Visualizing RDF Data for P2P Information Sharing

Arthur Ouwerkerk, Heiner Stuckenschmidt
AI Department
Vrije Universiteit Amsterdam
De Boelelaan 1081a
1081 HV Amsterdam, NL
{aouwerkerk, heiner@cs.vu.nl}

Abstract
We discuss the problem of assisting the users of RDF-based Peer-to-peer systems for information sharing with visual information about available information and conceptual structures. We focus on the problem of exploring the knowledge available in the P2P network and of formulating queries at the conceptual level. After defining requirements for these tasks, we briefly introduce existing visualization tools that work on RDF-based data and discuss their suitability with respect to the identified requirements. We conclude with a discussion of visualization needs that are not supported by available tools and propose some directions for future research.

Keywords
Peer-to-peer systems, Information sharing, query formulation, RDF

INTRODUCTION

Peer-to-peer architectures have been proven useful for large scale information sharing. Their benefits in terms of a low entrance barrier and a high degree of flexibility and fault tolerance have led to successful application like NAPSTER or Gnutella. One of the drawbacks of current peer-to-peer systems, however, is their limited search and querying capabilities. Most systems only support plain keyword queries that suffer from the lack of naming conventions and the ambiguity of natural language. Recently, different research projects attempt to overcome this limitation by integrating schematic and conceptual knowledge into peer-to-peer environments. Examples of such projects are Edutella[1], EDAMOK[2], Piazza[3] and our own project SWAP[4]. While the use of conceptual knowledge allows for more expressive and detailed queries it also causes problems in terms of additional complexity that clearly asks too much of a users not familiar with the model. If we still want to benefit from conceptual knowledge, we have to provide sophisticated support to the user of the system that helps him/her to deal with the existing complexity. Carefully designed interfaces with meaningful visualizations of data and conceptual knowledge can provide such support to a large extent.

In this paper, we discuss how visualization techniques could help to manage the complexity that arises from the use of conceptual knowledge in peer to peer information sharing (for an introduction to the aims and requirements of SWAP see [1]). We formulate a set of requirements with respect to a visualization of information and conceptual knowledge (section 2). We give a brief overview of existing visualization systems for RDF data and their functionality and special features (section 3) and evaluate these systems against the requirements of the SWAP project (section 4). We conclude with an analysis of further needs that are not yet addressed by existing tools.

VISUALIZATION REQUIREMENTS

The quality of a visualization depends on many factors. Some of them are linked to human cognition, our way of thinking and our mental limitations. We will not discuss these general requirements that hold for many kinds of visualizations, but rather concentrate on the specific requirements that can be derived from the application in the context of the SWAP project. Here we identify two main tasks that have to be supported. Both tasks are linked to different ways of accessing information. The first one is query formulation: here the user has to be guided in the process of choosing the right things to ask for in the right way. The second task is knowledge exploration with a more interactive process of accessing information in the network. We will discuss the requirements connected to these two tasks in the following.

Query Formulation

The SWAP system takes advantage of the knowledge of the whole network in order to answer the queries of users. Visualization techniques should support the following query-related tasks:

Query formulation query interfaces that contain a graphical presentation of the available knowledge (ontology), make the query formulation task much easier as the user is already acquainted with the used terminology. The ideal situation would be that queries can be formulated visually.

Query Result presentation inadequate presentation of query

[1] http://edutella.jxta.org/
[2] http://edamok.itc.it/
[3] http://data.cs.washington.edu/p2p/piazza/
[4] http://swap.semanticweb.org

answers shadows the performance of many search engines as the user is overwhelmed with results without being able to pick the part that is of interest for him. Graphical presentations can explain the relation of the answer to the original terms of the query.

Query reformulation Very often the user is not satisfied by the returned answer: there are either too many items or no items at all. Graphical presentation of results allows a user to perform:

- Query relaxation: in case of empty sets, by giving up some of the search terms and choosing the results that come closer to the original query.
- Query broadening: in the case of an empty answer set some terms can be replaced by their super-classes. thereby broadening the scope of the query.
- Query narrowing: if there are too many items, the query can be repeated with more specialized sub-classes of certain query terms, narrowing the scope of the query.

Visualizations should help the user in finding relevant information and in addition should make the formulation of queries easier by giving visual clues.

Network Exploration

The main feature of a peer-to-peer network is the ability to access not only the information that is stored locally, but also to take advantage of information in the whole network. In an ideal case, the user of a peer-to-peer system will not even notice the difference between local and remote information. In the presence of conceptual knowledge as a basis for searching and querying, however, we have to deal with the fact that different peers will use different conceptualizations to describe their information. This means that if we still want to provide the kind of query formulation support described above, we need a proper specification of how the different conceptual models relate to each other. A common solution is to define mappings between different models that formulate semantic relations between classes and properties in the different models. A number of methods have been proposed to generate these mappings, however, none of these methods is elaborate enough to produce mappings that do not require human inspection and improvement. Here, again, visualization techniques can play a vital role to support the user in creating and inspecting mappings between the models of different peers in the system. In order to support the exploration of the network, we formulate the following requirements on the supporting visualization.

Distribution of Knowledge: The visualization should display the distribution of knowledge in the network. It should link peers to thematic areas based on the information they host. On a higher level, the visualization should clearly show groups of peers specialized in a specific thematic area.

Personal Views on External Data: In order to make the distribution of knowledge clear, it has to be possible to explore information in the whole network based on a single conceptual model. More specifically, this requires to be able to create personal views on external data in terms of a local conceptual model.

Semantic Relations between Conceptual Models: The creation of personal views on external information requires semantic mappings between the models of different peers. In order to better understand and to maintain these mappings, a visualization of existing mappings between two conceptual models is a further requirement.

Comparison of Conceptual Models: In order to judge the quality of existing mappings or to create new ones visualizations that help to compare different conceptual models are required. Approaches that highlight differences between models are useful in this context.

In summary, the wish to explore a whole network of information rather than a single peer poses new requirements on supporting visualizations. Here, visualizations that display more than one model a time are required.

VISUALIZATION TOOLS FOR RDF DATA

With the increasing importance of RDF as a standard language to encode conceptual knowledge, a number of visualization tools for RDF have been developed and existing visualization tools have been adapted to deal with RDF. We found the following tools that fulfilled the criteria of being available for free and focussing on models of conceptual knowledge.

Spectacle

The Spectacle Cluster Map is an application for the visualization of instantiated taxonomies such as class and concept hierarchies. A user can select the instances of which class to display from a list of all classes. The Cluster Map visualizes overlap between classes through shared instances. This way, the user can see clearly how the classes relate to each other, because of the instances that they share. This makes it very useful for analysis of hierarchically organized datasets [2].

Figure 1 shows the Cluster Map Viewer displaying an example repository. All classes are represented by large green spheres. The smaller yellow spheres represent instances. All instances are connected to one or more classes through the balloon-shaped edges, which indicate the class membership. This is just a simple example, because it consists of a limited number of classes and instances, and there is not a lot of opportunity for overlap between classes (e.g., a resource is unlikely to be both an artist and an artefact).

Jambalaya

Jamabalaya [7] is a plugin for Protégé which allows the SHriMP visualization to be used on Protégé projects. The Simple

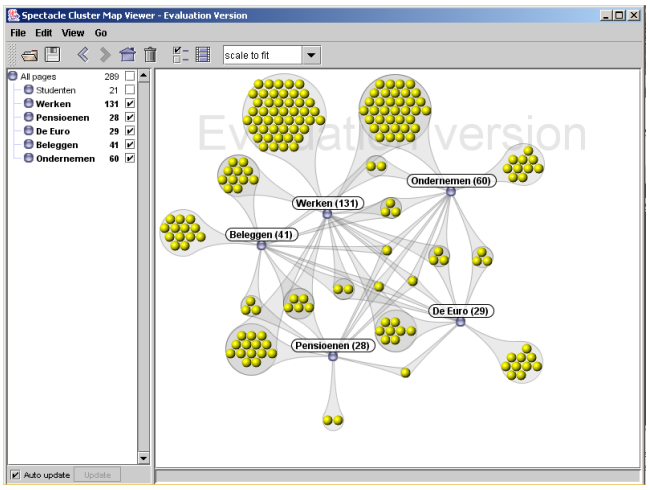

Figure 1: Spectacle

Hierarchical Multi-Perspective (SHriMP) visualization technique proposes that for exploring software an integrated approach which combines both pan+zoom and fisheye-view approaches is preferable for supporting a variety of comprehension strategies. Pan+zoom allows browsing a graph at various levels of detail, while the fisheye view displays critical information larger than less relevant information. Filtering, abstraction, and layout algorithms are used to better convey important structural information in the graph. Nested graphs display hierarchical structures such as containment relationships and graph abstractions. For example, related elements may be collapsed into a single subsystem node, with nesting to show a containment relationship between this node and the original artifacts. Figure 2 gives an impression of Jambalaya's user interface.

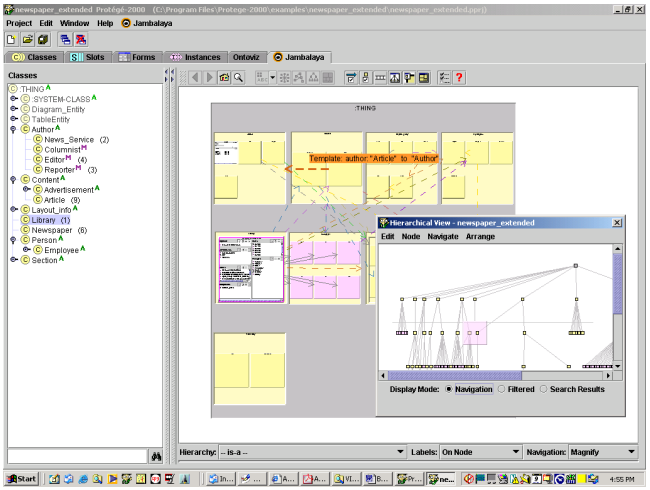

Figure 2: Jambalaya

KAON OI modeler

KAON is an open-source ontology-management infrastructure targeted at semantics-driven business applications, developed and maintained at the FZI (Research Center for Information Technologies) and AIFB (Institute of Applied Informatics and Formal Description Methods) at the University of Karlsruhe. It comes with a tool suite designed for ontology management and application. KAON focuses on integrating traditional technologies for ontology management and application with those typically used in business applications, such as relational databases. It is based on an ontology model, derived as an extension of RDF Schema, with some proprietary extensions (such as inverse, symmetric, and transitive relations), cardinalities, modularization, metamodeling, and representation of lexical information [4]. While not really relevant to the SWAP project, KAON can also be used to edit ontologies. Figure 3 shows the user interface of this tool.

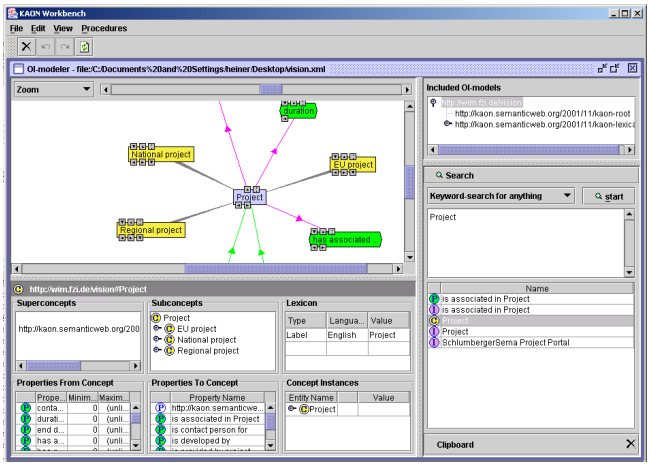

Figure 3: OI modeler

The OI modeler is modelling and visualization tool for ontologies used in KAON. It can be used to inspect and edit multiple ontologies linked by import statements combining a form-based a interface with a dynamic visualization of parts of an ontology that is implemented using the Touchgraph library. The system includes a query engine that can be used to retrieve schema elements and instance data.

OntoRAMA

A disadvantage of using a tree metaphor for browsing RDF is that an RDF file may contain multiple trees (or simply objects and attributes that disconnected). To overcome this problem, ONTORAMA [6] allows to browse through a forest rather than through a tree. The various components of the tree are made accessible by a pull-down menu called 'unconnected nodes list', here the tree components are named after the root node name of each of the component trees. ONTORAMA must display two tree structures: one for the object or class hierarchy and the other for the property hierarchy. To illustrate, consider the following RDF at the top of page 4.

The solution color and shape are useful to draw attention to the points where the interactions between the RDF Resource hierarchy and the Property hierarchy inter-connect. This is shown in figure 4, note the green Property Relation imme-

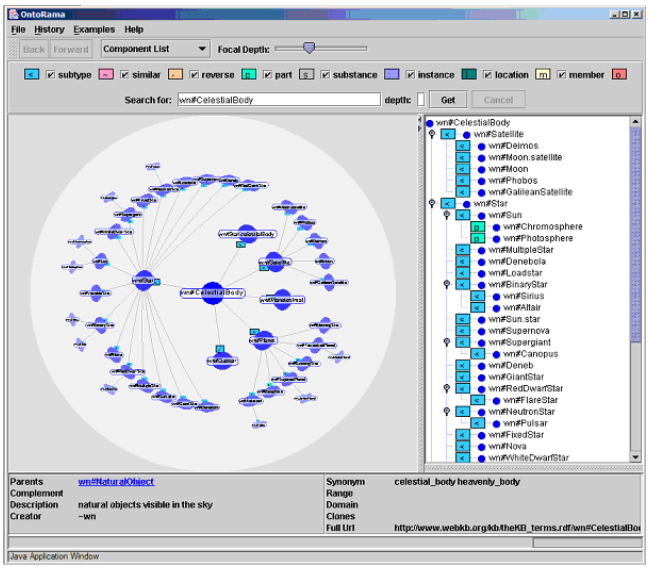

Figure 4: ONTORAMA

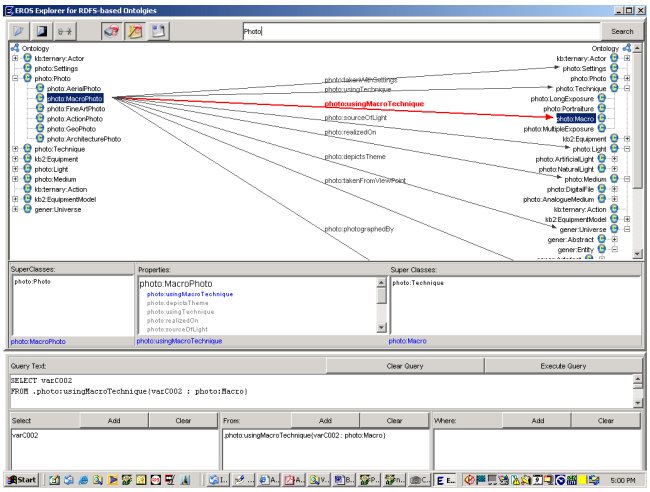

Figure 5: EROS

diately to the right of Thing, the notation of WEBKB-2 indicates that these are exclusive types, namely that the Resource/Object hierarchy is exclusive of the Property/relation hierarchy. The authors advocate multiple visual views over an ontology, and ONTORAMA includes both the hyperbolic-style view and a tree view widget.

EROS

The authors of Eros [8] believe that due to peculiarities of the RDFS language, a new interface is needed to convey RDFS-based ontologies to the end-user in a comprehensible form. The two main approaches currently used, the tree-based approach and the graph-based approach, do not fulfill the above requirement completely. The tree paradigm does not help the user in grasping other concept relationships than that used to construct the tree structure (most of the time being the rdfs:subClassOf relationship). The graph approach, on the other hand, displays all concept relations but as a result introduces the full complexity of a directed labelled graph in which it is very difficult to spot the hierarchical structure of the ontology hidden behind the special kind of edges.

EROS is a combination of the tree-based and the graph-based approach, trying to preserve the good aspects of both: the simplicity of the first and at least a part of the expressiveness of the second. The main idea behind EROS is to consider properties as partial mappings that relate (some) elements (classes) from the class hierarchy to either other (possibly identical) elements within the same hierarchy or to a special element called Literal. Hence, the set of all elements from the hierarchy serves two purposes: first as a (potential) domain of all properties, second as their (potential) range. This inspired EROS makers to have two (almost) identical hierarchy trees in our interface, the left tree being the domain (from) tree, and the right tree being the range (to) tree extended with the Literal element. Properties themselves are depicted as arrows connecting the classes from the domain tree with the classes from the range tree. Cases of multiple inheritance are handled by displaying a list of super classes for the selected node.

EVALUATION

We evaluated the above mentioned tools by reading the accompanying documentation and using the tools to explore and query different ontologies available on the web. In the course of the evaluation, we had to notice very early, that none of the tools provides the possibility to actually compare different ontologies. Only the KAON OI modeler was able at all to deal with import relations. We therefore restrict our evaluation to the query formulation problem. We summarize our impressions with respect to requirements of supporting query formulation in the following.

Spectacle

Query Formulation Formulating queries with Cluster Maps is simply a matter of clicking on checkboxes. All classes are shown in a list and the user can select which classes should be inserted in the query result. This approaches the ideal situation where queries can be formulated visually, but it remains a question whether this method allows for enough expressivity. An average SWAP user will probably want to do more than just search for classes.

Query Result presentation The fact that changes in the query are immediately carried through in the result presentation makes the relation between the query and the results very clear. The optional use of animation enhances this even more.

On the other hand, the method of result presentation employed by Spectacle can become overwhelming very quickly. This is demonstrated by figure 1. Only 5 classes are selected, but especially in the center of the Cluster Map it becomes very difficult to tell to which classes the different instances belong.

Query Reformulation Cluster Maps lend themselves very well to reformulating queries. Removing one or more classes from the selection relaxes the query and the new results are shown immediately. This prevents users from relaxing the query more than they need. Query broadening and Query narrowing are simply matters of checking super- and or subclasses.

Cluster Maps seem to be quite a good solution for SWAP. There are a few problems that need to be addressed, but this is a very good start.

Jambalaya

Jambalaya has a search facility which allows users to search for strings. Besides that, the layout represents a fisheye view of a tree. By clicking on relevant classes and/or instances a user can approach a relevant item.

Query Formulation When entering a search string, there is no graphical representation which helps the user in finding good keywords. It is simply a question of entering a simple or complex search pattern after which a list of relevant results is returned. Only after clicking on one of these results the visualization comes into play again, when a user is directed to the selected result.

Instead of searching for a string, it's also possible to click on class and instance names. With this method the visualization does help a lot because the user can always see where he is in the ontology. This is further aided by the good use of animation. A downside of this approach is that the user will have to be familiar with the ontology to know where to click and find his desired result.

Query Result presentation When a list is returned after searching for a string, there is no visual clue as to where a result belongs in the ontology. This may be important when a result is not very clear and more information is needed to decide whether this result is relevant or not. If a huge list is returned, the results can become overwhelming for the user. On the other hand, if more complicated and user friendly query mechanisms have failed, it's good to have such a simple yet powerful search engine as an alternative.

Clicking on a class or instance name shows the relevant part of the visualization, but when a user is unfamiliar with the ontology finding a good result is more a question of trial and error. The visualization won't really help with that.

Query Reformulation When searching for a string it is possible to refine the search by using the option 'Search within results'. This type of searching does not take a class hierarchy into account, so it is not possible to broaden or narrow searches by using sub- and super-classes.

The class hierarchy is visualized in a separate table, so by clicking there it is possible to go to super- and subclasses. While Jambalaya is a fine piece of software for exploring Protégé projects, its lacks a more elaborate query mechanism.

KAON OI modeler

The OI modeler can be used to query ontology models using the KAON query language. Query templates are predefined for searching schema elements with a certain name.

Query formulation The OI modeler allows the user to state different kinds of queries. The user can search for the name of an element. The search string may include wild cards. This keyword based querying is supported by the possibility to select the type of element (class, property, instance) and, as the system supports multilingual ontologies, the language to be used. This helps to formulate simple queries, but the creation of complex queries, which is also possibly, is only supported in terms of showing parts of the ontology.

Query Result presentation The results of a query are represented in a plain list. Entries in the list are names of elements and an icon indicating their type. Further information about the result elements can be obtained by selecting an element and inspecting its properties using descriptive forms or by dragging an element into the graph window which allows for further exploration of the model starting from that element. A disadvantage of the latter is that there is no distinction between elements in the result and other elements of the ontology any more.

Query reformulation The OI modeler does not provide a direct support for query reformulation. It is left to the user to inspect the ontology or the properties of the result in order to find a more suitable query.

Eros

The idea of EROS is to use an external RQL query engine, which makes this tool potentially quite powerful when it comes to querying the ontology.

Query formulation The property-centered visualization of the schema can be used to select concept-relation-concept triples and add them to a query template. In this way, a number of useful queries can be constructed by the user without having to know the syntax of the query language. For advanced users, there is the possibility to modify the resulting RQL description.

Query Result presentation The prototype version of EROS we used for evaluation did not support the execution of queries. Therefore we do not have sufficient empirical evidence to judge the suitability of this representation.

Query reformulation Queries can be relaxed by deleting items from the WHERE-clause of the query. This poses less restrictions on the results and should therefore have greater chance of returning an answer. Broadening and narrowing queries is not directly supported by EROS, but it is very clear from the ontology view which classes are sub and super-

classes of others. This helps the user in reconstructing the query.

EROS provides a view on a schema that is centered around properties. This view coincides with most existing RDF query languages and therefore provides good support for query formulation, while it makes exploration harder due to the conceptual difference to the view normally used to present schema information.

Ontorama

Ontorama is focused on browsing through ontologies and as such has no query engine to speak of. It is possible to search for classes, but that only works when the full class name is provided. In case of SWAP it should be possible to make queries without knowing exactly what everything is called.

Discussion

P2P information sharing on the basis of conceptual knowledge has strong requirements for user support in terms of tools for querying, exploring and visualizing information and conceptual structures. Our aim in this paper was to get an impression of the support that existing tools for visualizing RDF-based data can provide to this end. In the evaluation, we tested system with respect to two general tasks: support for query formulation and exploration of knowledge in a network. Our major finding is that existing tools if at all only address the first task. More specifically, we come to the following conclusions:

Most of the examined tools rely on a mixture of querying and exploration for supporting information disclosure. The tools differ in the relative importance of these two paradigms. On the one side Ontorama, Jambalaya and Spectacle focus on exploration and only provide basic querying support for finding elements in the model that can be used to further exploration. On the other side EROS and OImodeler are linked to RDF query engines and provide some support for formulating queries in a formal query langauge.

In all cases, the visual support for exploration and query formulation is mainly based on the conceptual model underlying the actual data. The tools differ in the focus on specific model elements. Spectacle for example emphasizes class membership, Ontorama focusses on the subclass relations and EROS on relations in general. All tools tackle the problem of information overload by only showing parts of the complete model.

Our greatest disappointment was to see that none of the existing tools really addresses the problem of exploring a complete space of linked ontologies. The only tool that is actually able of handling more than one model at a time is the OI modeler that also includes imported models. However, it is only possible to view models that are explicitly connected by an import statement and threats them as one ontology using different colors to distinguish different namespaces. What is needed is a view that focusses on existing or potential mappings between different models more in the style of the EROS visualization. Alternatively, support for comparing different models could focus on shared instances in the style of Spectacle.

We conclude that there is a need to extend existing visualization tools with the ability to handle different ontologies at the same time and allow to compare them. In the area of ontology merging and evolution management, some tools have been developed that address this problems. Examples are ontoview [3] and PROMPT [5]. Unfortunately, these tools have only very limited visualization support. Therefore a combination of these two technologies would be a big step into the direction of supporting semi-manual ontology alignment for P2P information sharing.

REFERENCES

1. M. Ehrig, C. Tempich, J. Broekstra, F. van Harmelen, M. Sabou, R. Siebes, S. Staab, and H. Stuckenschmidt. Swap: Ontology-based knowledge management with peer-to-peer technology. In *2. Konferenz Professionelles Wissensmanagement*, Luzern, 2003.

2. C. Fluit, M. Sabou, and F. van Harmelen. Ontology-based information visualisation. In ladimir Geroimenko, editor, *Visualising the Semantic Web*. Springer Verlag, 2002.

3. M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov. Ontoview: Comparing and versioning ontologies. In *Collected Posters ISWC 2002*, Sardinia, Italy., 2002.

4. A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz. Ontologies for enterprise knowledge management. Technical report, Research Center for Information Technology, University of Karlsruhe, 2002.

5. N. F. Noy and M. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *In Proceedings of AAAI-2000*, Austin, Texas, 2000. MIT Press / AAAI Press.

6. P.W.Eklund, N.Roberts, and S.P.Green. Ontorama: Browsing an rdf ontology using a hyperbolic-like browser. In *Proceedings of the First International Symposium on CyberWorlds (CW2002)*. IEEE Press, 2002.

7. M.-A. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson, and N. Noy. Interactive visualization to enhance ontology authoring and knowledge acquisition in protege. In *Proceedings of the Workshop on Interactive Tools for Knowledge Capture at K-CAP 2001*, Victoria, B.C. Canada, 2001.

8. R. Vdovjak, P. Barna, and G.-J. Houben. Eros:explorer for rdfs-based ontologies. In *Proceedings of Intelligent User Interfaces*, Florida, USA, 2003.