Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management

Maria Teresa Pazienza, Noemi Scarpato, Armando Stellato and Andrea Turbati
University of Tor Vergata, Dept. of Computer Science, Systems and Production, Via del Politecnico 1, Rome, 00133, Italy

Abstract. Born four years ago as a Semantic Web extension for the web browser Firefox, Semantic Turkey pushed forward the traditional concept of links&folders-based bookmarking to a new dimension, allowing users to keep track of relevant information from visited web sites and to organize the collected content according to standard or personally defined ontologies. Today, the tool has broken the boundaries of its original intents and can be considered, under every aspect, an extensible platform for knowledge management and acquisition. The semantic bookmarking and annotation facilities of Semantic Turkey are now supporting just a part of a whole methodology where different actors, from domain experts to knowledge engineers, can cooperate in developing, building and populating ontologies while navigating the Web.

Keywords: semantic browsing, semantic annotation, semantic bookmarking, ontology development

1. Introduction

The Semantic Web is becoming ever and ever a concrete reality: with SPARQL reaching W3C recommendation in 2008 [1], languages for data representation and querying have finally completed standardization, closing the gap on usability of Semantic Web technologies in real case scenarios. At the same time, initiatives such as Linked Open Data [2] have boosted the process of data provisioning on the Web. Finally, interests and research in SW technologies have definitely migrated from mere ontology development (which has now met industry standards) to the discovery and provision of applications which can exploit full Semantic Web potential: homogenous access to distributed information providers, connecting conceptual and information resources on the Web of Open Data.

With this scenario in mind, we have worked towards the definition of a Semantic Web browser extension which is two-fold in its offer: first, it is of interest for ontology developers and domain experts (since it aims at facilitating the process of knowledge acquisition and development even for non-technology-savvy users). Second, it provides an extensible infrastructure over which SW applications, needing and relying on rock-solid web browsing functionalities as well as on RDF management capacities, can be developed and deployed.

These objectives have been pursued during a two-year period of finalization and reengineering of Semantic Turkey [3], a Semantic Web extension for the popular Firefox® web browser.

In this paper, we describe the original application for knowledge acquisition and management as it was conceived in the beginning, and introduce and discuss the main innovations which transformed the new incarnation of Semantic Turkey into an open and extensible platform for Semantic Web development.

2. Related Work

Due to the multifaceted nature of our platform, an overview of related research should embrace diverse

1 Following from definition of information/non-information resources given in: http://www.w3.org/2001/tag/doc/HttpRange-14/2007-05-31/HttpRange-14

fields such as ontology editing and visualization, Semantic Web browsing, (social/semantic) bookmarking solutions and semantic annotation. In this section we recall the main works in these areas, and provide insight readings for a thorough view.

2.1. Ontology Editing Tools

Probably the most used and widely known ontology editing platform is Protégé [4,5]. Realized at the Center for Biomedical Informatics Research of the University of Stanford, Protégé has been for years the leading environment for ontology management and has also contributed to the first spread of Semantic Web Technologies in research communities and industries. The Protégé project is currently active, with the Stanford team carrying on development and maintenance of Protégé 3.x, and the University of Manchester developing the next version: Protégé 4.x, which is still in beta development. Another interesting framework is offered by the Neon toolkit [6]: an extensible ontology engineering environment, which has been developed inside the homonymous integrated project co-funded by the European Commission’s Sixth Framework Programme. Today, ontology development has reached industry standard, as witnessed by commercial off-the-shelf products such as Topbraid Composer.

2.2. Information Visualization/Semantic Browsing

Regarding information visualization through Semantic Web technologies, or “semantic browsing”, the first reference which comes to mind is probably the Haystack web client [7]. Developed at the MIT laboratories, it 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 Haystack user navigate over aggregation of data projected from RDF repositories available from different arbitrary locations. The application has been built as an extension for the popular IDE Eclipse; this choice facilitates extension of the tool thanks to Eclipse’s flexible plug-in mechanism, but requires the user to adopt its framework as a platform for browsing the Web and collecting data from it: a negative impact for the average user, who would just prefer to rely on their trusted personal web browser and try out other features which are not too invasive for their usual way of working.

An opposite approach is being followed by Magpie [8], 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 traditional 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. Later work on Magpie [9] extended the platform more and more towards the vision of the Semantic Web as “an open Web of interoperable applications” [10], 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 some of the same authors as those of Haystack, comes Piggy-Bank [11], an extension for the Firefox web browser 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.

Finally, the father of the WWW shared his perspective on what a Data Browser should be with Tabulator [12]. Tabulator focuses on pioneering aspects related to navigation of linked open data, by following dereferenced URIs across their ontology definition sparse in the web of data, and being able to extract data from heterogeneous documents whenever these expose some GRDDL declaration for data extraction through document transformation. As for Semantic Turkey, Tabulator does not offer dedicated and intuitive UIs focused on a given domain (with the sole exception of geo-spatial and temporal coordinates inspection through calendar/timeline and maps views respectively), but rather presents itself as a domain-agnostic data browsing tool. Also, for SPARQL it allows both a generic SPARQL query tool and a basic query-by-example wizard.

---

1. http://topbraidcomposer.info/
2.3. Semantic/Social Annotation/Bookmarking

The most popular "social bookmarking" service, del.icio.us\(^5\), 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.

Regarding semantic annotation, research in this field is mainly addressing three aspects: how to set up an annotation environment, how to improve the process and extend to several media, and how to automate it. The Annotea W3C project [13], suggests RDF based standards for representation of annotations, and provides a general architecture for establishing client-server annotation frameworks. Several clients have been developed for this architecture, such as Amaya [14] and Annozilla [15]. Melita [16] and KIM [17] are probably the most prominent examples of applying decades of research on NLP to automate semantic annotation. AKTive Media [18], the successor of Melita, pushes forward the concept of annotation to cover different media other than text. A thorough overview on Semantic Annotation can be found in [19].

3. History and Motivations

It is our idea that access to and interaction with the greatest source of information available today (the Web), should be ideally integrated in tools for Knowledge Modeling and (in particular) Acquisition. What lacks from the previous approaches is a really integrated solution which is able to combine the best of all worlds from visualization, annotation and ontology development.

Regarding annotation tools, as remarked in [19], though "there are signs that annotation systems are giving users more control of ontologies", still "ontology maintenance [...] is poorly supported, or not supported at all, by the current generation of [semantic annotation] tools".\(^4\)

Seen from the other side (ontology development tools), the RDF family (RDF, RDFS, OWL, SKOS) of models as well as many standard vocabularies such as Dublin Core, offer properties providing meta-knowledge about what is behind the creation of resources in an ontology (such as the RDF rdfs:seeAlso, or Dublin Core dc:relation, dc:source and dc:subject). This is because the specification of a domain should be naturally connected with the process of acquiring knowledge from external sources, and thus of documenting references to them, to better qualify the nature of formalized concepts. However, ontology development tools seem to live in a world of mere algebraic representation, requiring lot of manual work or parallel use of different tools if different actors need to cooperate and make reference to existing information (re)sources.

Semantic Turkey (ST from now on) differs from similar, previously described approaches, by mixing ontology development functionalities with the ease of use of a system for acquiring knowledge from the web. This way, instead of working on different frameworks and producing different kind of data which need to be integrated, domain experts may start to sketch ontologies and keep track of the information they get from the Web, leave comments and references which can then be reused and examined by knowledge engineers in continuous refinement circles.

3.1. The origins

Semantic Turkey had been initially developed as a prototype for a Web Browser extension with advanced bookmarking capabilities [20]: its mission was to go beyond the vague semantics (with respect to information organization) of traditional links&folders bookmarking, and promote a new paradigm, aiming at “a clear separation between (acquired) knowledge data (the WHAT) and their associated information sources on the Web (the WHERE)”.\(^6\)

We thus gave meaning to Semantic Bookmarking as to indicate the process of eliciting information from (web) documents, to acquire new knowledge and represent it through knowledge representation standards, while keeping reference to the original information sources.\(^6\)

The main difference with Semantic Annotation resides in the focus: the term “Semantic Annotation”, though being subject (as underlined in [17]) to slightly different interpretations, which are in some cases too much bound to the specific research settings where the term has been adopted (e.g. in [21,22] and, again, in [17]), has converged in literature towards the definition of “the process of associating portions of text of analyzed documents to predefined sets of semantic descriptors”. So, the text is the focus of Semantic Annotation, whereas the first objective of Se-

---

\(^{4}\) This definition is aligned with the one of Social Semantic Bookmarking provided in [32], though the social aspects are not explored in this work.

\(^{5}\) http://delicious.com/
Semantic Turkey was (and still is) to facilitate users in acquiring and organizing their knowledge, while keeping at the same time references to the source of information which are being consulted. Also, in a ever-changing setting as the WWW, keeping and maintaining precise reference (pointers to position in documents) to textual content would produce information doomed to corrupt, due to modifications of the bookmarked pages: for this reason, pointers to pages as a whole (i.e. bookmarks) were considered a good compromise for this task\textsuperscript{7}. This idea thus translated into a series of functionalities for the user, which, through very easy-to-use drag’n’drop gestures, could select textual information from web pages, create objects in a given domain and annotate their presence on the Web by keeping track of the selected text and of its provenance (web page url, title etc…).

An example is given in Fig. 1 where the user is adding the musician Steve Morse as an object in their ontology, while at the same time decorates it with a bookmark to their homepage and provides further details about him (the instrument he plays, the musical genre etc…) getting them from that same page.

3.2. From Semantic Bookmarking to Knowledge Management and Acquisition

Standing on the shoulders of mature results from research and development on Semantic Web technologies, such as Sesame \textsuperscript{23} and OWLim \textsuperscript{24} as well as on a robust platform such as the Firefox web browser, Semantic Turkey differs from other existing approaches which are more specifically tailored towards knowledge management and editing \textsuperscript{4}, semantic mashup and browsing \textsuperscript{9,11} and pure semantic annotation \textsuperscript{16,13} by introducing a new dimension which is unique to the process of building new knowledge while exploring the Web to acquire it.

By focusing on this aspect, we went beyond the original concept of semantic bookmarking and tried to amplify the potential of a complete knowledge management and acquisition system: we thus aimed at reducing the impedance mismatch between domain experts and knowledge investigators on the one side, and knowledge engineers on the other, providing them with a unifying platform for acquiring, building up, reorganizing and refining knowledge.

\textsuperscript{7}Though traditional Semantic Annotation is still made possible thanks to extensions though for this, such as: http://semanticturkey.uniroma2.it/extensions/rangeannotator/ which allows for precise reference to elements in the pages, through use of xpointers (http://www.w3.org/TR/xptr/)
4. User Interaction

ST is now an open editor for data modeled upon languages of the RDF family, allowing the exploitation of almost all of those language potentialities (currently, it does not allow editing of complex OWL descriptions, though it loads them and reasoners exploit their content; also, SKOS and SKOS-XL support is being provided in the next release).

Users can browse and edit (Fig. 2) the ontology by using ST like any other ontology editing (OE) tool. Unlike other ontology tools embedded in the web browser (such as Piggy-Bank [11]), which rely on web-based rendering of user interfaces, Semantic Turkey offers complete interaction with the ontology via the XUL interface completely integrated in the browser. The user is thus not diverted from web 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 focus over both the observed web page and the ontology.

To allow maximum flexibility, every element in the ontology can now be added through the advanced bookmarking/annotation functionalities or directly through the ontology editor (in both cases, further annotations can be added later to the created objects). Fig. 3 shows the different annotation/knowledge acquisition possibilities offered by the functionalities based on integration with the hosting web browser: the process is multifaceted in its possible outcomes, though very easy to carry out, since it depends on implicit, contextual factors, such as where in the ontology the user drops the element dragged from the page, as well as on simple interaction steps with the user (like choosing if adding new annotations for a previous element or adding a value for a property, followed by further possibilities depending on the kind of property).
User

Drag'n'drop text over resource

Choose which action to undertake

[resource is a Class]
[resource is an Individual]

show the Annotation Dialog

[resource is an Individual]

add a new Annotation for the selected individual

[resource is a Class]

add an Individual named after the selected text

[resource is an Individual]

add a new Annotation for the selected individual

Choose which property to fill

[property is DatatypeProperty]
[property is ObjectProperty]

[object is a resource]
[object is a literal]

ask for datatype

ask for language

[object is a literal]

[object is a resource]

add property Value

[object is a resource]

relate object with subject through selected property

[object is a resource]

annotate object

add new Individual named after selected text

[object is a literal]

[object is a resource]

bind & create or bind an existing object as value for the property
4.1. “Macroing” series of ontology editing operations

The drag’n’drop features for capturing data have been conceived to speed up the process of knowledge acquisition, allowing for complex series of ontology editing operations to be implicitly executed, depending on the specific action performed by the user. In the previous example, if we drag “Deep Purple” over the musician Steve Morse, and then select the playInBand object property, the following update operations on the underlying ontology are performed:

- creation of an instance with local name “DeepPurple” (taken after the selected text), if it is a new resource not yet present in the ontology
- assertion of a relation (identified by the chosen object property) between the selected object (the “Deep Purple” band) and the instance where the text has been dropped (Steve Morse)
- the assertion of the rdf:type relation between the object of the above relation and the class selected from the range of the object property (e.g. DeepPurple as a MusicBand, or even a RockBand subclass, because the user is prompted with class-trees rooted on classes featured in the ranges of the selected object property)
- creation of the bookmarked page (as an ontology individual) and associated data (title, url etc…)
- creation of a semantic annotation linking the created individual to the bookmarked web page

The cost for the above operations is just a drag&drop and a couple of intuitive choices among those proposed through the acquisition process.

4.2. Real “Open World Assumption”-Aware Approach to User Interface

Whereas constraint-checking approaches to UI exploit constraints defined in the underlying data model as a strict base for populating form-filling panels, not allowing any operation which could invalidate the constraints, a tool whose knowledge model is based on the open world assumption and on inferential capabilities, uses constraints to just suggest values to the user, or to optionally remove palesly incompatible values (that is, values which, by inference, would produce an inconsistency in the model) from choice lists, and give in any case complete freedom to users. For example, when, by following a drag&drop action, a value needs to be added to a resource, the range of suggested properties is first selected on those whose rdfs:domain is computed by inference to include at least one of (and be compatible with all of) the types of the subject resource. Much the same way, when a property has been selected for adding a value, resources can be selected from a class tree-view rooted on the rdfs:range of the property (with analogous considerations to the previous case). These suggestions can be bypassed (e.g. asking to display all the properties, or to explore the whole class tree instead of the suggested part), in that the user can go out of available boundaries, and introduce new “implicit” knowledge by adding ground facts which alter, by inference, the knowledge of the domain. This kind of interaction surpasses the limitations of (at least some of) current ontology editing tools, which are still not fully acquainted with the inferential aspects of the OWL language. For example, Protégé OWL 3.x [5], though offering advanced features and wizards for assisting users in adding entries to an ontology, is still bound to its original constraints-based model [4] which binds subject and object values of triples to the defined rdfs:domain and rdfs:range of the predicate. Protégé 4¹, being completely targeted for the OWL standard, abandons this constrained approach, though property-value editing is still in its infancy and, at present time, its authors preferred to not address at all classification-related issues and to show instead the (whole) list of available instances when the user asks for potential values to be added to object properties. Semantic Turkey thus makes ontology editing faster by proposing suggestions to the users, which rely on declared restrictions and on asserted (or inferred) types and values, but they can always break these boundaries and have access to the whole data, eventually letting further inference follow its actions.

4.3. Other features

Semantic Navigation. As an additional feature, the user may graphically explore the ontology, 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 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 popup a window, which contains the URLs of the pages where that instance has been annotated.

Extensibility. The drag’n’drop macros for ontology editing/annotation are just a nod to what can be

¹ http://protege.stanford.edu/download/registered.html#p4
done in a browser-embedded ontology editor: ST’s flexible extension mechanism allows for dedicated extensions to be realized, exploiting different interaction possibilities with the user and making it possible to deliver completely new applications based on the Knowledge Management infrastructure of ST (see section 6.2).

5. Knowledge Model

ST offers complete functionalities for importing ontological data coming from different RDF/OWL sources. Its internal Knowledge Model (KM) foresees a separation between the explicit (domain) knowledge managed by the user and the one which guides the system’s behavior. This last layer, defined as the Application Ontologies Layer, is kept invisible to the user, and is only exploited by the application to drive it knowledge based functionalities. Semantic Turkey currently includes one vocabulary in this layer, the Annotation Ontology: a set of concepts (and related properties) used to keep track of annotations from the Web. These include:

- ann:WebPage (rdfs:subClassOf ann:Document) concept for storing information about the annotated pages (such as ann:URL and ann: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
- ann:SemanticAnnotation containing the annotations performed by the user, and described by the bookmarked ann:WebPage, resource etc… these can be both ann:TextualAnnotation(s) (for text annotated from the web page) as well as ann:ImageAnnotation(s) (for future extensions with image media)

The textual annotations also keep track of the different possible lexical realizations (ann:text property) that a same object may have exposed into different web pages: they are not addressed as alternative labels for the resource, but are uniquely associated to that specific annotation, since they may also refer misspelled entries or other kind of references which the user may not want to associate to the targeted resource. The annotated text is used to retrieve the textual occurrence of the resource when the user gets back to the same page (a highlighter icon in the bottom will show the presence of previous annotations on a page and will allow the user to view them highlighted).

The Application Ontologies layer is not limited to include the sole Annotation Ontology, and can be dynamically extended to host new application ontologies according to the needs of Semantic Turkey extensions (see extension mechanism in the following section).

6. Architecture

The architecture (Fig. 4) 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 (also embedded in the extension) for administering the business logic and data access.

6.1. Architectural Layers

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

Presentation Layer. Everything relating user interaction is directly managed by the Firefox extension. An XPCOM component has been developed to link the presentation layer to the service layer, which is implemented in java. This direct link is actually performed just to wake up an embedded Java web server, which accepting further messages from the client. This layer is actually not limited to presentation responsibilities, since much of the web-related processing (e.g. accessing pages, browsing their content, extracting portions of text etc…) can be delegated to the web scripting engine of the web browser.

Service Layer. This layer offers an extensible set of OSGi services which may be invoked through XMLHttpRequest(s), following the Ajax [25] paradigm. Besides supporting the communication with the client, the middle layer provides the functionalities for definition, management and treatment of the data and the business logic of applications built on top of Semantic Turkey framework.

Data layer. It is mainly constituted by the component for managing the ontology. This has recently been rewritten as a series of dedicated middle-layer API for accessing ontological data: these offer both RDF triple-level access methods as well as more object-oriented tailored facilities, which have been appreciated in RDF libraries like Jena [26] (more details in the following section).

http://www.mozilla.org/projects/xpcom/
6.2. The extension mechanism

Semantic Turkey features an extension mechanism based on a proper combination of the Mozilla extension framework (which is used to extend the user interface, drive user interaction, add/modify application functionalities and provide javascript API for the whole set of Mozilla desktop utilities) and the OSGi java extension framework\(^\text{10}\) which provides extension capabilities for the service and data layers of the architecture.

OSGi compliance is obtained through the OSGi implementation developed by the Apache Software Foundation, called Felix\(^\text{11}\). Three main extension points have been introduced: a Service Extension, an OntologyManager Extension and a Data Extension, to provide respectively: new functionalities, support

\(^{10}\) http://www.osgi.org/  
\(^{11}\) http://felix.apache.org/
for other data management technologies and for introducing new application ontologies.

Both the Java business logic layer and the JavaScript layer for interaction with the browser provide API\textsuperscript{12} for accessing/manipulating RDF data as well as for interacting with the core system and the browser hosting this application. Target users of this integrated development framework range from developers of web browser extensions willing to add RDF-based functionalities without need to rewrite the whole infrastructure from scratch, to developers of knowledge acquisition tools, which get for free all the basic ontology management features and the possibility to interact with web content through a real robust web browser (and its associated development environment).

7. Conclusions

In this paper we presented (in its latest release) Semantic Turkey: a semantic extension for the web browser providing functionalities for Knowledge Management and Acquisition. We discussed the main innovations introduced with respect to its original prototype and showed the potentialities of this framework by presenting its extension capabilities.

7.1. Collected Experiences and Lessons learned

The experiences that we have recently undergone in the adoption of Semantic Turkey across different application scenarios have been a test bed for evaluating the real possibilities of such an extensible framework. The result is that, though far from perfect, the extension mechanism (combining both open service gateways and browser interaction) is flexible enough to allow for very different uses of the platform. For example, the UIMAST [27] extension, developed in the context of the UIMA Innovation Award 2007\textsuperscript{13}, brings into ST the document analysis capabilities of the UIMA platform (a framework originally developed by IBM on top of the OASIS standard for Unstructured Information Management Architecture\textsuperscript{14}, and lately devolved to the Apache Software Foundatio

\textsuperscript{12} Interaction with the business logic of the system is provided by direct Semantic Turkey API, access to RDF is provided by OWL ART API (http://art.uniroma2.it/owlart/); the hosting browser is accessible through Mozilla Javascript language while STscript API (http://semantic/turkey.uniroma2.it/documentation/jsdoc) allow for browser side access to the service layer functionalities.

\textsuperscript{13} https://www-304.ibm.com/cjct0002c/university/scholars/innovation

\textsuperscript{14} http://docs.oasis-open.org/uima/v1.0/uima-v1.0.html
tion), thus introducing functionalities for concept extraction from web pages and ontology learning.

UIMAST then allows users to literally interact through UI elements with the content of analyzed web pages.

ST extensions also range to totally new applications hosted on the web browser, which just rely on the underlying infrastructure for knowledge management. A success story in this sense is offered by STIA [28], an annotation environment for comparing web documents in the jurisprudence domain and for matching concepts from different laws, which completely hides underlying ontological details.

Developed inside our collaboration with the European Space Agency (ESA), in the context of their participation to the EU funded project Diligent (IST-004260), EOAnnotator [29] is another extension showing how to exploit the browser to ease acquisition of contents from the web (in this case, through extraction and projection of RDFa from the browsed pages, over working ontology).

The above experiences also made us better understand the added value given by the underlying ontology development framework, which comprehends high level data access and manipulation primitives going far beyond basilar RDF management, as it is commonly provided by triple store libraries/services such as Jena or Sesame.

Finally, one more lesson gained from these experiences is that the learning curve for extension developers is a bit steep due to the wide range of employed technologies and to their different levels of integration: this will require even stronger attention on solutions and support for an Aided Extension Development, which goes beyond extensive documentation and probably embraces the realization of dedicated tools and development frameworks. Supporting the growth of a dedicated open software development community has been in fact one of the key aspects in several successful experiences (e.g. Protégé).

7.2. User Feedback

We opened up tool evaluation to the user community through a questionnaire available at:

http://semanticturkey.uniroma2.it/questionnaire/

Contributions are still few to trace a statistically significant analysis (also because the questionnaire provides different questions depending on the user profile, which may vary from “Domain Expert” to “Semantic Web Application Developer”), though we collected most prominent results (homogeneous
across different users) which revealed Semantic Turkey’s strong points and flaws:

- **User interface** is considered friendly. All voted from “satisfactory” to “yes, sure!” upon the explicit question about friendliness of UI, and this has been remarked with comments – especially from domain experts – comparing it to other available tools.

- **Easiness of installation** is another strong point, though someone reported problems – in their comments to the answer – whenever Firefox java plugin is not properly setup: this is not something directly related to what can be done at application level, though we acknowledge that the underlying technology (java plugin, firefox xcom etc.) is not completely 100% guaranteed to work immediately on all machines, and may require some setup.

- **Extension Development**: the very few users who completed this part (rating themselves as Semantic Web Application Developers) rated the Extension Development learning curve as steep, thus confirming our considerations in previous section, though half of them really appreciated the possibilities of mixing different technologies and saw the learning phase as the necessary cost to pay for getting to them.

- **Semantic Bookmarking and Annotation**: the bookmarking feature of ST is seen as an added value with respect to existing tools: again, domain experts with no high computer skills provided most of the positive feedback. However, some of the users complained about lack of other bookmarking possibilities, such as bookmarking concepts other than instances: this feature has been requested to us especially by researchers working on Semantic Annotation who need to provide training datasets of pages tagged with respect to both entities and concepts.

### 7.3. Future Research Work

The next step which further development on this platform should take is to address the potentialities which have arisen by opening it up to full ontology development. In its new incarnation as a platform for development and acquisition of semantic web data, we cannot ignore important modeling axioms provided by the OWL language (restrictions, set operators etc...which are currently not available for editing, though being properly processed by the data/inference layer), and include explicit support for different modeling frameworks, such as SKOS.

On the other hand, while the above aspects are important in ontology development systems, there are other directions that, being by far more concerned with the contradistinguishing features of ST, could be properly investigated to push forward state-of-art research on this kind of framework. The presented architecture, thanks both to its modularity and web interaction features, could be lifted to a collaborative framework allowing knowledge engineers and domain experts to exchange information, opinions and data over the same working environment. Identification of proper user roles in the acquisition and development process could then give raise to a whole range of dedicated services being activated/hidden depending on the profile of the logged user. We are currently pursuing this objective [30], by introducing concepts close to (and inherited from) traditional solutions in Software Engineering: Bug-Tracking and Discussion, Issues Management, Versioning etc…

Another research line which naturally follows from the intrinsic connection between ontology and documents in ST, is related to the elicitation of knowledge from (web) resources: we are studying processes for automatically extracting knowledge from documents proactively collaborating with the user on how to use the collected information for populating/enriching managed ontologies (as for already cited UIMAST).

Finally, we found many overlapping points with current research on Semantic Desktops, especially in those modeling aspects which have been widely discussed and synthesized in the PIMO Ontology for Personal Information Models [31]. Interaction with this research field could be two-ways: by exploring assessed results in Semantic Desktop research, to better handle knowledge organization inside the current platform (e.g. by reusing PIMO ontologies in place of current annotation ontology), as well as by transforming ST into a browser end-point for Semantic Desktop interaction.

Semantic Turkey site (which reached now roughly 2700 downloads) can be reached at: http://semanticturkey.uniroma2.it/

**Footnotes**

16 Though 1) users of machines based on Mac OS experienced a few bugs due to idiosyncrasies in the Mozilla XUL language for UI description related to its Mac porting, and 2) the UI in general still has some flaws leaving room for improvement

17 These rare issues mostly affect Linux machines with non-SUN JVMs or not properly configured JVMs, thus happening to people with averagely more-than-average computer skills who know how to setup their system

18 Though being properly processed
References


[16] Fabio Ciravegna, Alexei Dingli, Daniela Petrelli, and Yorick Wilks, "User-system cooperation in document annotation based on information extraction..." in 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW'02, 2002.


[30] Daniele Bagni, Marco Cappella, Armando Stellato, and Maria Teresa Pazienza, "CONGAS: a COLlaborative ontology development framework based on Named GeAphs;" in International Conference of the Italian Association for Artificial Intelligence (AI*IA09), Reggio Emilia, Italy, 2009, December, 9-11.
