Gobbleing over the Web with *Semantic Turkey*

Donato Griesi, Maria Teresa Pazienza, Armando Stellato
DISP, Università di Roma "Tor Vergata"
{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 annotate information from visited web sites and organize this information 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 the acquired information and of the pages where it has been collected. This paper describes the architecture of the Semantic Turkey extension for Firefox, discusses its development in the context of the FILAS technophore project, shows its most interesting features and presents our plans for future improvements of the tool.**

***Index Terms*—Semantic Annotation, Semantic Desktop, Web Navigation, Human Computer Interface**

I. Introduction

WHILE new solutions for Semantic Desktops and Innovative Browsing of file systems ([13], [22]) 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 which is blowing from the Semantic Web, remains confined to its old fashioned interaction modalities. Despite a few initiatives to extend their standard functionalities, like Google Notebook [9], web browsers offer now 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 in almost a decade. The web is however not the same of ten years ago: today almost every information can be accessed even (if not solely) from the WWW so that every web user is exposed to a huge amount of information which is difficult to manage and retrieve. New paradigms are thus necessary to support web browsing and to aid in collecting and retrieving the data which is observed during navigation.

In this work we introduce Semantic Turkey, a Semantic Extension for the popular web browser Mozilla Firefox [6], 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), but it is instead intended as a personal desktop solution for organizing and managing the knowledge acquired during web navigation, an advanced replacement for the traditional "Favorites" menu, offering clear separation between knowledge data (the WHAT) and web links (the WHERE), allowing for innovative navigation of the acquired information as well as of the pages where it has been collected.

II. Related Works

As an evidence of the great interest for the matter, several actions have been undertaken in the last years towards the realization of so called Semantic Browsing solutions for the Web. In this section we will briefly recall a few of them.

Haystack [14], developed inside the laboratories of the MIT, 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 is built as an extension for the popular Integrated Development Environment Eclipse [5]; 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 strong demand to 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.

The opposite approach is being followed by Magpie [3], 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 [4] extended the platform more and more towards the vision of the Semantic Web as "an open web of interoperable applications" [1], 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 [10], an extension for the Firefox web browser [6] 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 together, 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.

III. Motivations

Our research work, funded by the FILAS (Finanziaria Laziale di Sviluppo) agency under contract C5748-2005, has been focused on innovative solutions for browsing the web and for organizing the information which is observed during navigation. The interest of the FILAS agency in this project has been motivated by their necessity of developing useful tools for assisting the work of the technophore, a particular figure which has the role of identifying and suggesting innovative technological solutions for industries. The technophore's effort is devoted to improve the business processes of companies involved in a technology innovation phase, as well as to suggest or promote new market directions for their activity. These suggestions walk through the discovery of proper technological solutions and possibly of the identification of the right contacts (academic institutions, research centers or even other companies) which reveal to be of interest for the objectives of the company.

The technophore has generally an high level education and a wide expertise in a given knowledge domain, though originality of mind, creativity and attitude to research are key aspects of his profile, as he is often exposed to strongly underspecified demands, where the problem must be identified even before solutions, and where interesting clues may emerge from unexpected information sources.

In our interviews with one of the technophores working for the FILAS agency, unexpected aspects emerged putting serious doubts on the kind of contribution we could make to aid their work: the interviewed technophore told us that she often got ideas by listening to the radio, or by watching a given spot on the TV (which were often unrelated to her original investigation), or simply by reading web sites and mailing lists. In many cases, good ideas were also accompanied by rapid flashbacks over things seen "somewhere" and "somewhen" in the past, but which were difficult to recall. No intelligent system could be able to behave in such a fashion, suggesting new ideas or interacting with such a wide variety of media (the scenario is very similar to the Semantic Web vision described by Tim Berners-Lee et al. in [1]), but, limiting to the Internet media, we could surely support technophores over the "where/when have I seen it?" aspect which is also fundamental for their work, and which often involves a lot of time required to retrieve the desired information.

Our experience is aligned with the outcomes of the empirical observations of Tauscher and Greenberg [19] (also cited in [3]) which reported the following statistics on the types of actions carried out by the typical web user:

- 58% of pages visited are revisits
- 90% of all user actions are related to navigation
- 30% of navigation actions are through the 'Back' button
- less than 1% of navigation actions use a history mechanism
- ~ 5% of navigation actions use bookmarks lists (also known as "hotlists", "favorites" etc...)

What emerges is a great need for efficient recovery of already visited pages (and, more in general, of already accessed knowledge), which is not matched by an adequate use of the available instruments (back buttons are only useful if the visited page is among the last visited pages in the same session, which can occur quite often, while the instruments to recover past knowledge, like history and bookmarks, are scarcely adopted).

IV. Approach: Requirements and Design Goals

We thus focused on finding innovative 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, except 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, but still maintain 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 to more than one folder (multiple instantiation).
- as a consequence of the two above, 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. This ontology may also be used in different contexts: for example, a personally populated FOAF ontology [7] could be used as a contact list, possibly exporting it to the personal mail browser
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 visited 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. This feature is described in detail in section VI on User Interaction.
4. Clearly separates 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.

In this sense, Semantic Turkey differentiates from similar, previously described, tools, as it offers a lightweight structure, which completely exploits the interface of the hosting web browser (with respect to, for example, the complex HTML interface of Piggy-Bank) and which grants the user maximum (and easy) control over its personal knowledge model (while Magpie adopts ontologies which have been defined elsewhere). Our experience with and feedback received by the FILAS technophores have been completely successful, in that they felt Semantic Turkey as a really ease-to-use add-on to their traditional web browser, with an intuitive interface and immediate response.

V. ARCHITECTURE

The architecture of Semantic Turkey consists in a web application, designed using a three layer approach.

The first layer, the presentation layer, has been developed as an extension for the web browser Firefox. Everything regarding 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 his 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" [11]. 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 every clients communicate with the same Jetty server.

The third layer, the persistence layer, is constituted by the component for managing the ontology, which is written in the OWL language [15]. It has been realized by using Sesame [2] and the OWLIM plugin [12]. 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 [16] dialect of the Web Ontology Language, the OWLIM plugin has been employed to provide OWL Lite reasoning to the Sesame component.

A. The three layers

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

1) 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 [25], XBL [23] and Javascript language.

The User Interface extension physically 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 [8] 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 an other 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 [24] 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 [18] has been used. This component permits to load java classes or jar packages, instantiate objects and to invoke static methods or methods of the object previously instantiated.

[1] "Taccuino" is the italian translation of the term "Notepad". In our lab, we hate so much the silly Italian expression "Taccuino Semantico" (Semantic Notepad) 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…

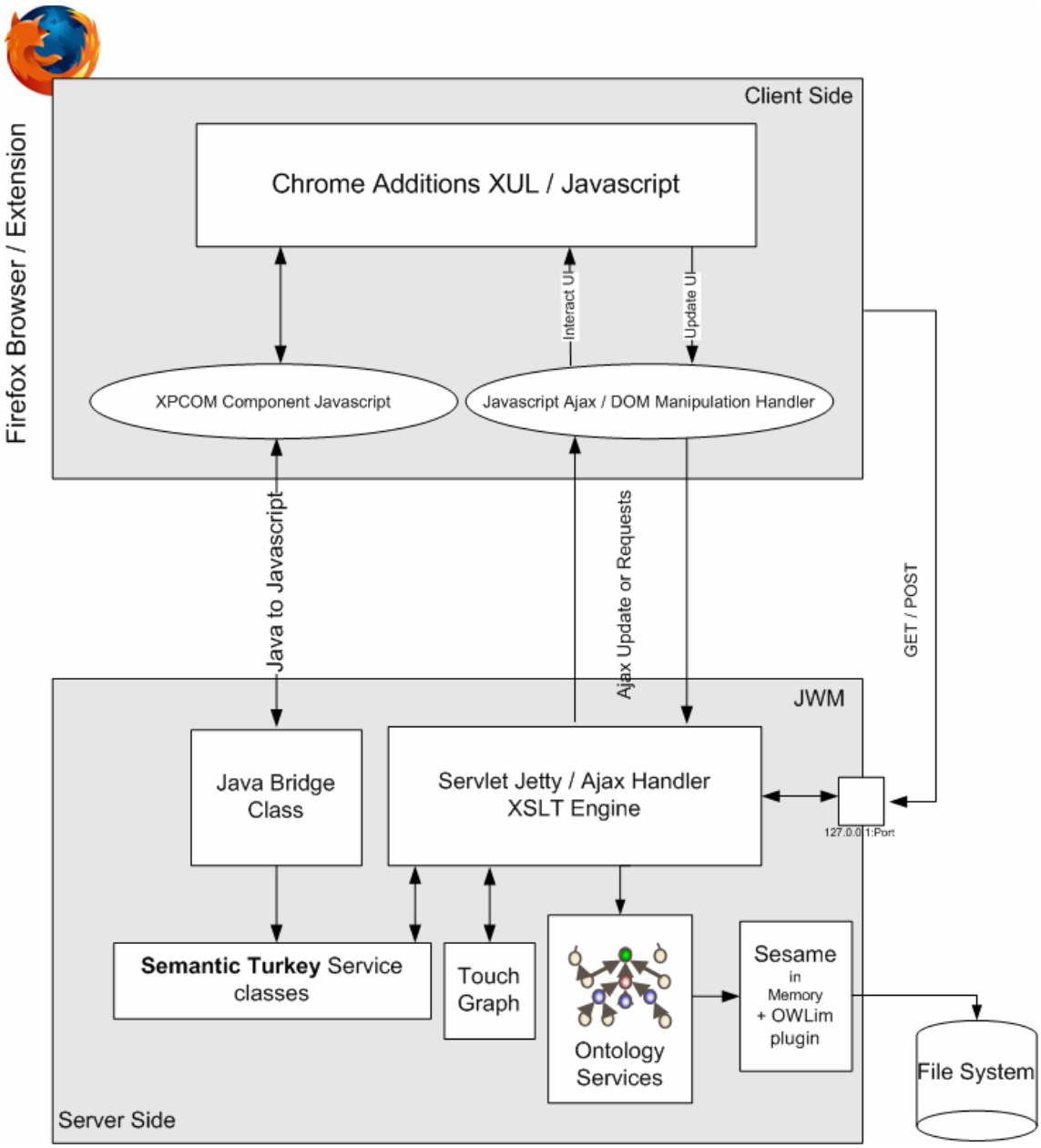

Figure 1: Semantic Turkey Architecture

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.

2) Service / Persistence 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

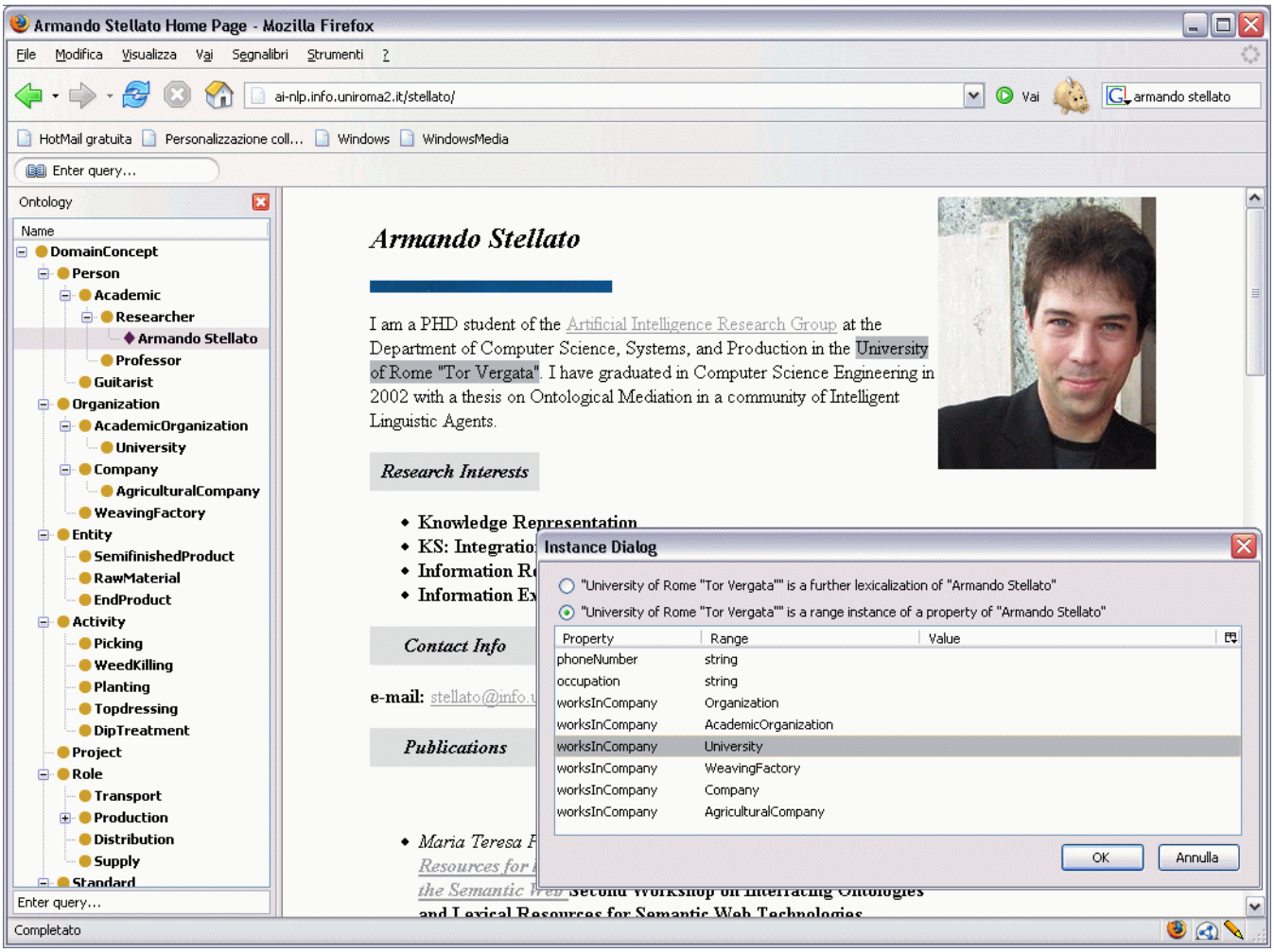

Figure 2: Annotating concepts from a web page and establishing relationships between them

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 [21]. 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.

In order to access TouchGraph from presentation layer, it has been used an apposite java applet, present in the library. Since the applet must be loaded on the server side, a specific Java servlet has been written. The servlet works like a proxy, redirecting the applet loaded, with the correct parameters, to the client side.

3) 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 site, thus offering different possibilities for decentralizing the application.

B. The Knowledge Model

The knowledge model of Semantic Turkey is based on 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
- `URI` which offers 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 `URI`s 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. 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 has to build from scratch its own conceptualization, which will be thus constituted by the sole *Personal Ontology*), or contain general purpose resources, like the already cited FOAF ontology, which could be adopted to maintain a list of contacts, possibly exchanging information with other applications (like a mail browser or a client for instant messaging).

VI. User Interaction

The user may interact with the tree in the ontology panel to modify the ontology, by performing the following operations:

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 lexicalization.
4. Direct editing of the ontology. In particular it is possible to modify the names or delete the created classes/individuals The last operation may be performed for the concepts only if they belong to the domain ontology (as described before). Furthermore a user may create a subclass of a preexisting class, defining new properties besides the ones that are inherited. Lastly, it is possible to directly create new individuals in the ontology, instead of generating them through annotations from the web. All the operations are being carried out through specific panels that may be selected by a dedicated menu which can be activated by right clicking on the nodes of the tree, in a way much similar to traditional ontology editing tools, like Protégé [17] or TopBraid Composer [20]. 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.
5. Add synonyms for the concepts. By the apposite menu, which can be activated by right click on the tree elements that represent concepts, it is possible to define other lexicalizations (in different languages) for the ontology concepts.

As an additional feature, the user may graphically explore the ontology (see

Figure 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. The nodes of the graph will be displayed

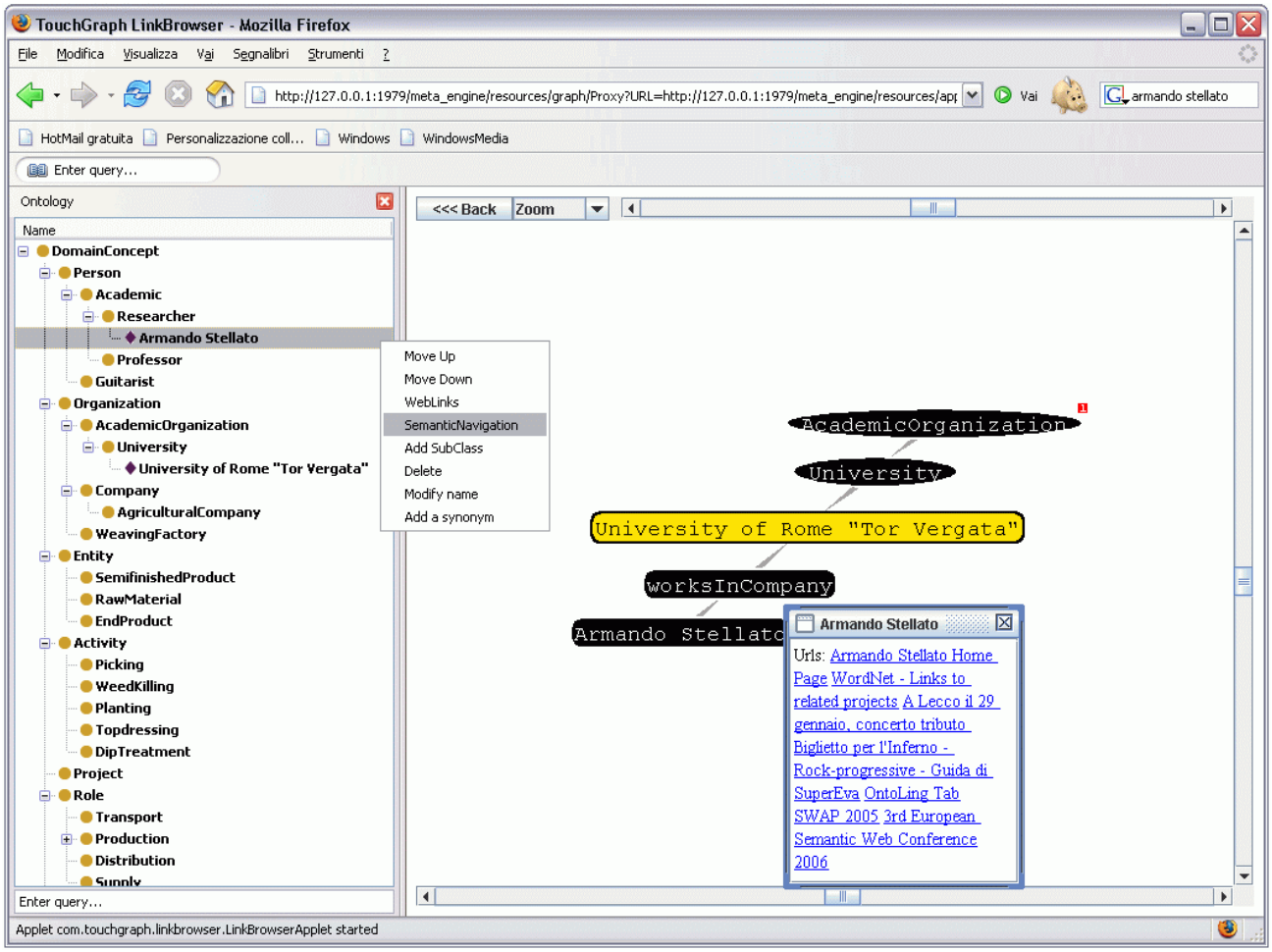

Figure 3: Semantic Navigation: recalling *ontology* and *web* links for the object "Armando Stellato"

in different manners, according to the nature of the ontological entity: nodes represent classes and individuals while properties are represented by the links between the nodes. For clarity of view, instances are displayed in a different manner compared to the classes. Dragging the mouse pointer on a node that represents an instance, it is possible to open a popup window, which contains the URLs of the pages where that instance has been annotated.

Finally, 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.

VII. Conclusions

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

The main objectives have been to allow users to extend and assist their "usual" web navigation with the possibility of annotating observed information, inside an ontological framework, to define a multilayered architecture in which platform independent software components have been merged (for portability purposes) and to realize and exploit a structured knowledge representation model in which different ontologies coexist, still maintaining their independence, while appearing as a personal unique ontological world description to the end user

The first developed system, based on our prototype of Semantic Turkey, received an enthusiastic feedback from its users (FILAS technophores), which mostly appreciated its robustness, easiness of use and non-invasive integration inside their usual way of searching over the web.

We are now in the direction of refining the overall architecture to support further needs of sharing different parts of represented knowledge among different users. This will require managing concurrent accesses to update different levels of the ontology. The next short-term objective, which we plan to reach in the next few months, is to make Semantic Turkey pass from its first release, specifically tailored over FILAS needs, to a publicly available browser extension, which

will be usable by anyone to semantically annotate web pages. This will require a mechanisms for creating and managing new ontologies and/or importing existing ones, and possibly adding further functionalities for ontology editing.

ACKNOWLEDGMENT

We would like to thank the FILAS agency for supporting our research in the development of the first prototype of Semantic Turkey.

Special thanks to the reviewers which gave valuable feedback for the final drawing of this paper and useful suggestions for improving our work.

REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "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): p.34-43. May 2001

[2] 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

[3] M. Dzbor,, J. Domingue and E. Motta. "Magpie: Towards a Semantic Web Browser". In proc. of the *2nd International Semantic Web Conference* (ISWC03), Florida, USA.

[4] M. Dzbor, E. Motta and J. B. Domingue. "Opening Up Magpie via Semantic Services". *In Proc. of the 3rd Intl. Semantic Web Conference* (ISWC04), November 2004, Japan

[5] Eclipse Platform Technical Overview: http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html

[6] Firefox home page: http://www.mozilla.com/en-US/firefox/

[7] Friend Of A Friend Ontology (FOAF): http://xmlns.com/foaf/0.1/

[8] J.J. Garrett. "Ajax: A New Approach to Web Applications". Feb. 18, 2005 http://www.adaptivepath.com/publications/essays/archives/000385.php

[9] Google Notebook: http://www.google.com/notebook

[10] D. Huynh, S. Mazzocchi & D.R. Karger "Piggy Bank: Experience the Semantic Web Inside Your Web Browser". In proc. of the *Fourth International Semantic Web Conference* (ISWC05), pages 413-430, Galway, Ireland, November, 2005

[11] Jetty Java HTTP Servlet Server. http://jetty.mortbay.org/jetty/.

[12] A. 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

[13] Leaftag project: http://www.chipx86.com/wiki/Leaftag

[14] D. Quan and D. Karger. How to Make a Semantic Web Browser. In Proc. Of the Thirteenth International World Wide Web Conference (WWW2004), New York City, USA, May, 2004

[15] Web Ontology Language: http://www.w3.org/TR/owl-features/

[16] OWL Lite Description: http://www.w3.org/TR/2004/REC-owl-features-20040210/#s3

[17] 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

[18] Simile Java Firefox Extension: http://simile.mit.edu/java-firefox-extension/

[19] L. Tauscher, and S.Greenberg. "How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems". *International Journal of Human Computer Studies*, 2001. 47(1): p. 97-138.

[20] TopBraid Composer: http://topbraidcomposer.info/

[21] Touchgraph Development Page: http://touchgraph.sourceforge.net/

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

[23] Extensible Binding Language: http://www.mozilla.org/projects/xbl/xbl.html

[24] XPCOM. http://www.mozilla.org/projects/xpcom/

[25] XML User Interface Language (XUL) Project. http://www.mozilla.org/projects/xul/