A Desktop-Integrated Semantic Platform for Personal Information Management

Maria Teresa Pazienza, Noemi Scarpato, Armando Stellato, Andrea Turbati

ART Research Group, Dept. of Computer Science,
Systems and Production (DISP) University of Rome, Tor Vergata
Via del Politecnico 1, 00133 Rome, Italy
{pazienza, scarpato, stellato, turbati}@info.uniroma2.it

Abstract

The Semantic Web dream of a real world-wide graph of interconnected resources is – slowly but steadily – becoming a concrete reality. Still, the whole range of models and technologies which will change forever the way we interact with the web, seems to be missing from every-day technologies available on our personal computers. Ontologies, annotation facilities and semantic querying could (and should) bring new life to Personal Information Management, supporting users in contrasting the ever-growing information overload they are facing in these years, overwhelmed by plethora of communication channels and media.
In this paper we present our attempt in bringing the Semantic Web Knowledge Management paradigm at the availability of diverse personal desktop tools (Web Browser, Mail clients, Agenda etc…), by evolving Web Browser Semantic extension Semantic Turkey to an extensible framework providing RDF data access at different levels: java access through OSGi extensions, HTTP access or dedicated JavaScript API for the whole range of tools from the open source suite of Mozilla applications

1. Introduction

The Semantic Web is becoming ever and ever a concrete reality: with SPARQL reaching W3C recommendation early this year (Prud'hommeaux, 2008), languages for data representation and querying have finally reached standardization, while interests and research in Semantic Web technologies have definitely migrated from mere ontology development (which has now met industry standards) aspects to the discovery and devise of applications which can both show and exploit Semantic Web full potential.
Despite this encouraging trend of Semantic Web models and technologies, these seem to be missing from applications which we use every day on our personal desktop computers. Hopefully, they could surely contribute to improve the quality of personally managed data by supporting users with powerful vocabularies (ontologies) which can be extended (by adapting them to personal needs) and shared through different applications and with other people.
Recently, several efforts have been spent towards definition of applications and solutions for implementing the so called Semantic Desktop (Iturrioz, Díaz, Fernández Anzuola, & Azpeitia, 2003; Sauermann, 2005; Groza, et al., 2007).
All the Semantic Desktop approaches cited above usually aim at centralizing an RDF Semantic Repository as a local information management resource, which can be accessed by diverse applications on the desktop sharing common data but providing different services over them.
In this work, we present our proposal for a Semantic Integrated Environment for the Mozilla suite (though it can be exploited also by other applications) of desktop utilities (Firefox, Sunbird, Thunderbird etc…). This project originated from our ontology tool Semantic Turkey (Griesi, Pazienza, & Stellato, 2007), which was originally thought as a Semantic extension for the Firefox Web Browser and lately evolved into a multi-layered extensible framework for Knowledge Management and Acquisition.
The current framework which still backbones Semantic Turkey, is two-fold in its offer: by first, being of interest for ontology developers and domain experts, since it aims at facilitating the process of knowledge acquisition and development, and, on the other side, providing 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. In this paper we present the different service layers which are exposed by current version of Semantic Turkey, and how they can be accessed by Mozilla-based and other external applications to give life to a new multimodal Semantic Desktop.

2. Other works

Beside the main research stream which is conducted in this field, other researchers are focusing on finding new powerful and versatile ways of interaction with the user, which can exploit the advanced possibilities given by the Semantic Desktop. as in (Iturrioz, Díaz, & Fernández Anzuola, 2008) where the seMouse (Semantic Mouse) offers a Mouse extension (cabled at Operating System level) allowing for easy classification, authoring, retrieval etc… of files on the desktop and of their textual content. Since it is acting at OS level, this mouse extension is not limited to any specific working environment/application: no matter whether the user is working with Word, Power-Point, Netscape, etc, the semantic button is available for annotation/authoring and the user does not have to move to a new dedicated editor when annotating.
Though intuitions such as the one of seMouse centered the limitations of past approaches with respect to their concrete usability in real life, most recent trends tend to favor the centralization of core knowledge services, thus giving the possibility to all desktop applications to feature even very specific and advanced functionalities while interacting together with (and possibly be coordinated by) the central semantic repository.
The most recent (and sensible) effort following this trend has been represented by the FP6 EU funded project NEPOMUK (Groza, et al., 2007) where a massive range

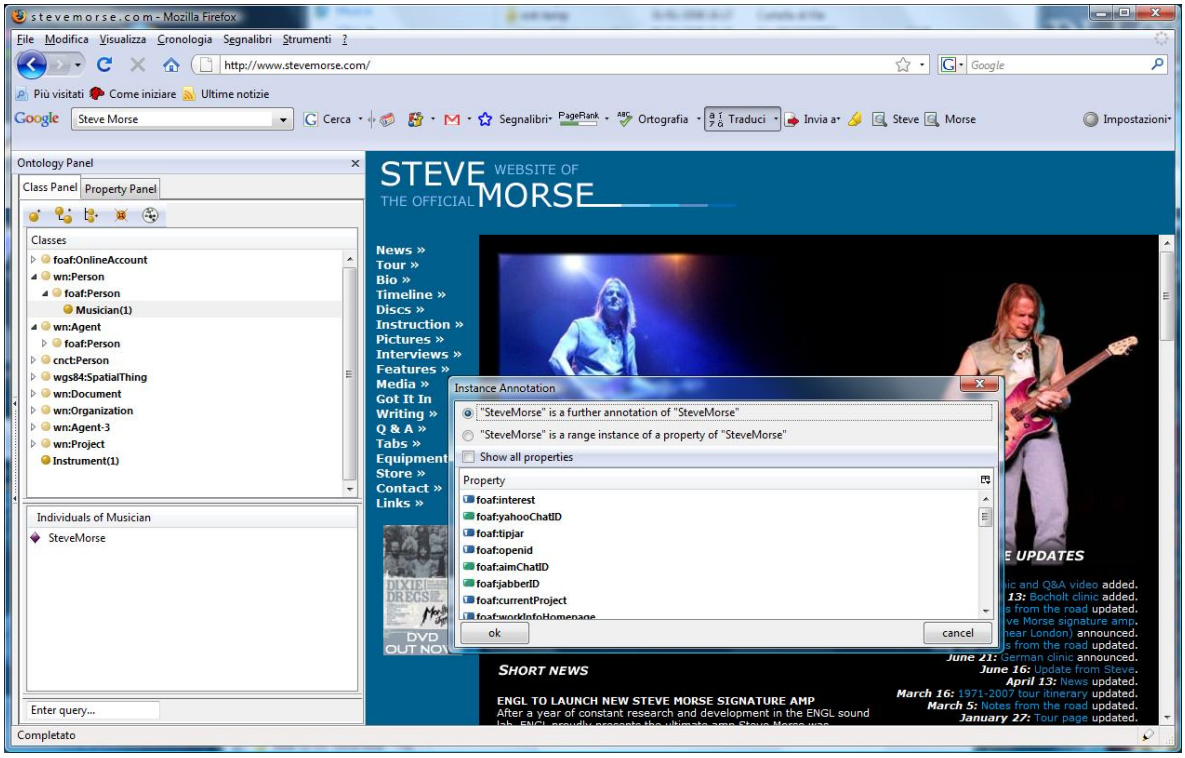

Fig. 1 Semantic Bookmarking with Semantic Turkey

of technologies comprehended several extensions for existing applications centered around an RDF Data server activated by the Operating System.

Eventually, a Semantic Desktop could probably rely on a combination of both approaches, which are not in contrast with each other.

Another important aspect of research is the definition of the metamodels which should contradistinguish such powerful organization systems: in PIMO (Sauermann, van Elst, & Dengel, 2007) a multilayered ontology model is presented. The PIMO (Personal Information Models) Ontology offer a first distinction between three conceptual categories: Native Resources (files, e-mails, contacts etc…), Native Structures (representing organizational schemas for the above, such as folders, bookmark folders, tags etc…) and lastly the Mental Model provides a cognitive representation of the knowledge a user is intended to manage, which is indipendent of (though may be linked to) the above.

PIMO is the structured according to five layers which account for different levels of specification (such as for the first three levels: *PIMO-Basic*, *PIMO-Upper* and *PIMO-Mid*) as well as for the specific exigencies of the user (*PIMO-User*) and of the working/social environment where he acts (*Domain ontologies*).

The necessity for addressing different facets of knowledge in organization systems is also present (though in a less general perspective, which is specifically aimed at enterprise organizations) in (Apostolou, Mentzas, & Abecker, 2008), where a single Knowledge Object (KO) may be characterized according to descriptors which are provided by different facets of the whole ontology. These facets are: Business, Domain, Community, Context and Content, describing *where* a KO may be used, according to *which conditions* its use is suggested, the *range of users* which may be interested in it, and the like.

3. From Semantic Bookmarking to Knowledge Management and Acquisition

Semantic Turkey was born inside a national project – funded by the FILAS agency (Finanziaria Laziale di Sviluppo) under contract C5748-2005 – focused on innovative solutions for browsing the web and for collecting and organizing the information observed during navigation.

The prototype for the project immediately took the form of a Web Browser extension allowing users to annotate information from visited web sites and organize it according to a personally defined domain model: Semantic Turkey paradigmatic innovation was in fact to "obtain a clear separation between (acquired) knowledge data (the WHAT) and web links (the WHERE)" pointing to it. That is, to be able, through very easy-to-use drag'n'drop gestures, to *select* textual information from web pages, *create* objects in a given domain and *annotate* their presence in the web by keeping track of the selected text and of its provenience (web page *url*, *title* etc…). We coined the expression "semantic bookmarking" for this kind of activity.

Due to its proverbial extendibility, the Firefox platform (http://www.mozilla.com/en-US/firefox/) had been chosen as the hosting browser for our application, while Semantic Web standards and technologies were the natural candidate for representing its knowledge model.

Semantc Turkey (Fig 1) was thus born. Standing on top of mature results from research on Semantic Web technologies, like Sesame (Broekstra, Kampman, & van Harmelen, 2002) and OWLim (Kiryakov, Ognyanov, &

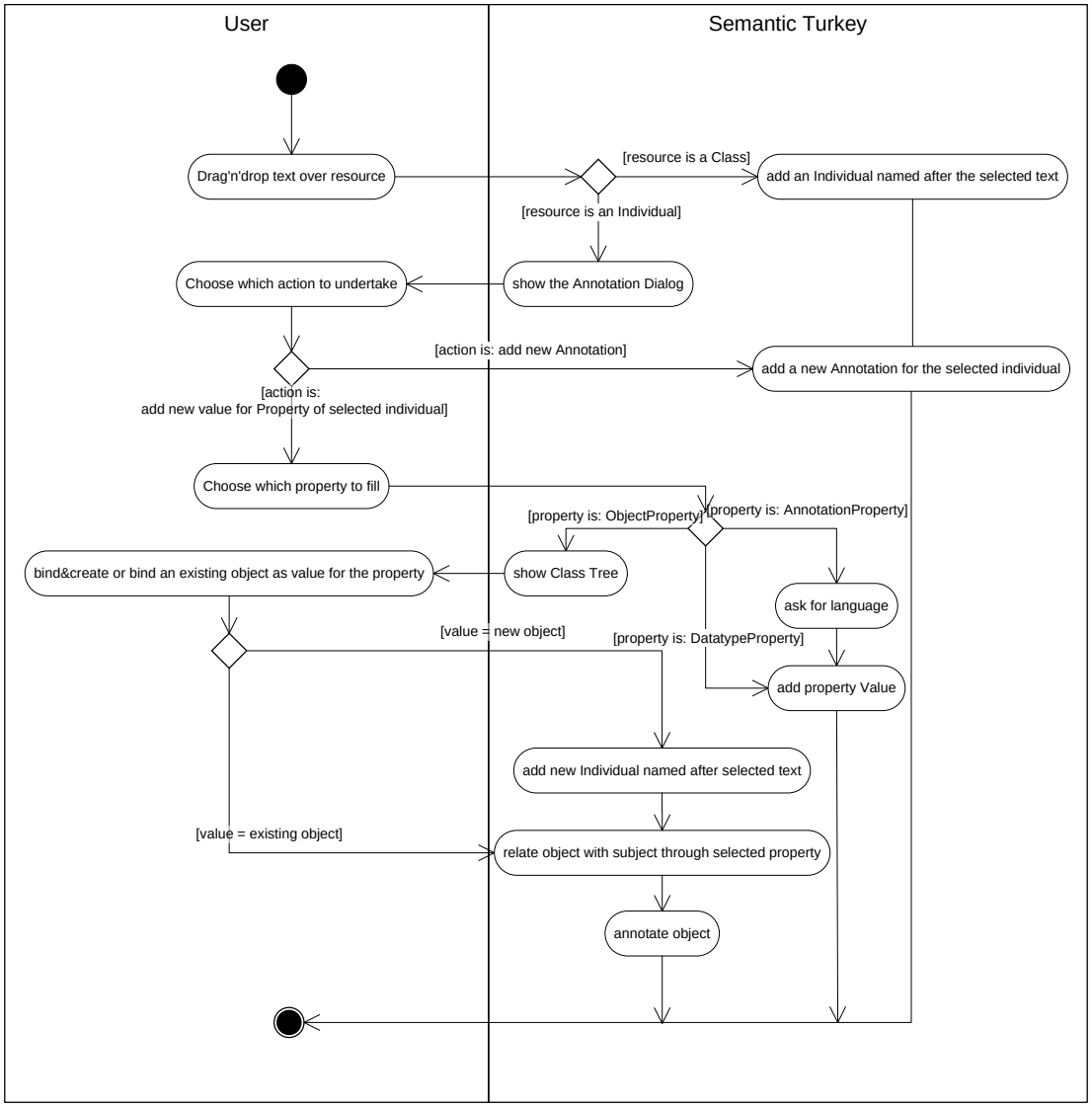

Fig. 2 Activity diagram for semantic bookmarking/annotation

Manov, 2005) as well as on a robust platform such as the Firefox web browser, ST (Semantic Turkey) differentiated from other existing approaches which are more specifically tailored respectively towards knowledge management and editing (Gennari, et al., 2003), semantic mashup and browsing (Dzbor, Domingue, & Motta, Magpie: Towards a Semantic Web Browser, 2003; Huynh, Mazzocchi, & Karger, 2005) and pure semantic annotation (Ciravegna, Dingli, Petrelli, & Wilks, 2002; Kahan & Koivunen, 2001), 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 new 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.

Fig. 2 shows the different annotation/knowledge acquisition possibilities offered by the functionalities based on interaction with the hosting web browser. In the new version of ST, support for all kind of properties has been introduced and reflected in the bookmarking facility: when a portion of text is selected from the page and dragged over an individual, the user may choose (as in the old version) to add a new annotation for the same individual or to use the annotation to fill one property slot for it. In the second case, the user can now choose from a list of properties (see small window in) the one which will be filled: this list includes those properties having their rdfs:domain including one of the types of the selected instance, but may be extended to cover all properties (letting the inference engine do the rest). If the property selected for enrichment is an object property, the user is prompted with a class tree (rooted on the rdfs:range of the selected property) and is given the possibility of creating a new individual named after the text selected for the annotation or to choose an existing one: in both cases the selected individual is bound –

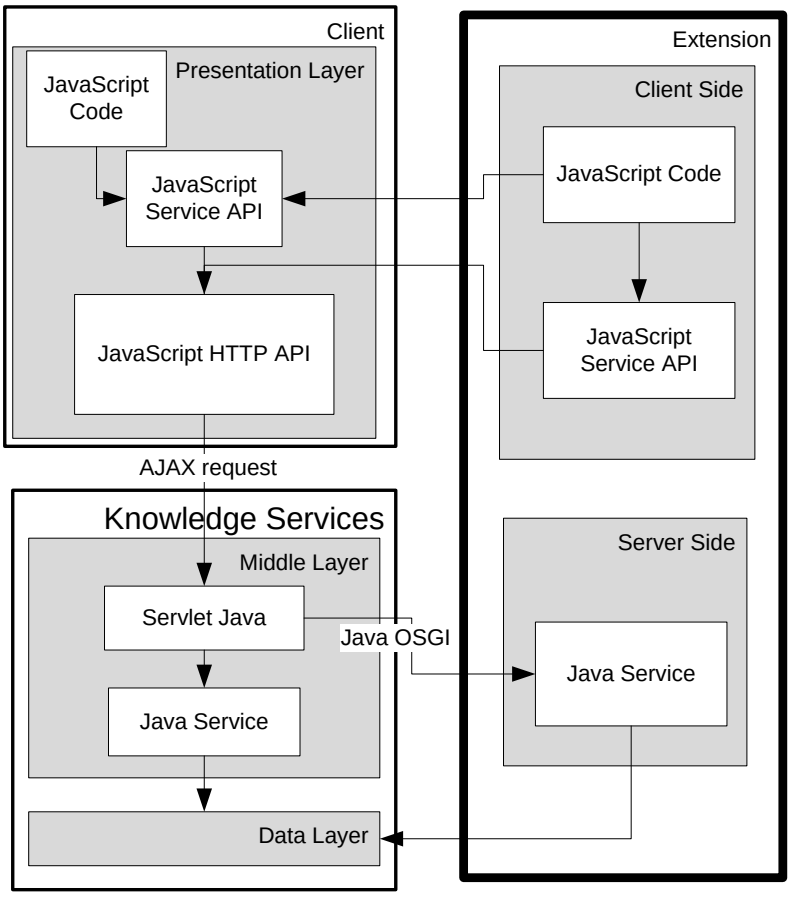

Fig. 3 Architecture of the different Access layers for Mozilla Semantic Desktop

through the chosen property – to the one where he originally dropped the text; a bookmark is also added for it, pointing to the page where the object has been observed. Even in this case, the user may choose to visualize the entire class tree and not the one dominated by the range of the property: the inference engine will automatically assign the pointed instance to that range.

The above interaction modalities for knowledge acquisition/annotation/bookmarking can be used in the main Ontology Editor tool, as well as be exported as *pluggable functional objects*, into other client applications willing to adopt them in simpler user-centered environments for Personal Data Management. The next sections describes the different service layers which are available through Semantic Turkey and how they can be used to propel Semantic based desktop applications.

4. Service Layers for Applications

The main underlying application consists of an RDF framework made of an HTTP application server (which in Semantic Turkey is automatically started through Firefox) based on Java technologies and of a set of client layers facilitating access by users or third party applications.

The whole extension mechanism of the framework is implemented through 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 (OSGi RFC0112, 2005) which provides extension capabilities for the service and data layers of the architecture. A comprehensive description of Semantic Turkey architecture can be found in (Griesi, Pazienza, & Stellato, 2007) and in (Pazienza, Scarpato, Stellato, & Turbati, 2008). In this section we focus instead on the different layers (see Fig. 3 above) and extension points which characterize Semantic Turkey as an open RDF framework with specialized functionalities for Personal Information Management.

4.1. Javascript extensibility

Thanks to javascript dynamic programming paradigm, where functions are first-class citizens of the language, functionalities such as the annotation resolver described in section 3, can be dynamically imported and associated to logically coherent events in different client applications of the Mozilla suite. The *pluggable functional objects* mentioned in section 3 can thus be considered independent components which can be exported and be reused in web browser as well as in email clients. For example, highlighting text in a web page within Firefox, and dropping it over a class, could invoke the same behavior when selecting text in emails from within Thunderbird. Conversely, reacting to changes in the underlying knowledge could produce different effects depending on the client platform which is connected to the

Semantic Desktop: finding RDFa (Adida & Birbeck, 2007) data on a web page from within the web browser, detailing scheduled events, could lead to the import of that data inside the semantic desktop's ontology, and the consequent export of this data inside other desktop applications for calendar management such as Lightning or Sunbird[1].

4.2. OSGi extensibility

OSGi compliance is obtained through the OSGi implementation developed inside the Apache Software Foundation, called Felix (felix.apache.org/).

Two main extension points have been introduced: an *OntologyManager Extension* and a *Service extension.*

The *OntologyManager* Extension point allows different triple-store technologies implementing low level RDF data storage, to be plugged to the system. Current implementations provide support for Sesame2, OWLIM and Jena (McBride, 2001) – through its NG4J extension (Bizer, Cyganiak, & Hartig) supporting named graphs – technologies.

The *service extension* point allows new java services to be plugged to the system, this way further desktop applications can automatically deploy and add their functionalities to the main service.

The set of services offered by the Knowledge Server provide high-level, macro operations, other than standard ontology management ones. The pure triple-level RDF data layer is not obfuscated by macro-operations, and is directly accessible through java API as well as replicated in a set of basic knowledge services for RDF manipulation.

A third extension point allows for the registration of plug-ins: these act as collectors for set of services sharing a common logical ratio. While standard service extensions are sort of add-ons to the main application and are always available unless deactivated or removed, extensions bound to plug-ins are activated/deactivated according to the status of the plug-in. Plug-ins are assigned to projects and their status and persistent information is stored with the metadata for each project.

The project-based behavior of the platform comes from its ontology-editor ancestry, while when it is being used as Semantic Desktop Server, a single project (called *main-project*), is always active and automatically started at system initialization. Each application based on the Semantic Desktop and needing customized services thus registers itself as a plug-in and installs all of its required services via OSGi.

Finally, a data extension point allows for the declaration of *support* and *application* ontologies which are loaded by the system to drive its behavior and the one of its extensions and connected applications. These ontologies are not treated the same way as imported domain/user ontologies and are explicitly registered for their role. Registering an ontology through this extension point has a variety of consequences: these are loaded automatically at system startup even if they are not explicitly imported by the edited domain ontologies and application ontologies' content (and content classified after application ontologies' concepts) is not shown explicitly but only managed and exposed indirectly through applications' services.

We enabled this classification of ontologies since all the data which is available through the Mozilla Semantic Desktop (MSD from now on) is available as RDF triples: it was thus mandatory to separate the knowledge which is being managed by the user, from the one which is being used by the Semantic Desktop to coordinate its activities. Despite this "conceptual separation" – ontology spaces are managed through the use of *named graphs* (Carroll, Bizer, Hayes, & Stickler, 2005) – having a single RDF cauldron where all triples are being stored allows for more tight connection between these spaces, so that, for example, data in the application space could be used to organize the domain information according to different *facets*, or add annotations which should not be available as domain ontology. As an example of application ontology, the basic version (i.e. no extensions installed) of MSD declares an *application ontology* called *Annotation*[2] describing the textual occurrences from which entities submitted by the user have been annotated, together with details about the document (type of document, url for web pages, title etc…) where these annotations have been taken. An example of *support ontology* is instead provided by the Sesame2 implementation of the *OntologyManager* extension point: Sesame2 library does not support OWL reasoning nor includes the OWL vocabulary; since Mozilla Semantic Desktop relies on the OWL vocabulary, this is being declared as a support ontology and dynamically added to the core knowledge.

Data defined upon vocabulary from the Annotation ontology (since it is an application ontology) is thus not shown by default in all ontology editing interfaces, and its content is made available to the user through MSD's functionalities (such as those for retrieving documents associated to ontology resources, or for highlighting all the annotations taken in a document), while resources from the OWL vocabulary (being it a support ontology) are shown but are kept separate from user data (owl vocabulary is not saved together with user data nor it is explicitly imported by user ontology).

4.3. HTTP Access

All of OSGi services are available via AJAX through HTTP request. The response to these requests is codified in XML or (in some cases) in JSON, depending on request type, available standards and compactness of the content.

Due to its complete platform/technology independence, this is the layer which can be exploited by any application which has no direct connection with the service layer and is not compatible with Mozilla technology.

4.4. Mozilla JavaScript API

Upon the above layer, a set of JavaScript API, completely hiding the HTTP request/response interaction, has been built by using Mozilla technology. These are the API which are currently used inside Semantic Turkey Semantic Web Browser.

These API are coded as exportable functions into Mozilla modules, a proprietary Mozilla solution for JavaScript allowing for persistence (JavaScript objects inside a module persist upon different imports of the same

[1] http://www.mozilla.org/projects/calendar/

[2] http://art.uniroma2.it/ontologies/annotation

module) and hiding/encapsulation (a module's developer must choose which objects/functions are exported by users of the module and which ones just serve as hidden internal machinery).
These JavaScript Modules (roughly paired with their service counterparts in the service layer) can thus easily be imported into any sheet of a Mozilla based application (or extension). In the following example:

```
Components.utils.import(
  "resource://stservices/SERVICE_Cls.jsm",semanticturkey
)
```

all the objects and functions exposed by the SERVICE_Cls module are imported into the variable semanticturkey: this is a good practice to prevent variable clashing, as Mozilla extensions share a common space where all script code (from main application and all of its extension) is pooled.
Once the above statement is explicated in a script document, API methods contained in SERVICE_Cls can be used in the same sheet, like in the following:

```
semanticturkey.STRequests.Cls.getInstanceList(clsName)
```

where all instances of class identified by clsName are retrieved and returned by the method.
HTTP masking is handled by a common module:

```
resource://stmodules/SemTurkeyHTTP.jsm
```

which is shared by all API methods. The SemTurkeyHTTP.jsm module contains convenience methods for composing GET and POST requests, for unmarshalling received XML/JSON over HTTP responses and recomposing them in terms of dedicated JavaScript objects.
Due to the masking of HTTP details by Mozilla JavaScript Semantic API, all of their methods return explicit JavaScript exceptions. These are classified as:

- *errors*: error JavaScript exceptions mask HTTP communication errors as well as exceptions thrown at run time by the invoked service and caught by the HTTP Server. Usually it is not easy for the common user to discover the problem which has been generated, and these kind of exceptions are considered as severe application faults
- *exceptions*: JavaScript exceptions marked as application exceptions are due to predictable java exceptions which occurred at server level. Usually they contain understandable messages which may be explicitly communicated to the user. Also, specific management of these exceptions depending on their type and the context where these occurred can be performed by the application invoking the method which threw them.

Developers of new applications based on the Mozilla framework can thus invoke the underlying services and handle exceptions depending on the context of invocation, thus following a traditional structured programming approach and producing readable "narrative scripting" code, instead of writing complex code for client-server interaction.
Application Developers willing to add further APIs for interfacing with their software, can extend the service layer through OSGi and then build new modules for the JavaScript API, relying on the common SemTurkeyHTTP.jsm infrastructure.

4.5. Reusable widgets for Semantic Applications based on this Mozilla Semantic Desktop

Applications exploiting the Mozilla Semantic Desktop which are based on the same Mozilla technology, can beneficiate of exportable widgets expressly dedicated to Ontology Management. We are currently expanding this aspect, which is currently limited to reusable widgets for class and property trees, and for resource editors (class, property, instance and ontology resource editor widgets) to cover a whole range of widgets for ontology maintenance and editing.
Also, to satisfy the more complex needs of end-user applications, which should hide the ontology editing aspects and show custom widgets more close to their specific nature, we are considering the addition of a dedicated UI generator based on the Fresnel model (Pietriga, Bizer, Karger, & Lee, 2006) for browser independent visualization of RDF graphs. Our UI generator will provide a Fresnel parser and UI generation facilities based on the XML User Interface Language XUL, which is adopted by the suite of Mozilla tools.

5. Conclusions

We have presented here our ongoing work for a fully-extensible RDF based platform realizing the Semantic Desktop paradigm.
The strength of Mozilla Semantic Desktop is not in the whole range of end-user services (which are currently limited to the Semantic Bookmarking services offered by its originating platform Semantic Turkey), but in the wide spectrum of connections that are exposed to future applications willing to interact with it.
A second point is on the depth and completeness of its ontology management capabilities, providing a solid platform with convenience methods for ontology editing, disburdening the application developer from the non-trivial effort of maintaining the underlying ontology. Keeping the RDF graph clean (free from potential redundancies and from dangling triples, i.e. triples out of the reachability of any application insisting on them) is in fact a non-trivial aspect from which applications should abstract and which is not supported by default triple-store systems. Advanced layers for RDF management should consider the kind of triple-store they are using, the level of reasoning which is supported (and, where necessary, the "trivial reasoning" which should be computed by them to present data in a readable way) etc.. to provide an homogeneous interaction layer for the application developer.
These "advanced management" requirements are not limited to pure graph maintenance. RDF/OWL pushed forward concepts such as explicit semantics, shareability and interconnectivity: platforms supporting shared knowledge for cooperation of RDF-based applications, should be able to provide powerful tools for meta-management: modularization, multi-faceted perspectives, visualization, are all fundamental aspects which should contradistinguish the layering of future RDF based frameworks, and in special case for Semantic Desktop Platforms.

References

Adida, B., & Birbeck, M. (2007, October 26). *RDFa Primer*. Retrieved from World Wide Web Consortium - Web Standards: http://www.w3.org/TR/xhtml-rdfa-primer/

Apostolou, D., Mentzas, G., & Abecker, A. (2008). Managing Knowledge at Multiple Organizational Levels Using Faceted Ontologies. *Journal of Computer Information Systems , Winter 2008-2009*, 32-49.

Bizer, C., Cyganiak, R., & Hartig, O. (Eds.). (n.d.). *NG4J - Named Graphs API for Jena*. Retrieved May 14, 2009, from NG4J - Named Graphs API for Jena: http://www.wiwiss.fu-berlin.de/suhl/bizer/ng4j/

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* (p. 54-68). Sardinia, Italy: Springer Berlin / Heidelberg.

Carroll, J. J., Bizer, C., Hayes, P., & Stickler, P. (2005). Named Graphs, Provenance and Trust. *WWW '05: Proceedings of the 14th international conference on World Wide Web* (p. 613-622). New York, NY, USA: ACM Press.

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.

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

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.

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.

Groza, T., Handschuh, S., Moeller, K., Grimnes, G., Sauermann, L., Minack, E., et al. (2007). The NEPOMUK Project - On the way to the Social Semantic Desktop. In T. Pellegrini, & S. Schaffert (Ed.), *Proceedings of I-Semantics' 07* (pp. 201-211). JUCS.

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

Iturrioz, J., Díaz, O., & Fernández Anzuola, S. (2008). Toward the Semantic Desktop: The seMouse Approach. *IEEE Intelligent Systems , 23*, 24-31.

Iturrioz, J., Díaz, O., Fernández Anzuola, S., & Azpeitia, I. (2003). The Semantic Desktop: an architecture to leverage document processing with metadata. In S. Guier (Ed.), *Multimedia and Data Document Engineering (MDDE'03)*. Berlin, Germany.

Kahan, J., & Koivunen, M.-R. (2001). Annotea: an open RDF infrastructure for shared Web annotations. *WWW '01: Proceedings of the 10th international conference on World Wide Web* (pp. 623-632). Hong Kong, Hong Kong: ACM.

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.

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

OSGi RFC0112. (2005). Retrieved from http://www2.osgi.org/Download/File?url=/download/rfc-0112_BundleRepository.pdf

Pazienza, M., Scarpato, N., Stellato, A., & Turbati, A. (2008). Din din! The (Semantic) Turkey is served! *Semantic Web Applications and Perspectives*. Rome, Italy.

Pietriga, E., Bizer, C., Karger, D. R., & Lee, R. (2006). Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In I. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, et al. (Ed.), *The 5th International Semantic Web Conference (ISWC06). LNCS 4273*, pp. 158-171. Athens, GA, USA: Springer Verlag.

Prud'hommeaux, E. :. (2008, January 15). *SPARQL Query Language for RDF*. Retrieved from World Wide Web Consortium - Web Standards: http://www.w3.org/TR/rdf-sparql-query/

Sauermann, L. (2005). The Gnowsis Semantic Desktop for Information Integration. *1st Workshop on Intelligent Office Appliances(IOA 2005): Knowledge-Appliances in the Office of the Future*. Kaiserslautern, Germany.

Sauermann, L., van Elst, L., & Dengel, A. (2007). PIMO - A Framework for Representing Personal Information Models. In T. Pellegrini, & S. Schaffert (A cura di), *Proceedings of I-MEDIA '07 and I-SEMANTICS '07 International Conferences on New Media Technology and Semantic Systems as part of (TRIPLE-I 2007)*. Graz, Austria.