Semantic Turkey: A Semantic Bookmarking Tool (System Description)

Donato Griesi, Maria Teresa Pazienza, and Armando Stellato

AI Research Group, Dept. of Computer Science, Systems and Production
University of Rome, Tor Vergata
Via del Politecnico 1, 00133 Rome, Italy
{griesi,pazienza,stellato}@info.uniroma2.it

Abstract. In this work we introduce *Semantic Turkey*, a Semantic Extension for the popular web browser Mozilla Firefox. Semantic Turkey can be used to keep track of relevant information from visited web sites and organize collected content according to a personally defined ontology. Clear separation between knowledge data (the WHAT) and web links (the WHERE) is established into the knowledge model of the system, which allows for innovative navigation of both the acquired information and of the pages where it has been collected. This paper describes the architecture of the Semantic Turkey extension for Firefox, analyzes its development, shows its most interesting features and presents our plans for future improvements of the tool.

1 Introduction

In this work we introduce Semantic Turkey, a Semantic Extension for the popular web browser Mozilla Firefox [3], which can be used to annotate information from visited web sites and organize this information according to a personally defined ontology. Semantic Turkey should not be addressed as a "Semantic Web Browser" (whatever the nature of this term, which will probably take shape in the next future); it is intended as a personal desktop solution for organizing and managing the relevant information which is observed during web navigation, an advanced replacement for the traditional "Favorites" menu, offering clear separation between knowledge data (the WHAT) and web links (the WHERE), thus allowing for innovative navigation of the acquired information as well as of the pages where it has been observed.

2 Motivations and Approach Followed

Our research work, funded by the FILAS (Finanziaria Laziale di Sviluppo) agency under contract C5748-2005, has been centred on providing innovative methodologies and instruments for browsing the web and for organizing information of interest gathered during navigation. A specific point which emerged in our interviews inside FILAS is the emerging great need for efficient recovery of already visited pages (and, more in general, of already accessed knowledge): people are often exposed to large quantities of information, which are not always useful when seen for the first time, though difficult to recover when needed. The result is that people often become frustrated by the classical "I've seen it somewhere, but I don't remember where!" problem. We thus focused on finding interesting solutions for collecting, managing and retrieving data observed during web navigation. Our key goal was to overcome the limited usability of bookmarks lists, which:

- see weblinks as first class citizens. They can be categorized by implicitly adding them to a bookmarks folder, but they are no way separated from the knowledge they represent. More links could be related to the same subject, but there is no way to represent this aspect, except from considering the subject as a folder itself, thus betraying the intended equation: folder = category. Also, in some cases, it could be important to identify the portion of a page which contains the relevant information which caused it to be bookmarked. (e.g., "John Doe" is cited in a long web document which is very generic and not directly related to John Doe; we would like to take note of the page, still maintaining the focus on the real subject of our interest and immediately recognize where it has been identified).
- do not foresee any kind of multiple categorization. Any folder cannot belong to two or more different folders (a kind of multiple inheritance between categories), nor can any single weblink belong (with the possible exception of new systems adopting virtual foldering) to more than one folder (multiple instantiation).
- single knowledge resources cannot assume any kind of structure. It is not possible to further characterize a weblink, or to relate it with other ones (except putting them in the same folder/category).

Our project headed towards the development of a sort of "semantic notepad"[1] offering basic functionalities for:

1. capturing information from web pages, both by considering the page as a whole, as well as by annotating portions of their text
2. editing a personal ontology for categorizing the annotated information and, possibly, to exchange information with other people and exporting to other tools.
3. navigating the structured information as an underlying semantic net which, populated with the many relationships which bind the annotated objects between them, eases the process of retrieving the knowledge which was buried by the past of time For example, a user could discover that two persons which he has kept track of in separate sessions (by annotating their presence and some aspects of their profiles appearing in different web pages), work in the same place, or have any kind of connection he would not recall with any kind of traditional bookmarking/annotation service.
4. clearly separating the business model from the user interface, by adopting a "knowledge service" architecture. This way, the same architecture could be exploited for an enhanced personal web browser as well as for a shared environment for collaborative semantic tagging of web pages.

[1] "Taccuino" is the italian word for the term "Notebook". In our lab, we hate so much the silly Italian expression "Taccuino Semantico" (Semantic Notebook) that we started to use any kind of misspelling of its name, the funniest (and most used) of which was "Tacchino Semantico" (Semantic Turkey). The rest is history.

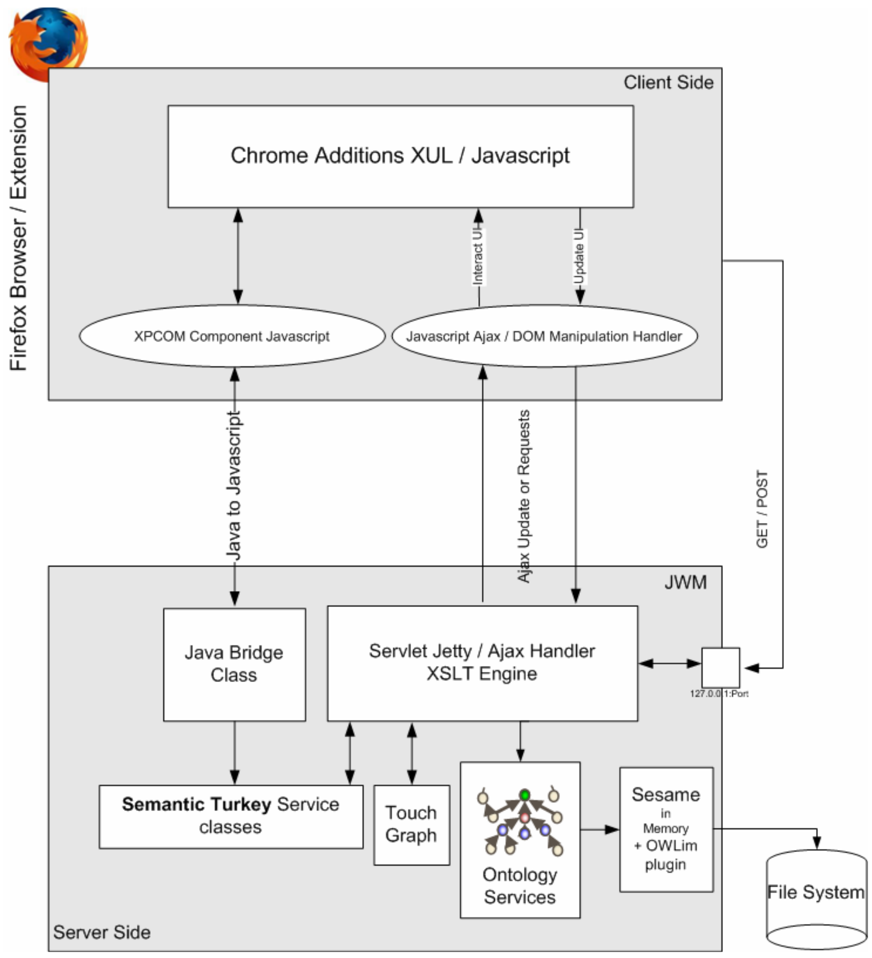

Firefox Browser / Extension
Client Side
Chrome Additions XUL / Javascript
Interact UI
Update UI
XPCOM Component Javascript
Javascript Ajax / DOM Manipulation Handler
Java to Javascript
Ajax Update or Requests
GET / POST
JWM
Java Bridge Class
Servlet Jetty / Ajax Handler XSLT Engine
127.0.0.1:Port
Semantic Turkey Service classes
Touch Graph
Ontology Services
Sesame in Memory + OWLim plugin
File System
Server Side

Fig. 1. Architecture of Semantic Turkey

3 Architecture

The architecture (Fig. 1) of Semantic Turkey consists in a web application, designed using a three layered approach.

The first layer, the presentation layer, has been developed as an extension for the web browser Firefox. Everything relating to user interaction is directly managed by the Firefox extension, thanks to a solution directly integrated in the browser. This approach has two main advantages: total reuse of the functionalities of a well assessed, stable and complete software for web browsing, and a non invasive offer for the user, who can still use the web browser he has been acquainted with.

The second layer, the service layer, is realized through a collection of Java Web Services, published through the Web Server “Jetty” [8]. Jetty is implemented entirely in Java, and the architecture foresees its use as an embedded component. This means that the Web Server and the Web Application run in the same process, without interconnection overheads and other sort of complications. This solution also allows for a flexible use of the tool, since it can both be adopted as a completely autonomous web browser extension, as well as a personal access point for collaborative web exploration and annotation: in the latter case, a centralized solution is being adopted, in which clients communicate with the same server.

The third layer, the persistence layer, comprehends the component for managing the ontology, which is represented in the OWL language [10]. This layer has been realized by using Sesame [1] and the OWLIM plugin [9]. Sesame is an open source RDF database with support for RDF Schema inference and querying. Since the Knowledge Model of Semantic Turkey is expressed in the OWL Lite [11] dialect of the Web Ontology Language, the OWLIM plugin has been employed to provide OWL Lite reasoning to the Sesame component.

3.1 Architectural Layers

The following sections describe more in detail the three layers which constitute the architecture of Semantic Turkey.

Presentation Layer. As previously mentioned, the presentation layer has been realized as an extension to the web browser Firefox. The User Interface has been created through a combined use of the XML User Interface Language XUL [17], XBL [15] and Javascript language. Physically it appears as a sidebar, containing the ontology tree, which may be shown on the left side of the window by selecting dedicated “ontology” item added in “Tools” menu. The icons that represent the nodes of the tree distinguish between classes and instances that belong to the ontology.

The ontology is loaded/updated through calls to the server, carried out using the Ajax [5] technique: the data – in XML format – is thus mainly exchanged between the two layers in an asynchronous way, to preserve good performance and to not penalize the activity of the browser.

The extension has also another prerogative, which is not an ordinary feature of the presentation layer: it has to assure that the web server is being loaded as an embedded component, at the start of the browser process. To do that XPCOM [16] components, written in JavaScript, have been developed for linking the chrome part and the Java part.

In order to load the Java component, the Simile Java Firefox Extension [12] has been used. This component allows to load java classes or jar packages, instantiate objects and to invoke static methods or methods of the object previously instantiated.

At the start of the browser process, after loading the java components (the java server code and the required libraries), a static method is being invoked with the role of instantiating the web server. This solution makes it possible to install all the application simply as a Firefox extension, without configuring other software.

Service Layer. This layer offers services which may be invoked through http requests submitted according to the Ajax paradigm, thus enabling communication between the client (Firefox extension) and the server. The server receives the requests coming from the client by GET or POST http calls, carries out the operations associated to these calls, and in case replies with an XML response. If a call implies the return of a XHTML page, a XSLT transformation is being performed, in order to decouple the data model with its manifestation in the presentation layer.

The majority of invocations to the server are being completed in an asynchronous way, so that, independently from the workload that is subjected the server, the browser can continue to respond to the user. This is a crucial issue for the usability of the application: expensive computations blocking normal behavior of the browser would otherwise not be tolerated by the user.

Besides supporting the communication with the client, the service layer provides the functionalities for definition, management and treatment of the data. Several objects are described through an ontological model (see next section), to represent both pure conceptual knowledge as well as application required information.

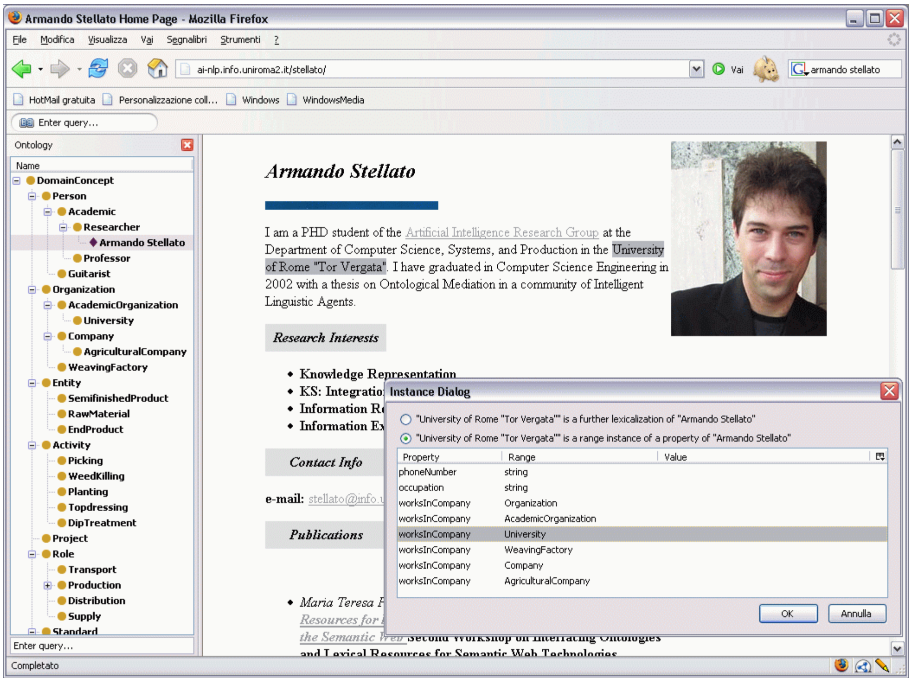
Finally, the service layer also provides another important functionality linked with the presentation layer. It allows for the capability of visiting the ontology through a graph view, using the TouchGraph library [14]. TouchGraph is an open source tool for visualizing networks of interrelated information. It renders networks of information concepts as interactive graphs that lend themselves to a variety of transformations. By engaging with the visual image, a user is able to navigate through large networks of information and to explore different ways of arranging the network's components on the screen. This functionality has been positively judged by the technophores, as it allows unexpected correlations to emerge from the network of information.

In order to access TouchGraph from presentation layer, a dedicated java applet and related servlet have been realized. The servlet works like a proxy, redirecting the applet loaded, with the correct parameters, to the client side.

Persistence Layer. Sesame provides the abstraction layer over ontological data. The foundation of the component is the Storage And Inference Layer (SAIL). This SAIL is an API that abstracts from the storage device used (in-memory storage, disk-based storage, RDBMS) and takes care of inference.

From the architecture perspective the Access APIs are the most important component. These APIs provide high-level access functionality to client applications, either locally or remotely (over HTTP or RMI).

Sesame can thus be deployed as an RDF database, with persistence in an RDBMS, or as a Java library for embedded use in applications. This last modality has been employed for the definition of the architecture. In our case, the ontology data is, by default, handled in memory and stored in the (local) File System, but it is possible to easily switch to the database storage backend for managing very large ontologies. Also, the ontology repository may be located in a different, remote, site, thus offering different possibilities for decentralizing the application.



Armando Stellato

I am a PHD student of the Artificial Intelligence Research Group at the Department of Computer Science, Systems, and Production in the University of Rome "Tor Vergata". I have graduated in Computer Science Engineering in 2002 with a thesis on Ontological Mediation in a community of Intelligent Linguistic Agents.

Research Interests

- Knowledge Representation
- KS: Integratio
- Information R
- Information Ex

Contact Info

e-mail: stellato@info.u

Publications

- Maria Teresa P
Resources for
the Semantic Web Second Workshop on Interfacing Ontologies and Lexical Resources for Semantic Web Technologies

( ) "University of Rome "Tor Vergata"" is a further lexicalization of "Armando Stellato"
(✓) "University of Rome "Tor Vergata"" is a range instance of a property of "Armando Stellato"

Property	Range	Value
phoneNumber	string	
occupation	string	
worksInCompany	Organization	
worksInCompany	AcademicOrganization	
worksInCompany	University	
worksInCompany	WeavingFactory	
worksInCompany	Company	
worksInCompany	AgriculturalCompany	

Fig. 2. Annotating concepts from a web page and establishing relationships between them

3.2 The Knowledge Model

The knowledge model of Semantic Turkey has been structured into four different layers of ontological knowledge:

1. *The Application ontology*: This ontology contains resources needed by Semantic Turkey to organize, retrieve and present information to the user.
2. *The Top Ontology* (which owl:import the Application Ontology): this ontology has originally been conceived inside our project for FILAS, and is thought for representing a minimal knowledge which should be shared across the different technophores. This ontology can simply be seen as a guideline for driving the personal annotations of each of the technophores, and could be used as well as a shared ontology for exchanging information between them.
3. *The Personal/Domain Ontology* (which owl:import the Top Ontology): The third ontological layer allows for a personalized organization of the knowledge which is extracted and collected from the web.
4. *The Knowledge Base* (which owl:import the Top Ontology), i.e. the set of instances which populate the personal ontology of the user.

The Application ontology is composed of resources useful for managing the annotation functionalities. These, among the others, include the classes:

– `Annotable` identifying the part of the ontology which can be annotated by the user
– `URL` which stores links to the visited pages

- `SemanticAnnotation` containing the annotations performed by the user, described by their URL, related concept etc…

and the properties:

- `has_location` linking URLs with *Annotable* concepts
- `observed_lexicalization` describing the form with which a given object appeared in a specific annotation. this property has been preferred to a more precise information, like reporting the byte offset of the annotation inside the page, to make retrieval of the annotated object more robust with respect to minor changes that occurred to the page over time.

The Application ontology is invisible to the user and is only exploited by the application to get the proper logic for administering the upper ontological layers. Key elements for the annotation process are expressed in terms of concepts from this ontology.

Resources originated from the Top ontology are read-only, and cannot be deleted as a consequence of any edit operation by the user. In a really general perspective, the Top Ontology could even be left empty (i.e. if there is no supposed shared conceptualization which must be adopted by users working on a common annotation framework; in this case, each user can build from scratch its own conceptualization, which will be thus constituted by the sole *Personal Ontology*), or external resources could be imported, possibly exchanging their content with other applications, like a mail browser (e.g. by adopting the FOAF ontology [4] for managing contacts) or a client for instant messaging. The *Personal Ontology* is the last conceptual layer which can be modeled according to personal preferences, perspectives and needs.

4 User Interaction

Semantic Turkey offers some basic editing operations for populating the *personal ontology* with annotations from visited web sites, as well as search and navigation functionalities which facilitate the recovery of already acquired knowledge.

4.1 Main Functionalities

The user may interact with the ontology panel to modify its personal ontology, through a series of operations, which we describe here, organized into categories.

Interaction with the browser. These mainly include drag&drop operations which allow to annotate information from the visited sites:

1. Drag and drop of a selection of a text from an html document displayed in the browser, on the icon that represents a class, in order to create an individual of that class. The selection will become the ID of the new individual and a new icon will be shown below the selected class
2. Drag and drop of a selection of text from an html document, on the icon that represents an individual, in order to characterize a property which that individual owns. A specific window will open, prompting the user to choose the fitting property. The selection will become the ID of a new individual that represents the

instance of the range of the property chosen. If the selected property is an object property, a new icon will be created relatively to the range class.

3. Drag and drop of a selection of text from an html document, on the icon that represents an individual, in order to define a further lexicalization for that individual. The user can choose, from the same panel described before, if the selection characterizes a range of a property or a new *observed lexicalization* (see section 3.2).

Direct Ontology Editing. These functionalities operate exclusively on the ontologies, as it should be important for the user to integrate its knowledge with information he would acquire through other media (communication with other people, radio, tv etc…). These include:

1. *Semantic Editing*. It is possible to create, modify and/or delete new classes/individuals/properties. All the operations are being carried out through specific panels that are activated by a context menu associated to the nodes of the tree, in a way much similar to traditional ontology editing tools, like Protégé [6] or TopBraid Composer [13]. By offering complete interaction with the ontology via the XUL interface (instead of an HTML interface, like in Piggy-Bank), the user is not diverted from his current navigation (i.e. the main browser panel is still focused on the visited web page, which would otherwise be replaced by the HTML UI) and may maintain its attention over the observed web page.

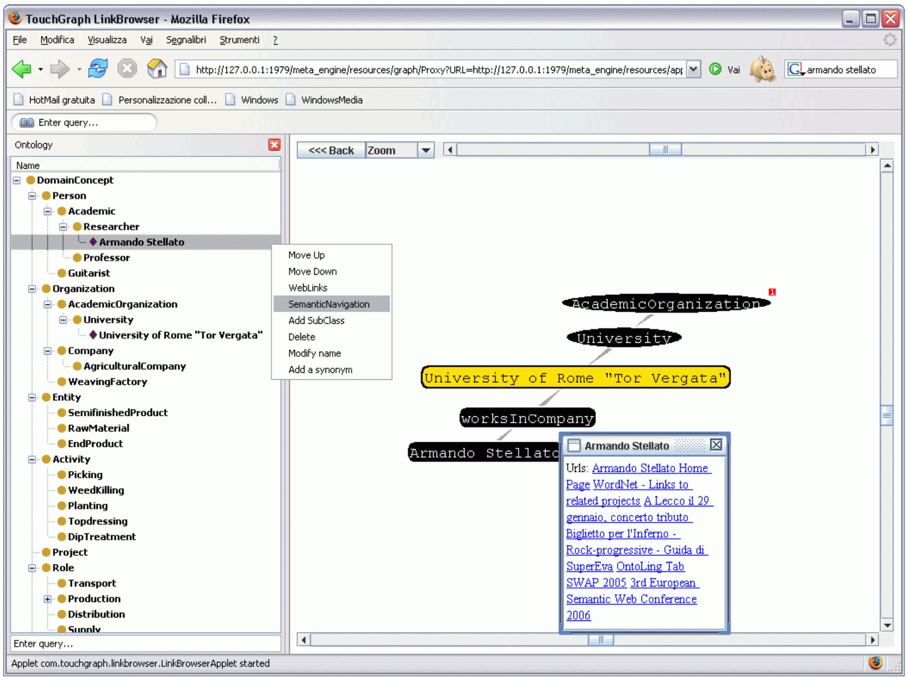

TouchGraph LinkBrowser - Mozilla Firefox
File Modifica Visualizza Vai Segnalibri Strumenti ?
http://127.0.0.1:1979/meta_engine/resources/graph/Proxy?URL=http://127.0.0.1:1979/meta_engine/resources/app
Vai
armando stellato
HotMail gratuita Personalizzazione coll… Windows WindowsMedia
Enter query…
Ontology
Name
DomainConcept
Person
Academic
Researcher
Armando Stellato
Professor
Guitarist
Organization
AcademicOrganization
University
University of Rome "Tor Vergata"
Company
AgriculturalCompany
WeavingFactory
Entity
SemifinishedProduct
RawMaterial
EndProduct
Activity
Picking
WeedKilling
Planting
Topdressing
DipTreatment
Project
Role
Transport
Production
Distribution
Supply
Enter query…
Move Up
Move Down
WebLinks
SemanticNavigation
Add SubClass
Delete
Modify name
Add a synonym
<<< Back
Zoom
AcademicOrganization
University
University of Rome "Tor Vergata"
worksInCompany
Armando Stellato
Armando Stellato
Urls: Armando Stellato Home Page WordNet - Links to related projects A Lecco il 29 gennaio, concerto tributo Biglietto per l'Inferno - Rock-progressive - Guida di SuperEva OntoLing Tab SWAP 2005 3rd European Semantic Web Conference 2006
Applet com.touchgraph.linkbrowser.LinkBrowserApplet started

Fig. 3. Semantic Navigation: recalling ontology and web links for "Armando Stellato"

2. *Lexical Editing*. Add synonyms and documentation for the concepts. These alternative lexicalization provide several anchors for referring the same ontological entries. This solution facilitates retrieval of knowledge objects when the ontology reaches a considerable growth, or simply when its knowledge is transferred to other users. Advanced search functionalities over the ontology objects and their lexicalizations in different languages, have been made available thanks to an embedded indexing engine [7] and the adoption of a library implementing different string matching algorithms [2].

Semantic Navigation. As an additional feature, the user may graphically explore the ontology (Fig. 3), thanks to the *SemanticNavigation* component. A Java applet will be loaded on a new tab of the browser displaying the graph view of the ontology, allowing the user to navigate its content and get back to the pages related to the annotated knowledge. Conversely, Semantic Turkey reports to the user, through a dedicated status bar, the pages which have been previously annotated. When the user visits an already annotated page, an icon with the shape of a pencil is being shown in the lower part of the browser. If the icon is being clicked, the html text entries that represent the past annotations will be emphasized (providing the page still contains those entries) with a light background color.

5 Conclusions

In this paper Semantic Turkey, a special environment for supporting end users in annotating information caught from visited web sites, has been described.

Main objective of our first experience in developing Semantic Turkey has been to extend "usual" web browsing modalities, with a particular focus on efficient and intuitive retrieval of information already observed during past navigation. A key characteristic of this approach has been to separate the role of site bookmarking from the more complex aspect of knowledge management and, at the same time, to interweave both of them in a homogeneous perspective over the two dimensions of the Web: traditionally exposed documents and the new web of data fostered by the Semantic Web. We are now in the direction of refining the overall architecture to meet more general requirements which would make Semantic Turkey an open and reusable platform. In particular, the multilayered approach in the knowledge model must be flexible enough to allow the user to import and reuse any number of available ontologies, while an extension mechanism should make it easy to produce specific add-ons for adding new functionalities to the browser. The flexibility offered by the client-server paradigm in the overall architecture should also be exploited to offer the possibility of performing and handling concurrent accesses to remote ontology repositories, effectively transforming the system in a client front-end for collaborative ontology management.

Acknowledgements

Our research work on Semantic Turkey has been funded by the FILAS (Finanziaria Laziale di Sviluppo) agency under contract C5748-2005.

References

1. J. Broekstra, A. Kampman & F.v. Harmelen I. Horrocks & J. Hendler (ed.) Sesame: “A Generic Architecture for Storing and Querying RDF and RDF Schema”. Springer Verlag, Proceedings of the First International Semantic Web Conference, Sardinia, Italy, pages 54-68, July 2002
2. W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In Proceedings of the IJCAI-2003.
3. Firefox home page: http://www.mozilla.com/en-US/firefox/
4. Friend Of A Friend Ontology (FOAF): http://xmlns.com/foaf/0.1/
5. J.J. Garrett. “Ajax: A New Approach to Web Applications”. Feb. 18, 2005 http://www.adaptivepath.com/publications/essays/archives/000385.php
6. J. Gennari, M. Musen, R. Fergerson, W. Grosso, M. Crubézy, H. Eriksson, N. Noy, and S. Tu. The evolution of Protégé-2000: An environment for knowledge-based systems development. International Journal of Human-Computer Studies, 58(1):89–123, 2003
7. Erik Hatcher and Otis Gospodnetić. Lucene in Action. Manning ed. 456 pages. 2004. ISBN: 1932394281
8. Jetty Java HTTP Servlet Server. http://jetty.mortbay.org/jetty/.
9. Kiryakov, D. Ognyanov & D. Manov OWLIM – a Pragmatic Semantic Repository for OWL. In Proc. of Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), WISE 2005, New York City, USA, 20 November 2005
10. Web Ontology Language: http://www.w3.org/TR/owl-features/
11. OWL Lite Description: http://www.w3.org/TR/2004/REC-owl-features-20040210/#s3
12. Simile Java Firefox Extension: http://simile.mit.edu/java-firefox-extension/
13. TopBraid Composer: http://topbraidcomposer.info/
14. Touchgraph Development Page: http://touchgraph.sourceforge.net/
15. Extensible Binding Language: http://www.mozilla.org/projects/xbl/xbl.html
16. XPCOM. http://www.mozilla.org/projects/xpcom/
17. XML User Interface Language (XUL) Project. http://www.mozilla.org/projects/xul/