Understanding Technological Spaces

Ralf Lämmel
Software Languages Team
University of Koblenz-Landau
Binging/Googling
“Technological spaces”
Technological spaces: An initial appraisal

citesex.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 ...

New-Approaching OWL and MDA Through Technological Spa...
citesex.ist.psu.edu/viewdoc/download?doi=10.1.1.58...rep...
by D Gašević - 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 | Research...
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/.../mag00007.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

citeeprx.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, ...)

A Silver Bullet for Knowledge Management and Electronic Commerce ...
Relational databases for querying xml documents: Limitations and ...

New-Aproaching OWL and MDA Through Technological Spa...
citeeprx.ist.psu.edu/viewdoc/download?doi=10.1.1.58...rep...
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 | Research...
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 Gavševic - 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 | Research...
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.
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"
Technical space comprises one or more modeling spaces

Let’s try “Bezivin”
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
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
This is not Jean Bezivin.
This is Jean-Marie Favre.
This is, in fact, an image of Jean-Marie Favre.
Google ➔ Bing
"technological spaces" "Jean Bezivin"

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
- ...

© 2012, 101companies
More technological spaces ...
More technological spaces ...

Data driven journalism

TOOLS 2007-2012

Specware

CSVware

Lambdaware

Javaware

Rubyware

???
We have a problem!
We have a problem!
Too much software technologies.
Too much software languages.
Too little time.
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?
What’s 101companies?

It is a knowledge resource for technological space travel.
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 “101 companies”?

Company X: Swing + JDBC

Company Y: SWT + Hibernate

Company Z: GWT + MongoDB

101 ways of building a HRMS.
Building a HRMS for 101 companies.

© 2012, 101companies
A Human Resources Management System

- Total salaries
- Increase salaries
- Cut salaries
- Edit employee data
- Import / export company data
The 101companies Repository

<table>
<thead>
<tr>
<th>#Files per implementation</th>
<th>Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages</td>
<td>LOC per implementation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Java</th>
<th>XML</th>
<th>Haskell</th>
<th>JavaScript</th>
<th>CSS</th>
<th>HTML</th>
</tr>
</thead>
</table>

442 LOC median (M=442)
The 101companies Wiki
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.
The 101companies Wiki
Feature model

Classification

- 101feature
- Behavioral 101feature
  - Data export
  - Data import
  - Logging
  - Structure-driven query
  - Type-driven query
  - Type-driven transformation
  - Visualization
- Meta 101feature
- Quality 101feature
  - 101design
  - 101execution
    - Access control
    - Data parallelism
    - Distribution
    - Persistence
    - Reliability
    - Serialization
    - Task parallelism
- Structural 101feature
  - Global invariant
  - Graph structure
  - Many-to-many
  - Tree structure
  - Type hierarchy
- UI 101feature

- All features of the 101companies System
  - An export operation for company data
  - An import operation for company data
  - Logging of salary changes
  - A query for the depth of department nesting
  - A query for totaling the salaries of all employees
  - A transformation for a salary cut
  - The visualization of company data

- Design qualities of the 101companies System
  - Execution qualities of the 101companies System
    - Access control for company data
    - Data parallelism for operations on company data
    - Distribution of company data and operations
    - Persistence for company data
    - Reliability of the system services
    - Serialization for company data
    - Task parallelism for operations on company data

- A constraint on salaries within the company hierarchy
- An association between mentees and mentors
- A friend relationship between employees
- Tree-like structure of companies and departments
- A common base type for departments and employees

© 2012, 101companies
Classification

- 101feature
  - Behavioral 101feature
    - Data export
    - Data import
    - Logging
    - Structure-driven query
    - Type-driven query
    - Type-driven transformation
    - Visualization
  - All features of the 101companies
    - An export operation for data
    - An import operation for data
    - Logging of salary changes
    - A query for the depth of a salary
    - A query for totaling the salary
    - A transformation for a salary
    - The visualization of company salaries

- Meta 101feature
  - Meta 101design
  - Meta 101execution
  - Design qualities of the 101companies
  - Execution qualities of the 101companies
  - Access control for companies
  - Data parallelism for operations
  - Distribution of tasks
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
     - Object/Text mapping for Java with ANTLR for parsing
     - Model/Object mapping for Ecore and Java with EMF
     - Object/Relational mapping for Java and SQL/HQL with Hibernate
     - Object/XML mapping for Java and XSD with JAXB

Category: 101theme
The 101companies Wiki
This is an implementation in the 101companies software corpus.

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

## Languages
- XML
- XSD
- Java
- JAXB annotations
- xjc POJOs
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
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:

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

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

```java
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);
    }
}
```
The 101companies Wiki
The 101companies CONTRIBUTORS

任何人开发的贡献将被添加到101companies项目中

© 2012, 101companies
The 101companies CONTRIBUTORS

Catharina 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 Wiki
The 101 companies Ontology

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

- a community and technology context
- a technological space focused on sequential
- a technological space focused on functional
- a technological space focused on object-oriented
- a technological space focused on object-oriented
- a technological space focused on relational
- a technological space focused on XML

- the base category of subordinated, g
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 $T$
- ...

Wednesday, June 20, 12
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
- Cut.java
- Serialization.java
- Total.java
- Operations.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
- data/employee
- Department.java

Source View

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

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

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

© 2012, 101companies
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);
}
```java
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);
}
```
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
  • ...

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

Great to work with you JM!
Understanding Haskellware
Haskell
ware
Haskell

Understanding

+1

ware
Understanding +1

Haskell

ware

© 2012, 101companies
Understanding +1

$ ghci -v0
$ ghci -v0

Prelude> let inc = (+) 1
Understanding \( +1 \)

$ ghci -v0

Prelude> let inc = (+) 1

Prelude> inc 41
Understanding `+1`

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

© 2012, 101companies
$ ghci -v0

Prelude> let inc = (+) 1

Prelude> inc 41
42

Prelude> :q
Understanding +1

- “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
- ...

$ ghci -v0
Prelude> let inc = (+) 1
Prelude> inc 41
42
Prelude> :q
<table>
<thead>
<tr>
<th>Impl.</th>
<th>Headline</th>
</tr>
</thead>
<tbody>
<tr>
<td>dph</td>
<td>Data parallelism in Haskell with DPH</td>
</tr>
<tr>
<td>haskell</td>
<td>Basic functional programming in Haskell</td>
</tr>
<tr>
<td>hdbc</td>
<td>Database programming in Haskell with HDBC</td>
</tr>
<tr>
<td>mvar</td>
<td>Concurrent programming in Haskell with MVars</td>
</tr>
<tr>
<td>parsec</td>
<td>Parsing in Haskell with Parsec</td>
</tr>
<tr>
<td>tmvar</td>
<td>Concurrent programming and STM in Haskell with TMVars</td>
</tr>
<tr>
<td>writerMonad</td>
<td>Logging in Haskell with the Writer monad</td>
</tr>
<tr>
<td>haskellCGI</td>
<td>CGI style Web programming in Haskell</td>
</tr>
<tr>
<td>hxt</td>
<td>In-memory XML processing in Haskell with HXT</td>
</tr>
<tr>
<td>wxHaskell</td>
<td>GUI programming in Haskell with wxHaskell</td>
</tr>
<tr>
<td>happstack</td>
<td>Web programming in Haskell with Happstack</td>
</tr>
<tr>
<td>haskellDB</td>
<td>Schema-aware database programming with HaskellDB</td>
</tr>
<tr>
<td>hxtPickler</td>
<td>XML data binding for Haskell with HXTs XML pickler</td>
</tr>
<tr>
<td>syb</td>
<td>Generic programming in Haskell with SYB</td>
</tr>
</tbody>
</table>
Language usage of Haskell-based implementations

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

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

#haskell irc channel

http://hackage.haskell.org/

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

Haskell
ware

Wednesday, June 20, 12
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
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
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
There are two aspects of review. We can compare vocabulary contributions among the different sources. In this manner, we can determine whether the sources are essentially complementary. Also, we obtain a relatively objective means of profiling the sources. We can also compare vocabulary contributions between sources and the chrestomathy. In this manner, we can determine what aspects of the sources are not covered by the chrestomathy and vice versa.

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

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

Yet another megamodel!

Figure 1. An overview of model transformation

Figure 1 summarizes the full model transformation process. A model $M_a$, conforming to a metamodel $\text{MM}_a$, is here transformed into a model $M_b$ that conforms to a metamodel $\text{MM}_b$. The transformation is defined by the model transformation model $M_t$ which itself conforms to a model transformation metamodel $\text{MM}_t$. This last metamodel, along with the $\text{MM}_a$ and $\text{MM}_b$ metamodels, has to conform to a metametamodel (such as MOF or Ecore).

A simple transformation example

This section introduces the transformation example that is going to be developed in the document. The aim of this first example is to introduce users with the basic concepts of the ATL programming. To this end, this example considers two similar metamodels, Author (Figure 2) and Person (Figure 3), that both encode data relative to persons.

Figure 2. The Author metamodel

Figure 3. The Person metamodel

Both metamodels are composed of a single eponym element: Author for the Author metamodel and Person for the Person metamodel. Both entities are characterized by the same couple of string properties (name and surname).

The objective is here to design an ATL transformation enabling to generate a Person model from an Author model. The transformation to be designed will have to implement the following (obvious) semantics:

- A distinct Person element is generated for each source Author element;
- The name of the generated Person has to be initialized with the name of the source Author;
- The surname of the generated Person has to be initialized with the name of the source Author.

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?