knowledge*** representation and management
- ***Education*** on programming technologies
- ***Ontologies*** in the fields PL, SE, SL, ...
- ***Empirical research***
  - Language usage analysis
  - Technology usage analysis
- ***Generic language technology***
- ...

Acknowledgement

Acknowledgement

- Co-initiators

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)

Acknowledgement

- Co-initiators
    - Jean-Marie Favre (University of Grenoble)
    - Dragan Gasevic (Athabasca University)

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)
  - Dragan Gasevic (Athabasca University)
- Student of the first hour

Acknowledgement

- Co-initiators
    - Jean-Marie Favre (University of Grenoble)
    - Dragan Gasevic (Athabasca University)
- Student of the first hour
    - Thomas Schmorleiz

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)
  - Dragan Gasevic (Athabasca University)
- Student of the first hour
  - Thomas Schmorleiz
- Principle PhD student

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)
  - Dragan Gasevic (Athabasca University)
- Student of the first hour
  - Thomas Schmorleiz
- Principle PhD student
  - Andrei Varanovich

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)
  - Dragan Gasevic (Athabasca University)
- Student of the first hour
  - Thomas Schmorleiz
- Principle PhD student
  - Andrei Varanovich
- Students of the current hour

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)
  - Dragan Gasevic (Athabasca University)
- Student of the first hour
  - Thomas Schmorleiz
- Principle PhD student
  - Andrei Varanovich
- Students of the current hour
  - Martin Leinberger

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)
  - Dragan Gasevic (Athabasca University)
- Student of the first hour
  - Thomas Schmorleiz
- Principle PhD student
  - Andrei Varanovich
- Students of the current hour
  - Martin Leinberger
  - ...

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)
  - Dragan Gasevic (Athabasca University)
- Student of the first hour
  - Thomas Schmorleiz
- Principle PhD student
  - Andrei Varanovich
- Students of the current hour
  - Martin Leinberger
  - ...

Great to work with you JM!

Acknowledgement

- Co-initiators
  - Jean-Marie Favre (University of Grenoble)
  - Dragan Gasevic (Athabasca University)
- Student of the first hour
  - Thomas Schmorleiz
- Principle PhD student
  - Andrei Varanovich
- Students of the current hour
  - Martin Leinberger
  - ...

Great to work with you JM!

Thank you **Paul** for forming me at CWI back then!

Understanding Haskellware

Haskell ware

Haskell

Understanding

+1

ware

Understanding +1

Understanding +1

```
$ ghci -v0
```

Understanding +1

```
$ ghci -v0
Prelude> let inc = (+) 1
```

Understanding +1

```
$ ghci -v0
Prelude> let inc = (+) 1
Prelude> inc 41
```

Understanding +1

```
$ ghci -v0
Prelude> let inc = (+) 1
Prelude> inc 41
42
```

Understanding +1

```
$ ghci -v0
Prelude> let inc = (+) 1
Prelude> inc 41
42
Prelude> :q
```

Understanding +1

```
$ ghci -v0
Prelude> let inc = (+) 1
Prelude> inc 41
42
Prelude> :q
```

- “Data modeling” for numbers
- “Core functionality” for increment
- “De-/serialization” of numbers
- “Web/CL/GU interface” for incrementing numbers
- “Testing” for incrementing numbers
- “Performance profile” for incrementing numbers
- ...

http://101companies.org/
Haskell-based implementations

Impl.	Headline
dph	Data parallelism in Haskell with DPH
haskell	Basic functional programming in Haskell
hdbc	Database programming in Haskell with HDBC
mvar	Concurrent programming in Haskell with MVars
parsec	Parsing in Haskell with Parsec
tmvar	Concurrent programming and STM in Haskell with TMVars
writerMonad	Logging in Haskell with the Writer monad
haskellCGI	CGI style Web programming in Haskell
hxt	In-memory XML processing in Haskell with HXT
wxHaskell	GUI programming in Haskell with wxHaskell
happstack	Web programming in Haskell with Happstack
haskellDB	Schema-aware database programming with HaskellDB
hxtPickler	XML data binding for Haskell with HXTs XML pickler
syb	Generic programming in Haskell with SYB

http://101companies.org/

Language usage of Haskell-based implementations

Language	Headline	# Implementations
Haskell	An advanced purely-functional programming language	14
CSS	A style sheet language for Web programming	2
SQL	A query language for databases	2
XHTML	A markup language for documents on the Web	2
XML	An extensible markup language	2
Haskell 98	A standardized version of Haskell	1
JavaScript	A dynamic, prototype-based scripting language with first-class functions	1

http://101companies.org/

Technology usage of Haskell-based implementations

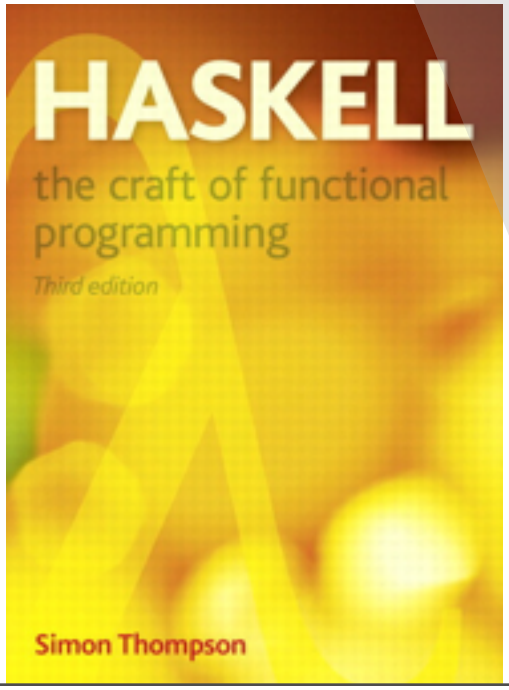
Technology	**Headline**	**# Implementations**
GHCi	The Haskell interpreter as part of GHC	12
GHC	A Haskell compiler	5
HDBC	A library for embedded SQL programming in Haskell	2
HXT	A toolkit for tree-based XML processing in Haskell	2
MySQL	A relational database management system	2
ODBC	A standard API for accessing database management systems	2
CGI	A standard for website generation on a web server	1
DBDirect	A program generator that derives Haskell types from database schemas	1
DPH	A GHC extension for data parallelism	1
Happstack	A framework for web programming in Haskell	1
HaskellDB	A combinator library for expressing DBMS queries in Haskell	1
Heist	An XHTML template engine for Haskell	1
Parsec	A parser combinator library in Haskell	1
XML pickler	An XML data binding technology for Haskell	1
wxHaskell	A wxWidgets-based GUI library for Haskell	1

http://www.haskell.org/

#haskell irc channel

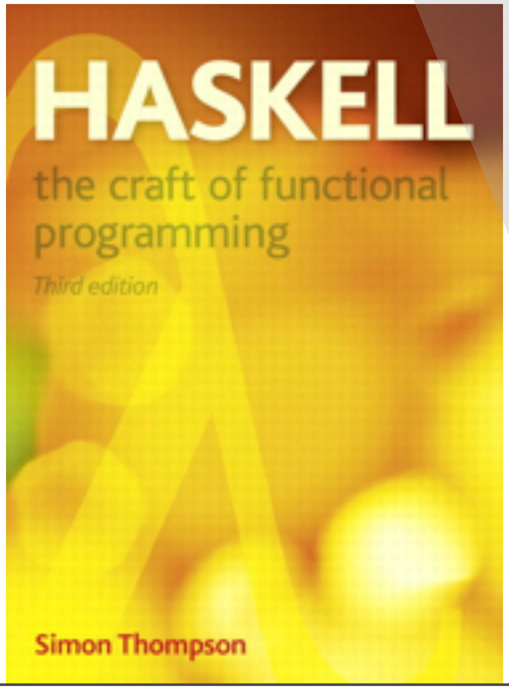
http://hackage.haskell.org/

http://www.haskell.org/haskell-symposium/



Chapters

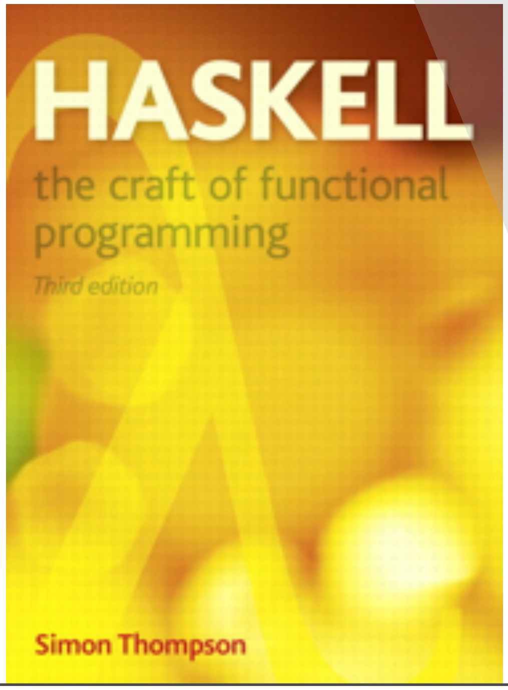
Getting started with Haskell and GHCi
Basic types and definitions
Designing and writing programs
Data types tuples and lists
Programming with lists
Defining functions over lists
Playing the game IO in Haskell
Reasoning about programs
Generalization patterns of computation
Higher order functions
Developing higher order programs
Overloading type classes and type checking
Algebraic types
Case study Huffman codes
Abstract data types
Lazy programming
Programming with monads
Domain Specific Languages
Time and space behaviour



- action → Action
- algebraic type → Algebraic data type
- base case → Base case
- bool → Boolean
- calculation → Calculation
- class → Type class
- code → Code
- coding → Programming
- …

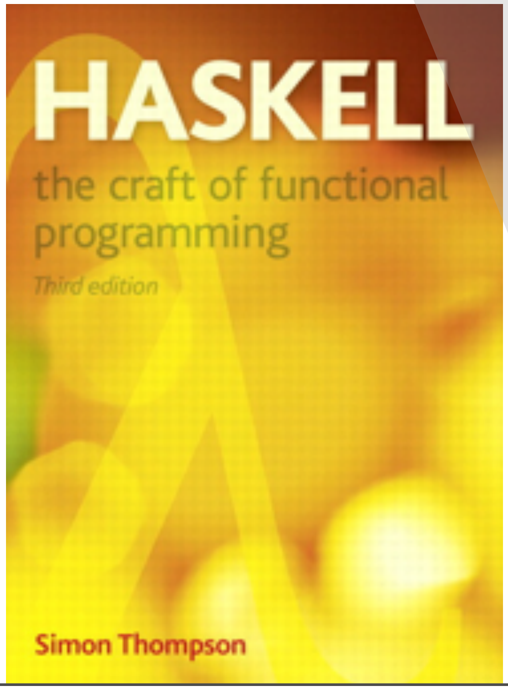
Terms

Terms	Getting started with Haskell and GHCi	Basic types and definitions	Designing and writing programs	Data types tuples and lists	Programming with lists	Defining functions over lists	Playing the game IO in Haskell	Reasoning about programs	Generalization patterns of computation	Higher order functions	Developing higher order programs	Overloading type classes and type checking	Algebraic types	Case study Huffman codes	Abstract data types	Lazy programming	Programming with monads	Domain Specific Languages	Time and space behaviour
action							●												
algebraic													●						
algebraic type													●						
base case								●											
bool												●							
calculation																●			●
class												●							
code														●					
coding														●					
command	●																		
complexity																			●
constructor				●									●						
database																			


HASKELL
the craft of functional programming
Third edition
Simon Thompson

Terms

Accumulator, Action, Algebraic data type, Applicative functor, Association list, Base case, Bit, Boolean, Calculation, Catamorphism, Character, Code, Command, Complexity, Condition, Core, Data constructor, Data structure, Data type, Database, Declaration, Directory, Eager evaluation, Equality, Equation, Equational reasoning, Evaluation strategy, Exception, Expression, Factorial, File, Filter function, Float, Fmap function, Fold function, Foreign function interface, Function application, Function definition, Functor, Guard, Haskell package, Haskell script, Head, Higher-order function, I/O system, Identity element, Import, Induction, Infinite list, Input, Integer, Lambda abstraction, Language:XML, Lazy evaluation, List comprehension, Local scope, Loop, MVar, Map function, Maybe type, Module, Monad, Monad transformer, Monadic value, Monoid, Operator, Operator precedence, Output, Parser, Parser combinator, Parsing, Partial application, Pattern, Pattern matching, Performance, Pointer, Polymorphism, Predicate, Prelude, Process, Product function, Profiling, Program design, Program optimization, Programming, Proof, Property, Pure function, Query, Queue, Random number, Recursion, Regular expression, Reverse function, Set, Stack, State, String, Sum function, TCP, Table, Tail, Technology:GHC, Technology:GHCi, Technology:Glade, Technology:HPC, Technology:Parsec, Testing, Text, Thread, Tree, Tuple, Type checking, Type class, Type definition, Type signature, Type system, Type-class instance, UDP, User interface, Zipper



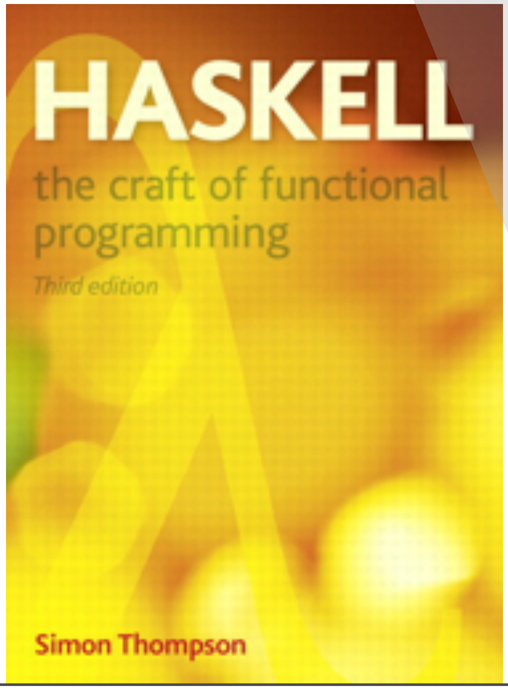
Terms

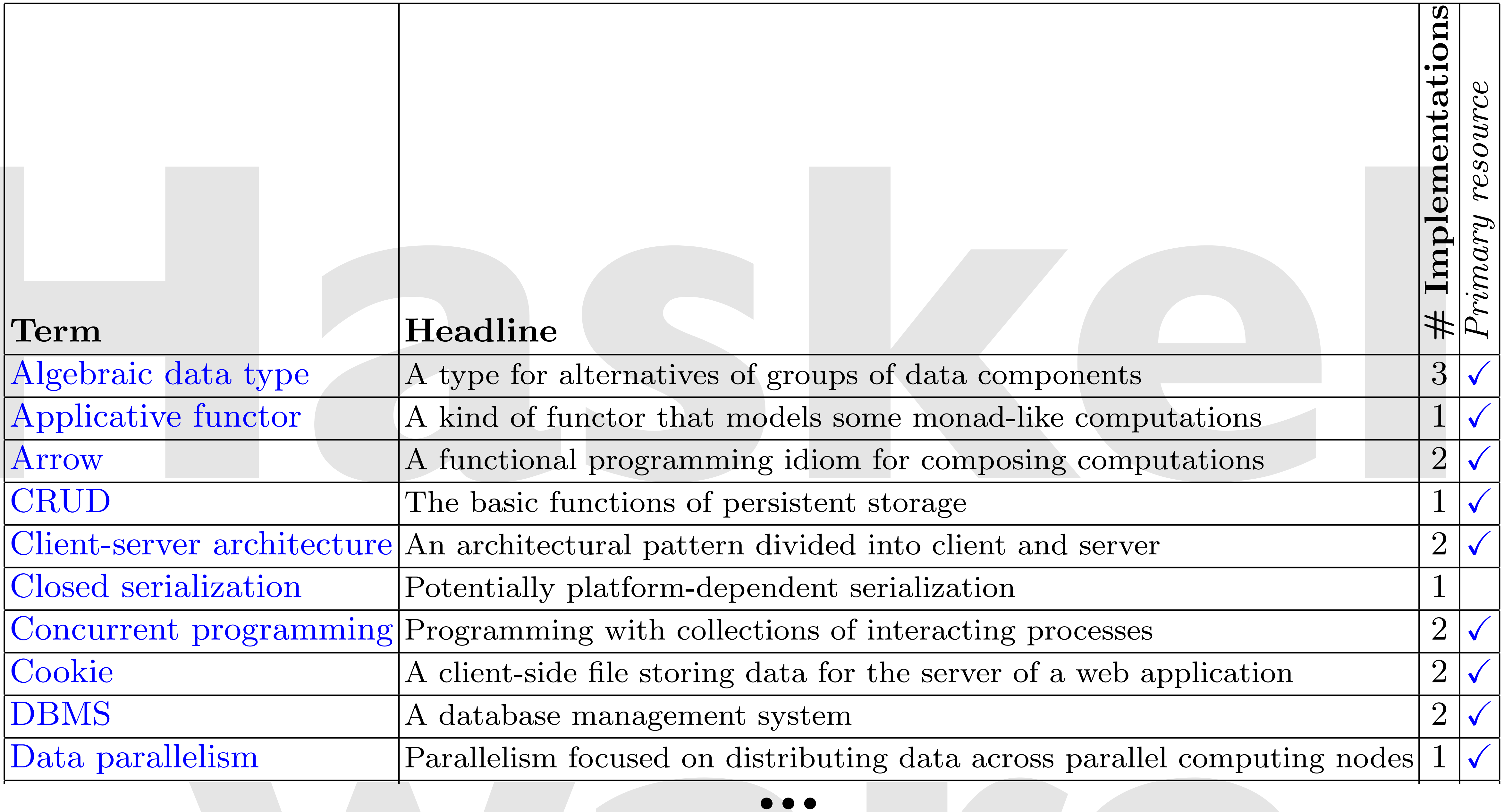
CRAFT only: Algebraic data type, Base case, Calculation, Code, Complexity, Equality, Equational reasoning, Float, Head, Higher-order function, Infinite list, Local scope, Partial application, Program design, Programming, Proof, Queue, Set, Tuple, Type checking

PIH only: Declaration, Equation, Function application, Function definition, Haskell script, Identity element, Lambda abstraction, Parser combinator, Product function, Reverse function, String, Type-class instance

RWH only: Association list, Core, Data type, Directory, Exception, Foreign function interface, Language:XML, Loop, MVar, Monad transformer, Operator precedence, Output, Parsing, Performance, Pointer, Polymorphism, Predicate, Process, Profiling, Program optimization, Property, Pure function, Query, TCP, Table, Technology:GHC, Technology:Glade, Technology:HPC, Technology:Parsec, Thread, Type definition, Type signature, Type system, UDP, User interface

LYAH only: Accumulator, Applicative functor, Condition, Data structure, Expression, Factorial, Fmap function, Functor, Import, Input, Monadic value, Sum function, Zipper





Term	Headline	# Implementations	*Primary resource*
Algebraic data type	A type for alternatives of groups of data components	3	✓
Applicative functor	A kind of functor that models some monad-like computations	1	✓
Arrow	A functional programming idiom for composing computations	2	✓
CRUD	The basic functions of persistent storage	1	✓
Client-server architecture	An architectural pattern divided into client and server	2	✓
Closed serialization	Potentially platform-dependent serialization	1	
Concurrent programming	Programming with collections of interacting processes	2	✓
Cookie	A client-side file storing data for the server of a web application	2	✓
DBMS	A database management system	2	✓
Data parallelism	Parallelism focused on distributing data across parallel computing nodes	1	✓

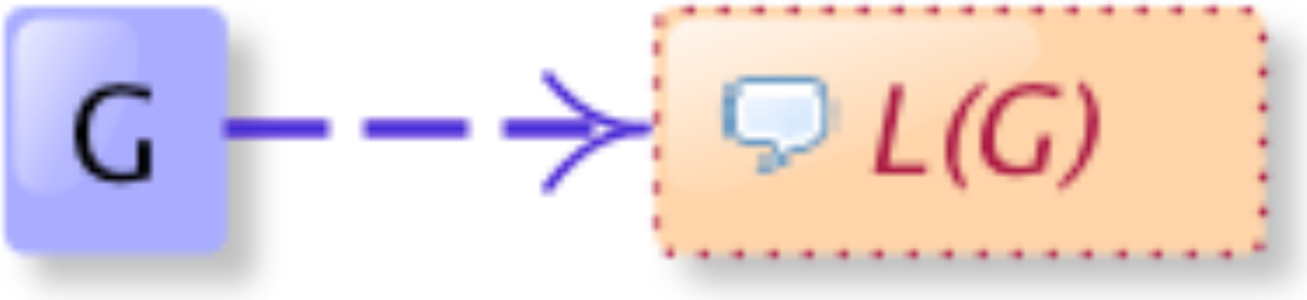
...

We can compare vocabulary coverage of the textbooks and the 101companies Wiki!

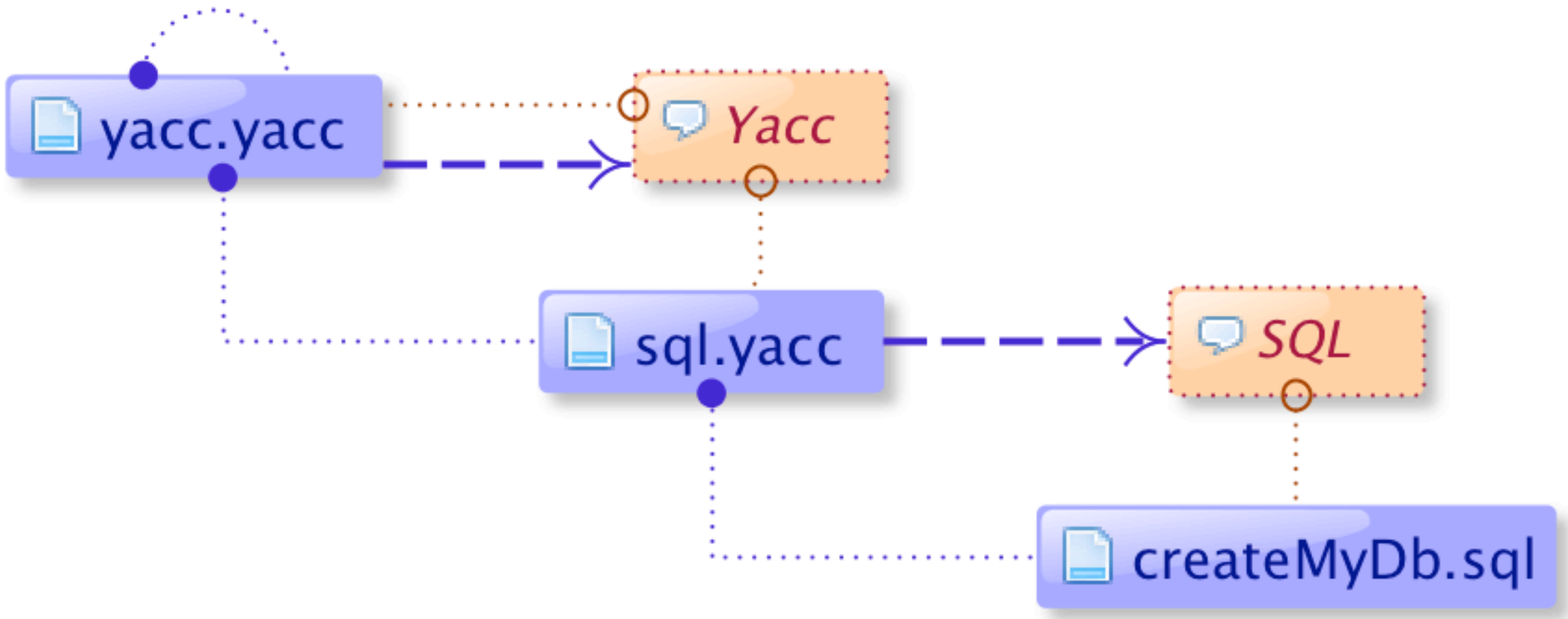
Megamodels to the rescue

What’s the essence of a technology?

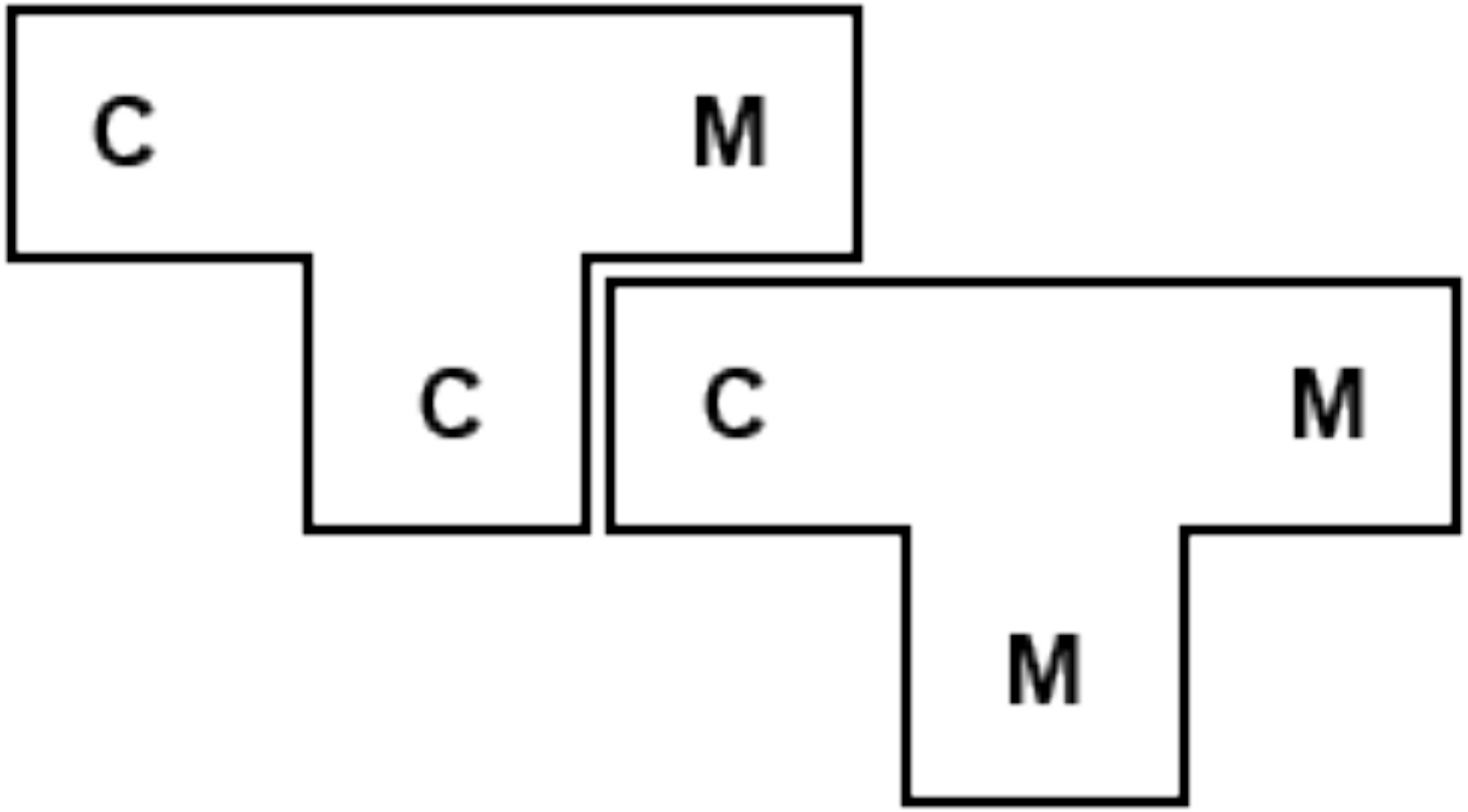
What’s the essence of a language?



What's the essence of a technology?

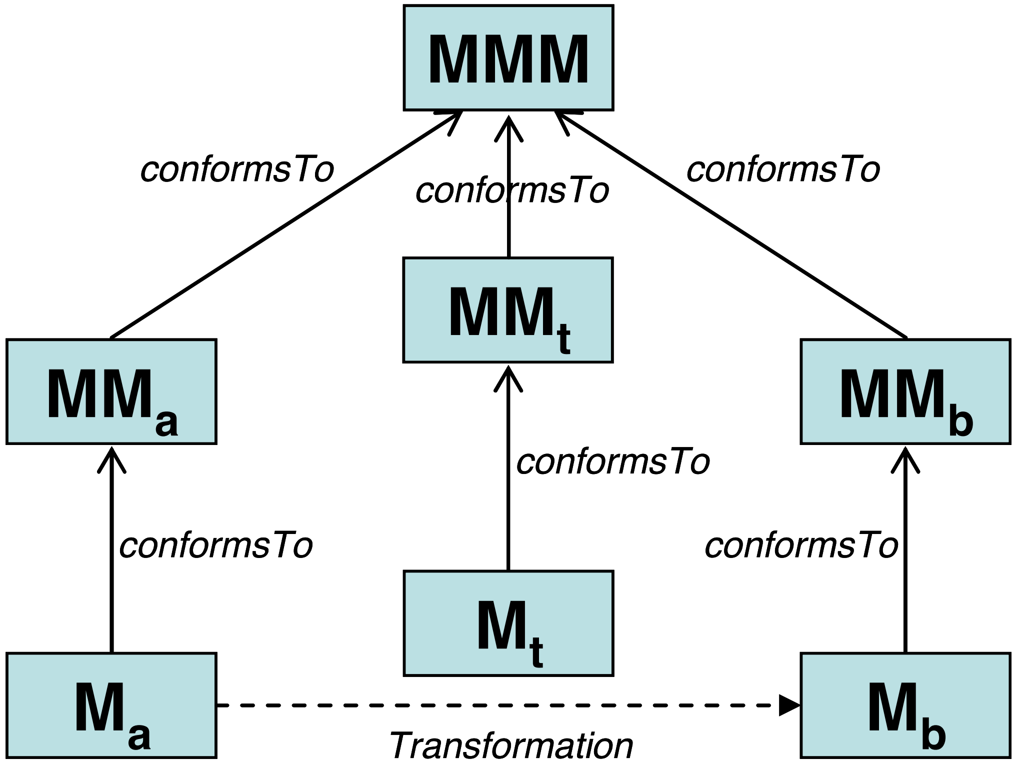


That’s a megamodel, too!



http://en.wikipedia.org/wiki/Tombstone_diagram

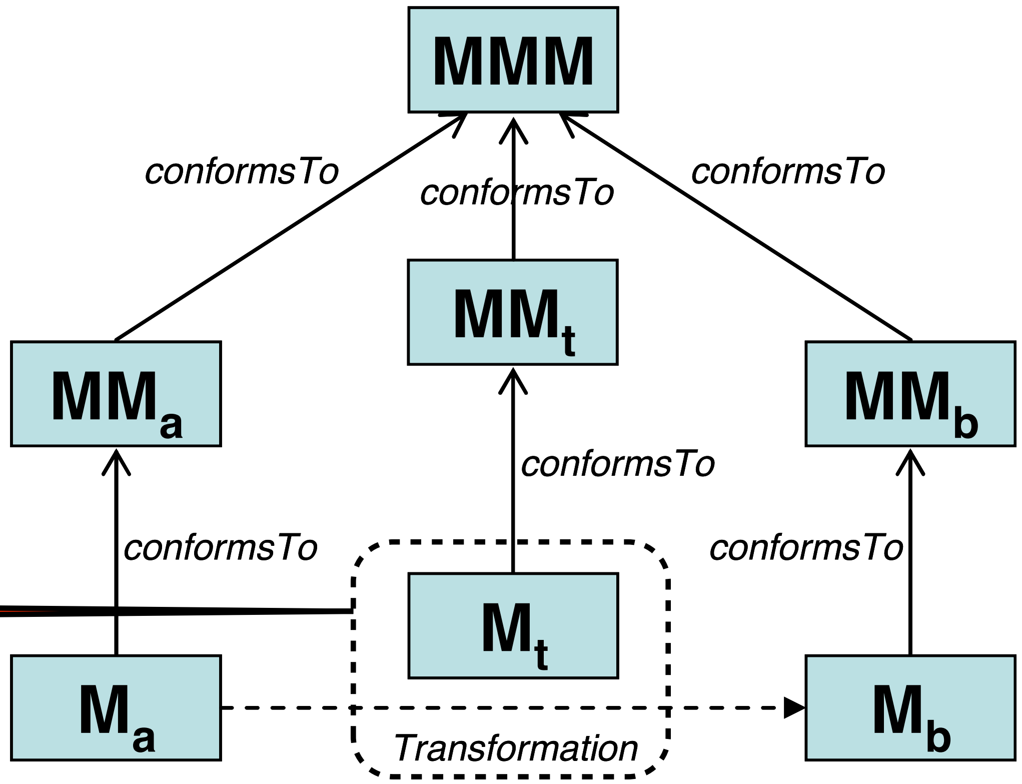
Yet another megamodel!



Model transformations with ATL

http://wiki.eclipse.org/ATL/Concepts#Model_Transformation

Yet another megamodel!

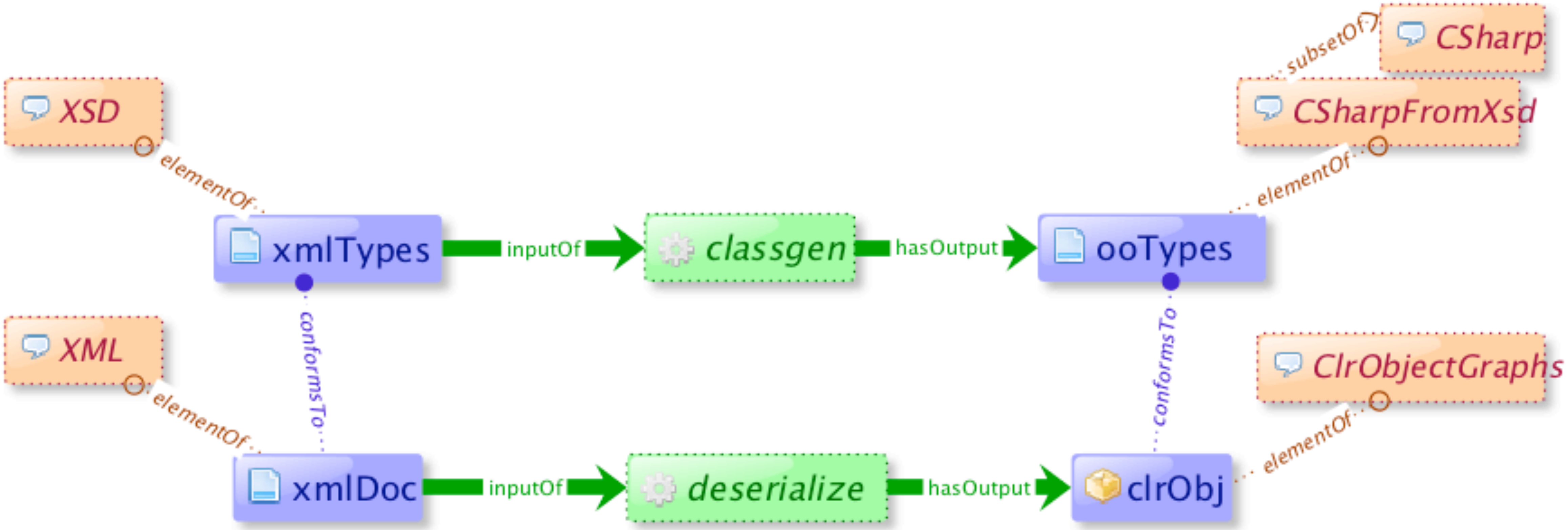


What to think of this part?

Model transformations with ATL

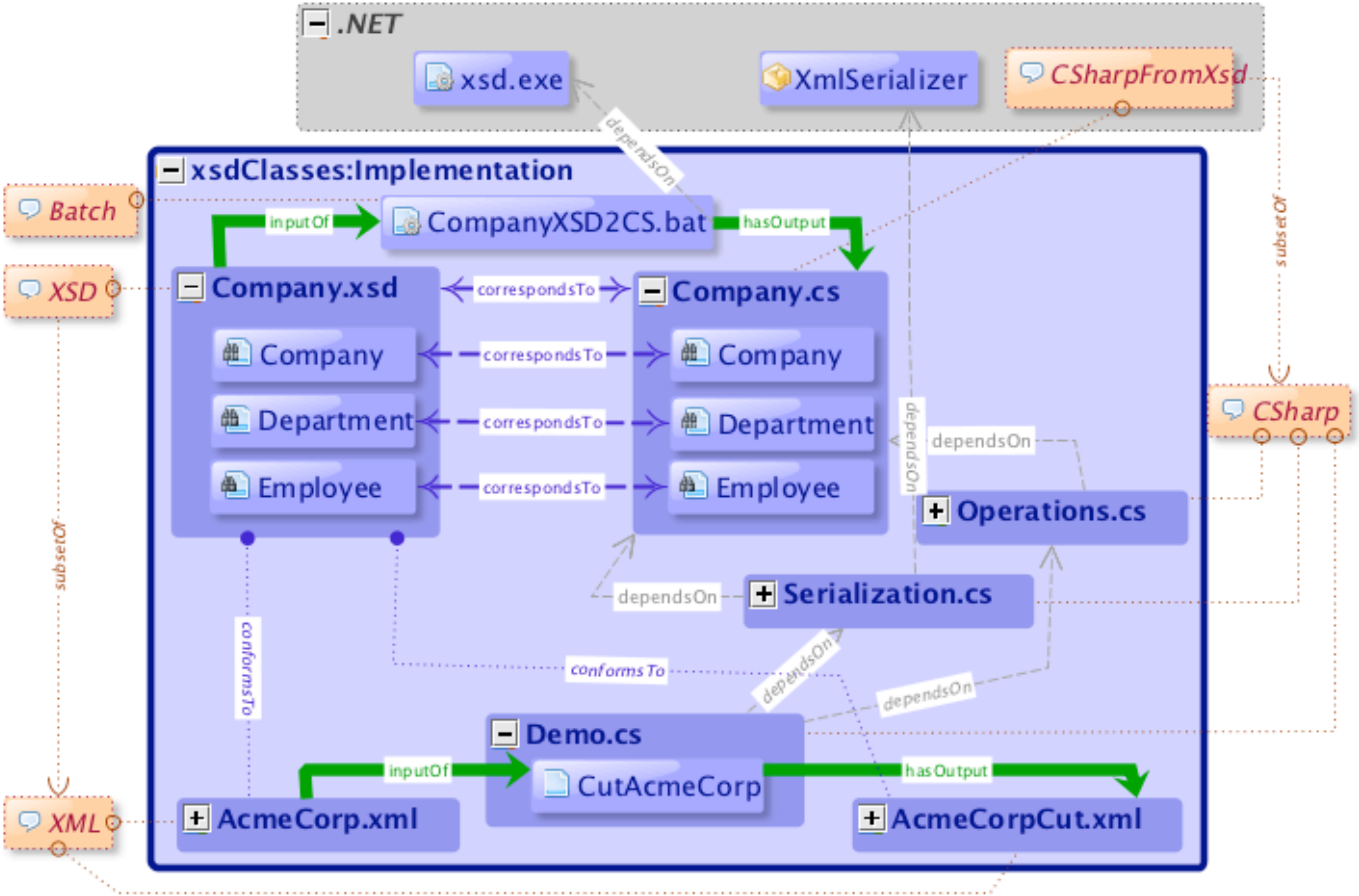
http://wiki.eclipse.org/ATL/Concepts#Model_Transformation

Megamodel of O/X mapping with xsd.exe



http://softlang.uni-koblenz.de/mega/

Megamodel of a product using xsd.exe



http://black42.uni-koblenz.de/production/101worker/MegaModels/implementations/xsdClasses/

Conclusion

- Please:
  - Have a look and spread the message.
  - Don't expect perfect material "yet".
  - Contribute or encourage others to do so.
  - Consider using the project in teaching.
  - Engage in collaboration.

Thanks!
Questions?