SEMANTEXPLORER: A Semantic Web Browser

Simon Scerri, Charlie Abela, and Matthew Montebello

Department of Computer Science and Artificial Intelligence
University of Malta
Malta

Abstract. The Semantic Web [17] 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

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 [18], an Ontology Language that is built on top of the Resource Description Framework [13], the associated schema [14] 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 [18]. 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 [5]. 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.
– 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 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 4 presents a discussion and comparism of semantExplorer with similar current technologies, while ideas for future work are discussed in section 5, after which we give some concluding remarks.

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.

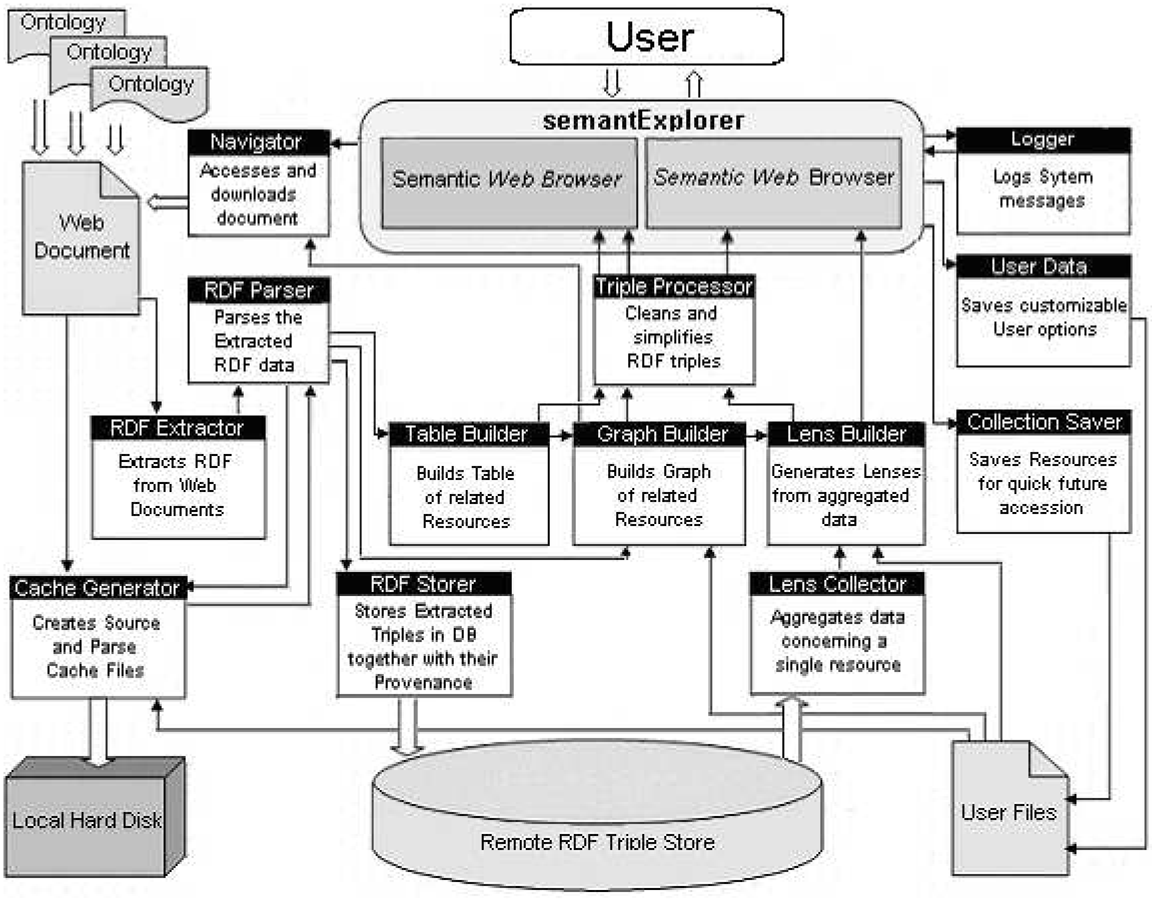

Ontology
Ontology
Ontology
User
semantExplorer
Semantic Web Browser
Semantic Web Browser
Navigator
Accesses and downloads document
Web Document
Logger
Logs Sytem messages
User Data
Saves customizable User options
RDF Parser
Parses the Extracted RDF data
Triple Processor
Cleans and simplifies RDF triples
RDF Extractor
Extracts RDF from Web Documents
Table Builder
Builds Table of related Resources
Graph Builder
Builds Graph of related Resources
Lens Builder
Generates Lenses from aggregated data
Collection Saver
Saves Resources for quick future accession
Cache Generator
Creates Source and Parse Cache Files
RDF Storer
Stores Extracted Triples in DB together with their Provenance
Lens Collector
Aggregates data concerning a single resource
Local Hard Disk
Remote RDF Triple Store
User Files

Fig. 1. 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 'University-OfMalta' 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 view each 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.

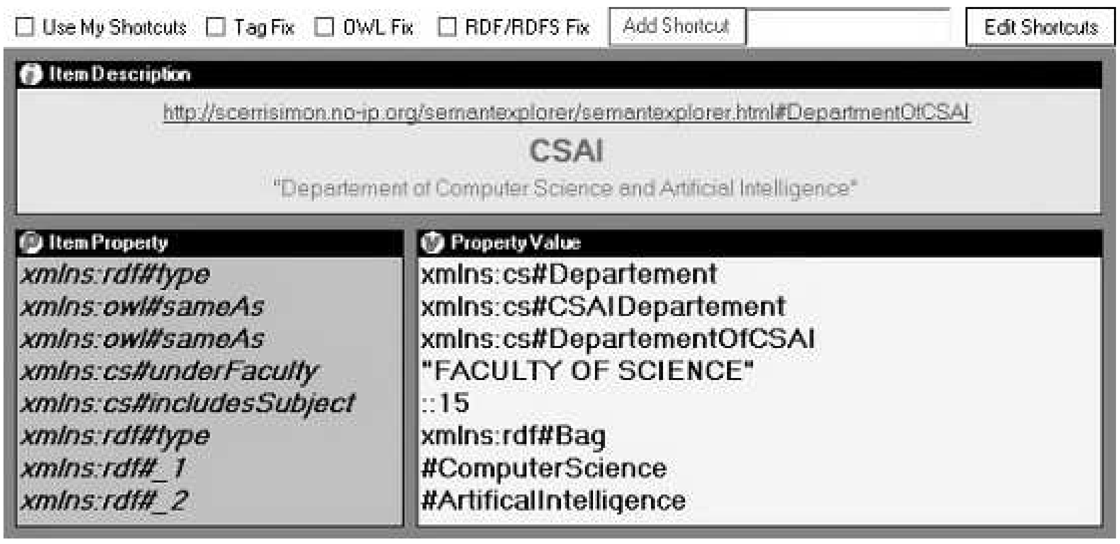
3 Evaluation of the system

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 [15] 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 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.

The 'Item Description' view, Figure 2, 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.



Item Description

http://scerrisimon.no-ip.org/semantexplorer/semantexplorer.html#DepartmentOfCSAI

CSAI

"Departement of Computer Science and Artificial Intelligence"

Item Property	Property Value
xmlns:rdf#type	xmlns:cs#Departement
xmlns:owl#sameAs	xmlns:cs#CSAIDepartement
xmlns:owl#sameAs	xmlns:cs#DepartementOfCSAI
xmlns:cs#underFaculty	"FACULTY OF SCIENCE"
xmlns:cs#includesSubject	::15
xmlns:rdf#type	xmlns:rdf#Bag
xmlns:rdf#_1	#ComputerScience
xmlns:rdf#_2	#ArtificalIntelligence

Fig. 2. Item Description showing information on a resource

Figure 3 shows the resulting graph output for the 'DepartmentOfCSAI resource. Comparing this with the data seen in Figure 2, 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 2 is simply shown as a multi-object node in Figure 3. 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.

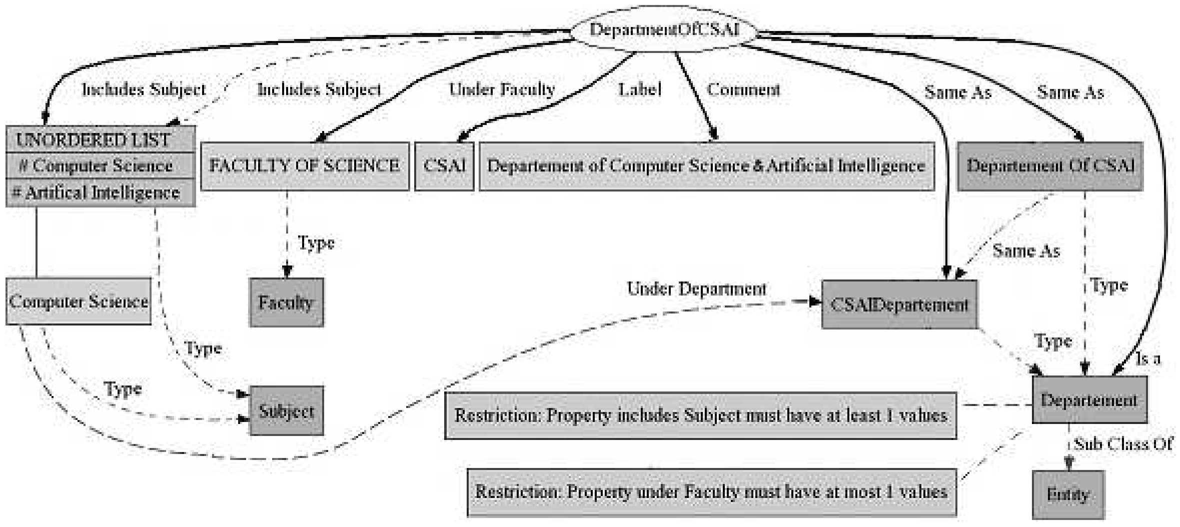
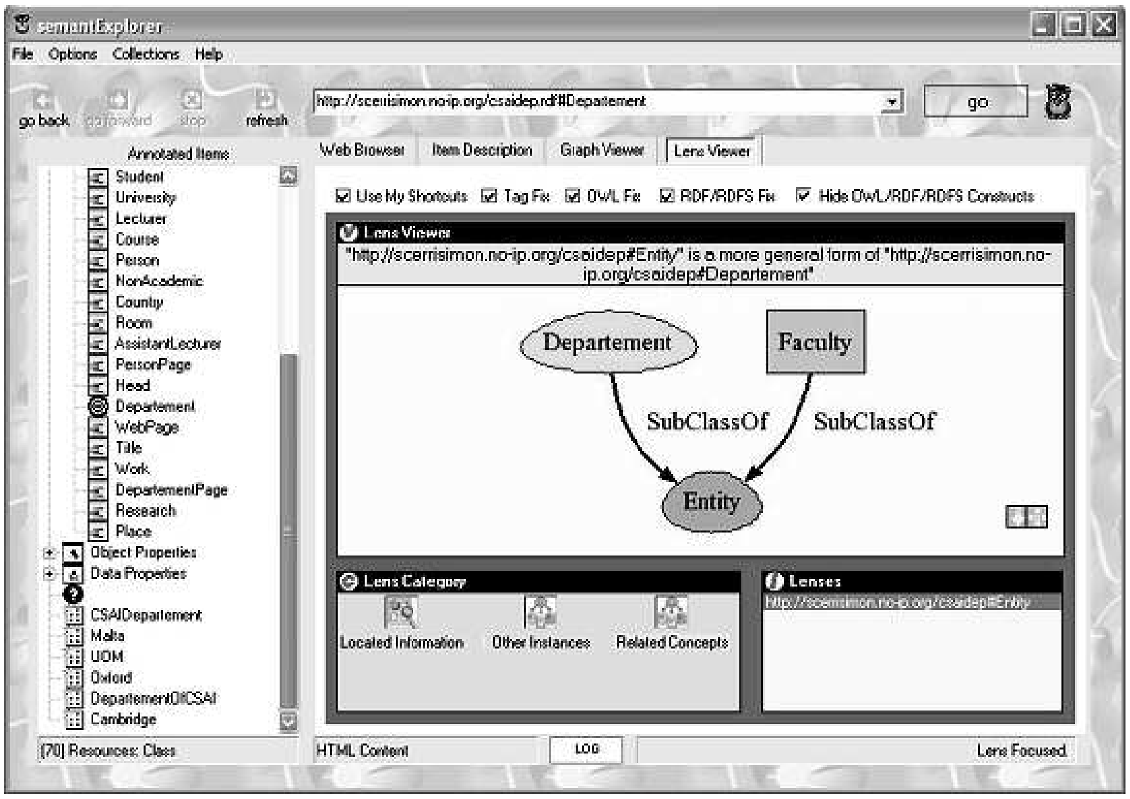


Fig. 3. Graph viewer output showing a detailed graph on a resource

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.



[✓] Use My Shortcuts [✓] Tag Fix [✓] OWL Fix [✓] RDF/RDFS Fix [✓] Hide OWL/RDF/RDFS Constructs

"http://scerrisimon.no-ip.org/csaidep#Entity" is a more general form of "http://scerrisimon.no-ip.org/csaidep#Departement"

Fig. 4. Lens Viewer within semantExplorer

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 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.

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 [8] is a tool that extends standard web browsers with the ability to visualize hidden metadata. It provides two methods for browsing [6]. 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. 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 [4] 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 [11] 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 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 [16] 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--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.

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.

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.

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, analyse and integrate the metadata they are annotating or working with.

References

1. Albrecht A.J., "Measuring Application Development Productivity", Proceedings of IBM Application Development Symposium, 1979
2. Arthur, L.J., Measuring Programmer Productivity and Software Quality, Wiley-Interscience, 1985
3. Jones C., "Applied Software Measurement", McGraw-Hill, 1991
4. Damian Steer, 2002, BrownSauce RDF Browser, (Online), Available from: http://brownsauce.sourceforge.net/
5. Dennis Quan, et al, May 2004, How to Make a Semantic Web Browser, World Wide Web Conference 2004, New York, New York, USA, pp. 2-3
6. Dzbor, Dr Martin et al, 2003, Magpie - towards a semantic web browser, Proceedings 2nd International Semantic Web Conference (ISWC2003), Sundial Resort, Sanibel Island, Florida, USA. pp. 10-11
7. Graph Visualization Software, (Online), Available from: http://www.graphviz.org/
8. Knowledge Media Institute, 2002, Magpie - The Semantic Filter, The Open University, (Online), Available from: http://kmi.open.ac.uk/projects/magpie/main.html
9. MbUnit, QuickGraph - Generic Graph Library for .NET, (Online), Available from: http://mbunit.tigris.org/
10. Microsoft, AxSHDocVw Web Browser Component, (Online), Available from: http://msdn.microsoft.com/library/default.asp?url=/workshop/browser/webbrowser/reference/objects/webbrowser.as
11. Haystack Project, (Online), Available from: http://haystack.lcs.mit.edu/
12. Drive: An RDF Parser for .NET, (Online), Available from: http://www.driverdf.org
13. RDF, 1997, Resource Description Framework, (Online), Available from: http://www.w3.org/RDF/
14. RDFS, 2003, RDF Vocabulary Description Language 1.0: RDF Schema, (Online), Available from: http://www.w3.org/TR/rdf-schema/
15. semantExplorer, 2005, semantExplorer A Browser For The Semantic Web, (Online), Available from: http://staff.um.edu.mt/cabe2/supervising/undergraduate/swbrowsing/semantexplorer.html
16. SIMILE, 2004, Piggy-Bank, (Online), Available from: http://simile.mit.edu/piggy-bank/
17. World Wide Web Consortium, 2001, Semantic Web, (Online), Available from: http://w3c.org/2001/sw/
18. World Wide Web Consortium, 2003, Frequently Asked Questions about RDF, (Online), Available from: http://www.w3.org/RDF/FAQ
19. World Wide Web Consortium, 2004, Web Ontology Language, [Online], Available from: http://www.w3.org/2004/OWL/