SEMANTIC TURKEY
A New Web Experience in between Ontology Editing and Semantic Annotation

Francesca Falluchi, Maria Teresa Pazienza, Noemi Scarpato and Armando Stellato
DISP, University of Tor Vergata, Via del Politecnico 1, Rome, Italy

Keywords: Semantic Browsing, Semantic Annotation, Semantic Bookmarking, Ontology Editing.

Abstract: In this work we describe *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. In this sense, Semantic Turkey can be seen both as an advanced semantic bookmarking system as well as an ontology editing assistant, which allows domain experts and ontology developers to build ontologies starting from the very raw source of information which they find on the web. The open architecture of Semantic Turkey and the specific three-layered approach of its design also allows for scaling the basic personal desktop application embodied by this tool up to a distributed framework for collaborative semantic annotation and ontology editing. This paper describes the architecture and the functionalities of the Semantic Turkey extension for Firefox, and describe possible evolutions for future improvement of the tool.

1 INTRODUCTION

While new solutions for Semantic Desktops and Innovative Browsing of file systems (Leaftag project; Wheeler, 2005) are arising in these days, one of the instruments which is mostly correlated to WWW – the Web Browser – and which should be more prone to the wind of innovation blowing from the Semantic Web, remains confined to its old fashioned interaction modalities. Web browsers currently offer no more than the classical services – for collecting and organizing bookmarks and for retrieving the history of past navigation activity – which we have been used to since the beginning of the Web. The web is however not the same of ten years ago: today much of the information which is intended to be public can be accessed even (if not solely) from the WWW so that every web user is exposed to a huge amount of knowledge and data which is difficult to manage and retrieve. New paradigms are thus necessary to support users in web browsing and in the process of collecting and retrieving the data which is observed during navigation.

In this work we describe Semantic Turkey, a Semantic Extension for the popular web browser Mozilla Firefox (Firefox home page), 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 near future): in its current form, it is intended as a personal desktop solution for organizing and managing the relevant information which is observed during web navigation. So, considering it under different perspectives and according to different needs, it could both be adopted as an advanced replacement for the traditional "*Bookmarks*" menu, offering clear separation between knowledge data (the WHAT) and web links (the WHERE), as well as an assistant for ontology developing, allowing users to easily grab information from the web and build new concepts, objects and relations to produce new ontologies or populate existing ones. As an additional possibility, the architecture of the tool has been conceived to allow for an easy scale-up to a distributed and collaborative framework for semantic annotation and ontology editing.

2 RELATED WORKS

As an evidence of the great interest in the matter, several actions have been undertaken in the last

Falluchi F., Teresa Pazienza M., Scarpato N. and Stellato A. (2008).
SEMANTIC TURKEY - A New Web Experience in between Ontology Editing and Semantic Annotation.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 90-97
DOI: 10.5220/0001526600900097

years towards the realization of so called *Semantic Browsing* solutions for the Web. In this section we will briefly recall a few of them, also addressing other topicalities like *advanced bookmarking* systems and services and *semantic annotation* tools.

The Haystack (Quan & Karger, May, 2004) web client, developed at the MIT laboratories, was conceived as an application that could be used to browse arbitrary Semantic Web information in much the same fashion as a Web browser can be used to navigate the Web. Standard point-and-click semantics let the user navigate over aggregation of RDF repositories from different arbitrary locations. The application has been built as an extension for the popular Integrated Development Environment Eclipse (Eclipse Platform Technical Overview); this choice facilitates extension of the tool thanks to Eclipse flexible plug-in mechanism, but requires the user to adopt Eclipse as a platform for browsing the web and collecting data from it: a negative aspect for the average user, who would just prefer to rely on his trusted personal web browser and try out other features which are not too invasive for his usual way of working.

An opposite approach is being followed by Magpie (Dzbor, Domingue, & Motta, 2003), which is deployed as a plug-in for the Microsoft Internet Explorer Web Browser. In its first incarnation, Magpie allowed for semantic browsing, intended as the parallel navigation of purely "exposed" web content and of its associated semantic layer (an ontology associated to the web resource, which semantically describes its content). Magpie also allows for collaborative semantic web browsing, in that different persons may gather information from the same web resource and exchange it on the basis of a common ontology. Recent work on Magpie (Dzbor, Motta, & Domingue, 2004) extended the platform more and more towards the vision of the Semantic Web as "an open web of interoperable applications" (Berners-Lee, Hendler, & Lassila, 2001), by allowing bi-directional exchange of information among users and services, which can be opportunistically located and composed, either manually (web services) or automatically (semantic web services).

From (part of) the same authors of Haystack, comes Piggy-Bank (Huynh, Mazzocchi, & Karger, November, 2005), an extension for the Firefox web browser (Firefox home page) that lets Web users extract individual information items from within web pages and save them in RDF, replete with metadata. Piggy Bank then lets users make use of these items right inside the same web browser. These items, collected from different sites, can then be browsed, searched, sorted, and organized, regardless of their origins and types. Piggy-Bank users may also rely on Semantic Bank, a web server application that lets them share the Semantic Web information they have collected, enabling, as for Magpie, collaborative efforts to build sophisticated Semantic Web information repositories from daily navigation through their enhanced web browser.

Though not being directly related to the category of "semantic browsing" solutions, it is however impossible to not mention recent trends in "social bookmarking" tools. The most popular one, del.icio.us (del.icio.us.), is a service for building personal collections of bookmarks and access them online. It is possible, through the same service, to add links to a collection of bookmarks, to categorize the related sites with keywords, and to share the personal collection with other users. Google recently offered a similar solution with its Google Notebook (Google Notebook). The idea is quite simple: open a scratch electronic paper from within your web browser, and let the user add not only bookmarks, but write complete multimedia comments (by using Google Page Creator technology (Google Page Creator)), which can also be shared with other people (currently, this sharing service is not yet active).

What lacks from the previous approaches (with the possible exception of Magpie which, on the other hand, realizes different objectives) is a really integrated environment extending a web browser with (light) knowledge management facilities and efficient retrieval of acquired information, all put at the hands of the user on its workstation (in opposition to current trends promoting service-based utilities which suppose an always-online working environment and entirely "webbed" user interfaces).

3 MOTIVATIONS AND APPROACH

Semantic Turkey had been initially developed as a prototype for an advanced bookmarking system (Griesi, Pazienza, & Stellato, 2007) with information management capabilities centered around modern Semantic Web knowledge representation models and technologies.

Our idea was to offer a sort of "semantic notepad" with basic functionalities for:

1. capturing information from web pages – both by considering the page as a whole, as well as by selecting portions of their text – and annotating it with respect to a personal ontology
2. editing the above ontology for classifying the annotated information and for better characterizing its interests according to its descriptive properties (attributes and relations)

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

In particular, we coined the expression *Semantic Bookmarking* to indicate the process of *annotating* information from (web) documents, to *acquire* new knowledge and *represent* it through knowledge representation standards. In this sense, Semantic Turkey differentiated from similar, previously described, tools, as it offered a lightweight structure, which completely exploits the infrastructure of the hosting web browser (with respect to, for example, the complex completely-web based interface of Piggy-Bank) and which grants the user a good control over its personal knowledge model (while, e.g. Magpie only adopts ontologies which have been defined elsewhere). In the process of reengineering the original prototype and releasing it to the community, we made a further step in strengthening its unique combination of semantic annotation and editing functionalities, endowing it with full ontology editing and import capabilities, an even more modular and flexible architecture allowing for different ontology technologies to be plugged at need and a versatile knowledge model for coordinating user/domain and application-driven information.

4 ARCHITECTURE

The architecture (Fig. 1) of Semantic Turkey follows a three layered design, with the presentation layer embodying the true Firefox extension and the other two layers built around java technologies for administering the business logic and data access.

Everything relating 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" (Jetty Java HTTP Servlet Server). 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.

4.1 Architectural Layers

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

Presentation Layer. The User Interface has been created through a combined use of the XML User Interface Language XUL (XML User Interface Language (XUL) Project), XBL (Extensible Binding Language) and Javascript language.

The UI physically appears as a set of Firefox sidebar, representing ontological information. User requests are handled through the Ajax (Garrett, 2005) paradigm: 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.

Javascript XPCOM (XPCOM) components have been developed and the Simile Java Firefox Extension (Simile Java Firefox Extension) has been adopted for linking the chrome part and the Java part to start the Jetty embedded java server.

Middle 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 middle 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.

Data Layer. It is mainly constituted by the component for managing the ontology. This has recently been rewritten as a series of dedicated API for accessing ontological data: these offer both RDF

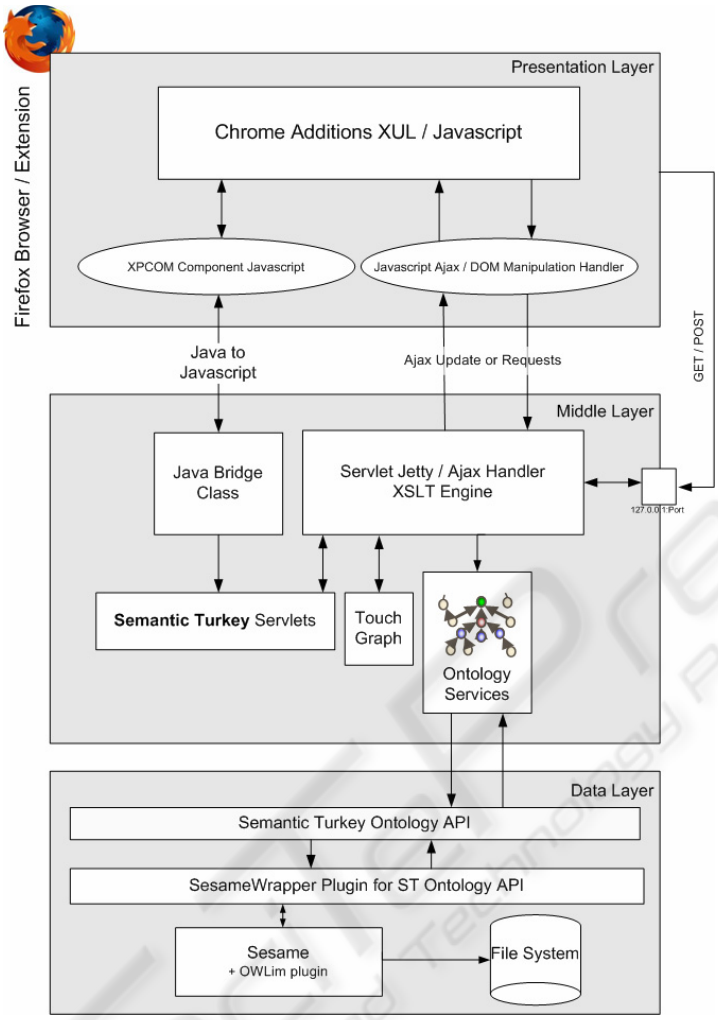

Figure 1: Semantic Turkey Architecture.

triple-level access methods as well as more object oriented facilities, which have been appreciated in RDF libraries like Jena (McBride, 2001). Semantic Turkey API constitute an interface which can be implemented by building wrappers for existing ontology libraries, so that we could easily select those which best fit the needs of a given situation (like working with small or large repositories, on a local or collaborative environment etc…) without having to modify the whole application. The first implementation of these API has been developed as a wrapper for Sesame (Broekstra, Kampman, & van Harmelen, 2002) and the OWLIM plugin (Kiryakov, Ognyanov, & Manov, 2005), which has been added for reasoning over OWL (Web Ontology Language webpage) data.

The above architecture easily allow for scaling the basic personal desktop application embodied by this tool up to a distributed framework for collaborative semantic annotation and ontology editing, as either (or both) the middle and data layer could be ported with little or no modification at all on a distributed environment, with extended Firefox clients accessing remote semantic repositories for concurrently accessing, annotating and editing ontological data.

We are currently pre-testing this possibility inside the Diligent (IST-004260) project, where data from web pages coming from different sites relating to the geospatial domain, will be semantically annotated according to a set of reference ontologies defined in the Impect portal (http://dl14.di.uoa.gr/gridsphere/gridsphere).

The objective is to populate Diligent internal repository with data coming from multiple sources thanks to mass collaborative work of several annotators. Keeping track of annotations' provenance is also necessary so that eventual reputation/validation mechanisms could be subsequently raised on top of this knowledge.

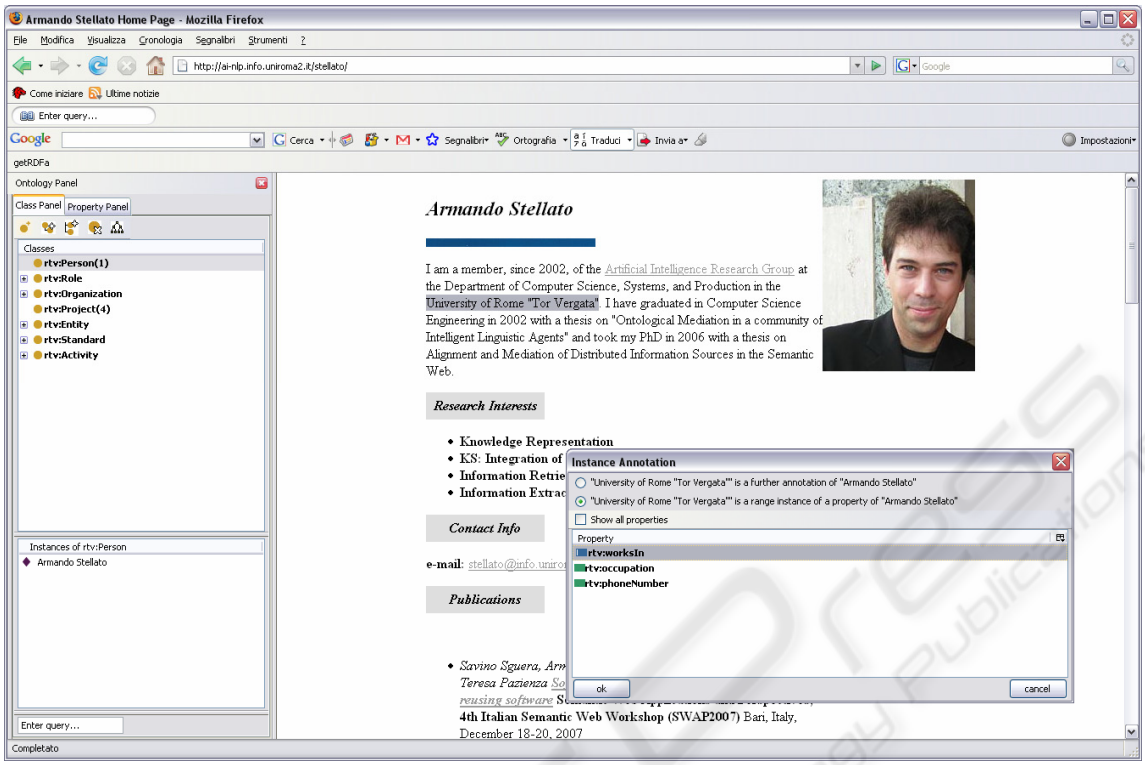

Figure 2: Building an ontological knowledge base through semantic annotations.

4.2 Knowledge Management

With respect to its former prototype, Semantic Turkey offers now a complete ontology editing environment, allowing users to import, edit and merge ontological data coming from different RDF/OWL sources. What has been maintained in this version is the main separation between the explicit knowledge managed by the user and the one which guides system's behavior. This last set, defined as *Application Ontologies*, at present state solely includes the *Annotation Ontology*: a set of concepts which are used to keep track of user annotations from the web, but could be expanded according to future extensions of Semantic Turkey. These, among the others, include the concepts:

- `WebPage` (subClassOf `Document`) concept for storing information about the annotated pages (mainly their `URL` and `title`), that is, the pages where part of the text is annotated with respect to the ontology and thus added to it as a new individual
- `SemanticAnnotation` containing the annotations performed by the user, described by their `URL`, annotated object etc… these can be both `TextualAnnotations` (for text annotated from the web page) as well as `ImageAnnotations` (for grabbed images)

The annotations also keep track of the different possible lexicalizations that a same object may have exposed into different web pages. Application Ontologies are invisible to the user, their content is however made implicitly accessible through dedicated functionalities offered by the tool (or viewed explicitly through settings of the platform).

5 USER INTERACTION

Semantic Turkey offers 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.

5.1 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 (Figure 2) 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 local name of the new individual, which will be shown inside the instances panel.
2. Drag and drop of a selection of text from an html document, on the icon that represents an individual, in order to ad a further bookmark for that individual, or to characterize a property which that individual owns. A specific window will open, prompting the user to choose the appropriate functionality. In the first case, a new semantic annotation is taken for the individual, with a new webpage as a bookmark for it and the new textual occurrence of that individual in the observed page. In the other case, the user can choose a property for enriching the description of the chosen individual through the selected text. If the selected property is an *owl:ObjectProperty*, the selection will become the name of a new individual created as an instance of the range class of the chosen property, or a further annotation for an existing individual. In both cases, the two individuals are bound through the selected property. In case of an *owl:Datatype* or *owl:AnnotationProperty*, a new value will be added.
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 0).

These functionalities have been conceived to speed up typical series of operations which characterize both the worlds of ontology development and semantic annotation. For example, the second one which has been described above performs, in case of an object property, the creation of a new instance, its annotation with the current web page and the assertion of a relationship between the new individual and the selected one, at the cost of just a drag&drop and a selection.

Direct Ontology Editing. These functionalities operate exclusively on the ontologies, as it should be important for the user to integrate the knowledge

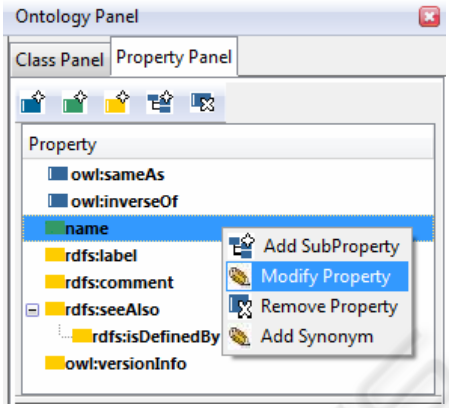

Figure 3: Editing properties in ST.

acquired through semantic bookmarking with information he could get through other media. All typical ontology editing operations (Figure 3) are carried out through buttons and context menus associated to the nodes of the tree, in a way much similar to traditional ontology editing tools, like Protégé (Gennari, et al., 2003) or TopBraid Composer (TopBraid Composer). 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, at the same time, maintain its attention over the observed web page. Extended support for natural language descriptions of ontology objects is also present in the system, allowing for explicit representations of the same objects through different synonimical expressions, or translation for different idioms, thus accounting for multilinguism.

Semantic Navigation. As an additional feature, the user may graphically explore the ontology, thanks to the *SemanticNavigation* component (Figure 4): a customized version of the TouchGraph library (Touchgraph Development Page). 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. The nodes of the graph will be displayed in different manners, according to the nature of the ontological entity: classes, properties or individuals. By dragging the mouse pointer on a node that represents an individual, it is possible to open a popup window, which contains the URLs of the pages where that instance has been annotated.

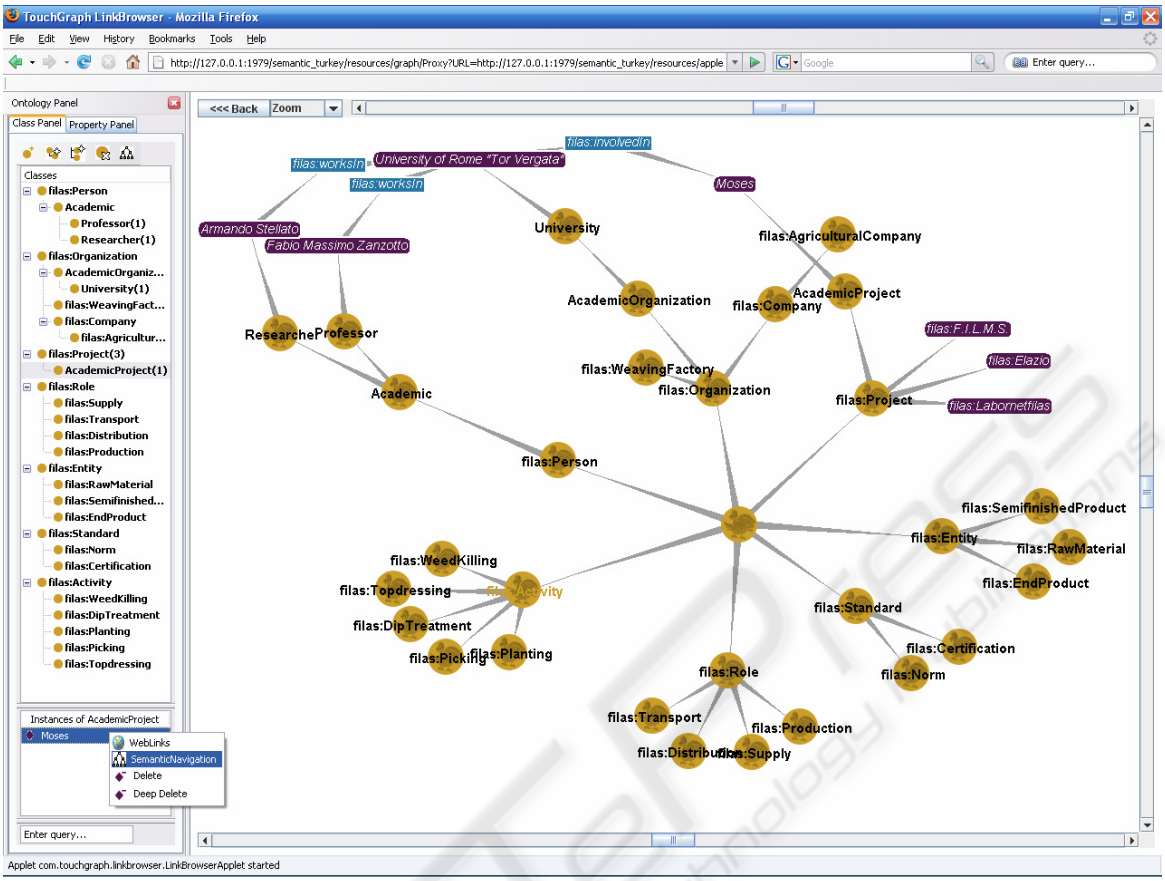

Figure 4: Semantic Navigation of Ontology Data.

6 CONCLUSIONS

In this paper we have described Semantic Turkey, a web browser semantic extension which could support ontology developers, domain experts, annotators or just simple web users in a variety of tasks which require managing information extracted from web sources.

ST's possible areas of intervention, which include semantic annotation, ontology editing and semantic bookmarking, are the result of a unifying solution which considers the definition, collection, organization and management of knowledge not completely separated from the analysis of the information sources where the same knowledge is acquired and then elicited. Trying a direct comparison with other recent similar works is thus a difficult task, as ST distinguishes for the unique characterization of its approach: it is obviously not an ontology editing tool, though we felt that adding basic ontology editing functionalities was a key feature for a semantic bookmarking system which considers customization to user needs as much a strong point as reuse of existing material. The annotation aspect is limited to bookmarking (reporting just the page where a given object has been annotated), though it could be extended in extracting specific information about the exact location – inside the page – of the annotated text, thus making ST suitable for tasks like semantic/linguistic annotation. Again, it would differentiate from its many predecessors, e.g. Melita (Ciravegna, Dingli, Petrelli, & Wilks, 2002), for the completeness of its annotation possibilities – whereas most of them just allow for annotating text with respect to a hierarchy of classes – and in the robust web content rendering, as it is implicit in the adoption of a popular and affirmed web browser.

Probably, a pitfall of so a versatile system would reside in the danger of becoming a jack-of-all-trades and a master of none: ST captures a good trade-off on knowledge representation and user interaction to offer a solid Semantic Web bookmarking platform, whereas its predecessors bear identifying functionalities associated to their main objectives: user-driven learning for Melita, web services for

Magpie, annotation repositories for Piggy Bank etc... all of these feature are difficult to maintain in a single platform even when it potentially allows for their integration. For this reason, we decided to open up the system to possible extensions, by introducing a dedicated plug-in framework which accounts for, and is able to coordinate at best, the hybridization of the several technologies which characterize the architecture of Semantic Turkey.

Next research directions will exploit the new extension framework to add advanced functionalities for automatically extracting knowledge from web sources and interact with the user on how to use it for populating/enriching ontologies; we intend also to explore innovative solutions expressly dedicated to a collaborative working environment and also consider diverse kind of media sources.

ACKNOWLEDGEMENTS

We would like to dedicate this work to the memory of the late Donato Griesi, who implemented the first prototype of Semantic Turkey. We hope you can be proud of where it has gone now...

REFERENCES

Berners-Lee, T., Hendler, J. A., & Lassila, O. (2001). The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American , 279* (5), 34-43.

Broekstra, J., Kampman, A., & van Harmelen, F. (2002). Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. *The Semantic Web - ISWC 2002: First International Semantic Web Conference* (pp. 54-68). Sardinia, Italy: Springer Berlin / Heidelberg.

Ciravegna, F., Dingli, A., Petrelli, D., & Wilks, Y. (2002). User-system cooperation in document annotation based on information extraction. *13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02*. Springer Verlag.

del.icio.us. (n.d.). Retrieved from : http://del.icio.us/

Dzbor, M., Domingue, J., & Motta, E. (2003). Magpie: Towards a Semantic Web Browser. *2nd International Semantic Web Conference (ISWC03)*. Florida, USA.

Dzbor, M., Motta, E., & Domingue, J. B. (2004). Opening Up Magpie via Semantic Services. *3rd Intl. Semantic Web Conference (ISWC04)*. Hiroshima, Japan: November.

Eclipse Platform Technical Overview. (n.d.). Retrieved from Eclipse: http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html

Extensible Binding Language. (n.d.). Retrieved from http://www.mozilla.org/projects/xbl/xbl.html

Firefox home page. (n.d.). Retrieved from http://www.mozilla.com/en-US/firefox/

Garrett, J. (2005, February 18). *Ajax: A New Approach to Web Applications*. Retrieved from http://www.adaptivepath.com/publications/essays/archives/000385.php

Gennari, J., Musen, M., Fergerson, R., Grosso, W., Crubézy, M., Eriksson, H., et al. (2003). The evolution of Protégé-2000: An environment for knowledge-based systems development, . *International Journal of Human-Computer Studies , 58* (1), 89–123.

Google Notebook. (n.d.). Retrieved from http://www.google.com/notebook

Google Page Creator. (n.d.). Retrieved from http://pages.google.com

Griesi, D., Pazienza, M. T., & Stellato, A. (2007). Semantic Turkey - a Semantic Bookmarking tool (System Description). *4th European Semantic Web Conference (ESWC 2007)*. Innsbruck, Austria.

Huynh, D., Mazzocchi, S., & Karger, D. (November, 2005). Piggy Bank: Experience the Semantic Web Inside Your Web Browser. *Fourth International Semantic Web Conference (ISWC05)*, (pp. 413-430). Galway, Ireland.

Jetty Java HTTP Servlet Server. (n.d.). Retrieved from http://jetty.mortbay.org/jetty/

Kiryakov, A., Ognyanov, D., & Manov, D. (2005). OWLIM – a Pragmatic Semantic Repository for OWL. *Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2005), WISE 2005*. New York City, USA.

Leaftag project. (n.d.). Retrieved from http://www.chipx86.com/wiki/Leaftag

McBride, B. (2001). Jena: Implementing the RDF Model and Syntax Specification. *Semantic Web Workshop, WWW2001*.

Quan, D., & Karger, D. (May, 2004). How to Make a Semantic Web Browser. *Thirteenth International World Wide Web Conference (WWW2004)*. New York City, USA.

Simile Java Firefox Extension. (n.d.). Retrieved from http://simile.mit.edu/java-firefox-extension/

TopBraid Composer. (n.d.). Retrieved from http://topbraidcomposer.info/

Touchgraph Development Page. (n.d.). Retrieved from http://touchgraph.sourceforge.net/

Web Ontology Language webpage. (n.d.). Retrieved from W3C: http://www.w3.org/TR/owl-features/

Wheeler, S. (2005, April 6). *Tenor: A Contextual Linkage Framework for KDE*. Retrieved from http://websvn.kde.org/*checkout*/trunk/playground/base/tenor/docs/tenor-architecture.pdf?rev=475778

XML User Interface Language (XUL) Project. (n.d.). Retrieved from http://www.mozilla.org/projects/xul/

XPCOM. (n.d.). Retrieved from http://www.mozilla.org/projects/xpcom/