Resources Annotation, Retrieval and Presentation: a semantic annotation management system

Marc Albert, Ceddrik Allery, Nicolas Freiss, Grégory L'Azou, Alban Moreau, Jonathan Piron, Vincent Ribaud, Philippe Saliou

Resources Annotation, Retrieval and Presentation: a Semantic Annotation Management System

M. Albert
C. Allery
Software Engineering by Immersion Masters
mxalbert@gmail.com
ceddrik.allery@gmail.com

N. Freiss
G. L'Azou
Software Engineering by Immersion Masters
nicolas.freiss@gmail.com
gregory.lazou@gmail.com

A. Moreau
J. Piron
Software Engineering by Immersion Masters
moreaualb@gmail.com
jonathanpiron@gmail.com

V. Ribaud
Ph. Saliou
Computing Department
University of Brest
ribaud@univ-brest.fr
saliou@univ-brest.fr

ABSTRACT

This paper addresses the problem of the management of resources metadata. A variety of responses are discussed, and we describe one possible way forward, which uses a semantic annotation management tool. The term 'semantic' describes the ability to create, retrieve, query and navigate knowledgeably about things identified by a Web URI. The support for this semantic tool is RDF, through the integration of Jena, an open-source RDF API provided by HP laboratory. Thanks to RDF capabilities, this tool offers new search features such as hierarchical browsing based on the structure of RDF vocabularies and faceted-browsing using properties lists defined by the end-user. The navigation inside annotations uses intuitive modes such as left/right and backward/forward movements. Presentation is controlled by the user using a subset of the Fresnel language to specify how RDF graphs are presented. This work is ongoing; certain open issues are raised.

Categories and Subject Descriptors

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *Web-based interaction.*

General Terms

Languages.

Keywords

RDF, semantic web, annotation.

1. INTRODUCTION

Tim Berners-Lee interviewed by [9] about his vision of the Semantic Web answered that "the goal of the Semantic Web initiative is to create a universal medium for the exchange of data where data can be shared and processed by automated tools as well as by people". As [3] pointed out, one of the basic milestones on the road to a Semantic Web is the linking of metadata to content. The Dublin Core Metadata Initiative is one of the most important organizations dedicated to promoting interoperable metadata standards and developing specialized metadata vocabularies for describing resources that enable more intelligent information discovery systems [1]. According to [1], a metadata record consists of a set of attributes, or elements, necessary to describe the resource in question. For example, a metadata system common in libraries - the library catalogue - contains a set of metadata records with elements that describe a book or other library item: author, title, date of creation or publication, location of the item on the shelf …

RDF (Resource Description Framework) [6] is a W3C (World Wide Web Consortium) standard intended for the management of metadata. RDF models metadata as 3-tuples (triples) which assert that a resource (identified by URI - Uniform Resource Identifier) has a property (identified by URI) which has a value identified either by URI, or given literally. RDF is suitable for the management of metadata records, each attribute of the record being represented by one or more triples. The linkage between a metadata record and the resource it describes may take one of two forms: elements may be contained in a record separate from the item, as in the case of the library's catalogue record; or the metadata may be embedded in the resource itself [1]. Hence, there are two main solutions for the management of metadata records, either the building of an independent system or the addition of an extension to the resource management system itself.

Expressing metadata records in RDF allows us to see metadata as semantic annotations. RDF annotations allow expression of valued properties on resources and/or typed links between resources. From a conceptual viewpoint, it is possible to organize properties, links and concepts in a vocabulary through the use of the RDF Schema (RDFS) language.

The Semantic Web - as hypertext - is based on the idea that knowledge must be represented in a formal way and cannot be centrally created and stored. The motivation behind our work is to provide a group of persons with an easy-to-use metadata management system. This system will support the process of building and sharing collective knowledge related to the group's concerns.

The KnowKnow software described in this paper is intended to manage metadata records as an independent system. Thanks to RDF capabilities, this tool offers new search and retrieve features allowing views, navigations and queries along semantic annotation rather than simply keyword and text searching and indexing. The KnowKnow software has been developed by a team of 6 graduate students on a Masters program in Software Engineering work placement. They worked half-time from September 2007 to May 2008 and delivered a first version of the system.

The main contribution of this software system is to provide hierarchical and faceted browsing; intuitive navigation inside the metadata space, and to offer the end-user the possibility of controlling presentation of results. This paper presents related work, the main features of the software, current limitations and perspectives.

Post-print submitted to *CSTST'08*, October, 2008, Cergy-Pontoise, Paris, France.

2. METADATA MANAGEMENT

The semantics of Dublin Core have been established by an international, cross-disciplinary group of professionals and we must understand the issues that they were faced with.

The Dublin Core basic element set comprises fifteen elements. Each element is optional and may be repeated. Most elements also have a limited set of qualifiers or refinements - attributes that may be used to further refine (not extend) the meaning of the element. Metadata record management should take into account the following features:

1. property-centred: annotating resources is not classifying resources; users have some knowledge about a resource and this can be expressed with an element (a property) but it does not mean that the resource belongs to an identified class of a taxonomy.
2. multi-valued: zero, one or several values can be provided for the same property (e.g. author) and there is often no way of knowing the authorized cardinality of a property in advance.
3. sub-properties: different communities of metadata experts will create and administer different metadata sets, specialized to the needs of their communities; hence, a mechanism has to be provided for extending a common element set for additional resource discovery needs.
4. value types: we need schemes that aid in the interpretation of an element value; these schemes include controlled vocabularies and formal notations or parsing rules.

Relational or object-relational management information systems are not suitable for the implementation of these features. Fortunately, RDF-based systems are fully compliant with these needs and can be used as the core of metadata management systems.

3. RELATED WORK

Semantic Web tools fall into several categories: semantic browsers, semantic annotations tools and semantic search tools are of particular interest in our context. Semantic annotation stores information about resources using semantic information from domain ontologies (called vocabularies in RDFS). Searches exploit ontologies to orient information retrieval and to make inferences about metadata. Semantic browsers present interfaces from a combination of relevant information, ontological specifications, and presentation knowledge [4].

3.1 RDF Browsers

Browsing an RDF-based repository requires the presentation of many small pieces of information linked together through named relationships. Rutledge et al. [8] provide a good survey of semantics browsers classified into three categories: global, local or integrated interfaces. Our system uses an integrated interface with a global view based on the underlying structure (queries or vocabularies or facets) and a local view focused on the selected resource and those directly hyperlinked with this focal point.

[8] states that most assume the Semantic Web can have no such immediate accessibility as with general-purpose browsers, being instead accessible only indirectly through user interfaces encoded for specific domains. One key factor in this assumption is that RDF lacks the document structure HTML and other XML formats have: primarily, that of hierarchy and sequence. Converting RDF structure to document structure in a domain-independent manner would give the information it encodes the same accessibility and approachability HTML enjoys. Our system follows a different approach: the hierarchical structure (vocabulary/class/property) is always available to the user but he/she mostly defines and uses his/her own structure.

3.2 Semantic Annotation Tools

Uren et al. [10] provide a complete survey of semantic annotation for knowledge management. They state that a document centric process must handle three classes of data: ontologies, documents and annotations and that they need to be supported by different kind of tools: semantic search tools, ontology maintenance tools, annotation tools required to cope with the re-versioning and reuse of documents, the evolution of ontologies and the users' and rights management. In our case, documents are managed outside our system. The required tools are provided but the evolution of both ontologies (vocabularies) and annotations related to the evolving vocabulary represents a weak point of our system, and needs to be improved.

3.3 Semantic Search Tools

In a recent work, Uren et al. [11] review the four modes of user interaction in existing semantic search systems, namely keyword-based, form-based, view-based and natural language-based systems. [11] concludes that future development should focus on multimodal search systems, which exploit the advantages of more than one mode of interaction, and on developing search systems capable of searching heterogeneous semantic metadata on the open semantic Web. Our system uses a combination of keyword-based and view-based searches. It needs to be enhanced with a pseudo-natural language based on vocabularies' structure which would guide the user in formulating searches.

4. AN ANNOTATION SYSTEM

4.1 Semantic Annotation of Resources

An annotated system is a system which "knows about" its own content in order that automated tools can process annotations to improve use of the system. For example, semantic annotations can describe documents' authors and their relationships, as well as including traditional metadata, such as the document subject and date of publication. With statements written in RDF, we can then support SPARQL queries like "give me all the married people who have written documents on the Semantic Web".

4.2 Overview of the Project

4.2.1 The Team

The Master program called "Software Engineering by Immersion" provides software engineering learning by doing, with a long-term project as the foundation of all apprenticeships. Young engineers make up two teams of 6; each team is led by one associate professor acting as project manager. This year, one team's project aims to provide a semantic annotation management system called KnowKnow.

4.2.2 Project Objectives and Outcomes

Each person or group manages a space of information that covers all information items (or resources) that he/she uses. The use of semantic annotations can considerably help to manage spaces of information. Annotations help to acquire, organize, maintain and retrieve information for everyday use. The main goal of KnowKnow is to enhance existing resources management systems with semantic annotations. KnowKnow use several kinds of tools:

- Vocabularies maintenance tools are used to input vocabularies to the system and maintain vocabularies.
- Semantic search tools are used to connect and exploit annotations on resources. They are based on the SPARQL language and the building of queries relies on vocabularies managed by the system.
- Semantic annotation tools also rely on the system's vocabularies, and allow the annotation of resources using RDF triples. Import of existing triples is provided.
- The presentation tool aims to display part of RDF graphs and link to other parts with a detail level parameterized by the end-user.

4.3 Components of KnowKnow

4.3.1 Jena: a Semantic Web Toolkit

RDF and RDFS management is based on the Jena API. Jena is a leading Semantic Web programmers' toolkit. It is an open-source project, implemented in Java, and available for download from http://jena.sourceforge.net/.

4.3.2 Triples

The small chunks of information called triples form the basis of an RDF-based system. A semantic annotation in RDF is a triple <subject, property, object>. The object value can be a literal or a URI; if the object is a URI , it can be used as the subject of another triple. We get a graph of annotations linked together (called a Model in Jena).

4.3.3 Vocabularies

A vocabulary (or schema) contains properties. RDF provides no mechanisms for describing these properties, nor does it provide any mechanisms for describing the relationships between these properties and other resources. That is the role of the RDF vocabulary description language, RDF Schema. RDFS provides constructs to describe groups of similar resources (classes) and to describe links between resources or between resources and literals (valued properties).

The RDFS system of classes and properties is very similar to the type system of a language like Java but a fundamental difference in RDFS is that a vocabulary describes properties relative to the classes that properties apply to, rather than describing classes relative to the properties that class instances share. This enables the addition of new properties to existing resources without the need to update a centralized description. RDFS is a frame-centred language and not a class-centred language. Vocabularies without classes (with properties only) such as the Dublin Core are very common.

4.3.4 SPARQL Queries

SPARQL is a query language for getting information from RDF graphs. SPARQL contains capabilities for querying by triple patterns, conjunctions, disjunctions, and optional patterns. Results of SPARQL queries can be ordered and presented in several different forms. Through the integration of the Jena SPARQL engine, KnowKnow offers triple querying and queries management.

4.3.5 Profiles

A profile is an execution context for a user, which the user wishes to keep, so as to be able to re-apply it in similar situations. Profile management enables it to be deleted, duplicated or to request that it be treated as the default profile.

5. SOME KNOWKNOW FEATURES

5.1 Architecture

5.1.1 Sub-systems

The KnowKnow system uses a three-tier architecture in which the user interface, functional process logic, computer data storage and data access are developed and maintained as independent modules, on separate platforms. Sub-systems are: Oracle database, Hibernate persistent layer, Spring framework running on Tomcat, JSF for the user-layer. The architecture achieves a clean separation of business logic, page navigation, and user interface by adhering to a model, view, controller (MVC) architecture.

5.1.2 Domain Model

An RDF-based system is, at least partially, domain-independent and provides features available on virtually any RDF repository. One of the implications of this is that no domain (data) model is hardly-coded in the system providing users with predefined concepts and associations of a given domain such as clients, commands, etc.

End-users import vocabularies to the system and use either new imported concepts, or existing ones, to annotate resources. For example, the vocabulary from INRIA used as an example throughout this paper defines the class Animal, its subclasses Person, Male, Female ... and their relationships with the use of RDFS (a UML class-based representation is given in figure 1).

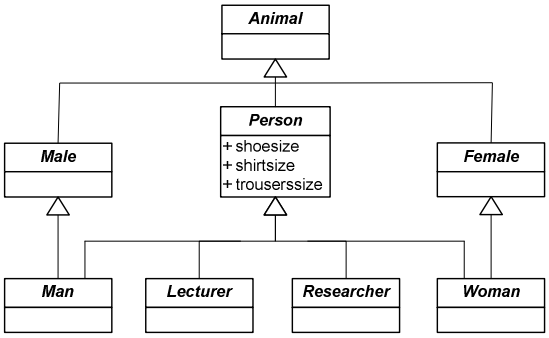

Figure 1. Class-based representation of the sample vocabulary

5.1.3 Storage Mismatch

Triples are stored in the Oracle database as a large and flat data store but the Jena API provides object classes ('model' in the MVC sense) to represent graphs, resources, properties and literals allowing easy object-oriented access to the Jena RDF store.

RDFS constructs need to be treated differently. An RDFS vocabulary description is itself expressed in RDF with the use of concepts such as class, property, domain, range, etc. RDF provides no easy distinction between a triple describing values and one which is describing type information. It may be difficult for the end-user to figure out the underlying domain model (the structure) of the annotations he/she manages.

The structure of vocabularies is widely used in the KnowKnow software to adapt user interface and present annotations in different ways described below: focal point, hierarchical searches, faceted browsing, and queries management. Although RDFS constructs exist in the Jena storage sub-system as triples, they need to be reified and stored in a conventional manner - such as a data dictionary in a database management system.

This 'normal' structure facilitates the development of software interfaces, especially in a web-based system. Unfortunately, this means that information about vocabularies exists in two separate and distinct places: the Jena RDF store and a vocabulary dictionary that is part of a 'normal' Oracle database. In addition, special attention has to be paid to consistency between these two representations.

5.2 Search modes

One kind of search is classical and exploits properties used as annotations. Search criteria can be combined through Boolean operators. One issue is to facilitate the building of queries.

Searches can use taxonomies. In a taxonomy, a controlled vocabulary (defined by a group or a community of practice) is hierarchically organized. Taxonomies are a kind of "a priori" indexation. The visual presentation of a taxonomy - the user interface - is a reliable representation of the semantic organization of the domain.

Faceted classification provides a way to design hierarchies which are simpler and more lightweight. Facets organization (criteria and values) is no longer hierarchical, but multi-dimensional. Classification schemes are not predefined but built by users. Faceted search provides navigation throughout different dimensions or "facets" of searched objects.

As pointed out by [8], a user - browsing an RDF repository storing many small chunks of information with many explicit relationships among them - cannot succeed without the help of an interface which makes the underlying structure explicit. The display giving a global view of these many relationships is referred to as the global interface, while the display that presents a small part of the RDF graph related to the current interest of the user (in greater or lesser detail) is called a local interface.

These different search modes will be illustrated on a set of annotations controlled by the vocabulary presented in figure 1.

5.2.1 Multi-criteria Search

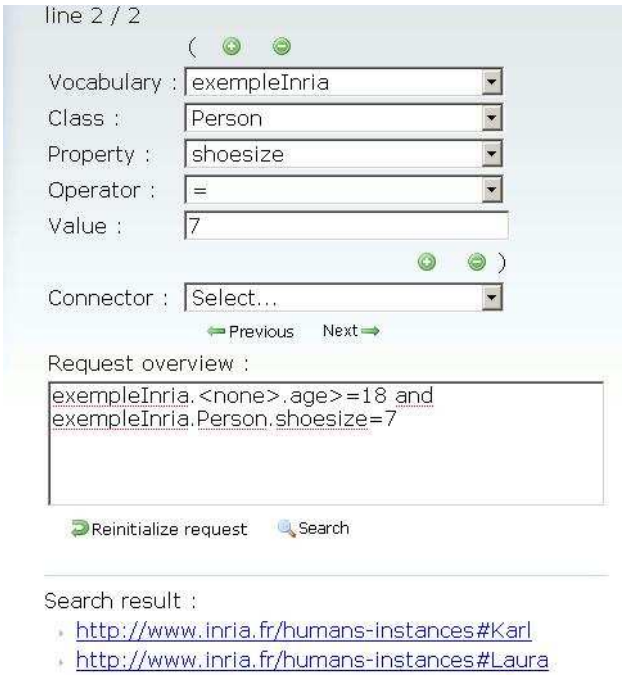

Figure 2. Multi-criteria search

A multi-criteria query defines search criteria with property-value couples (search predicates) associated with a comparison operator; couples (predicates) being linked with logical connectors and a priority system. Logical aspects (connectors and priority) have to be simplified because we argue that most queries use one or two predicates. Building of the query is dynamic and based on information stored in a dictionary of vocabularies. The KnowKnow system displays a textual representation of the query under construction.

In the example shown in figure 2, we search persons aged 18 or over and wearing size 7 shoes.

Asking for a query processing yield to generate and then process a SPARQL query, and finally the display of a list of URIs (e.g. #Karl and #Laura) matching the search criteria. The global interface is the query under construction. This request, once processed, populates the local interface (below the query) with a list of matching URIs.

5.2.2 Hierarchical browsing

Web information portals provide a point of access onto an integrated and structured body of information about a domain. Many portals use a hardwired navigation structure based on a single rich classification scheme (e.g. Dublin Core) coupled to hyper linking of related items and free text search. Yahoo is a canonical example.

In an RDF-based repository, the classification scheme is dynamic. Each time a user inserts triples using properties from a new vocabulary, the classification scheme is enhanced with the structure of this new vocabulary. Hierarchical browsing provides users with access to resources through the taxonomy of classes that resources belong to.

KnowKnow displays the classification in the same way as many code browsers or ontology browsers do it, and an example is provided in figure 3. In the global interface (the left part of the frame), the hierarchical structure is presented to the user. If a class has subclasses, the sub-tree can be deployed / undeployed. If a selected class has resources, URIs are presented in the local interface (the right part of the frame).

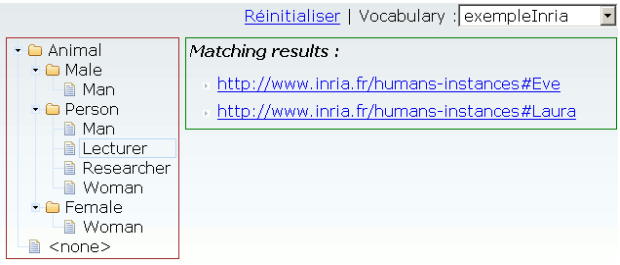

Figure 3. Hierarchical browsing

5.2.3 Faceted browsing

Annotations use a richly structured internal descriptive schema (the structures of different vocabularies) and KnowKnow offers a rich search interface which can exploit this schema. This allows search to be tied to specific facets of the descriptive metadata and to exploit controlled vocabulary terms - leading to searches that are far more precise [7].

Faceted classification allows the grouping (and retrieval) of resources into different categories. Each category (or faceted-list) is organized with the help of a hierarchical list of properties. The first property in a list constrains the grouping at the higher level: all resources having the same value for this highest-level property belong to the same group (within a given category). Each category builds its own groups in an

independent manner. Thus a single resource belongs to several different groups, in accordance with its properties values.

Let us take an example of two faceted-lists: the former "Vital record" uses the a*ge* property then the n*ame* property; the latter "Fashion" uses the sequence of *shoesize*, *trousersize*, *shirtsize* properties. The resource identified by the URI #Laura is 20-years old, wears size 7 shoes and size 28 trousers. Thus, this resource belongs simultaneously to the group of twenty-year olds for the first facet and the group of size 7-shoe for the second facet. Inside this latter group, #Laura belongs to the subgroup of size 28-trouser.

If a user chooses to visit a given group, only resources from this group are selected for the next step. The following properties in the faceted-list are used to separate the selected group into sub-groups with the same rule: each sub-group is constituted with resources of the original group having the same value for the second property. Thus, the user can browse concepts through facets that are in fact successive filters on sets of resources.

In figure 4, a user has first selected the value *7* for the property *shoesize*, then the value *28* for the property *trousersize* (as represented in the History sub-frame). #Lucas and #Laura are the only matching URIs. The "Fashion" facet thus proposes the third property of the list (*shirtsize)* in order to refine the browsing. Properties of the first list (Vital record) were never used, hence the first property a*ge* is available for and to build further subgroups (in this example, a 20-year old group and a 12-year old group). The Browse subframe together with the History subframe yield the global interface; while URIs of resources whose property values match the values selected are presented in the local interfaces - the Results subframe.

Figure 4. Faceted browsing (1)

Faceted browsing is not restricted to a unique faceted-list: at every moment, the current property of any facet can be used to restrict results; e. g. in the figure 5, a user selected the value *7* for the property *age*, leading the system to display a restricted result set in the local interface and the next property (*name*) of the "Vital record" facet.

Figure 5. Faceted browsing (2)

5.3 Navigation and Presentation Modes

Navigation provides a route inside annotations along different roads that link resources. The user can either flit between links, or browse facets, or explore the hierarchy, or search with criteria (see previous sections). In order to help him/her to flit about, it may be necessary to orient the user and to provide beaconed roads.

The Haystack framework [4] provides a Semantic Web-based personal information management system, integrating (Semantic) Web browsing. Haystack authors argue in favour of separating content (that is under the publisher's responsibility) from presentation (an issue for the end user, aware of what he/she wants to be displayed).

5.3.1 Presentation Choices

W3C's answer to the presentation problem is the Fresnel initiative [2]. Fresnel is a simple, browser-independent vocabulary for specifying how RDF graphs are presented. The concept is very close to the use of CSS style sheets for the rendering of HTML pages. Fresnel lenses define which properties of an RDF resource, or group of related resources, are displayed and how those properties are ordered.

Lenses are defined by users and stored in visualization profiles. In the example of figure 6, a lens is defined on the class *Person* and may indicate that only *shoesize* and *age* properties are of interest to the user of this profile and should be displayed.

```
:exempleInriaLens
      rdf:type fresnel:Lens ;
      fresnel:classLensDomain ex:Person ;
      fresnel:showProperties
                ( ex:shoesize
                  ex:age ) .
```

Figure 6. A visualization profile lens

5.3.2 Resource Details Display

When a URI appears in the local interface, clicking on the URI displays details of the resource.

Let's take the example of the resource identified by #Laura, where figure 7 displays Laura's details. Each line of the upper frame shows the information in an RDF triple. Each triple appears as a "duple", because the current resource displayed - Laura - is the subject of these triples.

Figure 7. Resource display - default case -

Note that wherever the object is a literal (*shoesize* 7) or a URI (*hasFriend* #Alice), the value is displayed. Applying a profile containing a lens for the class of a given resource indicates that the user wants to display properties selected in the lens and literals values only. For example, applying the lens presented in figure 6 leads to the result presented in figure 8 (*shoesize* and *age* properties only).

Figure 8. Resource display - applying a lens -

The 'Navigation Mode' button switches the navigation mode (§5.3.3) and the 'Edit' button links to the annotation management features (§5.4).

5.3.3 Navigation

When two resources are in a relationship through a property, there is a kind of hyperlink. In this case, we have to proceed with a different kind of search, which must be oriented. The user starts from a plausible point and expects to follow hyperlinks, flitting from one object to another until the searched object is attained. The issue is to provide multiple and visible entry points together with the possibility of taking any route – left, right, backwards or forwards.

When the user is in a navigation mode, the system understands that subject or object values which are URIs should be used as links. Figure 9 shows how these linked URIs are displayed. When regarding triples <subject, property, object> related to a given resource (e.g. #Laura), the link (here #Alice) is placed to the right of the resource when the resource is the subject of the triple (e.g. < #Laura, hasFriend, #Alice > and to the left of the resource when the resource is the object of the triple (< #aWebSite, dc:creator, #Laura). Only values that are literals are displayed under the URI of the resource.

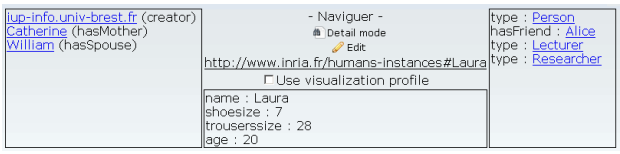

Figure 9. Navigation possibilities

Displayed URIs (left or right) are navigable and clicking on a URI shifts the focus to the resource selected. Left and right links are re-computed according to the new current resource. For example, in figure 10, a user has selected a left link of #Laura, which is a Web site created by #Laura. Thus, the focal point is now this Web site and its properties are displayed and Laura appears to the right of this focal point because Laura is the object of the triple (< #aWebSite, dc:creator, #Laura).

Figure 10. A left navigation

Note that a new 'Previous' button has appeared. It works like the navigation history of a Web browser allowing previous (and next) access to visited resources.

If the check box "Use visualization profile" is selected, the system looks for a lens that could apply to the current resource. If there is a lens, it will be used to display only selected properties. If there is no lens available, all the literal properties are displayed.

5.4 Annotation Management

The KnowKnow system provides only basic features for the addition, modification, and removal of annotations of a given resource. Figure 11 shows the edition of triples related to a given resource (e.g. #Laura). Properties are drawn from different vocabularies, allowing the user to permanently enhance the metadata associated with resources. Manual annotation is a tedious task, especially with this kind of interface. We expect the editing of annotations (RDF triples) to be performed through a normal editor and then imported to the system - a facility to load (import) a set of RDF annotations is therefore provided.

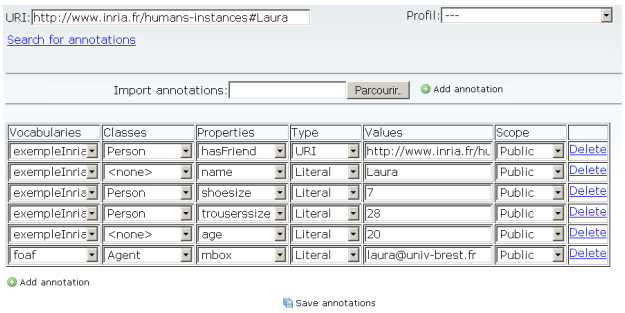

Figure 11. Annotation management

6. PERSPECTIVES AND CONCLUSION

6.1 Limits

A special effort has to be made to provide an easy language querying access to RDF metadata. The multi-criteria search (§ 5.2.1) is an attempt to hide the complexity of a query language, but it needs to be improved.

Providing display facilities to the end-user is a real challenge (§4.3.2). The whole Fresnel language specification was not implemented, and we use only a small subset of its features. Further work is required.

6.2 Consistency

Humans are the metadata creators and they use vocabularies in different manners. So it may be difficult to find a common meaning between annotated resources, even those belonging to neighbouring domains. There is a need to use a domain-independent manner in order to provide the user with guidance. Two particularly informative types of literal are: the rdfs:label - "a human-readable version of a resource's name" [5], and the rdfs:comment - "a human-readable description of a resource" [5]. As noted by [8], inferring label and comment properties from domain-specific ones will provide an efficient way to make repositories more accessible to generalized RDF(S) browsers, without any knowledge of the domain being necessary.

6.3 Automation

The provision of facilities for automatic mark-up of resources is easing the metadata acquisition bottleneck. No-one wants to spend time producing metadata, especially if it is still present (in one way or another) with the resources. Creators, dates, description, and keywords are provided within most resource formats (word processors, spreadsheets, images, etc.) and metadata are present on many resources. Annotations should be automatically gleaned where possible.

6.4 Evolution and Extension of Information Structure

Information requirements change over time. This sort of evolutionary change requires us to change the metadata and any associated database schema, not just the descriptive ontology. This can be complex. We need to permit metadata to be added, without invalidating existing metadata. This is greatly facilitated by use of RDF semi-structured data representation. RDF enables incremental additions of properties and relations by virtue of its property-centric (rather than record-centric) approach to representation [8].

6.5 Conclusion

KnowKnow is a first step towards a semantic environment intended to facilitate metadata management. We brought together various modes of searching and navigating. A major challenge is to provide a uniform interface to the user which is easy to understand and which hides the underlying complexity of RDF graphs.

The next step is to use this system in order to collectively build knowledge. A new team has to use it in order to manage the day-to-day events of their software project from meeting schedules to document management. The implementation of some parts of the Semantic Web vision is a contribution to its achievement. We hope to have taken a step in the right direction.

7. REFERENCES

[1] Dublin Core metadata element set, version 1.1. July 1999. URL: http://dublincore.org/documents/1999/07/02/dces/

[2] Fresnel - Display Vocabulary for RDF URL: http://www.w3.org/2005/04/fresnel-info/manual/

[3] Kahan, J. et al. 2001. Annotea: an open RDF infrastructure for shared Web annotations. In *Proceedings of the 10th international Conference on WWW* (Hong Kong, May 01 - 05, 2001). WWW '01. ACM, New York. DOI= http://doi.acm.org/10.1145/371920.372166

[4] Quan, D. A. and Karger, R. 2004. How to make a semantic web browser. In *Proceedings of the 13th international Conference on World Wide Web* (New York, NY, USA, May 17 - 20, 2004). WWW '04. ACM, New York, NY, 255-265.DOI= http://doi.acm.org/10.1145/988672.988707

[5] RDF Vocabulary Description Language 1.0: RDF Schema. URL: http://www.w3.org/TR/rdf-schema/

[6] Resource Description Framework (RDF): Concepts and Abstract Syntax. URL: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[7] Reynolds, D., Shabajee, P., Cayzer, S. SWAD: Semantic Portals - URL: http://www.w3.org/2001/sw/Europe

[8] Rutledge, L., van Ossenbruggen, J., and Hardman, L. 2005. Making RDF presentable: integrated global and local semantic Web browsing. In *Proceedings of the 14th international Conference on WWW* (Chiba, Japan, May 10-14, 2005). WWW '05. ACM, New York, NY, 199-206. DOI= http://doi.acm.org/10.1145/1060745.1060777

[9] Updegrove, A. The semantic web: an interview with Tim Berners-Lee. Consortium Standards Bulletin, 2005. URL: http://www.consortiuminfo.org/bulletins/semanticweb.php

[10] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F. 2006. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Web Semantics: Science, Services and Agents on the World Wide Web* 4, 1 (Jan. 2006), 14-28. DOI=10.1016/j.websem.2005.10.002

[11] Uren, V., Lei, Y., Lopez, V., Liu, H., Motta, E., Giordanino, M. The usability of semantic search tools: A review. *The Knowledge Engineering Review*, 22, 4, (Dec. 2007), 361-377. DOI=10.1017/S0269888907001233