A3-D5

Testbeds and Prototypes

Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Project number:	IST-2004-506779
Project instrument:	EU FP6 Network of Excellence (NoE)
Project thematic priority:	Priority 2: Information Society Technologies (IST)
Document type:	D (deliverable)
Nature of document:	P (prototype)
Dissemination level:	PU (public)
Document number:	IST506779/Hannover/A3-D5/D/PU/a1
Responsible editors:	Ingo Brunkhorst
Reviewers:	Charlie Abela, Stefania Ghita, Sergey Chernov
Contributing participants:	Hannover, Malta
Contributing workpackages:	A3
Contractual date of deliverable:	August 30th, 2005
Actual submission date:	September 15th, 2005

Abstract
This report documents the achievement of working group A3 - "Personalized Information Systems" to design and develop an early prototype of the Personalized Semantic Portal for REWERSE. The implemented software is used as a testbed for further development and evaluation of the developed framework and scenarios.

Keyword List
semantic web, reasoning, personalization, portal, information systems

Project co-funded by the European Commission and the Swiss Federal Office for Education and Science within the Sixth Framework Programme.

Testbeds and Prototypes

Fabian Abel[1], **Charlie Abela**[2], **Ingo Brunkhorst**[1], **Nicola Henze**[1]

[1] ISI- Semantic Web Group, University of Hannover,
Appelstrasse 4, D-30167 Hannover, Germany
henze,abel,brunkhorst@l3s.de

[2] Department of CS & AI, Faculty of Science,
Msida, Malta University Of Malta
charlie.abela@um.edu.mt

September 15th, 2005

Abstract
This report documents the achievement of working group A3 - "Personalized Information Systems" to design and develop an early prototype of the Personalized Semantic Portal for REWERSE. The implemented software is used as a testbed for further development and evaluation of the developed framework and scenarios.

Keyword List
semantic web, reasoning, personalization, portal, information systems

Contents

1 Outline of Report

This report documents the achievement of working group A3 - "Personalized Information Systems" to implement a prototype of the Personalized Semantic Portal for REWERSE as a testbed for further development and evaluation of reasoning for the Semantic Web. Section 2 starts with a description of the semantic portal prototype built according to the outline of the previous report [2]. A short summary of the used technology and data is provided therein. We selected the SWAD-E [13] portal software as the platform to re-use and extend for our portal approach. Section 3 gives a brief explanation of the futures of the chosen software platform, which is followed by a more detailed description of the extension we implemented using the SWAD-E software as the underlying infrastructure. Section 4 describes another approach for browsing semantically enriched data. Section 5 contains references to the developed prototype as well as pointers to additional media about the portal like videos, etc. After a conclusion, we list in the Appendix the peer-reviewed scientific papers about the research in this report.

2 Overview

Semantic Portals allow integrated and syndicated data views on information by using ontological knowledge and machine processable semantic descriptions. In particular, Semantic Portals should allow the user to customize the access to information by making advanced use of the semantic descriptions of the information, and should provide individualized views on the data, enhance user awareness and orientation, help the user in detecting relevant information or relations, etc. In the last report [2] we already outlined a first realization of a Personalized Portal for REWERSE (*REWERSE-PP*): The Personal Publication Reader [3]. The Personal Publication Reader focussed on displaying publications, contextual and related information, and shows how it relates to the other parts of the project. We have identified several scenarios where the support of a *personalized* portal will be helpful. These scenarios deal with specific functionality for the REWERSE portal and for REWERSE research achievements like publications, but also for project-related events, like working group meetings, deliverables, etc. For this, the portal relies heavily on the already introduced Semantic Web Ontologies:

Researcher Ontology: This ontology models persons and their involvements in projects, working groups or other organizations. At the moment it contains about 200 instances which represent members of the REWERSE project. The Researcher Ontology is accurately described in [2].

Semantic Web Glossary Ontology: Models Semantic Web terms and their relations between each other. As Figure 1 illustrates, the relation between the concepts is realized by a class hierarchy whereas human readable information is added by *Annotation Properties*.

Bookmark Ontology: This ontology contains concepts that are necessary to describe and evaluate websites. It allows statements like:

```
(Person, isAuthorOf, Note)
(Note, isReviewOf, Bookmark)
(Bookmark, hasTopic, Topic)
```

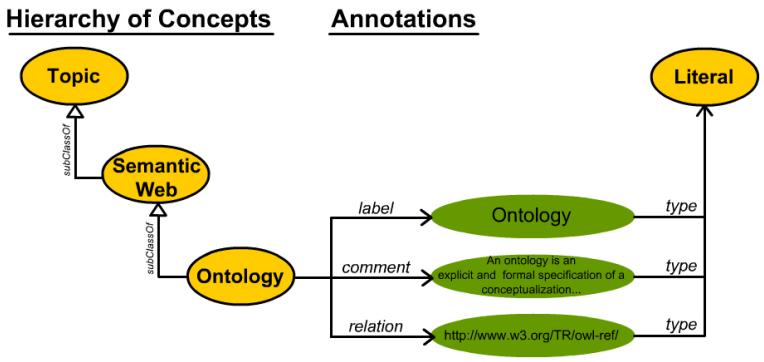


Figure 1: Extract of the Semantic Web Glossary Ontology

The aim of our research work was to build a portal for the REWERSE project, which enables users to access information provided by these ontologies and portal specific data in a personalized way and further let the user benefit from the schema information which is defined in the ontologies. These benefits should include context based navigation through the RDF Data and the ability to query the RDF Data in a schema-specific way. These ontologies are already applied to several applications, such as the Researcher Ontology which is used by the Personal Publication Reader to gather additional information about authors of specific publications. In addition to this project-wide information, we made use of *portal specific data* to represent News, Appointments and TODOs, and provide *personalized* information to the user, the RADAR application.

3 Semantic Portal Prototype

To realize the described kind of portal, we used a portal software which was developed for the SWAD-E demonstrator called *"The Semantic Web Environment Directory"* (SWED[1]). The *portal specific data* we define as:

News: actual information of a Project, Person, Working Group, etc.

Appointments: dates which have a certain group of participants

TODOs: tasks that have to be done until a specific deadline

All of these concepts are special *RSS Items*[4] which refer to our ontologies. News, Appointments and TODOs are exhibits of the interoperability between our different ontologies, as illustrated in Figure 2.

SWAD-E Semantic Portal A slightly abstracted architecture overview of the SWAD-E Semantic Portal is shown in Figure 3.

The *Content Aggregator* (Harvester) is used to read in distributed RDF Data via the Jena API. This data is encapsulated in a special model (*Datastore*), which is accessible through

[1]SWED: `http://www.swed.org.uk/` (08/2005)

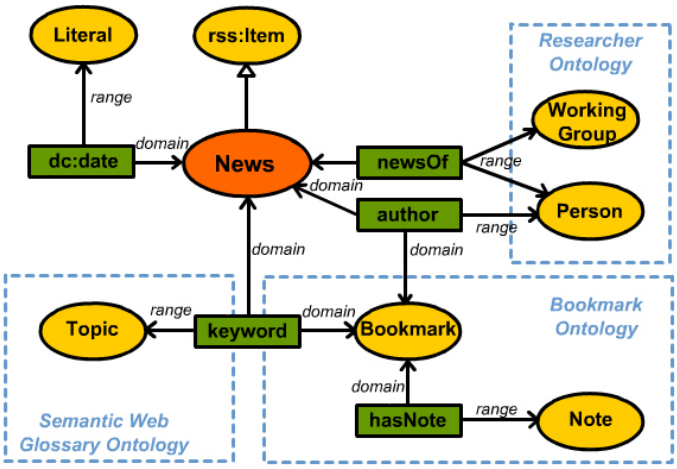


Figure 2: News refer to several ontologies

the portal definition (*Datasource*). This data source provides access to the whole dataset including the schema definitions (Domain-specific Ontologies). Using queries and filters it is possible to extract specific data subsets from the model. *Visualization Templates* based on Velocity[2] are used to visualize those data subsets which match specific filters. Besides the *Core Portal Software*, the SWAD-E Semantic Portal also offers data creation functionality which was originally implemented for the *Semantic Blogging*[3] project. Designed as a web interface it guides the *Content-Creator* (see Figure 3) through the creation of new portal content and on completion it notifies the Harvester of the portal.

[2]Apache Jakarta - Velocity: `http://jakarta.apache.org/velocity/` (08/2005)
[3]Semantic Blogging: `http://jena.hpl.hp.com:3030/blojsom-hp/blog/` (08/2005)

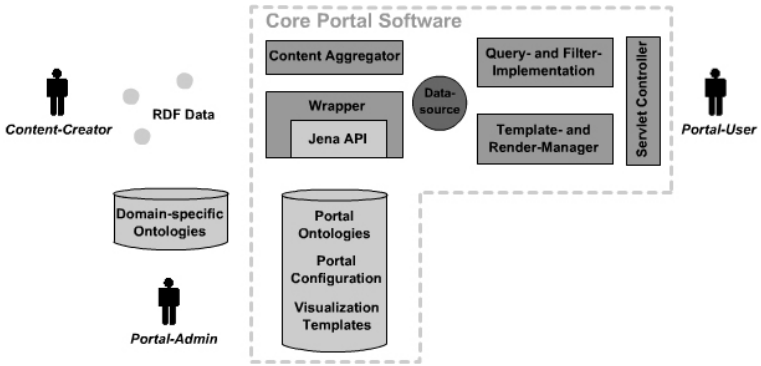


Figure 3: SWAD-E Semantic Portal Architecture

3.1 Portal Customization

To customize the SWAD-E portal software three main tasks had to be performed: Definition of the Datasource, of the Facets and the Visualization Templates. Modification of the portal configuration is done by modifying entries in the portal's configuration files, which are essentially RDF files using the N3 [5] syntax:

1. **Defining the Datasource:** Specifying the attributes of the portal (i. e. name) and announcing which Ontologies and Data should be imported: i

```
01:  [] rdf:type pcv:DataSource;
02:              rdfs:label "REWERSE Portal";
03:     ...
04:               pcv:sourceURL
05:                   <../ResearcherInstances.rdf>;
06:               pcv:ontologySourceURL
07:                   <../ResearcherOntology.owl>;
08:     ...
```

   Lines 5 and 7 refer to the location of the files containing the metadata resources and the associated ontology.

2. **Defining Facets:** Facets are certain attributes of a filter (*matching patterns*). A filter is composed of an arbitrary number of stateful facets. The following excerpt illustrates a facet that allows to filter instances of the Researcher Ontology by *rdf:type*:

```
01: pcv:facet [
02:         a pcv:HierarchicalFacet;
03:          rdfs:label "Persons.. ordered by type";
04:           pcv:linkProp  rdf:type;
05:           pcv:widenP    rdfs:subClassOf;
06:           pcv:facetBase rewerse:ResearcherConcept;
07:     ...
08: ];
```

3. **Defining Visualization Templates:** Visualization Templates have to be written for the three different states the portal can be in:

   Initial: This is the initial status. At this starting point the portal user has not selected any filter to extract data.

   Matching Results: After applying a filter the portal user can refine his search or select a specific *RDF Resource*.

   RDF Resource: Having selected a certain RDF Resource the portal user can catch up about this resource by applying different views (e. g. *graph view*: pictures the RDF Resource as a graph).

   A *Velocity Template* is a mix of HTML and the *Velocity Scripting Language* which allows to call Java Functions from within the template. The following code is an extract of such a template (researcherSummary.vm) in which the transformation of an instance of the Researcher Ontology into viewable HTML is described:

```
01: ...
02: <a href="$resource.getPageLink($request)">
03:   $resource.getProperty("rewerse:name").value.name
04: </a>
05: <br/>
06: <b>Type:</b> $resource.getPropertyValue("rdf:type")
07: <br/>
08: #set($email=${resource.getPropertyValue("rewerse:eMail")})
09: #if ($email != "")
10:   <b>E-Mail:</b> <a href="mailto:$email">$email</a>
11: #end
12: ...
```

The variable `$resource` is a RDF Resource whose properties (e.g. `rdf:type` and `rewerse:eMail`) should be displayed to users of the portal. If this template is processed it generates HTML-Code which is visualized as shown in Figure 4.

Figure 4: Result of a processed Velocity Template

In the portal configuration file the defined templates can be applied to a special context and a particular type of a RDF Resource:

```
01: pcv:template [
02:     a pcv:Template;
03:       pcv:templateContext "default";
04:       pcv:templatePath <portal://templates/researcherSummary.vm>;
05        pcv:templateClass rewerse:Person;
06: ];
```

The *template context "default"* (line 03) corresponds to the portal state for *Matching Results*. Thus this template visualizes RDF Resources of type `rewerse:Person` (line 05) during the search process, where it is useful to visualize only a short summary of the resources.

After these three steps are fulfilled and the Look And Feel is adjusted in a Cascading Stylesheet the portal customization is finished. Additionally the *Data Creation Functionality* of the Semantic Portal is customized. The types of objects the creation component should be able to produce is configured in a main configuration file:

```
01: # News
02: []
03:   a itemtype:ItemType ;
```

```
04:   rdfs:label "News";
05:   itemtype:hasConfigFile "/config/news.n3" ;
06:   itemtype:hasScopeNote
07:      "<b>News</b> can be news of a Working Group, Project, Person etc.";
08:   itemtype:hasPriority "1";
09:   itemtype:hasConfigLanguage "N3" .
10:
11: # ToDos
12: []
13:   a itemtype:ItemType ;
14:   rdfs:label "ToDo";
15:   ...
16: # Appointments
17: ...
```

Only portal specific data is defined (*News*, *TODOs* and *Appointments*) for data creation purposes. For each of this data type we further had to compose a specific configuration file which outlines the data type and its properties in detail. The following listing is an extract of the configuration file for News Items:

```
01: ...
02: itemtype:hasField
03: [
04:   a itemtype:Field;
05:   itemtype:controlsProperty rdf:type ;
06:   itemtype:hasStyle "simpleField" ;
07:   itemtype:hasDefault rnews:NewsItem;
08:   rdfs:label "news type " ;
09:   itemtype:visible "false" ;
10:   ...
11: ];
12: ...
13: itemtype:hasField
14: [
15:   a itemtype:Field;
16:   itemtype:controlsProperty rss:title ;
17:   itemtype:hasStyle "simpleField" ;
18:   rdfs:label "News Title<em>*</em>" ;
19:   itemtype:hasScopeNote
20:     "This should be a significant news title" ;
21:   itemtype:hasPriority "2";
22:   itemtype:hasScope "Item";
23:   itemtype:hasValue "Literal";
24:   itemtype:visible "true" ;
25:   itemtype:hasUIType itemtype:Field ;
26:   itemtype:isRequired "true" ;
27: ] ;
28: ...
29:     itemtype:hasField
30: [
```

```
31:   a itemtype:Field;
32:   rdfs:label "Author";
33:       itemtype:hasStyle "tree" ;
34:   itemtype:controlsProperty rnews:hasAuthor ;
35:   itemtype:hasScopeNote
36:             "Who is the author of this news-feed item?
    Please select a Person from the tree list.";
37:       itemtype:hasValue "Resource";
38:   itemtype:hasUIType itemtype:Tree;
39:   itemtype:hasVocab
40:   [
41:     a itemtype:Vocab;
42:     rdf:value "vocabs/Researcher.rdf";
43:     itemtype:hasConceptType rewerse:Person;
44:   ];
45:   ...
46: ];
47: ...
```

Description of the configuration file excerpt

02-11 - rdf:type The type of a News is `rnews:NewsItem` (line 07). This value is default and cannot be edited by the user (`itemtype:visible "false";` - line 09).

12-27 - rss:title Each NewsItem is a special *rss:Item* and thus has the property `title` which has to be entered by the user (*required property* - line 26). A label and a description (`rdfs:label` and `itemtype:hasScopeNote`) guide the user by entering a value (of type *Literal*) into an input field (`itemtype:hasUIType itemtype:Field`).

29-46 - rnews:hasAuthor The property `rnews:hasAuthor` refers to `rewerse:Person` (line 43), thus an instance of the Researcher Ontology. This instance can be selected by the user from a tree (`itemtype:hasUIType itemtype:Tree;`) which is filled with all Persons of the Researcher Ontology (line 39-44).

According to these configurations the data creation functionality generates RDF files from the users entries:

```
01: <rdf:RDF ... >
02:   <rnews:News rdf:about="http://www.personal-reader.de/.../data-entry.html">
03:     <dc:date>2005-01-08</dc:date>
04:     <rss:title>Data Entry with SWAD-E Semantic Portal</rss:title>
05:     <rss:description>
06:          If you want to submit new data to the portal you can...
07:     </rss:description>
08:     <rdf:type
    rdf:resource="http://rewerse.net/ontologies/RewerseNews#NewsItem"/>
09:     <rnews:hasAuthor
    rdf:resource="http://www.personal-reader.de/rewerse#abelFabian"/>
10:     <rnews:keyword
    rdf:resource="http://www.personal-reader.de/rdf/Glossary.owl#Semantic_Web"/>
11:   </rnews:News>
12: </rdf:RDF>
```

The generated RDF Data is aggregated by the Content Aggregator of the SWAD-E Semantic Portal (*Harvester* - see Section 3) but it is also stored to be harvested by other content aggregation software, such as a common *News Aggregators*.

3.2 Portal extension

Next to the common portal customization we also extended the functionality of the SWAD-E portal software, e.g. with implementing the *Date Facet* which is used to filter News according to the creation date (`dc:date`). In a second iteration packages were developed to accomplish personalization. These packages provide:

User Management: Enables the portal to determine which registered users are online. Registered users are associated with the corresponding instance of the Researcher Ontology, which is a necessity for the RADAR to work.

Personal Filters: When a user is logged in, he can use predefined personal filters to navigate to e.g. News of Working Groups he is involved in.

The RADAR: Calculates and visualizes the distance between two Persons. We distinguish between:

- *Browsing Distance:* Browsing distance is counting the number of clicks it would take for a user to navigate to the same pages (representing RDF Resources) other users have just visited (see Figure 5)

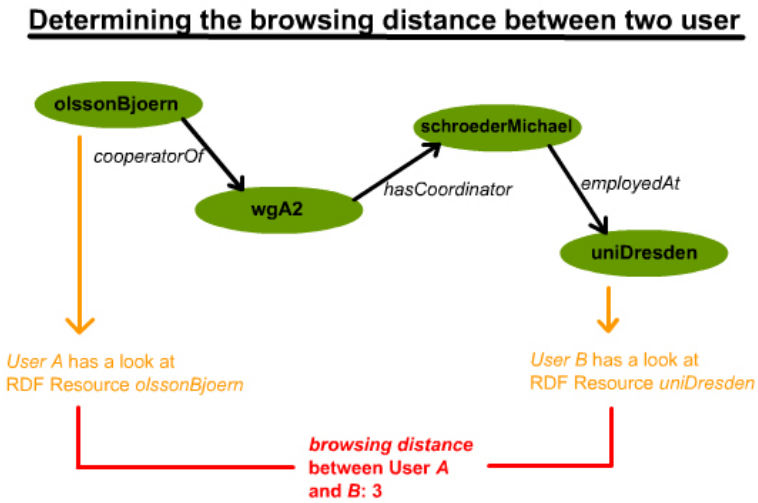
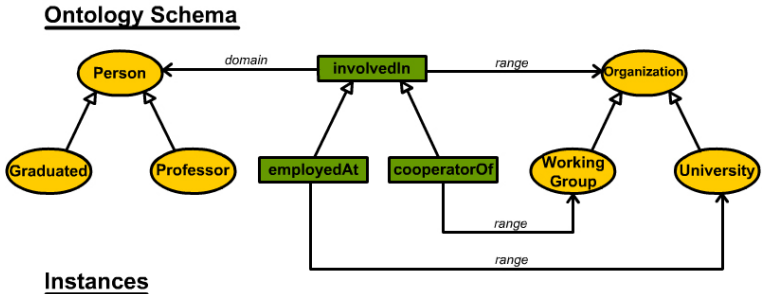


Figure 5: Determining the browsing distance

- *Professional Distance:* Professional distance is a measure for the relation a user has to other users of the portal (and members of the project). The process of determining the professional distance by using the Researcher Ontology is outlined in Figure 6. In the example, the relations of the user `abelFabian` are compared to the relations

of two other users. Each matching element contributes to the final score, a measure for similarity.



Instances

id	type	[...]	employedAt	cooperatorOf	involvedIn
abelFabian	Graduated		uniHannover	wgA3	REWERSE, uniHannover, wgA3
brunkhorstIngo	Graduated		l3s	wgA3	REWERSE, l3s, wgA3
henzeNicola	Professor		uniHannover	wgA3	REWERSE, uniHannover, wgA3

Professional Distance between *abelFabian* and other users

	Number of Matches				
target	[...]	employedAt	cooperatorOf	involvedIn	scoring
brunkhorstIngo		0/1	1/1	2/3	0.66
henzeNicola		1/1	1/1	3/3	1.0

Figure 6: Determining the professional distance

4 The semantExplorer

Web Page annotation, using domain specific ontologies is the basis of the Semantic Web, and besides RDF and RDFS, OWL has emerged as the most common language for defining relationships between resources in a web page. A Semantic Web Browser will deal with the annotations embedded in the Header of the HTML code for a web page, and is able to relate and aggregate information about resources located in different web documents. On the other hand a Semantic Web Browser can be described as a special web browser, that augments standard web browsers with the ability to visualize hidden metadata. The approaches are not limited to present WWW resource sharing technologies, but could involve special repositories collecting RDF triples from various accessed locations over time. Such triple stores could largely improve the efficiency of locating information on some resource of interest. The semantExplorer is based on a number of architectural components including an extractor and parser for retrieving the metadata from the head of the document and a database to store or update freshly discovered

triples. It features a number of builders, to generate different visualizations for the user based on the current document and the triple database. Table and graph visualization provide different representations of the viewed document, while the “Lens” Builder enables navigation to related data.

Figure 7 shows the components of the semantExplorer.

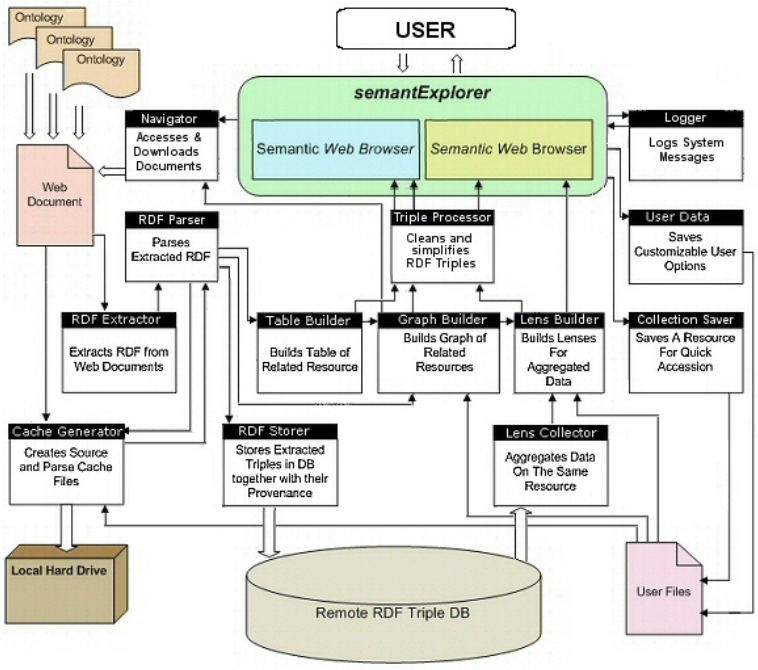


Figure 7: Architectural components of the semantExplorer

Like any other browser, the semantExplorer includes a Navigation Panel (Back, Forward, Stop, Refresh and the Address Bar) that provides standard document navigation for ’.html’ web pages, and ’.rdf’ and ’.owl’ ontologies. Once a document is loaded successfully, any RDF content is extracted and parsed. Resources described through this metadata are listed in the ’Defined Objects’ List. In the case of Ontologies, defined classes and properties will also be listed. Figure 8 shows the Web Browser View of the semantExplorer, which will display standard html documents, as well as RDF and OWL ontologies and provide standard navigation via hyperlinks.

Additional views include *Item Description*, *Graph Viewer* and *Lens Viewer*. Information from a selected item is displayed in table form with the Item Description View, while in the Graph View mode, the same information is displayed as a RDF graph. The Lens Viewer tries to aggregate data relating to a singular resource and displaying it to the user. Lenses widen views on a particular resources by using additional (known) information about the resource. This is based on a remote unique triple store that stores metadata from resources the user encounters on the web.

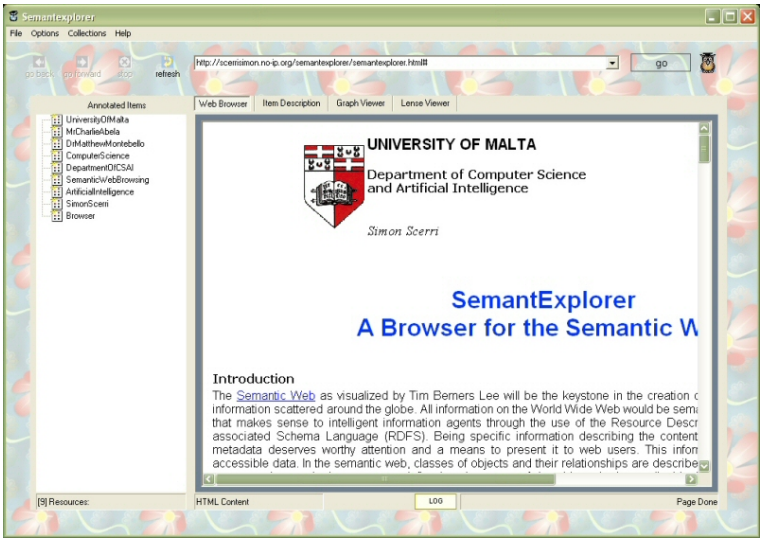


Figure 8: semantExplorer after navigation to a web document containing annotated items (shown in list on the left). Tab selected is the Web Browser.

5 Prototype and Videos

5.1 Prototype

The prototype portal is reachable at `http://personal-reader.de:8080/portal/` Further information, including information about accounts and passwords are available on the portal documentation page at `http://www.personal-reader.de/portal/`.

The sample user account with the *name* `guest` and *password* `guest` can be used to access to the portal.

5.2 Videos

A number of demonstration and tutorial videos are available that show the basic functionality of the portal and the extensions:

Data Entry: Creating a new news item using the portal and triggering the harvester to import it: `http://www.personal-reader.de/portal/blog/movies/data-entry.html` (08/2005)

Radar: A demonstration of the RADAR application and the login process: `http://www.personal-reader.de/portal/blog/movies/radar-film.html` (08/2005)

Browsing Experience: The different filters and visualizations at work: `http://www.personal-reader.de/portal/blog/movies/browsing.html` (08/2005)

6 Conclusion

In this report we have presented our application prototypes for the personalized semantic portal and a semantic browser application. A semantic portal allow the user to customize access to information by using semantic descriptions and helping the user to detect relevant information or relations. In addition to providing an easy to navigate interface to the data available with the REWERSE researcher and publication ontologies, we adapted the core basic portal software of the SWAD-E project, to aggregate news, appointments and TODOs. We extended the portal with a personalized application, called RADAR, which adds a notion of distance or relation between users browsing the portal. This distance is derived from the browsing location inside the portal or from the distance between the nodes representing the users in the underlying ontology. The second application we have developed is a semantic browser which helps to exploit the information stored in annotated web pages. The semantic data from all the pages the users has visited is aggregated in a knowledge base and used to provide additional information and relations about the resources the users is currently viewing.

References

[1] Baumgartner, R., Flesca, S., and Gottlob, G. Declarative information extraction, web crawling, and recursive wrapping with lixto. In *6th International Conference on Logic Programming and Nonmonotonic Reasoning* (Vienna, Austria, 2001).

[2] Baumgartner, R., Geisler, T., Gottlob, G., Henze, N., Kulas, A., Nejdl, W., and Schütz, H. Personalized portal for rewerse. Tech. rep., Hannover, Vienna, Webexcerpt, 2005.

[3] Baumgartner, R., Henze, N., and Herzog, M. The personal publication reader: Illustrating web data extraction, personalization and reasoning for the semantic web. In *The Semantic Web: Research and Applications: Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005. Proceedings* (2005), A. Gmez-Prez and J. Euzenat, Eds., vol. 3532 of *Lecture Notes in Computer Science*, Springer-Verlag GmbH, p. 515. `http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/11431053_35`.

[4] Beged-Dov, G., Brickley, D., Dornfest, R., Davis, I., Dodds, L., Eisenzopf, J., Galbraith, D., R.V. Guha, MacLeod, K., Miller, E., Swartz, A., and Eric van der Vlist. Rdf site summary (rss) 1.0.

[5] Berners-Lee, T. An rdf language for the semantic web - notation 3.

[6] Berners-Lee, T. Semantic web - keynote at xml 2000 conference, 2000. http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html.

[7] Berners-Lee, T. The semantic web - mit/lcs seminar, 2002. http://www.w3c.org/2002/Talks/09-lcs-sweb-tbl/.

[8] Brunkhorst, I., and Henze, N. User awareness in semantic portals. In *PerSWeb'05 Workshop on Personalization on the Semantic Web, Colocated with 10th International Conference on User Modeling (UM'05)* (Edinburgh, UK, July 24 - 30 2005).

[9] Domingue, J., and Motta, E. A knowledge-based news server supporting ontology-driven story enrichment and knowledge retrieval. In *Knowledge Acquisition, Modeling and Management* (1999), pp. 103–120.

[10] Dzbor, M., Domingue, J., and Motta, E. Magpie: Towards a semantic web browser, 2003.

[11] Gottlob, G., Koch, C., Baumgartner, R., Herzog, M., and Flesca, S. The lixto data extraction project — back and forth between theorie and practice. In *ACM Symposium on Principles of Database Systems (PODS)* (June 2004), vol. 23, ACM.

[12] Haase, P., Broekstra, J., Ehrig, M., Menken, M., Mika, P., Plechawski, M., Pyszlak, P., Schnizler, B., Siebes, R., Staab, S., and Tempich, C. Bibster — a semantics-based bibliographic peer-to-peer system.

[13] Hewlett-Packard Development Company. W3c semantic web activity. ERCIM (for W3C), ILRT (University of Bristol), HP Labs, RAL/CCLRC, and Stilo, `http://www.w3.org/2001/sw/Europe/`.

[14] Hewlett-Packard Development Company. Introduction to the swad-e portal structure, 2004. `http://www.swed.org.uk/swed/doc/portal-structure.html`.

[15] Institute, K. M. Magpie - the semantic filter, the open university, 2002. `http://kmi.open.ac.uk/projects/magpie/main.html`.

[16] The apache jakarta project, 2004. http://jakarta.apache.org/.

[17] Jena - A Semantic Web Framework for Java, 2004. http://jena.sourceforge.net/.

[18] Network of Excellence — Knowledge Web, 2004. http://knowledgeweb.semanticweb.org/index.html.

[19] Maedche, A., Staab, S., Stojanovic, N., Studer, R., and Sure, Y. SEAL – A framework for developing SEmantic Web PortALs. *Lecture Notes in Computer Science 2097* (2001), 1–??

[20] Ontoweb — ontology-based information exchange for knowledge management and electronic commerce, 2004. http://ontoweb.ontoware.org/.

[21] OWL, Web Ontology Language, W3C Recommendation, Feb. 2004. http://www.w3.org/TR/owl-ref/.

[22] OWL Web Ontology Language Guide, Feb. 2004. `http://www.w3.org/TR/owl-guide/`.

[23] Protege Ontology Editor and Knowledge Acquisition System, 2004. `http://protege.stanford.edu/`.

[24] Quan, D., and Karger, D. How to make a semantic web browser, 2004.

[25] Resource Description Framework (RDF) Schema Specification 1.0, 2002. `http://www.w3.org/TR/rdf-schema`.

[26] RDF Vocabulary Description Language 1.0: RDF S, 2004. `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`.

[27] Reynolds, D., and Shabajee, P. Swad-europe deliverable 12.1.5: Semantic portals - requirements specification. Initial Release.

[28] Reynolds, D., Shabajee, P., and Cayzer, S. Semantic information portals. In *Proceedings of the 13. WWW Conference* (New York, May 2004), vol. 13, ACM.

[29] Scerri, S., Abela, C., and Montabello, M. semantexplorer: A browser for the semantic web. In *IADIS International Conference, WWW/Internet 2005* (Lisbon, Portugal, October 19 - 22 2005).

[30] Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H.-P., Studer, R., and Sure, Y. Semantic community web portals. *Comput. Networks 33*, 1-6 (2000), 473–491.

[31] Steer, D. Brownsauce rdf browser, 2002. `http://brownsauce.sourceforge.net/`.

[32] SWRC - Semantic Web Research Community Ontology, 2001. http://ontobroker.semanticweb.org/ontos/swrc.html.

7 Appendix

The work described in this reports has been successfully published at international workshops and conferences:

7.1 User Awareness in Semantic Portals

[8] see Appendix A

Authors: Ingo Brunkhorst, Nicola Henze

Accepted for publication at the PerSWeb'05 Workshop on Personalization on the Semantic Web (PerSWeb 2005), Colocated with 10th International Conference on User Modeling (UM'05), Edinburgh, UK, July 24 - 30, 2005

7.2 User Awareness and Personalization in Semantic Portals

Authors: Fabian Abel, Nicola Henze

Accepted as demonstration paper at the International Semantic Web Conference, Galway, Ireland, 2005.

7.3 SEMANTEXPLORER: A Browser for the Semantic Web

[29] see Appendix B

Authors: Simon Scerri, Charlie Abela, Matthew Montobello

Accepted for publication at the IADIS International Conference, WWW/Internet 2005, Lisbon, Portugal, 19-22 October 2005

A User Awareness in Semantic Portals

Ingo Brunkhorst, Nicola Henze
ISI - Semantic Web Group, University of Hannover &
L3S Research Center
Appelstr. 4, D-30167 Hannover, Germany
{brunkhorst,henze}@kbs.uni-hannover.de

abstract In this paper we discuss the benefits of *semantic* information portals. We argue that creating semantic information portals using Semantic Web technologies pays off right from the beginning: The possibility to reason about the (semantically annotated) Web resources allows for realizing personalization functionality which otherwise, i.e. compared to conventional portals, requires additional overhead. We provide a proof-of-concept for our claims within a semantic information portal for a research community and demonstrate personalization algorithms for improving user awareness.

keywords: semantic web, personalization, reasoning on the semantic web, semantic portal

Introduction Information portals have been proven to be successful gateways to information in the World Wide Web: Information portals provide collections of relevant information on specific topics, group and structure the information, and support the user in retrieving, selecting and accessing electronic information. The idea of the semantic web as a layered architecture for realizing machine understandable meaning of Web resources holds particular new ways for building information portals: so called *Semantic Portals* can reason about the - now machine readable - semantic of Web resources, can search for relevant information in a more focused way by considering explicit semantic information, and can extract, rate and combine information resources in an advanced manner. In particular, the realization of user-adapted, *personalized* views on the data can profit from the provision of machine readable semantics, as user requirements can be more precisely considered in the retrieval, selection and presentation processes. At the current state, the creation of machine-readable semantic is not a fully automated process, but requires some overhead, e.g. in creating appropriate ontologies - for describing the application domain, for formalizing personalization attributes and requirements, etc. In this paper, we will demonstrate how the overhead of creating a semantic portal pays off right from the beginning. In Section A.1 we outline the basic ideas of *semantic* portals, and demonstrate these ideas in the realization of a semantic portal for the research community of the "REWERSE - Reasoning on the Web" project. In particular, we focus on the developed ontology which models the objects of discourse and the organization of research projects like partners, working goals and groups, co-ordination and participation, etc. This ontology is on the one hand the core model of the Semantic portal, and browsing the information in the portal is realized as projections of the information space according to (varying sets of) the concepts of the ontologies (see Section A.1.2). In fact, the resulting portal for REWERSE could have been accomplished by using different techniques. But what distinguishes the approach of semantic portals from other portals realized by standard techniques is that we have advanced, nevertheless easy to realize ways to improve user support and to realize personalized views on the data. As an example, we demonstrate how user awareness can be supported in semantic portals. This awareness is based on a collaborative approach. Users browsing the semantic portal of REWERSE can easily grasp when other users are currently interested in similar information,

or whether users with same interests are around. This is implemented by reasoning about the semantic descriptions of the Web resources with respect to the basic ontology of the portal, which models the community / project as a whole (see Section A.2). We end the paper with a comparison of our work with related work on semantic portals, and a conclusion.

A.1 Semantic Portal for REWERSE

Semantic Portals are the result of a continued development of existing portal systems, basically *information portals* [27, 28] such as YAHOO[4] and DMOZ[5], providing access to an integrated and structured body of information about a specific domain. Additionally the *community-based* [27, 30] portals support the collaboration between members of a community, e.g. by allowing the contribution of information and news into the portal.

Crucial for every portal is the design of the navigation and search tools, because they provide the interface for accessing the information. Contrary to traditional portals using a hardwired navigation structure and text search, semantic portals exploit the properties and classification of information items and relationships between them, commonly using external ontologies [27]. Of course, the separation of structure and content of the portal is not new [19, 13, 9]. However, the Semantic Web approach allows for significant advancements: Ontologies describe objects of discourse, model the domain of interest, etc., thus allocate structure but also purpose of a semantic portal in machine readable format. Furthermore, this structure is interlinked with the Web, e.g. via the references of Web resources with respect to some ontology, but also via the connections between ontologies (see the efforts in ontology engineering, especially ontology mapping and ontology merging). This enhanced space of - for machines meaningful - information can be used for improved information syndication strategies and personalization strategies, realized in the logical layer of the Semantic Web architecture [6] by reasoning over these ontological information, evaluated against the personal information of users, retrieved from their user profiles.

A.1.1 Use-Case: A Semantic Portal for a Research Project

This section describes the development of a semantic portal for the European Network of Excellence REWERSE. We have constructed an ontology for describing researchers and their involvement in REWERSE. This "REWERSE-Ontology" has been built with aid of the Protg-tool [23]. It extends the Semantic Web Research Community Ontology (SWRC) [32]. Like in the SWRC, the REWERSE-Ontology has three subclasses: *person*, *organization*, and *project*. Due to the extension of the SWRC, some more subclasses appear in it, e.g. university, department and institute as subclasses of organization. Additionally, the bibliographic metadata for publications is automatically retrieved from the members' public web pages, using the *Lixto* [1, 11] tool.

A synopsis of classes and properties used in the REWERSE-Ontology is depicted in figure 9. The current REWERSE-Ontology consists of 157 instances of persons and organizations using the 73 classes from the extended SWRC-Ontology. Figure 10 shows for an example how some classes and properties are instantiated.

[4]The Yahoo information portal at `http://www.yahoo.com/`
[5]The Open directory project at `http://www.dmoz.org/`

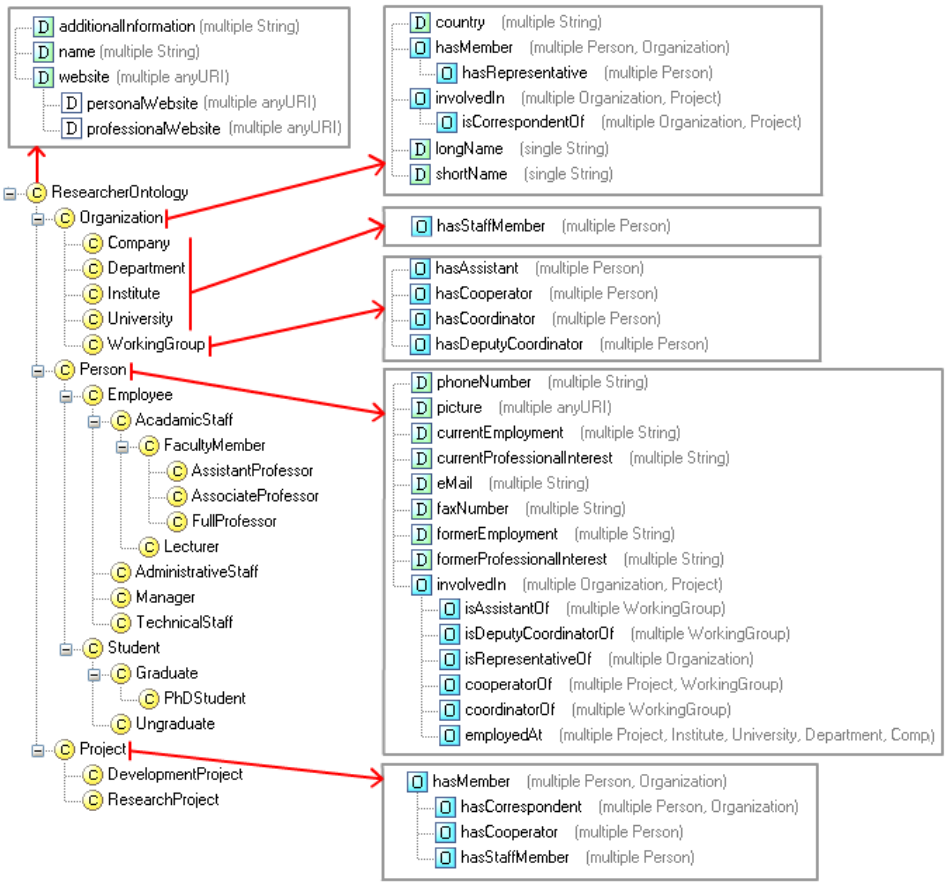


Figure 9: Classes and Properties of the REWERSE-Ontology

A.1.2 Realizing the Portal

As the base of the REWERSE portal we use the SWED Portal technology. SWAD-Europe has been an European project run from 2002 to 2004. The Semantic Community Portal Group developed a prototype for a semantic portal, which we extended to fit our needs. As illustrated in figure 11, the portal itself consists of mainly two parts, the portal viewer and an aggregator component to import new data. The viewer application uses a Jena [17] RDF data model and technology from the Jakarta [16] project for generation of the interface. The aggregator scans known data sources for new or changed metadata and updates the data model accordingly. Rendering the content of the pages is based on templates and so called “facets”, which are used to create browsable views of the underlying RDF data model, the REWERSE researcher ontology. Figure 12 shows the Portal starting page, including six facets, three of them based on the REWERSE ontology. The other three facets use a news data storage combined with the researcher ontology. In figure 13 the facet is used to select a certain view on the data, in this

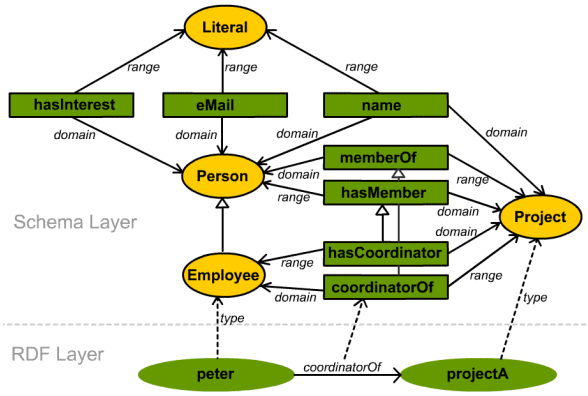


Figure 10: Classes and Properties of the REWERSE-Ontology

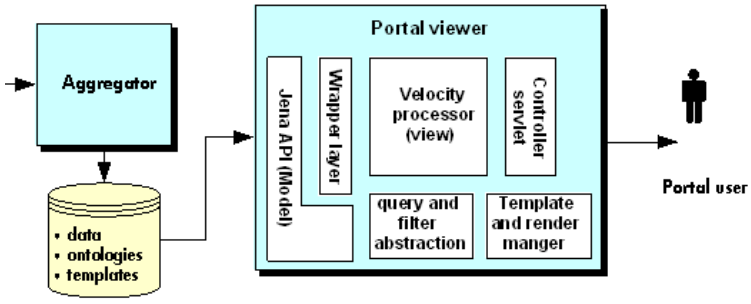


Figure 11: Structure of the portal (Source: [14])

case it displays a list of persons, organizations and projects ordered by type. The template used to render the web page allows for further refinement of the selection, or browsing the ontology. For every resource in the RDF graph, a separate web page is generated, including all of the properties of the node. If a property is pointing to another resource of the ontology, the relation is visualized as an HTML link. The simple news system represented by the three other facets is the first extensions of the portal developed for the REWERSE project. This is complemented by a web based form to input new articles and to annotate them with metadata based on the project ontology, as well as an interface to the news aggregator module to manage the import of new articles.

A.2 Example: Awareness Module

The potential of having a semantic portal instead of generic text or HTML based content is the possibility to exploit the additional information with algorithms taking into account the relationships between the entities in the ontology. These algorithms have only to be developed

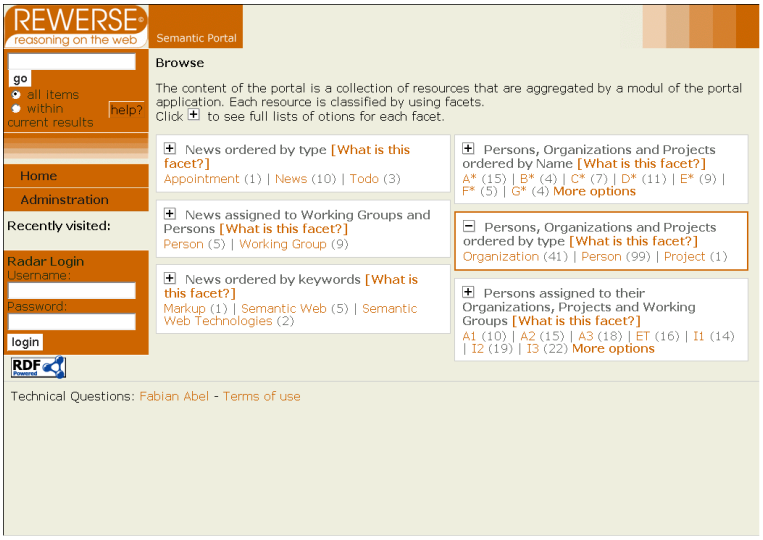


Figure 12: Portal: Starting Page

once, and can be applied to other ontologies as well (Ontology Mapping [22]). Currently implemented are two example algorithms, exploiting a measure for "semantic distance" between the nodes in the graph of the ontology. The distance is computed by a component running in the portal system and exported as an XML document. A flash Applet, running on the client browser retrieves the document and display the distance as a Radar Screen animation, as shown in figure 14. However, for realizing these algorithms it is necessary to have some type of user authentication available, to map the user to the corresponding node in the researcher ontology and to identify the visited pages. The first algorithm computes the *browsing distance* of portal users. Since every node of the RDF graph is presented as a web page, this distance is measured by calculating the shortest path through the graph, between the pages currently viewed by the users of the portal. The algorithm for *professional distance* uses not the visited pages, but the nodes representing the authenticated users. Hereby the distance is calculated by comparing the affiliation with the projects working groups and organizations, e.g. persons working for the same work package are grouped closer together than those working in different groups.

A.3 Related Work: Semantic Portals

Already a few Semantic Portals and Tools have been developed and deployed on the Web. Tools used for the creation of ontologies and collecting metadata include Protg [23] and Bibster [12]. Bibster is a Java-based system which assists researchers in managing, searching and sharing bibliographic metadata (e.g. from BibTeX files) in a peer-to-peer network. The bibliographic metadata for the REWERSE semantic portal is automatically retrieved from the web pages of the project partners using the Lixto tool [1][11], which originated at TU Wien, but is now distributed commercially by Vienna based Lixto Software AG. It is a system that assist the creation of rules and queries to extract metadata from web pages. For Semantic Portal Technologies, the REWERSE semantic portal will take advantage of the SWED portal software

Figure 13: Portal Facet "Persons, Organizations and Projects ordered by type"

developed by the W3C and its partners in course of the SWAD-Europe Project [13]. Another initiative is SEAL [19], developed by AIFB from University of Karlsruhe. The research on semantic web and ontologies that was started with the European Project *OntoWeb* [20] is now continued and expanded in the project *KnowledgeWeb* [18], with a working group dedicated to designing and developing a portal for the project, in close cooperation with the REWERSE project.

A.4 Conclusion and Future Work

Realizing *semantic* information portals using recent semantic Web technologies brings new and fascinating possibilities for improving personalized access to Web resources. In this paper, we have discussed a demonstrator application: a semantic portal for the European Network of Excellence REWERSE. The core of this semantic portal is an ontology which models the world of research communities (the SWRC ontology), which we have extended to also model important aspects of research projects like working groups, the different roles of persons and institutions in such projects, etc. (the REWERSE ontology).

We have shown how personalization can be realized by using ontological information about Web resources: a simple but effective personalized user awareness support: the visitors of a portal are grouped according to their relation to the beholder: are they currently interested in similar topics (e.g. is the information they are currently viewing on similar topics), or do they normally have similar interests (e.g. are they working on similar topics as the beholder)? The beholder can easily check it via the implemented radar which visually depicts the distance of the visitors of the portal to her/him.

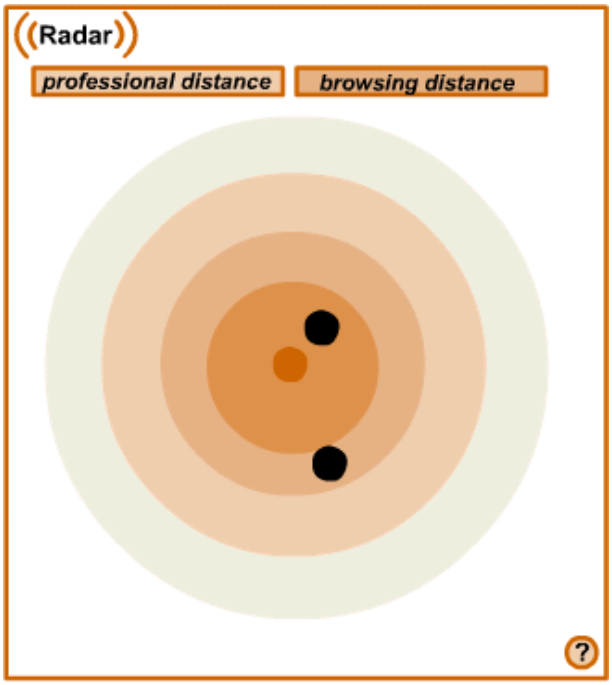


Figure 14: Visualization of Distance

Acknowledgments

This work has been partially supported by the European Network of Excellence REWERSE - Reasoning on the Web with Rules and Semantics[6]. We would like to thank Fabian Abel for his work on the semantic portal for REWERSE.

[6]The REWERSE project at `http://rewerse.net/`

B SEMANTEXPLORER: A Semantic Web Browser

Simon Scerri
Department of CS & AI, Faculty of Science,
Msida, Malta, University Of Malta
mailto:ssce001@um.edu.mt

Charlie Abela
Department of CS & AI, Faculty of Science,
Msida, Malta University Of Malta
charlie.abela@um.edu.mt

Matthew Montebello
Department of CS & AI, Faculty of Science,
Msida, Malta, University Of Malta
matthew.montebello@um.edu.mt

Abstract The Semantic Web [7] will be the keystone in the creation of machine accessible domains of information scattered around the globe. All information on the World Wide Web will be semantically enhanced with metadata that makes sense to both human and intelligent information retrieval agents. For the Semantic Web to gain ground it is therefore very important that users are able to easily browse through such metadata. In line with such philosophy we are presenting semantExplorer, a Semantic Web Browser that enables metadata browsing, provides visualization of different levels of metadata detail and allows for the integration of multiple information sources to provide a more complex and complete view of Web resources.

Keywords Semantic Web Browsing, Metadata, RDF Triple Store

B.1 Introduction

In the Semantic Web, classes of objects and their relationships are described in accessible Ontologies. In turn, resources in a Web document are defined as instances of the objects in the applicable Ontologies. Creating relationships between the resources is possible with the use of the Web Ontology Language [21], an Ontology Language that is built on top of the Resource Description Framework [25], the associated schema [26] and the eXtensible Markup Language (XML). The ultimate goal of the Semantic Web is to achieve a semantically enabled World Wide Web, by annotating online documents and services with semantic meaning. In this way it will be possible to relate between resources on the Web, thus making it easier for software agents to understand the content of the Web and ultimately for people to have better access to concept-oriented data.

Metadata Annotation is the process of attaching semantic descriptions to Web resources by linking them to a number of classes and properties defined in Ontologies. In general, metadata annotation methods fall under two categories: Internal and External annotation. Internal annotation involves embedding mark-up elements inside the HTML documents. On the other hand, external annotation involves storing the metadata in a separate location and providing

a link from the HTML document. W3C recommends the use of external annotations[7]. Other methods are increasing in popularity, one of which promotes the inclusion of the external annotation reference within the HTTP response header.

With more RDF metadata being created, the need for persistent metadata storage and query facilities has emerged. The Semantic Web could enable structured searches for search engines and automated agents, given a large database to manage metadata efficiently. With such a database, related resources are easily connected, irrelevantly of their location and provider. RDF triple stores can be used to store RDF triples so that document metadata becomes more accessible. This would result in quicker and more efficient metadata querying. A number of experimental RDF triple stores have been set up, however none handle provenance, that is, the original source of the triples is not stored

Currently there are two main approaches to creating Semantic Web Browsers [24]. A Semantic Web Browser has been described as a browser that explores the Semantic Web in its own right, and is able to relate and aggregate information about resources located in different Web documents. On the other hand,a Semantic Web Browser can be described as a special Web browser, which augments standard Web browsers with the ability to visualize hidden metadata. Although quite a number of projects have tackled these approaches singularly, few have attempted to merge them together and develop an appropriate tool that can browse and visualize the Semantic Web at the same time. The aim behind this project is to create a tool in the form of a Resource Oriented Browser that will help with the visualization of hidden metadata layers associated with resources in a Web document, as well as aggregate information related to a singular resource of interest from different locations. The latter possibility needs to be based on a unique RDF triple store, which stores RDF triples for each accessed Web document.

The objective in this paper is to show how the two named approaches are bridged to achieve a Semantic Web Browser, which is able to:

- Access a required Web document and return the list of resources described within it, if any.
- Navigate from resource to resource, irrelevantly of the Web document they are defined in, as well as standard document to document navigation.
- Visualize annotated metadata concerning some resource in a Web document in the simplest yet fullest manner.
- Collect metadata from all visited locations and store it in an appropriate database for future use.
- Aggregate information related to a resource of interest from multiple locations, and displays it to the user.

The rest of this paper is divided as follows. In section B.2 we provide some insight into the work carried out to bridge these techniques, where we discuss semantExplorer's architecture and major modules. In the evaluation section we discuss the ability or inability to reach the set objectives. Section B.4 presents a discussion and comparism of semantExplorer with similar current technologies, while ideas for future work are discussed in section refse:future, after which we give some concluding remarks.

[7]Frequently Asked Questions about RDF: `http://www.w3.org/RDF/FAQ`

B.2 System Overview

Figure 1 depicts the overall architecture of the semantExplorer. This includes the most important components, their interactions and the resulting output given to a user who is browsing the Semantic Web. The developed system is composed of four subsystems which we describe below.

The Navigation subsystem caters for document location, verification and accession or download. Navigation to particular URIs can be requested by the user. Contrary to standard web browsers, this subsystem provides a resource-oriented mechanism apart from the standard document-oriented navigation mechanism. Hence navigation requests can include locations pointing to resources defined through RDF data, and not just to web documents.

The Document Processing subsystem handles document processing and information extraction. When this subsystem receives a file, it is passed on to the RDF Extractor, which extracts any available RDF descriptions. The descriptions are then passed on to the parser to be transformed to a set of RDF triples and associated namespaces. Apart from being used by the Data Viewing subsystem, the generated triples are sent to the Cache Generator, and to the remote RDF triple store for storage. This is a unique, remote database, to which users can contribute freshly discovered RDF triples, or update them accordingly. The triple store caters for provenance, which is the original source of stored triples. The source of semantic web data is relevant when looking at the semantic web as an extension of the present Web. Therefore, since this project is also based on such a perspective, the original URLs containing gathered RDF data are also stored for reference.

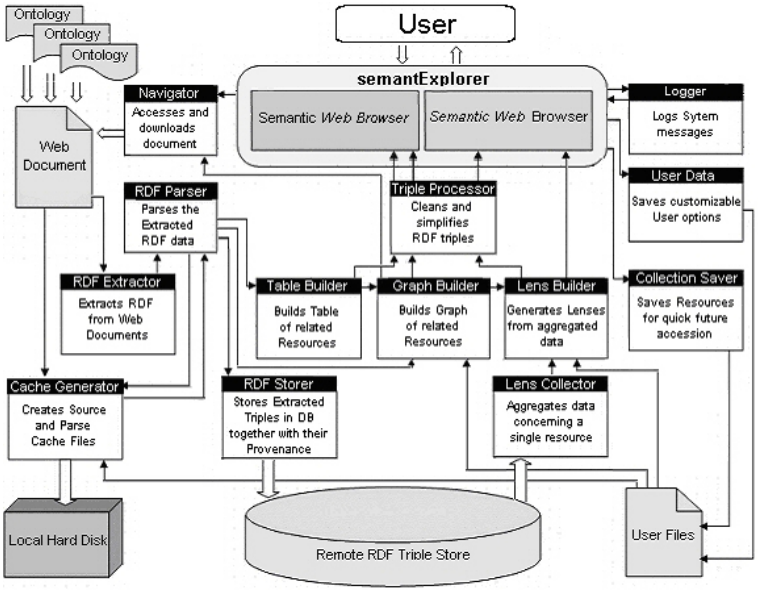


Figure 15: Overall system design architecture

The Data Viewing subsystem is responsible for all user output. After receiving the generated set of RDF triples, it creates a corresponding list of available resources, which the user can use

to request information about some particular resource. In such a case, three different views are generated and presented to the user. The Table Builder gathers information in a document concerning the selected resource, and provides different and simplified ways of displaying it to the user. This data is presented as a set of properties and property values relevant to a resource. Users can navigate to resources that are connected to the selected resource. The Graph Builder processes information as in the table builder, with the difference that such information is displayed to the user as a colour-coded graph. This component also provides an option to extract further relevant background data. This is achieved by processing ontologies that are linked to the current document by namespaces. Data from these ontologies that is relevant to the resource of interest is attached to the basic data, to obtain a more detailed graph. Some basic reasoning is performed by one of the Data Viewing classes. Triple predicates are applicable to a domain and a range. For example, a predicate `isLocatedIn` could have a domain of `Building` and a `Place` as a range. The resource `UniversityOfMalta` could be linked to `Malta` by such a predicate. Although `Malta` could be untyped and defined only as a datatype, it can be inferred that it represents a `Place` by checking the range of `isLocatedIn`. Although simple, this reasoning can enhance information about resources. The Lens Builder extracts data related to the resource of interest from the underlying triple store. These are then displayed to the user as a collection of 'lenses'. A lens can be described as a particular conceptual aspect of the required resource, which after being located can be `focused`. Such lenses can give the user a broader vision of the resource of interest. The user can vieweach lens separately as a graph similar to the one generated by the graph builder. Users can navigate to any generated Lens, since in reality such lenses are nothing other than resources. Before displaying RDF triples in the three generated views, triples are shortened and simplified as required by the Triple Processor. Some triples are irrelevant to the average user and therefore the user is presented with the option to simplify the triple set before it is processed by the Table, Graph and Lens builders for output.

The User Options subsystem handles customizable user options that are retained when the user exits the application. This class library is also responsible for managing collections. The collector component saves a selected resource for future reference. When such a saved collection is selected, the system navigates to that resource and as a result, the table, graph and lens builders process and present the relevant data.

B.3 Evaluation of the System

In this section we describe the capabilities of semantExplorer through the use of an example scenario. We will consider the situation where a user visits the Web page associated with this project[8] through a standard Web browser. A lot of information would be available on the project, but information on concepts behind the various terms and titles in the page are not available, unless given explicitly in the standard Web content. If on the other hand, the visitor is an automated computer agent, it is probable that it would not make heads or tails of what the project is about.

Nevertheless, semantExplorer's first provided view is the *Web Browser* view, showing standard Web content in standard format. In order for the average internet users to be introduced to the Semantic Web, we believe that viewing standard web content alongside its semantic context is crucial. Although the power of the Semantic Web is much greater than that of the

[8]semantExplorer: A Browser For The Semantic Web, `http://staff.um.edu.mt/cabe2/supervising/undergraduate/swbrowsing/semantexplorer.html`

Web, much headway has been made on the latter for it to be completely replaced by the former. However, hidden metadata associating terms in the page with concepts needs to be displayed to the user. Since these concepts are nothing other than resources, the user can request further information about them, or navigate to them. For the page in question, a list of annotated resources is drawn up and shown to the user. One such resource is `DepartmentOfCSAI`. If the user clicks on this resource, the three available views display the relevant information.

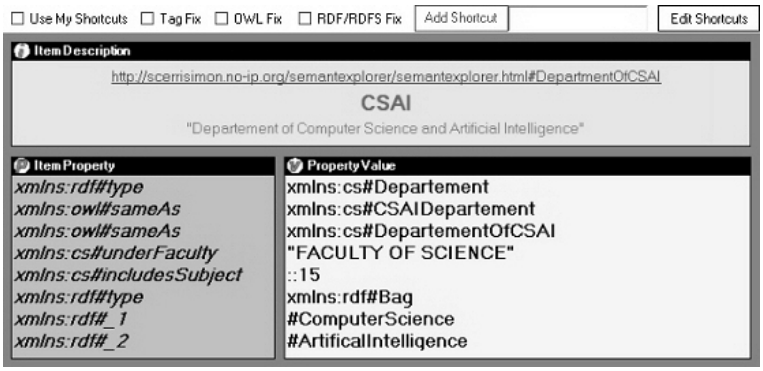


Figure 16: Item Description showing information on a resource

The *Item Description* view, Figure 16, shows data extracted from metadata in the page about the resource. A set of item properties and associated property values are listed. The user can navigate to any of the latter values. In this case, the presented data has not been processed and is not very readable. However the data could have been simplified if the user chose any number of available data fixes.

The *Graph Viewer* view displays the data given by the *Item Description* as a directed graph. Additionally this view can be used to extract more data from underlying ontologies that are referenced in the namespaces. This data is then attached to the basic set to get a more detailed graph. Figure 17 shows the resulting graph output for the `DepartmentOfCSAI` resource. Comparing this with the data seen in Figure 16, besides the fact that data is output in the form of a colour-coded graph, two other differences can be noted. First, the output has been simplified. Blank nodes within basic constructs, as well as references to namespaces, have been removed. In particular, the `Bag` of objects included in Figure 16 is simply shown as a multi-object node in Figure 17. The other difference concerns the inclusion of a lot of extra information on concepts somehow related to the resource of interest. These are attached to the basic data using dashed arcs.

Finally the *Lens Viewer* view responds to the selection by extracting a number of lenses related to the selected resource from the RDF Triple store. Lenses are resources which are directly or indirectly related to the selected resource. These lenses are categorized into lens categories. When a lens is selected, semantExplorer obtains the available information concerning this lens from the database and displays it to the user as a graph. The lens would be opening up on a particular conceptual aspect of the selected resource.

In our case, suppose that the user has navigated to the resource linked to the `xmlns:cs#Department` from the *Item Description*. All three views display information on the new resource `Department`. The *Lens Viewer* shows the three categories available to the user. The first contains a list of

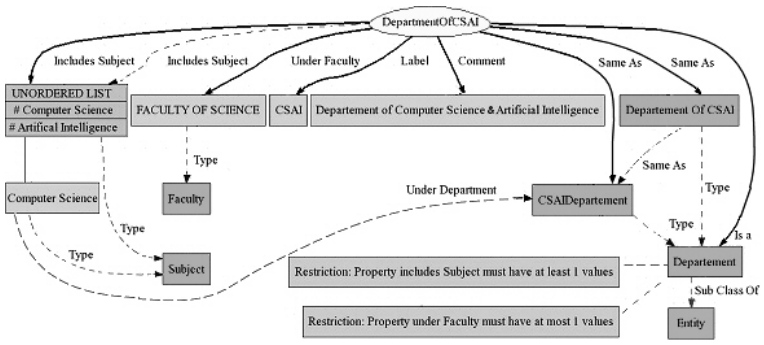


Figure 17: Graph viewer output showing a detailed graph on a resource

URLs containing data on the resource. The second contains other instances of the resource, that is, other resources defined as `Department`, while the third category contains a number of concepts related to the resource. The resource `Entity` is in fact a super class of `Department`. Figure 4 shows the resulting information after the user selected the resource `Department` and `Entity` lens. The user can decide to navigate to any lens, which in this case would trigger semantExplorer to navigate to the resource `Entity`.

The `Graph Viewer` and the `Item Description` fulfil the Semantic Web browser approach to Semantic Web browsing. The `Lens Viewer` caters for the Semantic Web browser facet of semantExplorer. Through these three views, semantExplorer enables Semantic Web data to be collected, visualized and browsed alongside the displaying and browsing of standard Web content.

B.4 Related Work and Comparisons

Existing projects and tools on Semantic Web Browsing have been given their due importance. A number of tools related to the subject are already available. Some of them are very basic, others are very promising and quite complex. Different applications take a different approach to such browsing and metadata visualization. The main ideas behind our research where taken from these projects. The following is a brief introduction to the tools that were most relevant to our work, and a consecutive comparism of these tools in relation to semantExplorer.

Magpie [15] is a tool that extends standard web browsers with the ability to visualize hidden metadata. It provides two methods for browsing [10]. The first provides browsing to physically linked content while the other method provides the semantic context of a particular resource. semantExplorer provides both these methods in its Semantic Web Browser facet. In fact, while browsing to physically linked content is provided by the *Web Browser* view, the semantic context is available to the user through the given list of available resources. This context can be visualized by selecting a desired resource, upon which, the *Item Description* and *Graph Viewer* views will display the relevant data. The advantages of semantExplorer over Magpie are the following. semantExplorer can simplify semantic data to improve interpretability. Secondly, the *Graph Viewer* can be set to extract further data from higher ontology levels, and in this way enhance the basic semantic data available for a resource in a Web page. Magpie provides trigger services called Magpie Collectors, which collect relevant data while the user is browsing.

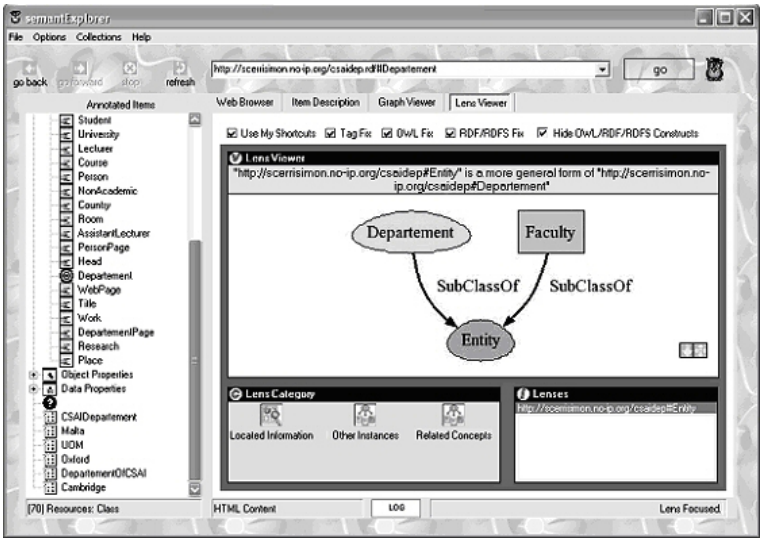


Figure 18: Lens Viewer within semantExplorer

Alternatively, semantExplorer collects RDF data from all over the Semantic Web and sends it to a unique RDF store, which can be subsequently used by any instance of semantExplorer. On the downside, semantExplorer does not provide any semantic data annotation mechanisms, and it does not tackle Semantic Web services.

Brownsauce [31] is a Java servlet application for browsing generic RDF data using a web interface through HTML and CSS. It breaks the problem into Coarse-Graining, where data is broken down into usable chunks consisting of a set of triples relating a singular resource, and Aggregation, where data chunks relating an underlying resource from multiple sources are merged together. The latter feature however, has not yet been implemented. semantExplorer's `Item Description` is basically an improvement over BrownSauce's Coarse-Graining approach. The output given by the `Item Description` is in fact similar to that given by BrownSauce, with the difference that the latter does not cater for blank nodes and no data simplification options are available. The Aggregation problem proposed by the BrownSauce developers has been implemented in semantExplorer through the use of an RDF triple store as discussed in the previous sections.

Haystack[9] employs a Semantic Web Browser that browses the actual metadata and is not just an extension to visualize metadata in conventional URIs. A person's haystack can be described as a repository of all information that the person has come across. RDF metadata concerning a resource from multiple locations is collected and the unified data is presented to the user after converting it to a human readable form. The user in turn can navigate from some piece of Semantic Web data to other related pieces. In this way, separate pieces of information about the same resource that used to require browsing through several different Web sites can be unified into one display. In semantExplorer we have adopted this strategy to achieve a Semantic Web browser. In fact, the `Graph Viewer` and `Lens Viewer` are both

[9]MIT, Haystack Project: `http://haystack.lcs.mit.edu/`

based on such ideas. The `Graph Viewer` attempts to gather further related semantic data than is originally associated with a URL, by gathering more information from ontologies whose definitions are being used to annotate data. In this way, the user is presented with unified information sets while being spared the tedious task of browsing to related resources to achieve better understanding of a resource of interest. Haystack's approach to Semantic Web browsing is based on the notion of the Semantic Web being almost a completely different technology than the present Web. In fact, it is perhaps too complex for the average internet user to consider using it instead of standard Web browsers. semantExplorer is better designed to facilitate such ease of use by users, while integrating the key innovative ideas presented by the Haystack project. The latter has introduced the concept of Lenses, which was subsequently adopted in the design of semantExplorer. A Lens in Haystack is defined as a list of properties that make sense being shown together, and is like a window opening up on certain aspects of a resource of interest, displaying a list of appropriate properties. Similarly, semantExplorer generates Lenses by aggregating information on a resource by querying the RDF triple store. A number of Lenses are in this way generated within four possible Lens Categories. The first contains a number of URLs that contain semantic information directly related to the resource of interest. The second category contains a number of resources similar to the one of interest. For example, if the user requested information about an object whose type is `Student`, this category will display a list of other instances of the class `Student`. The third category displays a number of definitions related to the chosen resource. In the previous example scenario, this category could yield the Lenses `Student`, `Person`, `UnderGraduateStudent` and `PostGraduateStudent`. The fourth category is based on the `rdfs:seeAlso` property, and URIs defined to be related to the selected resource will be included within. When the user selects one of the generated Lenses within any of these categories, the information gathered from the RDF triple store is conveniently merged and displayed as a colour-coded graph. Another notion adopted by semantExplorer from Haystack is the idea of Collections. In both applications, Collections are the Semantic Web browsers equivalent to the standard Web browser's Favourites. When a user selects a previously saved Collection, all semantExplorer's views will focus on that one specific resource. Haystack provides alternative naming schemes other than URLs, and it is based on presentation Recommendations, themselves defined in RDF. At this stage, these ideas where deemed unnecessary for semantExplorer. Contrary to the latter, Haystack also caters for Semantic Web Services.

Piggy-Bank[10] is an extension to the Mozilla Firefox browser, which enables the collection and browsing of Semantic Web data linked from ordinary Web pages. Users can collect and save useful metadata from within the browser pages which in turn can be browsed and searched through an appropriate built-in facetted browser. Piggy-Bank is similar in principle to the Magpie tool. In effect, the same ideas where used in the implementation of our Semantic Web browser. A basic difference in both Piggy-Bank and Magpie vis-a-vis semantExplorer is that while the former two are extensions to standard Web browsers, semantExplorer is a singular tool with its own independent Web browser, providing Semantic Web browsing and mechanisms for the gathering, simplification, integration and visualization of metadata.

B.5 Future Work

A number of ideas for future work have been brought up at the end of this project. Various components in the application can be improved to significantly make them more efficient.

[10]SIMILE - Piggy-Bank: `http://simile.mit.edu/piggy-bank/`

In particular, the Lens Viewer could be extended to fully extract data from a bottom-up triple search. Although the system handles document and parses result caching, this cache is very primitive. Proper caching mechanisms should be developed.

A component which could be included in the system is an RDF Reasoner. The Graph Viewer does provide some basic reasoning. Although simple, this reasoning infers some indirect information about resource, which would otherwise have been missed. With a full-fledged reasoner, data about resources could be significantly enriched for the benefit of the user.

Another idea involves creating a stand-alone Graph Viewer plug-in for standard web browsers. The Graph Viewer is the focal point of the whole application, and it is the component which provides the most important and easily interpretable data visualization. Since the Graph Viewer and Item Descriptor are intrinsically very similar, these two components can be integrated into one, resulting in a Graph Viewer which is also able to navigate to the resources it displays. In this way, all the powerful functions implemented in the Semantic Web Browser facet of this project, could be implemented into a single plug-in application that can be attached to a standard Web browser. This would augment such browsers with the ability to show hidden metadata and browse to related resources.

B.6 Conclusion

The idea to merge the two Semantic Web Browsing approaches was successfully realized. The integration of these two approaches, together with the useful external components and a suitable resource oriented navigation mechanism, resulted in semantExplorer: a Semantic Web Browser. Our browser can be useful both to Semantic Web beginners, to help them learn about the potential of this new generation of the Web, as well as to Semantic Web developers, to help them visualize, analyze and integrate the metadata they are annotating or working with.