



ABAP Software Ontologies Virtual Community Day 2009

Tobias Trapp

Agenda

Introduction

Knowledge as Sets of Facts

Ontologies and Reasoning

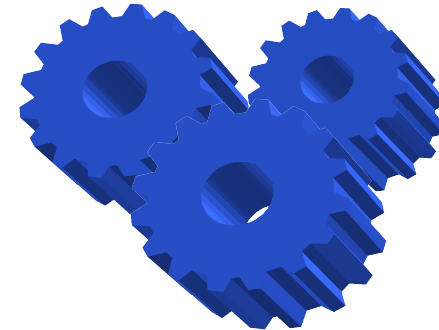
An Ontology for SAP Architecture



Use of Semantic Web Technology in Enterprise Architecture

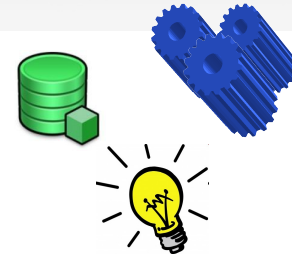
Elements of IT Systems are

- **DATA**
- **FUNCTIONS**
- **KNOWLEDGE**



Use of Semantic Web Technology in Enterprise Architecture

And what about SAP?

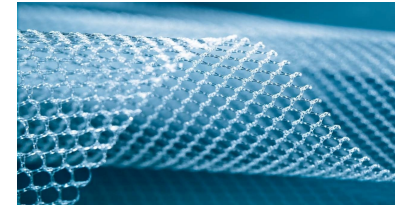


- SAP Business Suite is an integrated system containing **DATA & FUNCTIONS**
- It's the task of Business Process Experts to make IT „intelligent“ by doing customizing, process orchestration and implementing business rules
- In other words: he adds the aspect of business **KNOWLEDGE** to ERP

Semantic Web Technologies add Aspects of Knowledge Management to ERP & Netweaver

How can Semantic Web Technologies help?

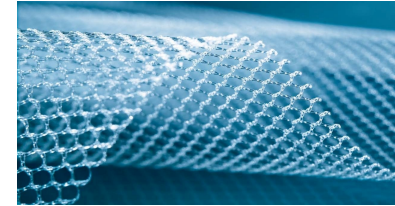
- Tagging, i.e. linking different data sources
- adding metadata to business objects for advanced process automation
- enabling collaboration
- visualization
- exploration & reasoning
- building repositories



Use of Semantic Web Technology in SAP Enterprise Architecture

In this lecture we learn how to

- visualize and explore the architecture of ABAP applications
- build repositories of systems & applications
- explore repositories by using methods of artificial intelligence

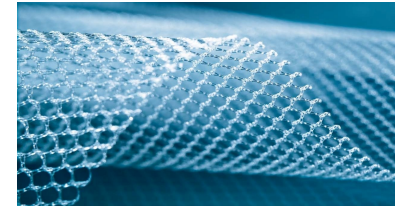


We support an enterprise architect

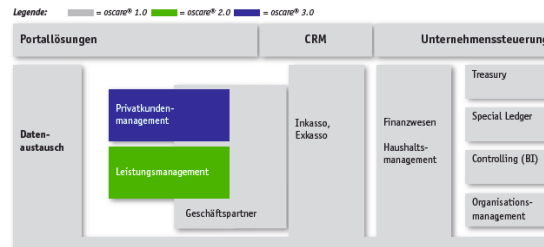
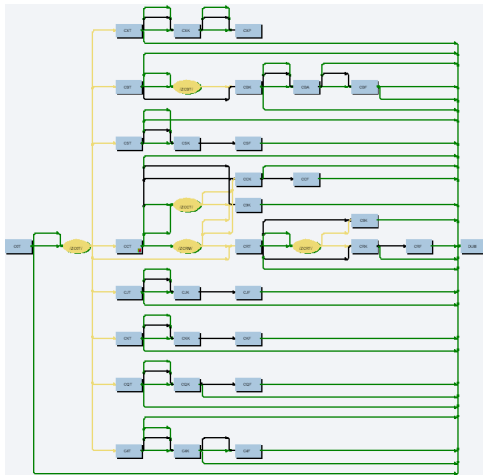
- doing architecting & change management,
- checking violations of architectural guidelines and
- putting his knowledge into a repository.

Status Quo

Current situation: information about systems, applications, software logistics are spread on different places:



- applications: SE80 (application hierarchy), excel sheets, bic pictures
- packages & dependencies: within SE80, on UML diagrams, PPT slides...
- software components: SLD, ST03, STMS...



Paket		CRM	gesichert
Eigenschaften			
Verwendungsdeklarationen			
Paketchnittstellen			
Erhaltene Pakete			
Pakethierarchie			
⊖	Pakethierarchie	Pake...	Beschreibung
⊖	CRM_APPLICATION	✗	Alle CRM Komponenten ohne spezielle Strukturpakete
⊖	_CRM_APPLICATION_DEFAULT	✗	Default Schnittstelle für CRM Application
⊖	_CRM_APPLICATION_FILTER	✗	Filter Schnittstelle für CRM Application
⊖	_CRM_APPL_VIRTUAL_DEFAULT	✗	virtuelle Default Schnittstelle für CRM Application
⊖	@BEV	✗	Bestandteil Schnittstelle Vertragsmanagement (BIC) SFA
⊖	@ICRMGOSTRUCTURES	✗	Generierte Strukturen und Tabellenplan
⊖	@BEABASICS	✗	Applikationsübergreifende Grundfunktionen
⊖	@BEACRM_IPM_DOCU	✗	IPM: Dokumenten von Billing-Engine-Applikationen
⊖	@BEACRM_IPM_DOCU2	✗	IPM: Dokumentation von Billing-Engine-Applikationen ()
⊖	@BEADOCU	✗	BE- Dokumentation Metadaten und zu generierende ()
⊖	@IBONAG_CUSTOMIZING	✗	Customizing für Bonusabsprachen
⊖	@IBONAG_CUSTOMIZING	✗	Bonusabsprachen: Pflege
⊖	@IBONAG_MAINTENANCE	✗	Bonusabsprachen: Pflege
⊖	@IBONAG_MAINTENANCE	✗	Bonusabsprachen: Pflege
⊖	@ICEMENTITLEMENTS	✗	Anrechte
⊖	@ICEMENTITLEMENTS_INTF	✗	Anrechte
⊖	@ICEMCRM_PROFILE_DET_USO	✗	Verwendung der Anrechtsprofilfindung
⊖	@ICEMCRM_PROFILE_DET_USO	✗	Verwendung der Anrechtsprofilfindung
⊖	@ICEMEE_COMMON	✗	Anrechtsmanagement- Allgemeine Objekte
⊖	@ICEMEE_COMMON_INTF	✗	Anrechtsmanagement- Allgemeine Objekte

Use Cases

Online demo

- We have a set of applications developed on different systems
- Some of these applications are cross section applications that are transported to other development systems
- We are looking for
 - an application, a certain package belongs to
 - packages developed on a certain system
 - packages which use a certain package indirectly
 - packages which are shipped in more than one software component
 - packages that do not use a certain package indirectly

Agenda

Introduction

Knowledge and Visualization

Ontologies and Reasoning

An Ontology for SAP Architecture



Modeling Knowledge using RDF

RDF consists of sentences coded as triples: subject – predicate - object

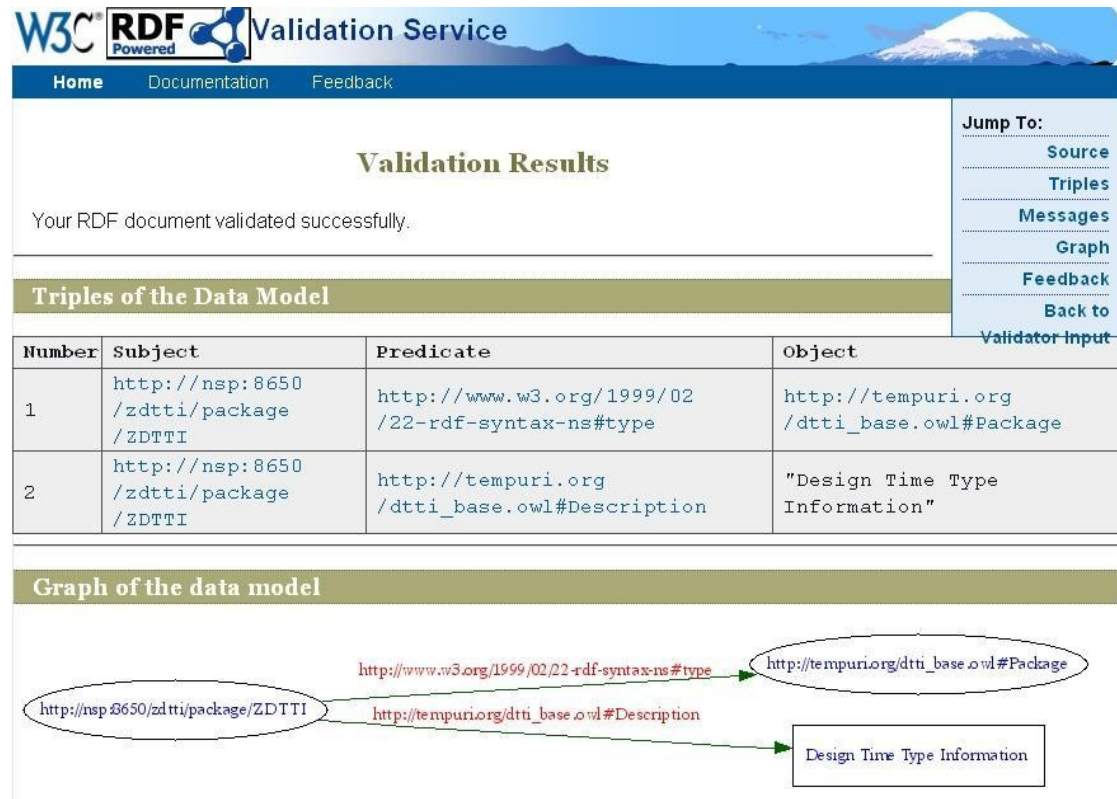
- *The ABAP development object ZDTTI is a Package with description 'design time type information' coded in RDF:*

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dtti_base="http://tempuri.org/dtti_base.owl#">
  <rdf:Description
    rdf:about="http://nsp:8650/zdtti/package/ZDTTI">
  <rdf:type
    rdf:resource="http://tempuri.org/dtti_base.owl#Package"/>
  <dtti_base:Description>
    Design Type Type Information</dtti_base:Description>
  </rdf:Description>
</rdf:RDF>
```



RDF Validation Service: <http://www.w3.org/RDF/Validator/>

- *validate RDF*
- *explore triples*
- *explore RDF graph*



W3C RDF Powered Validation Service

Home Documentation Feedback

Validation Results

Your RDF document validated successfully.

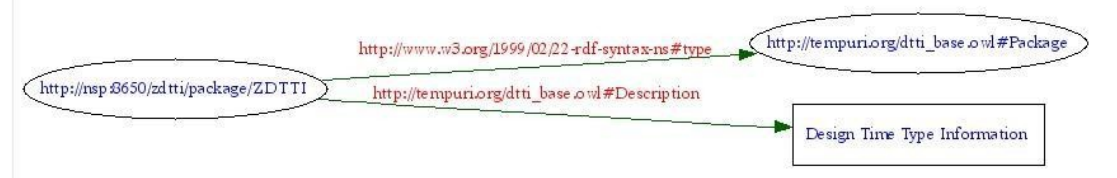
Jump To:

- Source
- Triples
- Messages
- Graph
- Feedback
- Back to Validator Input

Triples of the Data Model

Number	Subject	Predicate	Object
1	<code>http://nsp:8650/zdtti/package/ZDTTI</code>	<code>http://www.w3.org/1999/02/22-rdf-syntax-ns#type</code>	<code>http://tempuri.org/dtti_base.owl#Package</code>
2	<code>http://nsp:8650/zdtti/package/ZDTTI</code>	<code>http://tempuri.org/dtti_base.owl#Description</code>	"Design Time Type Information"

Graph of the data model



```

graph LR
    S1("http://nsp:8650/zdtti/package/ZDTTI") -- "http://www.w3.org/1999/02/22-rdf-syntax-ns#type" --> O1("http://tempuri.org/dtti_base.owl#Package")
    S1 -- "http://tempuri.org/dtti_base.owl#Description" --> O2["Design Time Type Information"]
  
```

RDF Basics

XML serialization of RDF data enables us to use the whole XML Middleware:

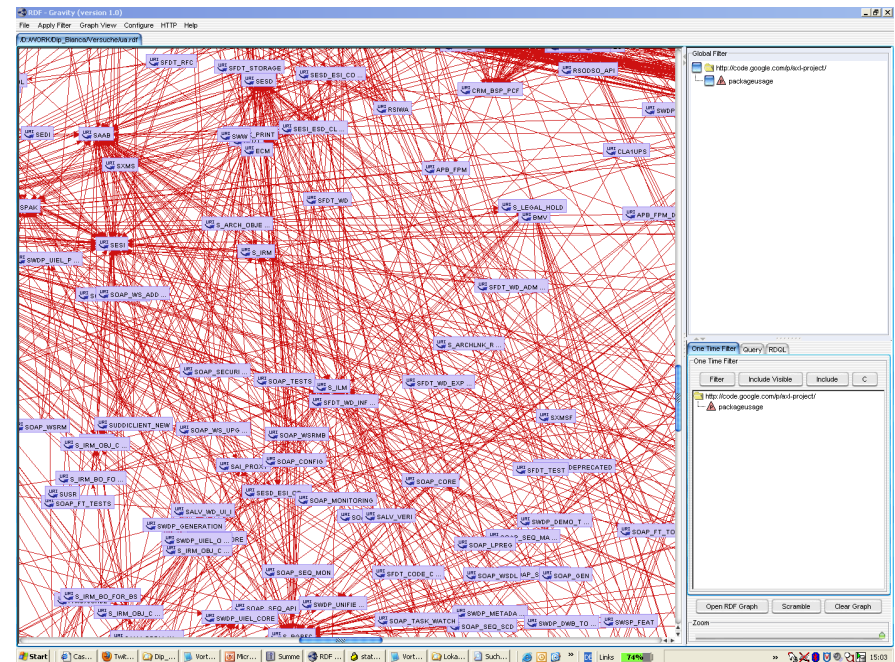
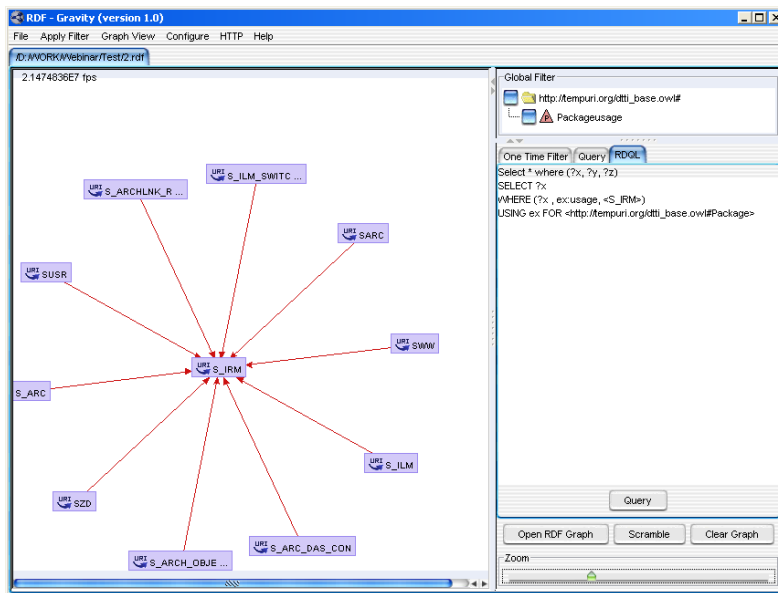
- generation and parsing of RDF using XSLT and Simple Transformations
- integration of RDF data in XML / XHTML documents:
 - add metadata to XML documents for process automation
 - tag your data for intelligent search
- save huge RDF datasets in XML databases
- expose Metadata by using REST Web Services:
 - you can see RSS 1.0 feed as a very simple REST Web Service.
 - the service provides metadata of as RDF triples.



Demo

Visualization :

- visualization of package dependencies within AS ABAP
- queries using RDQL and SPARQL



Agenda

Introduction

Knowledge and Visualization

Ontologies and Reasoning

An Ontology for SAP Architecture



Ontologies

Ontologies are concept models

- They allow us define classes (f.i. packages, systems), relations between classes (f.i. a packages uses another package) and individuals of a class.
- We can use methods of logics to define classes & queries:
„Computer, please give me all packages to use more then 10 other packages and that are used by more than 100 packages indirectly.“



Here we use OWL-DL to define Ontologies:

- OWL is a W3C standard based on RDF syntax
- OWL allows to import RDF facts and to classify them
- OWL-DL allows reasoning in conception models



Ontologies

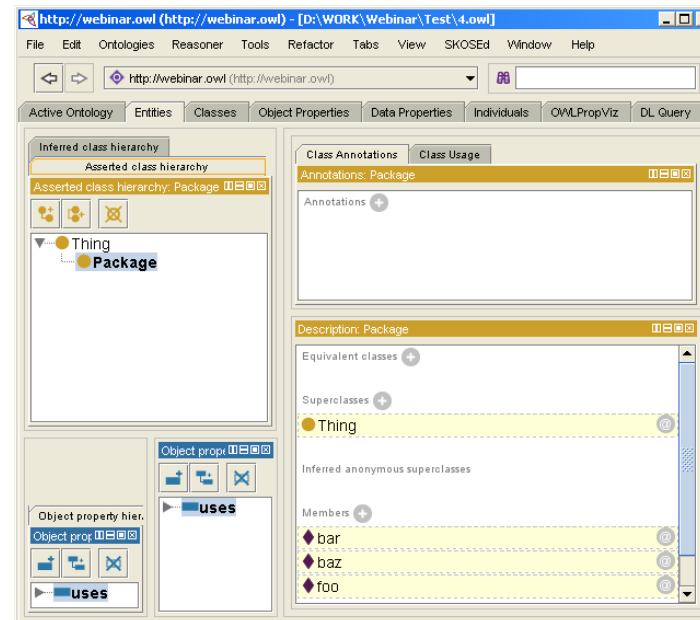
Creation of Ontologies



- Use an ontology editor and a reasoner, f.i.
 - *Protégé 4:* <http://protege.stanford.edu/>
 - *Pellet:* <http://clarkparsia.com/pellet>

Demo

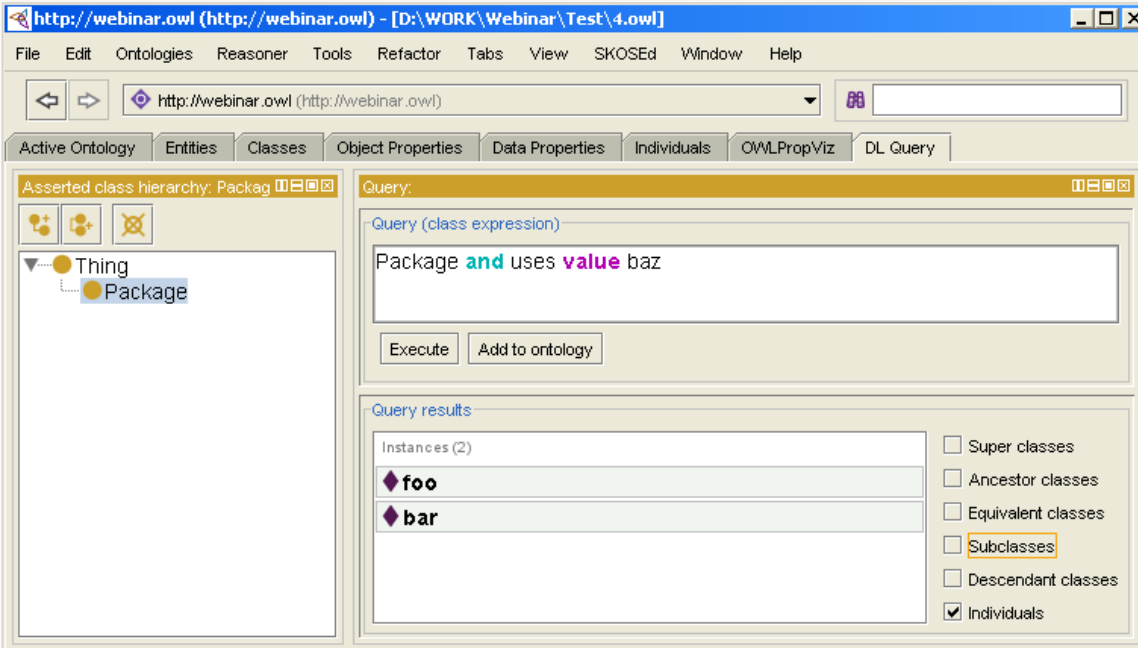
- Creation of a simple ontology
- We model packages and use accesses



Reasoning in OWL

Logical expressions

- We learn how to define classes with logical expressions
- We can use that mechanism for ad hoc queries

The screenshot shows the Protege OWL editor interface. The title bar indicates the file path: `http://webinar.owl (http://webinar.owl) - [D:\WORK\Webinar\Test\4.owl]`. The menu bar includes: File, Edit, Ontologies, Reasoner, Tools, Refactor, Tabs, View, SKOSEd, Window, Help. The address bar shows `http://webinar.owl (http://webinar.owl)`. The main interface has several tabs: Active Ontology, Entities, Classes, Object Properties, Data Properties, Individuals, OWLPropViz, and DL Query. The 'Classes' tab is active, showing an 'Asserted class hierarchy: Packag' with a tree view containing 'Thing' and 'Package'. The 'DL Query' tab is also active, showing a 'Query (class expression)' field with the text `Package and uses value baz`. Below the query field are 'Execute' and 'Add to ontology' buttons. The 'Query results' section shows 'Instances (2)' with a table listing 'foo' and 'bar'. To the right of the results are several checkboxes: 'Super classes', 'Ancestor classes', 'Equivalent classes', 'Subclasses', 'Descendant classes', and 'Individuals' (which is checked).



Agenda

Introduction

Knowledge and Visualization

Ontologies and Reasoning

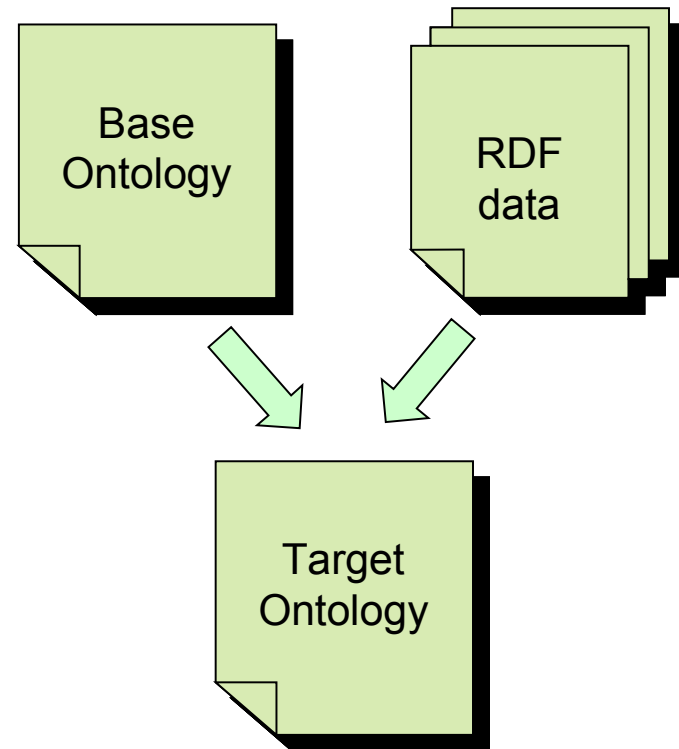
An Ontology for SAP Architecture



How to Design Software Ontologies

I suggest a hierarchical approach:

- Define an ontology of your main entities & properties: systems, software components, packages, dependencies...
- Import RDF data that is valid according to this base ontology. This data should be extracted from SAP systems automatically.
- Import all data to a target ontology:
 - use it to define your own concepts
 - model your system landscape
 - import additional data
 - do reasoning



DTTI Open Source Project

Design Time Type Information:

- DTTI stands for „Design Time Type Information“ and is an Open Source Project under Apache License.
- It contains a base ontology together with a set of REST Webservices based on ICF that expose RDF data about ABAP development objects.
- It can easily extended for other entities:
 - enhancements
 - switches
 - SAP frameworks like Business Data Toolset
 - development objects like transformations

Summary

Benefits of Ontologies

- Ontologies are approach to model system landscapes and architectures.
- They are based on open standards and supported by free tools.
- They can easily be extended and adapted to your specific needs.

We can use them for

- formal definition of architectural guidelines
- automated reasoning, f.i.
 - violations of architecture: are dependencies between packages properly so that we can install software components in linear order?
 - indirect dependencies
 - Are there dependencies between packages which are forbidden due to an architectural guideline?

Thank you for your Attention!

Questions? Criticisms ? Comments?

Reasoning in OWL

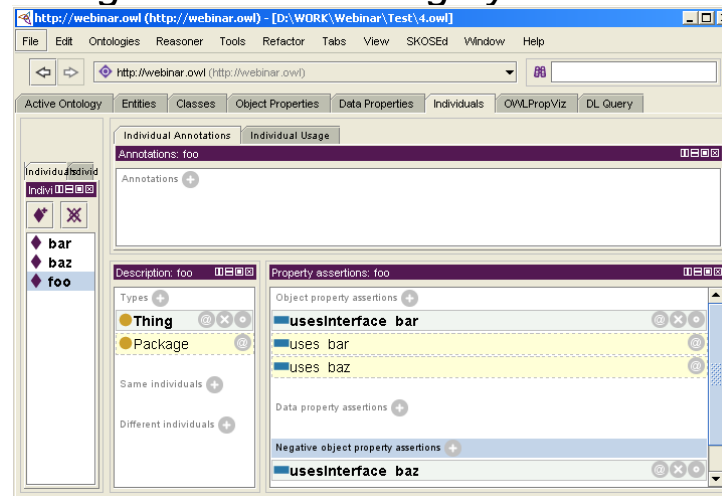
Open World Assumption

- If a set of facts doesn't contain a certain information it doesn't mean that it is false. As a consequence negations and „for all“ operators won't work.
- In database context we have complete knowledge – and so possibly in some parts of an ontology. In this case we should work with

- integrity constraints:

<http://clarkparsia.com/weblog/2009/02/11/integrity-constraints-for-owl/>

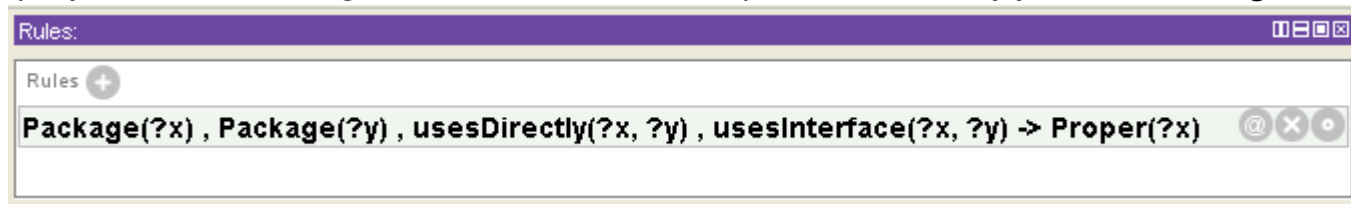
- negative properties



Advanced Topics

Rules & DL safe queries

- Use rules to simplify an ontology by using an decidable subset of SWRL (<http://www.w3.org/Submission/SWRL/>). There is support in Protegé:



- Rules are supported by Pellet reasoner:
<http://clarkparsia.com/weblog/2007/08/12/understanding-swrl-part-1/> and
<http://clarkparsia.com/weblog/2007/08/27/understanding-swrl-part-2-dl-safety/>
- You can use SPARQL-DL to query data:
<http://clarkparsia.com/weblog/2007/10/26/towards-sparql-dl-evaluation-in-pellet/>

Advanced Topics

Modularization

- You can modularize ontologies using `owl:import` command or using a tool like Protegé:

