Semantic Turkey goes SKOS
Managing Knowledge Organization Systems

Manuel Fiorelli
University of Rome, Tor Vergata
Via del Politecnico 1
00133, Rome
+39 0672597334
fiorelli@info.uniroma2.it

Maria Teresa Pazienza
University of Rome, Tor Vergata
Via del Politecnico 1
00133, Rome
+39 0672597378
pazienza@info.uniroma2.it

Armando Stellato
University of Rome, Tor Vergata
Via del Politecnico 1
00133, Rome
+39 0672597330
stellato@info.uniroma2.it

ABSTRACT
In this paper we describe a novel SKOS editor built on top of the web browser Mozilla Firefox. Our tool is targeted towards KOS developers and KOS consumers as well. Indeed, the ability to surf the Web with a standards compliant browser proves useful for both: the former may prove the soundness of a concept by associating it with a concrete set of web resources, whereas the latter may exploit a given KOS to effectively organize information collected from the Web. The editor has been designed as an extension of the knowledge management and acquisition tool Semantic Turkey. The proposed SKOS editor creates a dedicated perspective within an OWL compliant environment, which eases dealing with KOSs. By relying on such rich environment, the editor allows the user to exploit the subtle relationship between SKOS and OWL, thus opening it up to more elaborated modelling solutions, in contrast to other tools which are built on top of the SKOS direct semantics.

Categories and Subject Descriptors
D.2.13 [**Software Engineering**]: Reusable Software – *Domain Engineering*

I.2.6 [**Artificial Intelligence**]: Learning – *Knowledge Acquisition*

General Terms
Management, Documentation, Design, Standardization, Languages

Keywords
Thesauri, Knowledge Organization Systems, SKOS, Information Visualization, Knowledge Management and Acquisition, Semantic Bookmarking, Semantic Annotation

1. INTRODUCTION
The term Knowledge Organization System was introduced by the Networked Knowledge Organization Systems Working Group at its initial meeting at the ACM Digital Libraries '98 Conference in Pittsburgh, Pennsylvania [19].

Later on, a definition of Knowledge Organization Systems (KOS) has been provided [8], as this term is "intended to encompass all types of schemes for organizing information and promoting knowledge management. Knowledge organization systems include classification schemes that organize materials at a general level (such as books on a shelf), subject headings that provide more detailed access, and authority files that control variant versions of key information (such as geographic names and personal names). They also include less-traditional schemes, such as semantic networks and ontologies." In what follows we use the term KOS in a slightly narrower sense, since in our judgment ontologies and other formal resources are best treated on their own.

Librarians have used KOSs to assign a physical position on the shelf to each book and, later on, to easily discover books related to a given topic. In the modern virtual world of Information Organization the purpose of KOSs has not changed in its fundamentals: now that digital libraries are spreading over the Web, KOSs are even more relevant resources and there is great urge for their sharing and standardization.

KOSs are not only relevant to the organization of libraries, but are widely used in Information Retrieval in general and in numerous tasks related to Computational Linguistics and Machine Learning. In short, KOSs are used to categorize resources (whatever they are), in order to make it easier to retrieve them later.

In the context of the Semantic Web KOSs have progressively found their way: from the *époque* of "ontologies everywhere" now we are facing a new era where publishing interlinked repositories of mere data is the priority and the role of rigid formal vocabularies has been even debated (see [12]) or the many results on blogs and mailing list archives returned by searching the Web for: "Does the Semantic Web Need Ontologies?"). This switch has been facilitated by the technological progress in the field (most of modern triple stores are able to handle millions – if not trillions – of triples[1]) and pushed forward by the progressive adherence of companies and big organizations to the Linked Open Data [1] paradigm.

KOSs are thus even more relevant in this scenario: they generally require very weak (formal) semantics, thus easing their production by domain experts. This simplification also facilitates the massive reuse and export of the several concept schemes and thesauri already available inside several organizations. At the same time, KOSs provide mere conceptual indexes for organizing knowledge content. KOSs may thus highly innovate the concept of traditional search, as they provide much more than traditional controlled vocabularies or machine readable dictionaries [9]; their conceptualization supports the idea of a controlled set of indexes, their multilingual lexicalization enables high recall (but still

I-SEMANTICS 2012, 8th Int. Conf. on Semantic Systems, Sept. 5-7, 2012, Graz, Austria

[1] http://www.w3.org/wiki/LargeTripleStores

converging on the unifying concept set) and, when available, semantic relations enable the move from "search" to "discovery": users may in fact discover new topics they are interested in by exclusively navigating concept structures and only later retrieve the content associated to them.

Although there are several types of KOSs, there is a sufficient overlap among them so that the W3C has been able to create a dedicated language, SKOS [17], for modelling the features mostly recurring in existing KOSs. SKOS was thus created as an RDF vocabulary, defined on top of the Web Ontology Language (OWL). The need for a new language in the RDF family was motivated by the twofold nature of concepts in a KOS, being them objects of the domain of interest which have to be described (as if they were owl individuals) and, at the same time, play the role of categorizing terms (a role normally associated to classes). At the same time, KOSs do not need any class/instance differentiation as concepts represent the sole indexes of a domain description. The solution has thus naturally emerged: coin a specific class (skos:Concept) to describe KOS concepts, which consequently are individuals (in the OWL sense) and thus can be described. Finally, coin a dedicated vocabulary of properties which apply to these concepts and which can thus be universally adopted and managed by Semantic Web applications.

The subtle connection between SKOS and OWL allows interesting modelling solutions, in which resources described in terms of much more formal vocabularies are linked to SKOS concepts when further formalization is not required. For example, the Open Government Working group is editing a vocabulary, named Data Cube[2], for publishing (statistical) multi-dimensional data on the Web. Each dimension of the hypercube expresses a concept (e.g. the notion of reference region, age, gender, ...), which can be represented (optionally) as a SKOS concept.

The wide adoption of SKOS is largely dependent on the availability of conformant and practical implementations. In the transition period the relation with OWL can be exploited to manage a SKOS description as an ontology. However, the promotion of the standard to the intended audience, which comprehends mostly librarians and terminologists, requires technologies which are closer to such users– in terms of usability –than to knowledge engineers.

In this paper, we present a novel SKOS editing framework, hosted as an extension for the Firefox Web Browser[3], modelled over the existing Knowledge Management and Acquisition Framework Semantic Turkey [13] and designed specifically with the potential KOS user in mind. The editing framework in fact does not only support all of the editing requirements of SKOS (and SKOS-XL [18], its language extension for characterizing lexicalizations), but also enables a smart bookmarking/annotation feature for associating web pages content with related concepts in a SKOS scheme. Indeed, the ability to surf the Web with a standard compliant browser proves useful for both KOS developers and KOS consumers: the former may prove the soundness of concepts by associating them with concrete web resources, the latter may exploit existing KOSs to effectively organize information collected from the Web.

The paper is organized as follows: in section 2 we present a few of the most notable SKOS editors available, we then provide the motivations behind the realization of this tool in the following section. Section 4 provides details on the user experience and on the features provided by the tool, while section 5 provides insights over the system extensible architecture. Finally, the two last sections provide comments and hints for future research work on this open source tool.

2. RELATED WORKS

We have analysed existing SKOS editors, in order to identify use cases and how they are addressed.

The Food And Agriculture Organization[4] (FAO) has developed a tool for collaborative thesaurus and vocabulary management, called VocBench[5]. Originally built to manage the AGROVOC[6] [10] thesaurus, an indexing scheme containing over 32000 concepts related to the food and agriculture domain expressed in more than 20 languages, VocBench has recently evolved into a general purpose thesaurus and concept scheme management framework [10], featuring a distinctive collaborative nature.

The framework has a satisfactory coverage of SKOS features and has proved to be able to manage as a large thesaurus as AGROVOC is. One of its main drawbacks is the lack of advanced modelling solutions, as no OWL construct is available. However, the strongest point in favour of VocBench lies in a really interesting support for collaborative editing. The framework manages the maintenance workflow, assuring that proposed changes have been validated before they are consolidated into the mainstream KOS.

VocBench offer consists of a web application, which appears to be an appropriate choice in the highly distributed scenario where VocBench is called to operate.

VocBench persistence system is currently based on the API of Protégé 3 OWL [11], extended with the database backend, to be able to manage large amounts of data. Support for SKOS is not "direct", as VocBench is based on a proprietary inner model – based on OWL – for representing vocabularies. Offline utilities for exporting its content (by directly accessing its database backend) to SKOS and SKOS-XL are however available. VocBench is distributed under open source conditions and terms of reuse.

PoolParty[7] is another web-based application, distributed as a commercial solution. PoolParty features an advanced implementation of SKOS, including consistency checks for SKOS integrity constraints which are not bound to a formal OWL specification (e.g. disjointness between the set of literals used as preferred labels for a given concept, and those used for alternative or hidden labels).

Distinguishing features are: support for linking to other KOSs (following the principles of Linked Open Data), deployment of services related to the managed data (such as SPARQL endpoints) and other accompanying features which are actually applications on their own, such as text analysers (which may be employed to discover new concepts from documents) and text/concept indexing of managed documents to enable semantic search. PoolParty comprises a user management system, but has a support for collaboration not as evolved as VocBench has.

[2] http://www.w3.org/TR/2012/WD-vocab-data-cube-20120405/

[3] http://www.mozilla.org/firefox/

[4] http://www.fao.org/

[5] http://aims.fao.org/tools/vocbench-2/

[6] http://aims.fao.org/website/AGROVOC-Thesaurus/

[7] http://poolparty.biz/

SKOSEd[8] is a plugin for the ontology editor Protégé that provides an additional tab (named SKOS view) for editing SKOS entities. The layout of the SKOS view is a customized version of the standard Protégé Individual view providing different perspectives on the SKOS taxonomy (e.g. by filtering concepts depending on the skos:ConceptSchemes they belong to), like the Class Tree in the Classes View.

Further SKOS customization consists of dedicated widgets for specific SKOS properties, so that their access is facilitated with respect to other properties.

Differently from the previous tools, which are focused on thesauri development, SKOSEd benefits from the capabilities inherited from its hosting environment Protégé, allowing for an interwoven editing of SKOS and OWL constructs, to give maximum modelling power to the developer.

On the other hand, some ontology editing tools, such as TopBraid Composer[9], rely on their high level of customizability without offering a dedicated view/perspective over SKOS. In fact the user may configure the existing widgets and panels to host an appropriate perspective over SKOS data (e.g. the concept tree may be built by using the *Association View* which allows defining taxonomies over custom properties, and then setting the skos:broader/skos:narrower properties as the carriers for the taxonomical relation). A dedicated collaborative application (Enterprise Vocabulary Net[10]) for SKOS editing has however been published, based on the same backing technology.

3. MOTIVATIONS

Most of the tools analyzed in section 2 are concerned with the editing of KOSs regardless of their use for indexing or other purposes. PoolParty and Enterprise Vocabulary Net go partially beyond, since they provide facilities for automatic indexing and discovery of concepts. Nevertheless in our vision the content is not only relevant for the sake of statistical analysis, but even human terminologists may take advantage of it in order to ground the proposed terminology to a concrete set of examples. Under this perspective the importance of a tool for the interactive acquisition of knowledge from documents is clear. Furthermore, we believe it is important to allow the acquisition of knowledge directly from the Web, which has evolved into a very comprehensive source of information.

The experience shows that surprisingly such kind of tool dealing with third-party Web contents should not be deployed as web applications. Indeed, pure web-based solutions rely on frames and other approaches that are generally regarded as harmful[11] with respect to usability, maintainability and other concerns. The recently launched Volunia[12] social search engine uses frames to decorate third-party web sites with a toolbar assisting users during the navigation (e.g. providing sitemaps and social functionalities). Unfortunately, in numerous observers' opinion Volunia is excessively invasive and as restrictive as the "cages" of traditional Social Networks that it was expected to break.

It is actually the Web browser that should evolve to support new functionalities for enhancing the Web experience, going much further the simple resolution of addresses and the provision of Web content. The strong customizability and extendibility of current Web browsers are supporting this trend, though, on the other hand, the lack of unified environment for programming extensions is hampering it: in fact the main disadvantage in developing browser extensions and plugins rather than Web applications lies in the necessity to develop ad-hoc portings for each different browser being supported.

Thus, our tool has been designed as an extension for the Web browser Mozilla Firefox, enabling users to acquire knowledge while surfing the Web by using the tool they are more proficient with: the Web browser. A browser-hosted extension (though backed by robust and scalable Java technologies) satisfies the ideal requirement for an immediately usable tool. Nevertheless our solution has to be considered a desktop application, which is completely under the user's control and open to unlimited customization being an open-source and free-of-charge product.

The proposed tool has been developed as an extensions of Semantic Turkey, a knowledge acquisition and management platform for the development of applications targeted to such Semantic Web standards as RDF, RDFS, OWL and SKOS. Relying on such rich environment, our tool allows users to exploit the subtle relationship between SKOS and OWL, which proves useful in reusing SKOS conceptualization within other OWL based models.

4. USER EXPERIENCE

The role of a SKOS editor is to provide a more convenient perspective on the KOS content. Indeed, from the viewpoint of an OWL editor SKOS concepts are but individuals of the class skos:Concept and do not require any special treatment. The editor will probably show SKOS concepts in a flat list rather than a tree, since it has no understanding and special management of hierarchical relations between concepts. Hence, a dedicated editor is needed, able to hide unnecessary details (e.g. the fact that a concept is an instance of the class skos:Concept) and interpret SKOS specific constructs under a more direct interpretation.

The first step in this direction, discussed in section 4.1, is to provide the user with a convenient perspective on the KOS entities (e.g. concepts and schemes), paired with the basic editing functionalities. The ability to manage the content of a KOS is but the baseline, which is common to all tools analysed in section 2. The distinctive feature of our tool is the support of the interactive annotation of information resources from the Web (see figure 2). This capability, discussed in section 4.2, is important for two reasons (and two classes of users): terminologists may document their conceptualizations by means of concrete examples from the Web, while end-users may utilize an existing indexing scheme in order to categorize web resource accordingly.

The advanced user may still exploit the connection with OWL for more advanced modelling solutions by means of the functionalities provided by the hosting ontology editing platform.

4.1 Main Editing Functionalities

The user is able to manage the KOS content by means of a sidebar, which has been added to the browser.

The concepts within the edited KOS are shown in a hierarchical manner based on the SKOS taxonomic relations (i.e. skos:broader and skos:narrower).

Concepts may be organized into schemes, which enable a loose form of containment, not covering statements about concepts, which are instead asserted globally (i.e. statements about a

[8] http://code.google.com/p/skoseditor/

[9] http://www.topquadrant.com/products/TB_Composer.html

[10] http://www.topquadrant.com/solutions/ent_vocab_net.html

[11] http://en.wikipedia.org/wiki/Framing_(World_Wide_Web)

[12] http://www.volunia.com/

concept are not tied to any concept scheme). Managing multiple schemes is useful for the purpose of linking and mapping different KOSs, but may degrade the usability of the editor if the user wants to focus on one of them only. This problem has been solved by asking users to choose the scheme they want to restrict the concept hierarchy to.

An important contribution of SKOS to RDF lies in the definition of three properties allowing a more precise labelling of resources through natural language expressions[13]. The three properties differentiate which expressions are mostly adopted to represent a concept in a given language (skos:prefLabel), which ones are common synonyms or acronyms (skos:altLabel) and which ones (skos:hiddenLabel) can be used internally by an application for coverage of various linguistic phenomena (e.g. common misspells) or application specific needs (e.g. word stems). Furthermore, SKOS-XL introduces XLabels, that is reifications of labels: this means that labels become thus first class citizens of a domain description and, beyond their *lexical form*, they can be related to each other through lexical relation or further characterized through dedicated descriptors.

The SKOS editor of Semantic Turkey allows explicit management of these labels and exploits their content to provide a human-friendly representation of SKOS concepts based on their preferred labels instead of their URI or qualified name. Within a concept scheme in fact the preferred labels assigned to concepts can be safely considered local identifiers, since by convention they have to be unambiguous in a given language. Hence, it is sensible to present concepts and schemes by their preferred labels (if any) in the user local language instead of their qualified name (QName). This alternative presentation may give the user a better sense of the KOS content and it is crucial in those circumstances (e.g. AGROVOC) where concepts are not provided with a human friendly name.

Finally, for KOSs modelled after the SKOS-XL vocabulary, the indirection from the concept to the lexical form of its skosxl:prefLabel is automatically managed by the application.

4.2 Semantic Annotation and Bookmarking

Users may surf the Web with a standards compliant Web browser, associating information found in Web documents with concepts from the current KOS. The utility of this association is twofold: KOS developers may document a concept by attaching a set of web resources to it, whereas a KOS consumer may categorize information resources tagging them with concepts from the KOS. The nature of the association may also vary: the editor supports both the bookmarking of web pages as a whole and the annotation of the occurrences of concepts within a web page.

In the first case, the bookmarked page metadata are stored together with the link to a skos:Concept through the dcterms:subject property. That concept is not required to be explicitly mentioned in the bookmarked page, but it is assumed to represent a category of Web pages (e.g. related by the same topic). Bookmarking may support topic based IR or the creation of gold standards for tasks of document classification.

In the second case, the annotation of specific portions of text is triggered by drag'n'drop action performed by the user. When a portion of text is selected, dragged and finally dropped over a concept in the tree, several options are presented to the user. This functionality is a follow-up of the original (OWL) version of Semantic Turkey and, in general, the available options depend on the nature of the RDF resource where the text has been dropped on (i.e. classes or instances in the case of OWL).

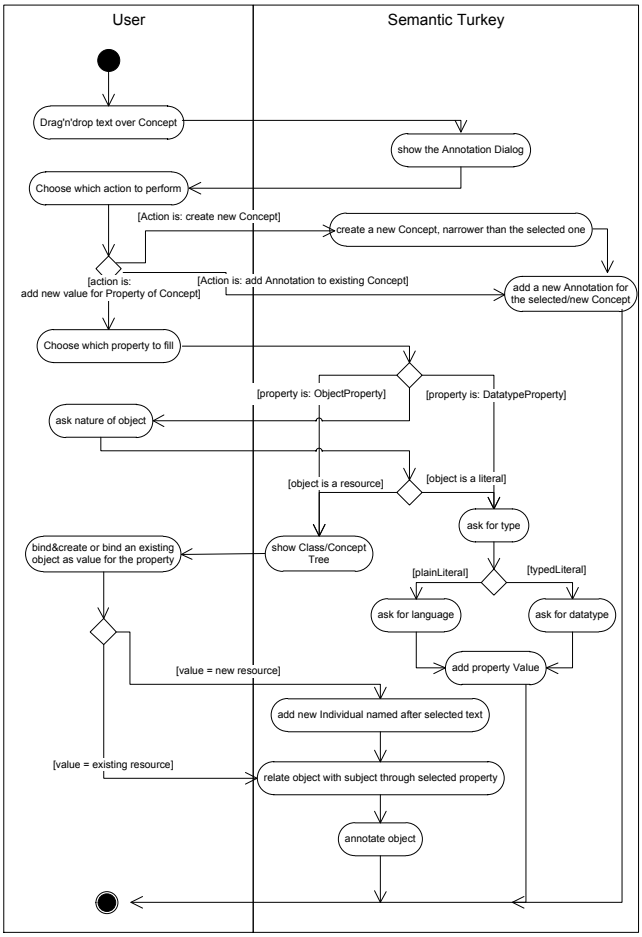

Figure 1. UML activity diagram for semantic bookmarking/annotation in SKOS

In SKOS there is only the notion of skos:Concept which covers the double role of realizing taxonomies (classes) and of holding descriptions of relevant domain objects (individuals). For this reason, we have provided different options combining those already available for OWL classes and individuals.

Figure 1 illustrates, through an activity diagram, the flow of actions which are performed when information is dropped on a skos:Concept. Firstly, the user is prompted with a dialog window listing the set of available options, namely:

1. add an annotation to the selected concept,
2. create a new concept (and annotate it),
3. add a new value for a property of the concept.

In the first case, an annotation is added to the concept where the text has been dropped on. The nature of the annotation may vary, depending on the annotation model which has been selected. In general, an annotation for a concept will include: the selected text as an occurring lexical form of that concept, a reference to the page where this text has been selected, metadata about the source

[13]This is actually a contribution to the whole family of RDF languages, as the subject of SKOS labelling properties is not restricted to skos:Concepts

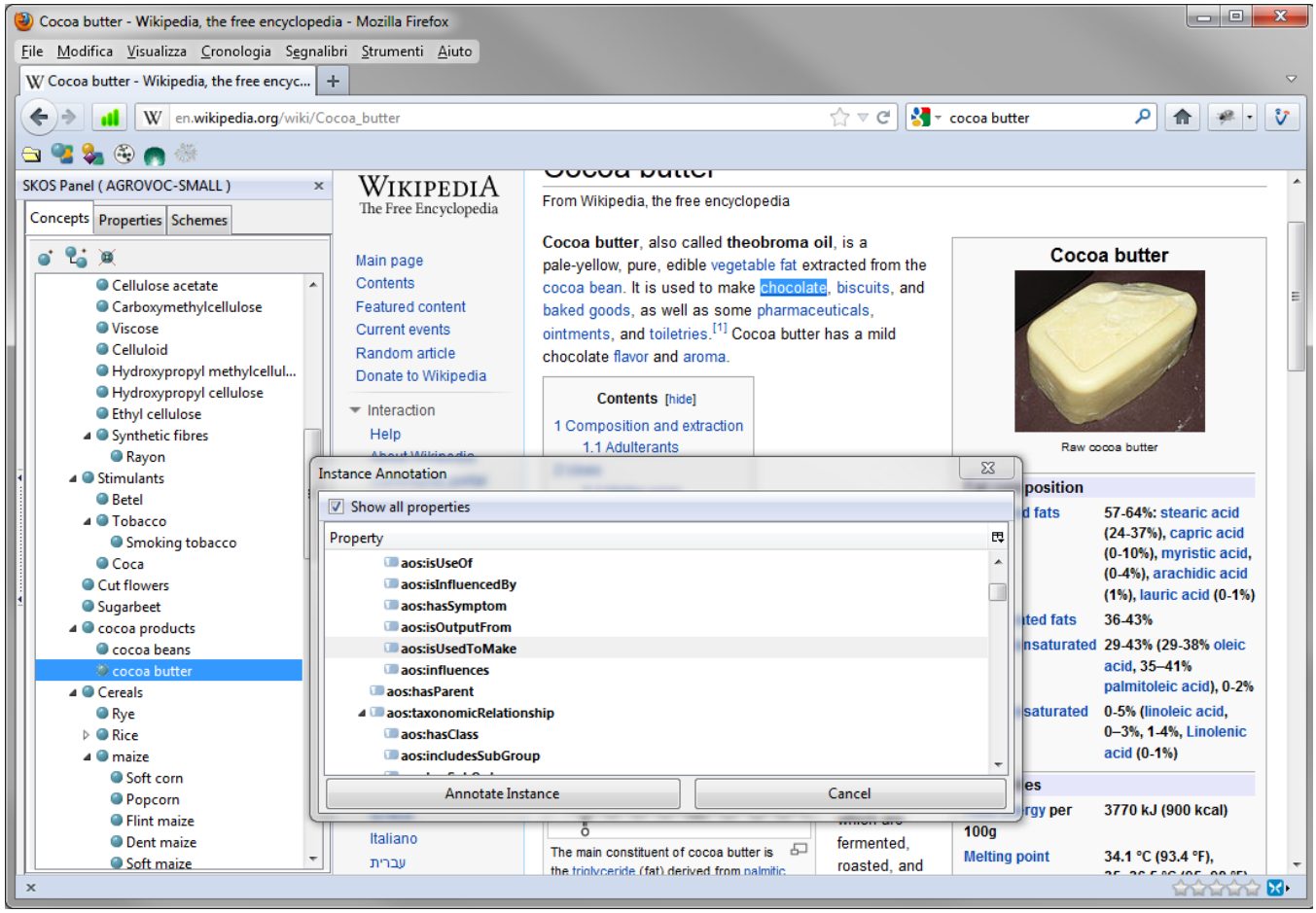

Figure 2. Annotating information about a concept in the hierarchy tree

page, and (optionally) a punctual reference to the position of the text in the page. The annotation model shipped with Semantic Turkey does not register the exact position of the selected text within the web page, resulting very similar to bookmarking.

Range Annotator[14] is an extension of Semantic Turkey providing an annotation model in which punctual references are stored as XPointers. There can be concerns about the stability of those annotations with respect to changes in web pages, but they do not apply when considering a stable corpus of web pages (e.g. Wikipedia assigns a distinguished URI to each version of every article).

The second option allows creating a new concept, named after the content of the selection, as a narrower concept of the one where the text has been dropped on. An annotation is also added to the newly created concept by using the same text content.

Finally, the third choice opens a more complex interaction to assert a new relation in the KOS, whose subject is the concept where the text has been dropped on. The user initially chooses a property (see Figure 2) and decides which kind (a Literal or a URI) of value will be associated to it (unless univocally determined by the property type). Finally, in case of a URI, the user can select an existing resource as the object of the relation or create a new one for that role, named after the content of the selection. In both cases, the object is annotated as described in point 1.

4.3 Further Features

Users may exploit Semantic Turkey's SPARQL [16] capabilities to query SKOS descriptions by means of a rich and standard query language, which mostly operates by unification of triple patterns with the underlying RDF graph.

The editor provides a graph view of the KOS, which has been implemented by extending the applet for ontology visualization already available in Semantic Turkey. The framework asks the extender to specify what properties will be considered (e.g. skos:narrower) and how vertices and edges will be rendered.

Finally, the editor inherits support for representation of metadata and recursive resolution of the owl:import directive, allowing the import of further schemes and vocabularies.

5. ARCHITECTURE

The offer of Semantic Turkey covers an extensible platform [13] for the development of RDF based applications. The platform features a three-layer architecture, reflecting the common separation among presentation, business and data management. Each layer bases on widely adopted technologies and offers specific mechanisms for extension development. Semantic Turkey's SKOS editor is composed of a set of extensions for each layer of the host platform. The resulting system is further

[14] http://code.google.com/p/st-rangeannotator/

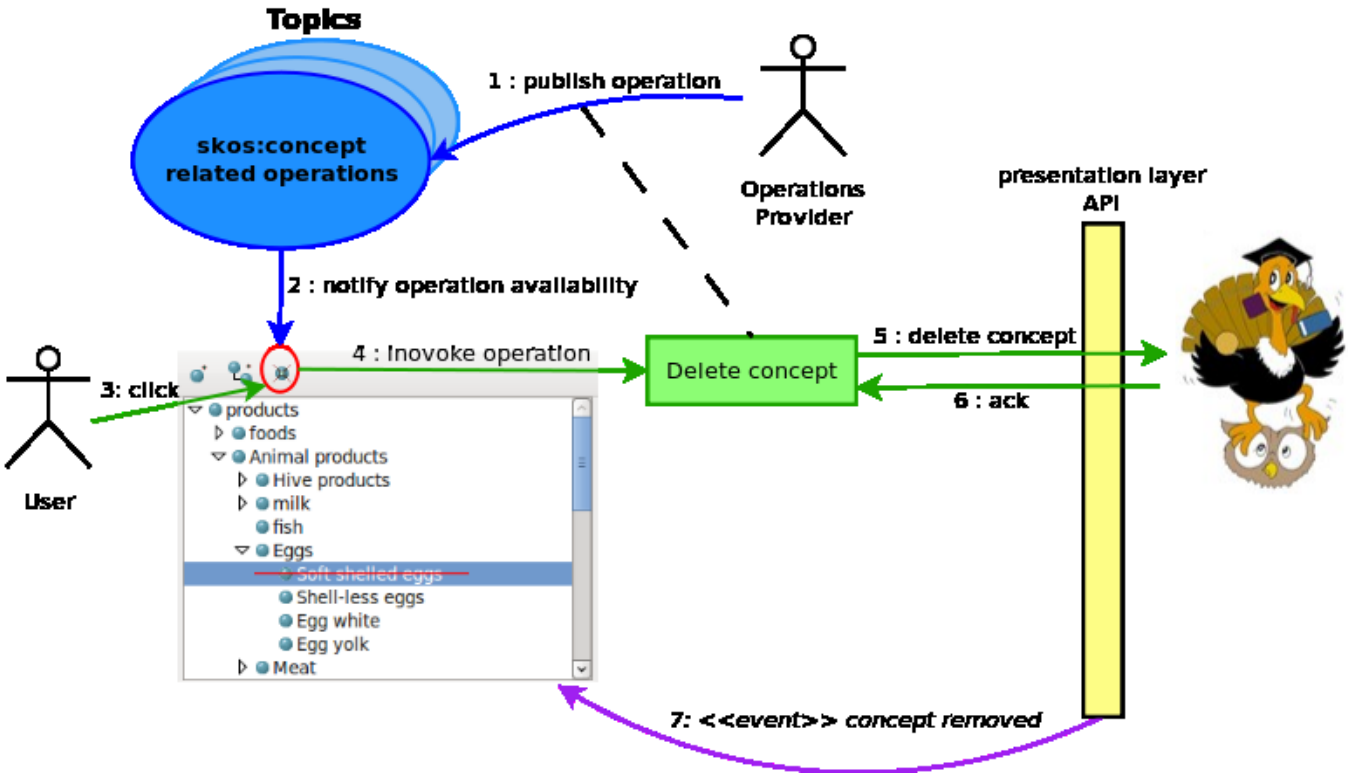

Figure 3. MVC Architecture

extensible, as more capabilities can be added freely and the SKOS related functionalities can be reused in different contexts.

5.1 Presentation

The presentation layer, associated with the Firefox extension, is primarily built over technologies adopted inside the Mozilla platform, such as XUL, JavaScript, CSS and XBL.

We have designed the presentation layer in a way that it would be easy to reuse graphic elements and to inject further capabilities into them. To fulfil this objective, we have broken the user interface into reusable parts, known as widgets [15], which can be imbued dynamically with operations (thus providing specific extensions points for third party developers).

We have extended XUL, the Mozilla user interface definition language, with a set of application specific widgets: these widgets have been implemented by means of XBL bindings and automatic attachment is guaranteed by a dedicated CSS style sheet.

Two widgets have been identified so far for SKOS: the concept scheme list and the concept tree. As lists and trees are strictly related (i.e. a list is a flat tree and a tree is recursively defined as a list of trees), we have a common abstract widget which models the interaction with a tree oriented data source. The abstract widget specifies a set of data related operations, such as root retrieval and children retrieval. Each widget implementation binds those abstract operations to concrete methods, which in turn depend on the appropriate middle layer service. The decoupling among widgets and operation providers is guaranteed by the employment of the well-known publish-subscribe [4] connector.

Figure 3 depicts the general architecture which guarantees the provisioning of operations to widgets and, ultimately, to users. A widget declares its interest in a particular topic (e.g. concept related operations) by subscribing to it. By doing so, it has to comply to a set of interfaces for that topic. When an operation (e.g. concept deletion) is published on a topic, the framework notifies its availability to registered widgets, which may decide how to render it (e.g. by means of a toolbar button). If the user activates the operation, it is invoked with a reference to the widget. The operation code uses that reference to access the provided interfaces, collect the needed information (e.g. the currently selected concept) and, finally, execute the request (e.g. invoke the middle layer service for the concept deletion). This mechanism effectively decouples widgets from operations, because neither the former know how many operations it will have, nor the latter have to know precisely which widgets they are attached to. Being a proxy for the underlying data (the *view* in a MVC perspective), widgets remain in sync with them, updating themselves when something interesting happens (see section 5.2).

5.2 Business

The functionalities offered to the end-user are either more elaborated or more constrained than those provided by the underlying OWLART API[16]. For each element of a KOS (e.g. concepts and schemes) the user is provided with a set of CRUD (Create, Read, Update, Delete) operations, the semantics of which is imperative more than assertive. For instance, our editor refuses to create a new concept when there is already one with the same name, whereas the underlying API accept to perform the operation, as the API semantics bear no issue with multiple assertions and just ignore redundant triples. Furthermore, there are many more operations which simply go beyond the sole scope of RDF management (e.g. project management, user preferences etc..). The gap between the underlying API and the operations

[16]OWLART API: http://art.uniroma2.it/owlart/

offered by the application is covered by the business level of the SKOS editor.

The business layer services may be invoked synchronously by means of the HTTP protocol. According to the front controller design pattern all requests are routed to a single servlet, which in turn will dispatch them to the appropriate handler, given the service name parameter inside the request. Actually, business functionalities are implemented by service handlers, which result largely independent from web technologies (e.g. the HTTP protocol), since those are taken in charge by the front controller.

The business level services are exposed to the presentation layer through a set of JavaScript modules, which hide the communication issues (e.g. the use of Ajax [7]) and takes care of some horizontal aspects, such as firing events.

Widgets and other components managing a certain kind of resources should listen to a small set of events which are fired by the business layer for relevant changes of those resource. For example, in Figure 3 it has been shown how the deletion of a concept is notified to a widget (possibly different from the one the request originated from) which has the chance to update itself in order to stay in sync with the data.

5.3 Data Management

Semantic Turkey accesses RDF repositories by means of an implementation neutral API, named OWL ART API, which effectively allow ST to wrap different triple stores for managing its projects content. The appropriate triple store may thus be plugged to address different requirements, such as memory footprint, scalability and so on.

The OWLART API provide several layers for managing RDF repositories, ranging from mere triple-oriented management to more advanced modelling primitives based on W3C vocabularies. In particular, they feature operations to directly manage SKOS and SKOS-XL entities – such as concepts, schemes and xlabels – instead of working at triple level. Applications may be written on top of these API to exploit SKOS descriptions for various purposes.

Besides the uniform API for accessing data, a dedicated section of the OWLART interface allows for inspection of the wrapped triple store, so that different characteristics may explicitly be exposed and thus taken into account by the application (e.g. if the triple store is persisted in real-time or not, if it supports reasoning, and in case which materializations, etc..), thus implying different interaction modalities with the user, or, conversely, the establishment of different strategies for exposing an homogenous behaviour to them.

Extending the API to a new triple store technology mainly includes an implementation of both their data access interface and the model factory, which is in charge of the construction of the model objects by means of implementation specific mechanisms. The API provide a default implementation of the higher level, (vocabulary oriented) interface, thus requiring minimal effort to enable support for a new triple store.

6. CONCLUSIONS AND FUTURE WORKS

In this paper we have presented an improvement of the Semantic Turkey framework for supporting development of KOSs through the SKOS and SKOS-XL languages of the RDF family.

For the sake of usability essential features have been made accessible from the main sidebar, whereas the user may learn more advanced functionalities later, in an incremental way. Both KOS developers and consumers may take advantage of the editor, as each one can focus on the functionalities they need. Moreover, the tight integration with a Web browser allows the user to surf the Web in a natural way. Functionalities for semantic annotation and bookmarking allow users to keep note of relevant information found on the Web (both pages as a whole and specific portions of text) and associate it with concepts from a KOS (or, conversely, to acquire new conceptual content and keep the information about the textual source where it has been observed).

We believe that the editor may benefit from capabilities for automated knowledge acquisition. Human supervision will always be required for the construction of critical KOSs (e.g. the reference vocabulary of a public institution), but automated tools could be employed for reducing the effort, by providing suggestions about new content to add.

We are currently building on top of Semantic Turkey an open knowledge acquisition layer, based on the CODA [3] [6] framework, allowing users to plug their own acquisition components, tailored to their objectives and requirements. Aiming at maximum compatibility and compliance to standards, CODA builds on top of the Unstructured Information Management Architecture [5] (UIMA) and features an orchestration mechanism for knowledge acquisition. A declarative language, PEARL [3] [14], for projecting UIMA extracted data into RDF triples is also provided by the CODA system. This approach will promote the reuse of state-of-art analytics and facilitate plug-and-play scenarios, as in [3], where users will just sit in front of the development environment, load their ontology/concept scheme, and automatically get knowledge extractors downloaded from the Web.

Another direction for improvement lies in the collaborative editing of KOSs. Semantic Turkey has already basic support for multi-user editing, although it is not possible to define access control policies nor these have been enforced consistently yet. In our vision, users should work in sandboxes, so that contributions are distinguishable from each other and from already assessed knowledge. Thus, mechanisms should be provided for integrating these contributions and for promoting them to the status of stable knowledge.

7. ACKNOWLEDGMENTS

This work has been partially supported by the EU-funded project INSEARCH[17]

8. REFERENCES

[1] Christian Bizer, Tom Heath, and Tim Berners-Lee, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems (IJSWIS), Special Issue on Linked Data*, vol. 5, no. 3, pp. 1-22, 2009.

[2] Caterina Caracciolo et al., "Thesaurus Maintenance, Alignment and Publication as Linked Data: The AGROOVOC Use Case," in *Metadata and Semantic Research*, Elena García-Barriocanal et al., Eds.: Springer Berlin Heidelberg, 2011, vol. 240, pp. 489-499.

[3] Andreea Diosteanu, Andrea Turbati, and Armando Stellato, "SODA: A Service Oriented Data Acquisition Framework," in *Semi-Automatic Ontology Development: Processes and Resources*, Maria Teresa Pazienza and Armando Stellato, Eds.: IGI Global, 2012, ch. 3, pp. 48-77.

[17] FP7-SME-2010-1, Research for the benefit of specific groups, GA n° 262491

[4] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec, "The many faces of publish/subscribe," *ACM Computing Surveys*, vol. 35, no. 2, pp. 114-131, 2003.

[5] David Ferrucci and A. Lally, "Uima: an architectural approach to unstructured information processing in the corporate research environment," *Nat. Lang. Eng.*, vol. 10, no. 3-4, pp. 327-348, 2004.

[6] Manuel Fiorelli, Maria Teresa Pazienza, Steve Petruzza, Armando Stellato, and Andrea Turbati, "Computer-aided Ontology Development: an integrated environment," in *New Challenges for NLP Frameworks 2010 (held jointly with LREC2010)*, La Valletta, Malta, 2010, 22 May, 2010.

[7] J.J. Garrett. (2005, February) Ajax: A New Approach to Web Applications. [Online]. http://www.adaptivepath.com/publications/essays/archives/000385.php

[8] Gail M. Hodge, "Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files," Washington, DC, ISBN 1-887334-76-9, 2000.

[9] Nancy Ide, Jean Véronis, and Aix en-provence Cedex, "Machine Readable Dictionaries: What have we learned, Where do we go," in *In Proc. of the post-COLING '94 intl. workshop on directions of lexical research, Beijing*, 1994, pp. 137-146.

[10] Johannes Keizer et al., "A Collaborative Framework for Managing and Publishing KOS," in *The 10th European Networked Knowledge Organisation Systems (NKOS) Workshop*, Berlin, Germany, September 2011.

[11] Holger Knublauch, Ray W. Fergerson, Natasha Friedman Noy, and Mark, A. Musen, "The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications," in *Third International Semantic Web Conference - ISWC 2004*, Hiroshima, Japan, 2004.

[12] Enrico Motta, Frank van Harmelen, Michael Witbrock, and Tom Heath, "Does the Semantic Web Need Ontologies?," in *Panel at the International Semantic Web Conference 2009 (ISWC2009)*, Joel Sachs, Ed., 2009, Fake reference creata per gli stili dove le interviste non sono disponibili. Per qls stile che le supporti, usate la vera reference di tipo "interview".

[13] Maria Teresa Pazienza, Noemi Scarpato, Armando Stellato, and Andrea Turbati, "Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management," *Semantic Web Journal*, vol. 3, no. 2, 2012.

[14] Maria Teresa Pazienza, Armando Stellato, and Andrea Turbati, "PEARL: ProjEction of Annotations Rule Language, a Language for Projecting UIMA Annotations over RDF Knowledge Bases," in *International Conference on Language Resources and Evaluation*, Istanbul, Turkey, 2012.

[15] Ralph R Swick and Mark S. Ackerman, "The X Toolkit: More Bricks for Building User-Interfaces or Widgets for Hire," in *USENIX Winter*, Dallas, 1988, pp. 221-228.

[16] W3C. (2008) SPARQL Query Language for RDF. [Online]. http://www.w3.org/TR/rdf-sparql-query/

[17] W3C. (2009, August) World Wide Web Consortium (W3C). [Online]. http://www.w3.org/TR/skos-reference/

[18] W3C. (2009, August) World Wide Web Consortium (W3C). [Online]. http://www.w3.org/TR/skos-reference/skos-xl.html

[19] Ian Witten, Rob Akscyn, and Frank M. Shipman III, Eds., *DL '98: Proceedings of the third ACM conference on Digital libraries*. Pittsburgh, Pennsylvania, United States, Pennsylvania, United States: ACM, 1998.