Transitioning Applications to Ontologies

Kalina Bontcheva
on behalf of
the TAO consortium

http://www.tao-project.eu

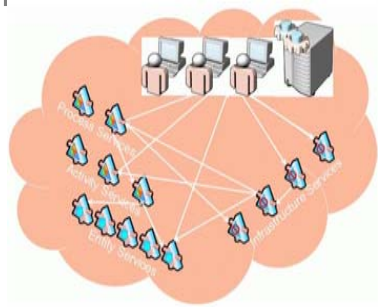
Funded by: European Commission – 6th Framework
Project Reference: IST-2004-026460

TAO

- Addressing the problem of **transitioning legacy applications to ontologies**
- What is a legacy software system:
  - “A large software system that is vital to [an] organisation, but resists modification and evolution to meet new and constantly changing business requirements”
- Towards **semantic-assisted software engineering**

Legacy systems: Main Problems

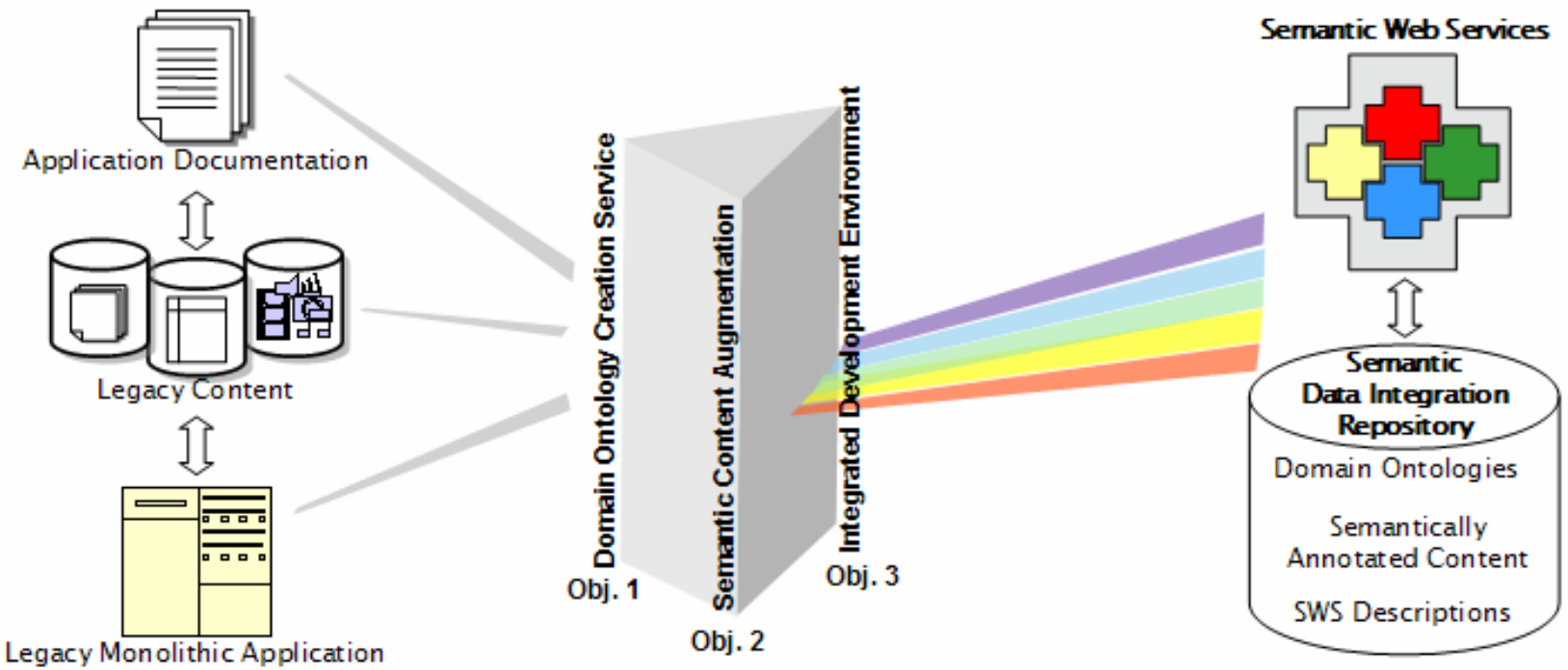
- Enterprise Application Integration
  - Built with languages and data models that are now **out-dated**
  - Badly structured and **hard to maintain**
  - **Badly documented and understood**
  - **Difficult to integrate with each other and with new systems**
  - **Need for migration towards Web 2.0 applications & services**



TAO: Towards a Low-Cost Migration Path

- Make transitioning to ontologies **fast and effective**
- Build a **reusable transitioning process**
- **Minimize** consulting **time** during migration and integration
- **Minimize costs**
- **Reduce integration** overheads and **limit risk**

Transitioning Process

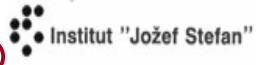


- **Semi-automatic learning of domain ontologies** from software artefacts and legacy content
- **Semantic augmentation of legacy content** and web service definitions
- **Heterogeneous & distributed** semantic repositories
- **Transitioning Infrastructure supporting IT providers**

Partner Contributions

- 

Semantic annotation of text and multimedia content; **GATE** – widely used open-source infrastructure for **text mining**
- 

Transitioning methodology
- 

Machine **learning** and data mining
- 

Transitioning **infrastructure**; exploitation
- 

Most scalable & efficient semantic repository
- 

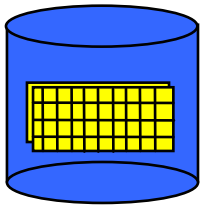
Content and knowledge management
- 

Aviation case study; aircraft manufacturer

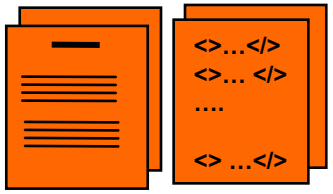
Transitioning Problems

- **From legacy databases to ontologies**
- Towards semantic-based software engineering
- Transitioning web applications towards ontologies and services

Aeronautics Case Study

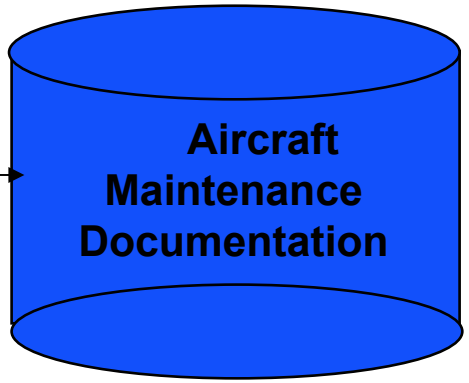


LSA Database



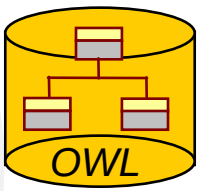
STANDARDS

Semantic Content Augmentation



Ontology Learning + Standards reuse

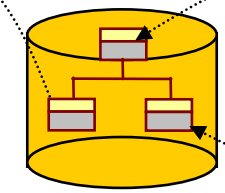
Common Logistic Ontologies in OWL



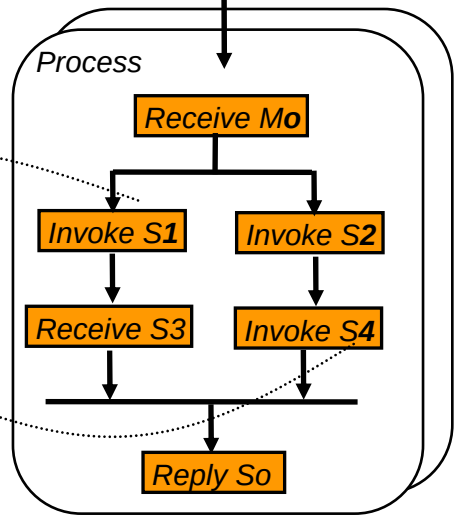
- **A/C Parts**
- **M & O Tasks**
- **Consumables**
- **Tools**
- **Workloads**
- **Maintenance Plans**
- **Due lists**
- **…**

Distributed Knowledge Repositories

Process Modelling

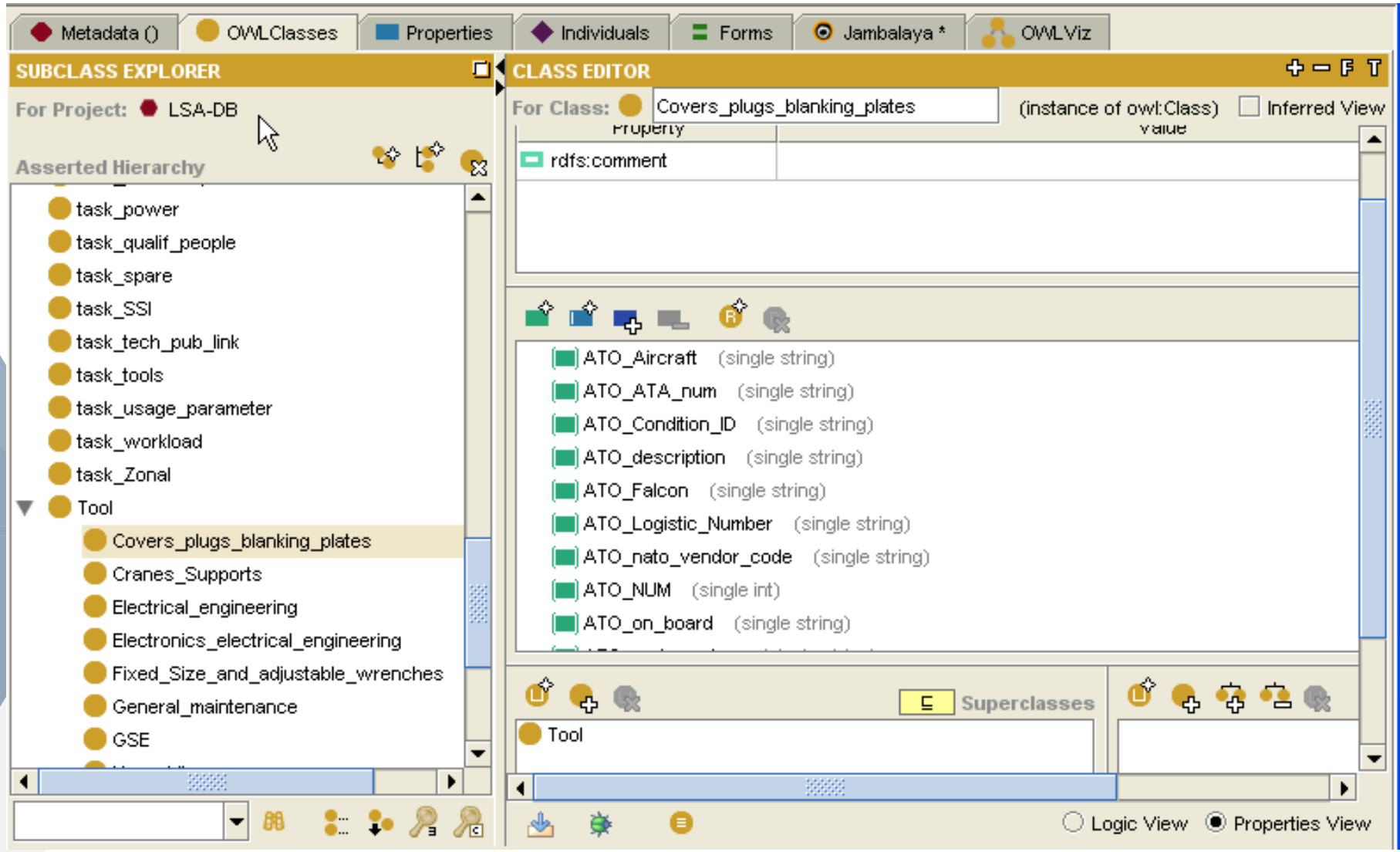


Service Registry

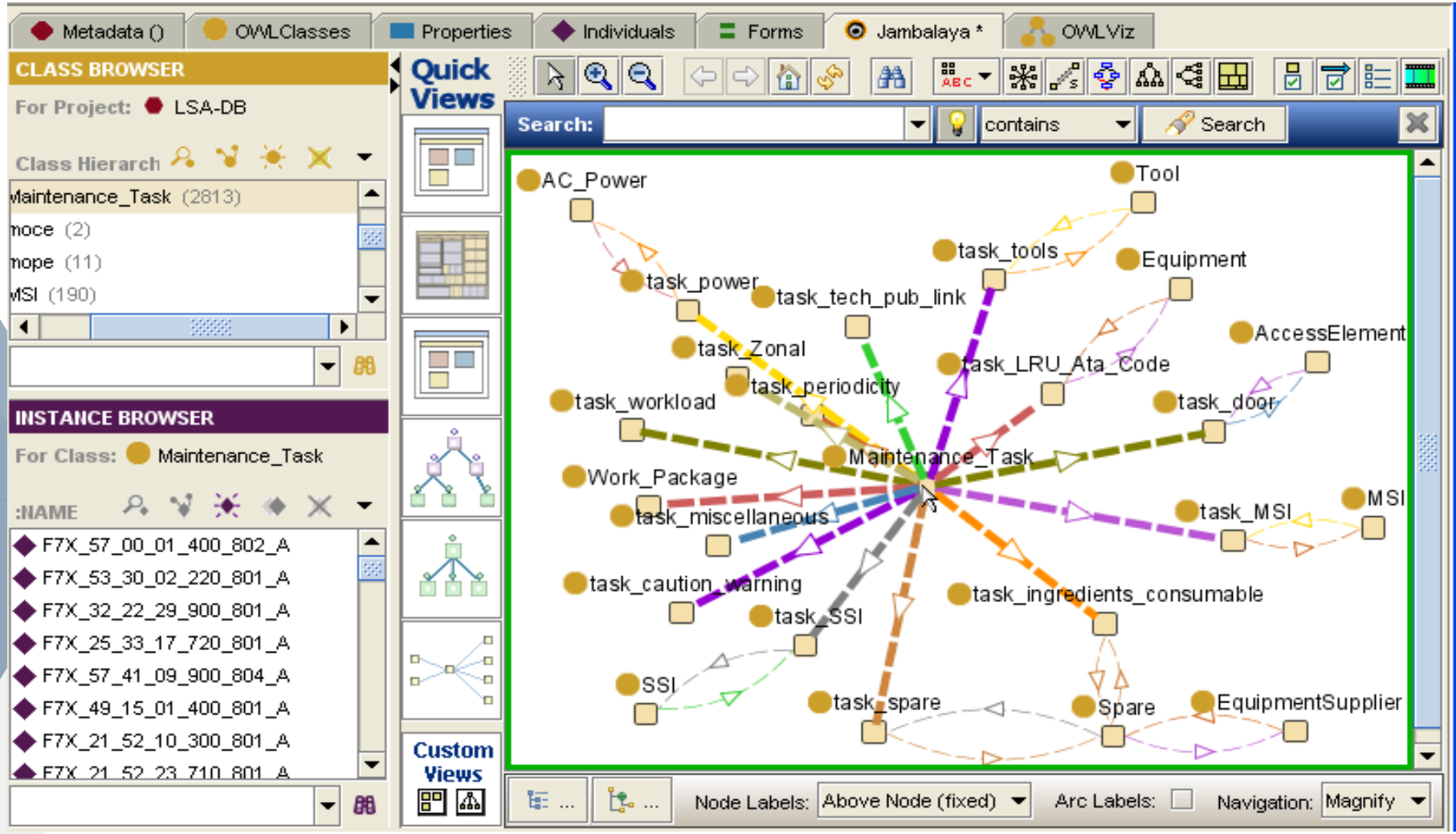


*Improving **semantic interoperability** in business processes through **Semantic technologies***

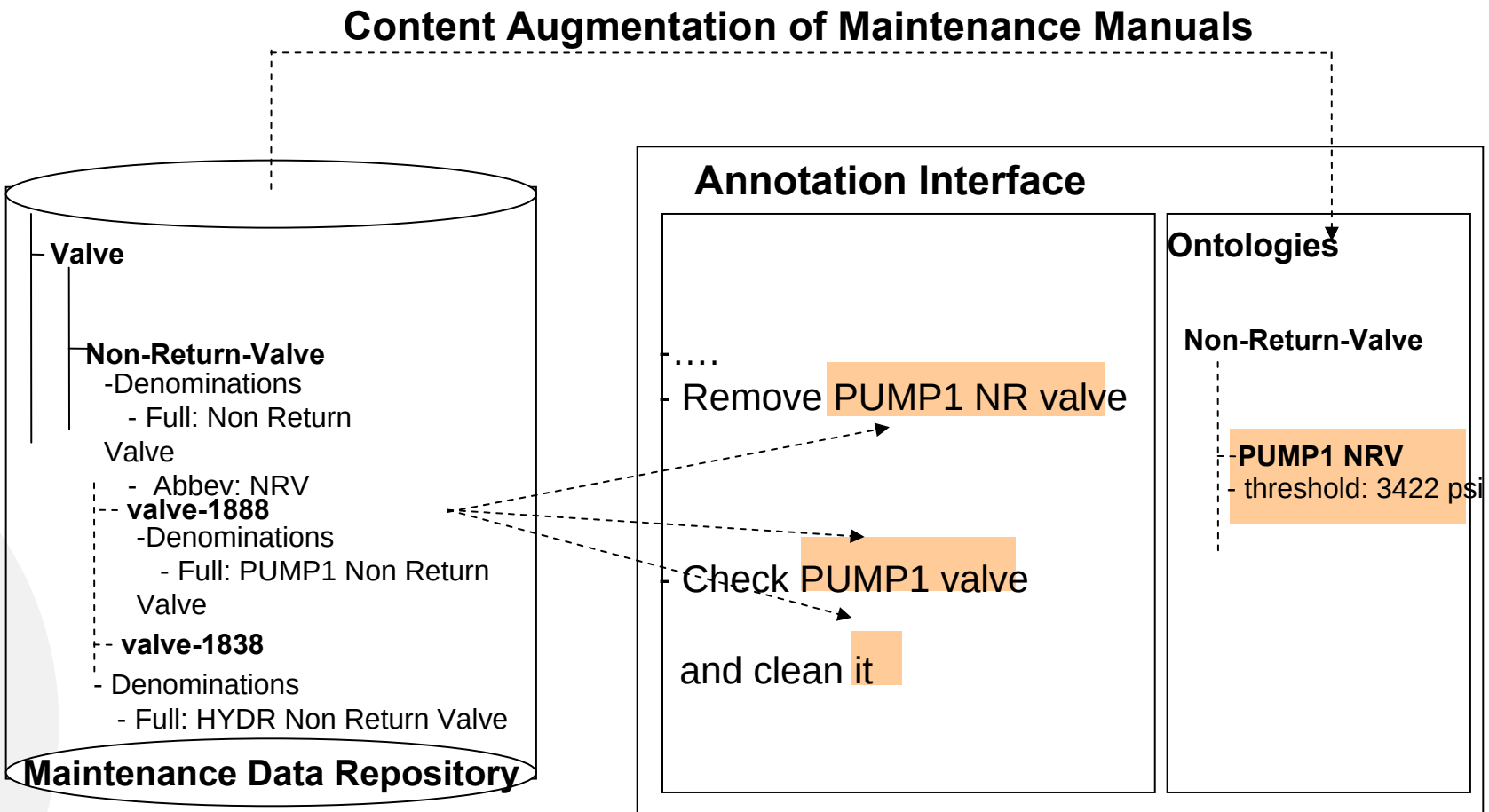
Example Learnt Ontology – 400k triples



Example Learnt Ontology (2)



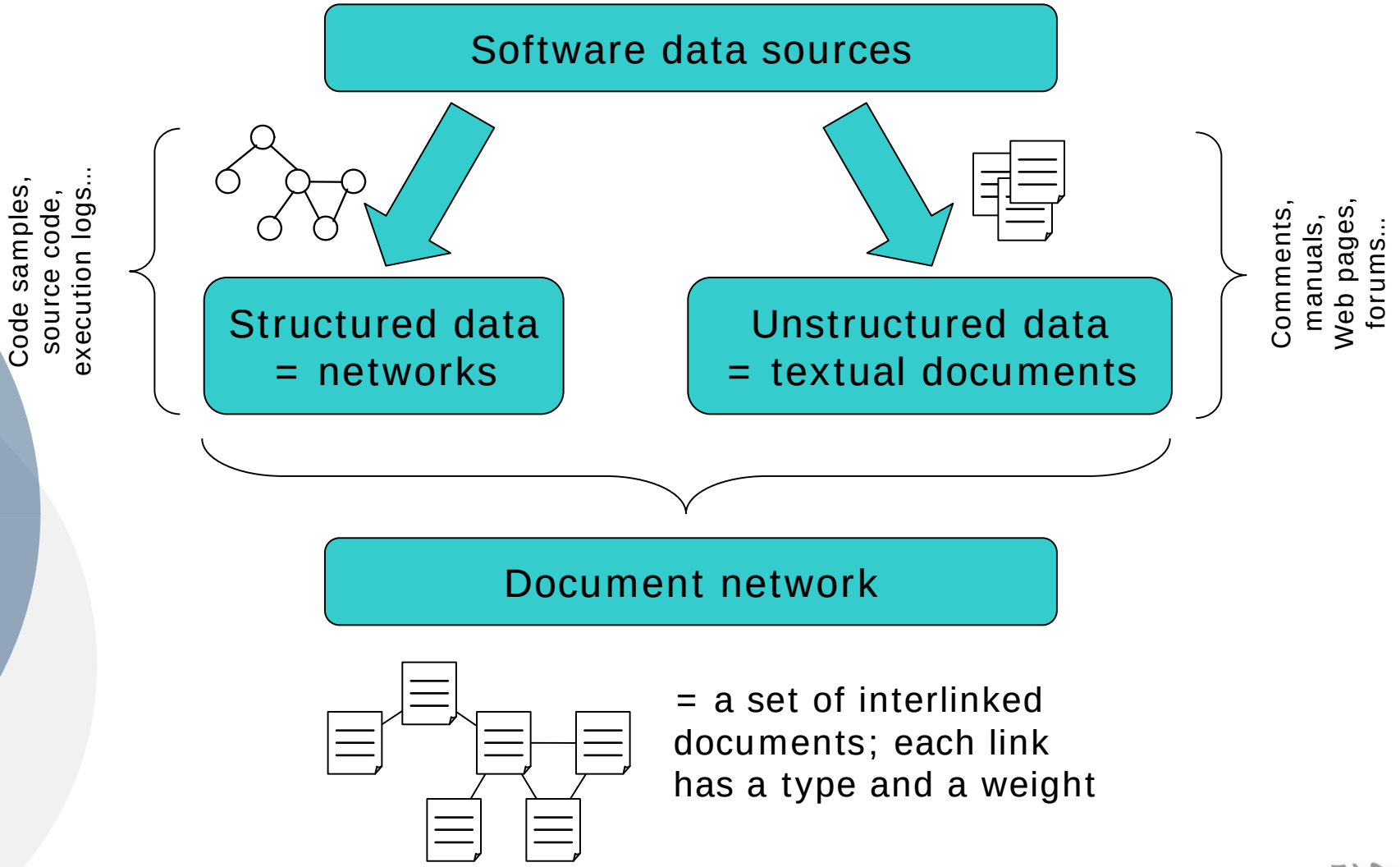
A Semantic Tagging Example



Transitioning Problems

- **From legacy databases to ontologies**
- **Towards semantic-based software engineering**
- Transitioning web applications towards ontologies and services

Learning Ontologies from Software Artefacts



Creating a Document Network

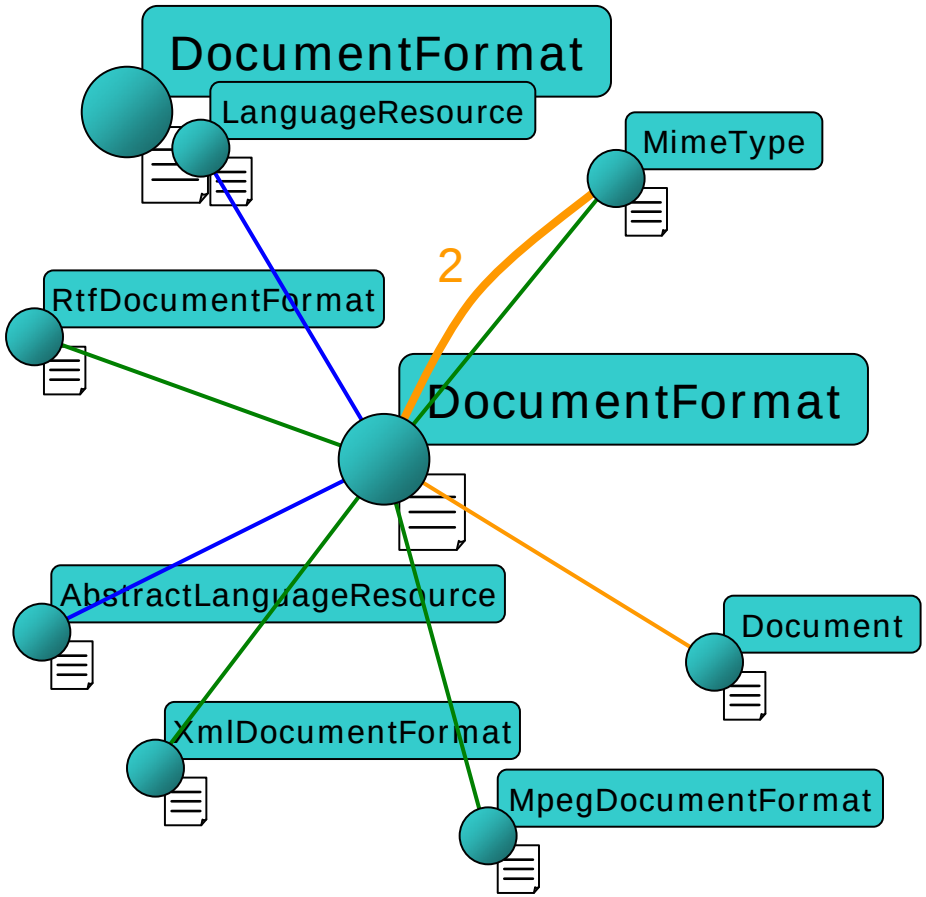
DocumentFormat.class

```
/** The format of Documents. Subclasses of DocumentFormat know about
  * particular MIME types and how to unpack the information in any
  * markup or formatting they contain into GATE annotations. Each MIME
  * type has its own subclass of DocumentFormat, e.g. XmlDocumentFormat,
  * RtfDocumentFormat, MpegDocumentFormat. These classes register themselves
  * with a static index residing here when they are constructed. Static
  * getDocumentFormat methods can then be used to get the appropriate
  * format class for a particular document.
  */
public abstract class DocumentFormat
extends AbstractLanguageResource implements LanguageResource{
```

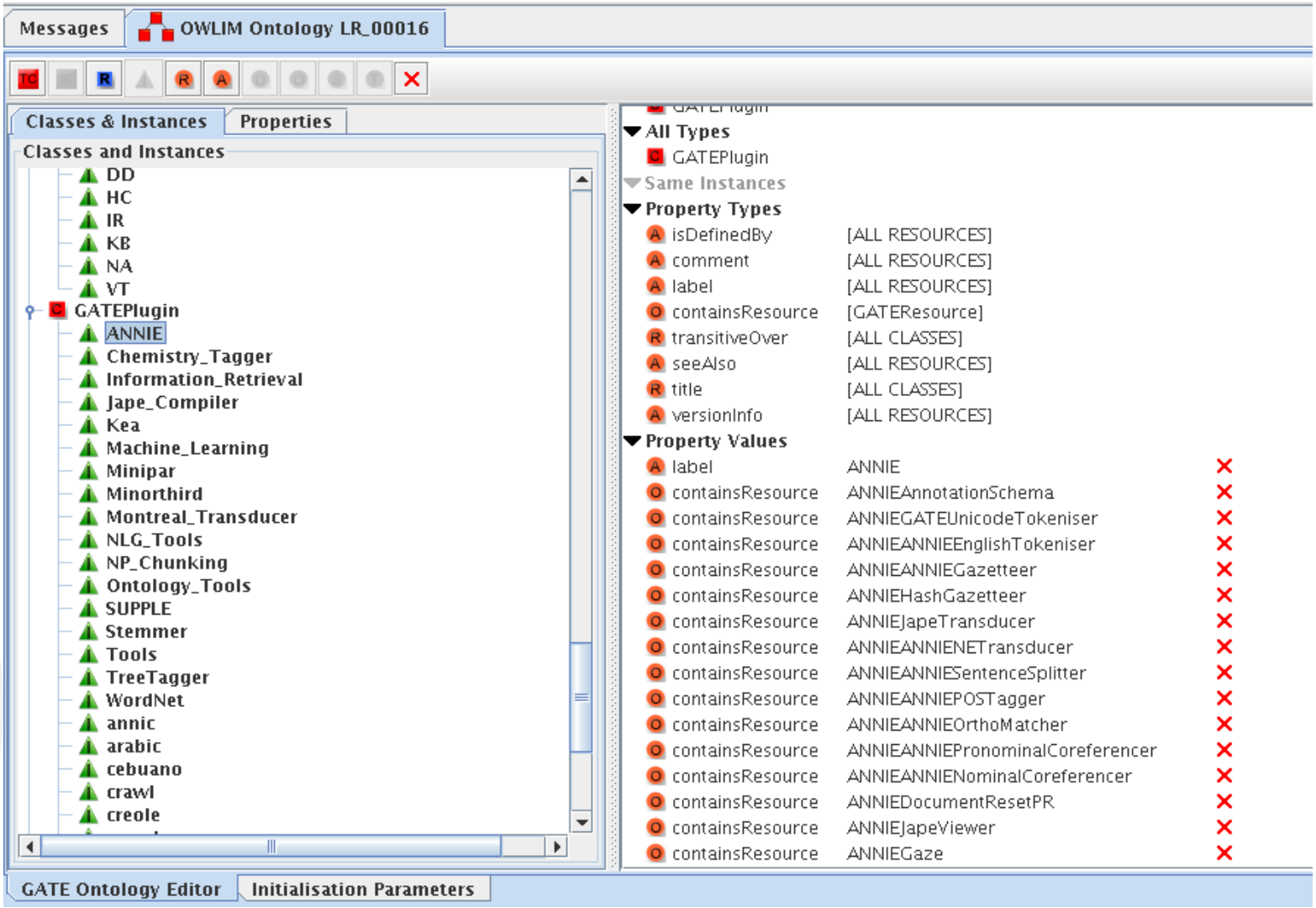
```
 /** The MIME type of this format. */
  private MimeType mimeType = null;
```

```
 /**
    * Find a DocumentFormat implementation that deals with a particular
    * MIME type, given that type.
    * @param  aGateDocument this document will receive as a feature
    *                       the associated Mime Type. The name of the feature is
    *                       MimeType and its value is in the format type/subtype
    * @param  mimeType the mime type that is given as input
    */
  static public DocumentFormat getDocumentFormat(gate.Document aGateDocument,
                                                 MimeType mimeType){
```

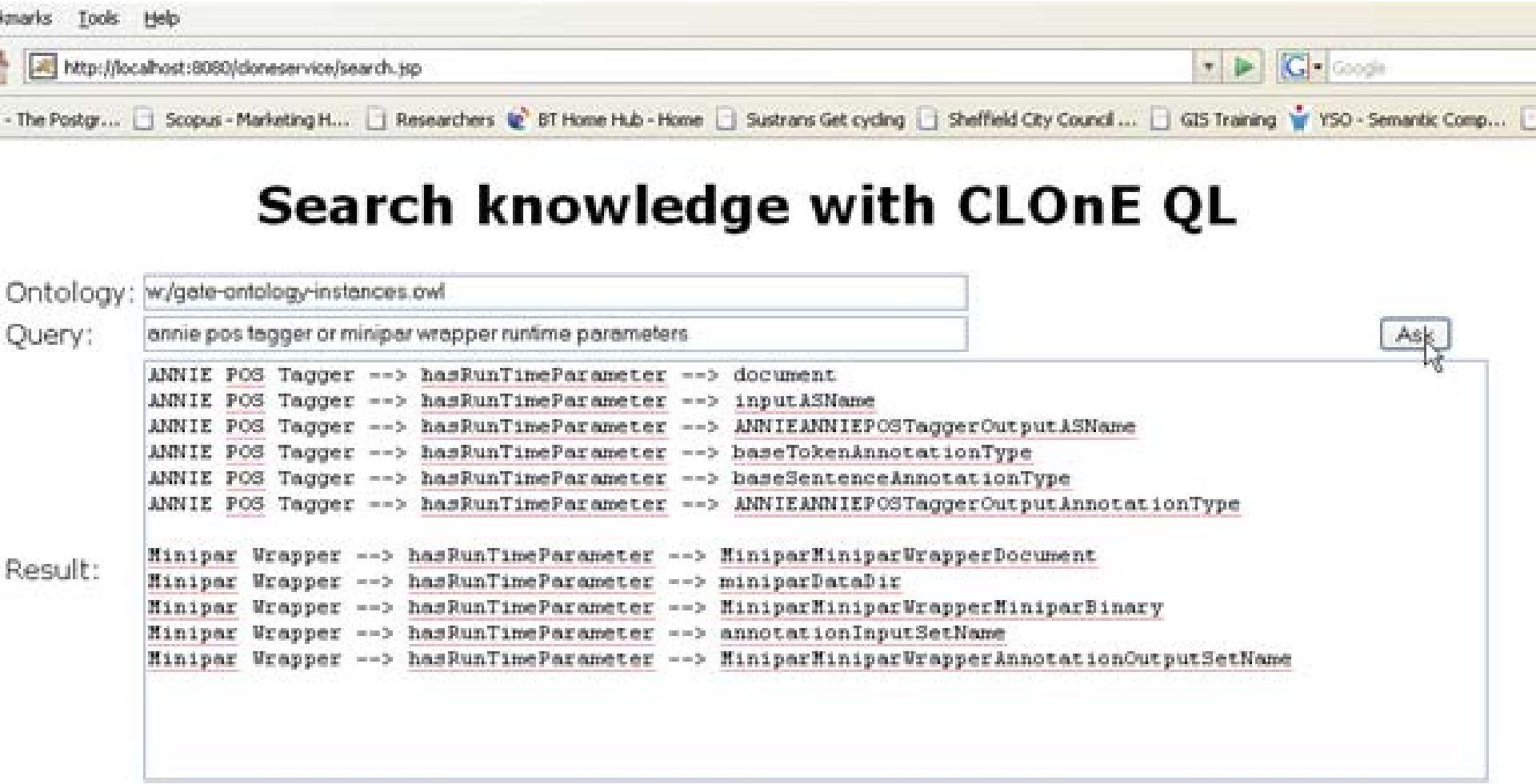
```
  } // getDocumentFormat(aGateDocument, MimeType)

} // class DocumentFormat
```



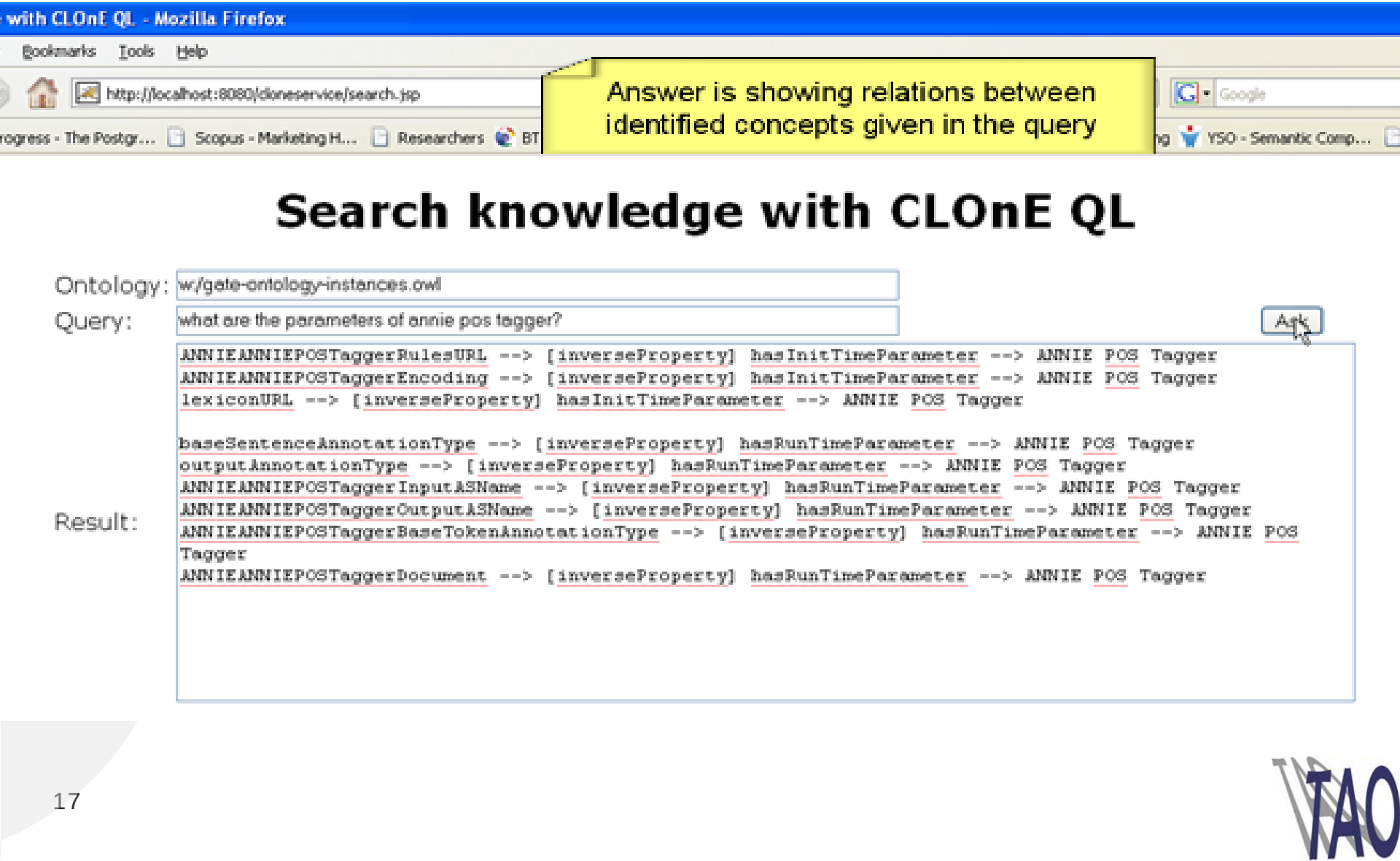
Ontology Learnt from Software Code



Google-like Conceptual Search (1)



Question-Based Conceptual Search (2)

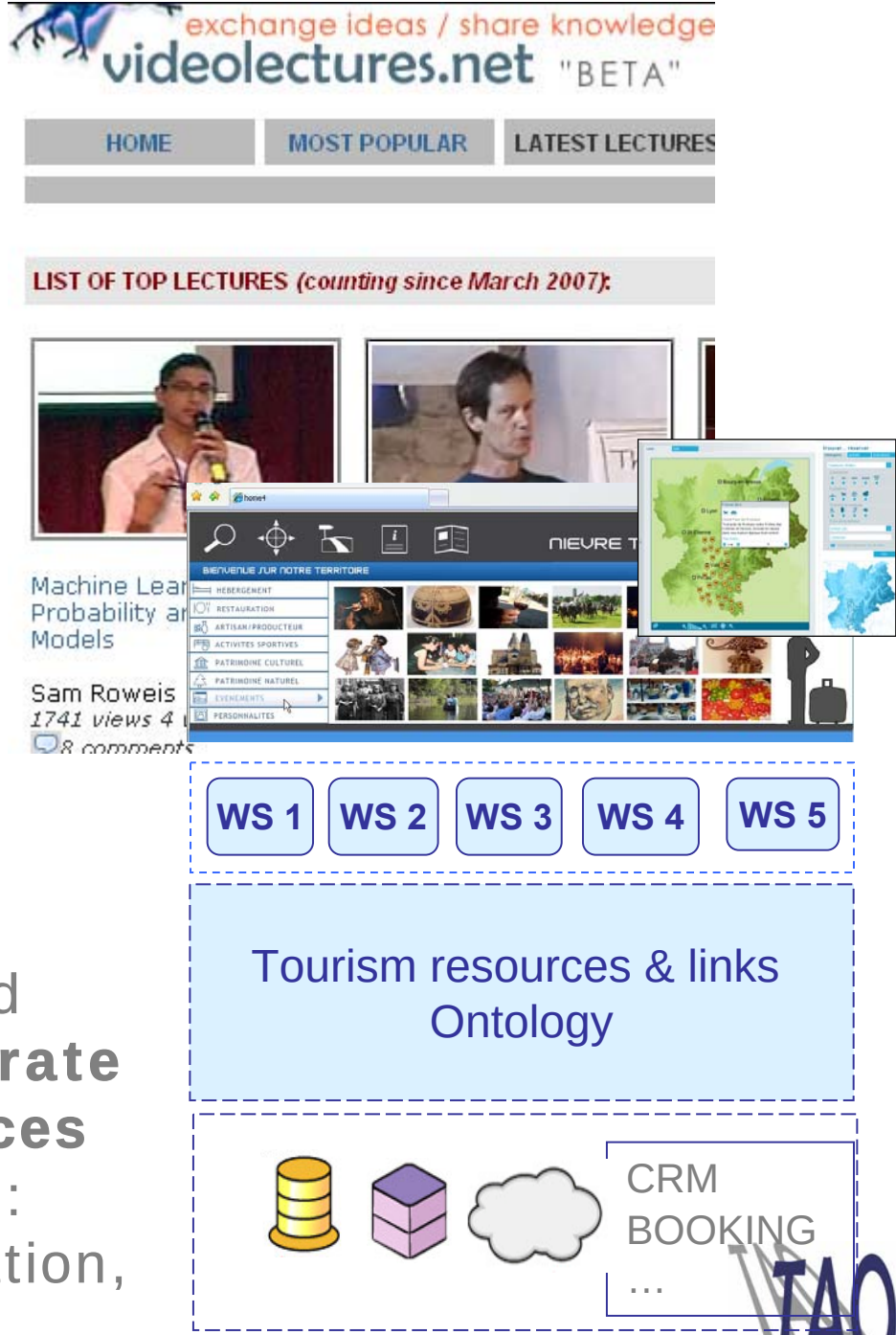


Transitioning Problems

- ✓ **From legacy databases to ontologies**
- ✓ **Towards semantic-based software engineering**
- ➢ **Transitioning web applications** towards ontologies and services

Transitioning Web Applications

- Legacy application:
  - database driven
  - no interoperability
- Ontologies + SOA:
  - **Learn ontologies**
  - Manage **complex resources** and **knowledge links**
  - Use Service Oriented Application to **integrate added value services** from other suppliers: cartography, translation, booking services…



Software Demonstrations on TAO Web site

- Learning ontologies from software code
- Supporting software developers with conceptual search
- RDB2Onto

Thank you! Questions?

This presentation + demos:

www.tao-project.eu/demos-dec07/

Kalina Bontcheva:

kalina@dcs.shef.ac.uk

Contact Information

For queries / further information, please contact the project co-ordinator:

Kalina Bontcheva

Department of Computer Science
University of Sheffield
Regent Court
211 Portobello Street
Sheffield S1 4DP

phone: (+44 - 114) 222 1930
fax: (+44 - 114) 222 1810
email: K.Bontcheva@dcs.shef.ac.uk