INTELLIGENT SYSTEMS IN ACCOUNTING, FINANCE AND MANAGEMENT
Intell. Sys. Acc. Fin. Mgmt. **22**, 65–79 (2015)
Published online 27 January 2015 in Wiley Online Library (wileyonlinelibrary.com) DOI: 10.1002/isaf.1362

A FLEXIBLE APPROACH TO SEMANTIC ANNOTATION SYSTEMS FOR WEB CONTENT

MANUEL FIORELLI,[a*] MARIA TERESA PAZIENZA[b] AND ARMANDO STELLATO[b]
[a] *Department of Civil Engineering and Computer Science Engineering, University of Rome "Tor Vergata", Rome, Italy*
[b] *Department of Enterprise Engineering, University of Rome "Tor Vergata", Rome, Italy*

SUMMARY

In this paper we propose a flexible approach that supports heterogeneous requirements on systems for the semantic annotation of web content. The flexibility of the approach originates from a model based on the definition of abstract events, which captures at the logical level the main interactions occurring in a system for combined management of ontologies and web content. Application-specific semantics is then provided operationally as an assignment of handlers to these events. While the abstract events are rather coarse-grained to reduce prior commitment, preconditions on the handlers express application-specific distinctions based on contextual information associated with each specific event. Although the possibility to define completely new handlers guarantees the generality of our approach, we foster convention over configuration by providing a set of default handlers, which can be customized by filling their extension points. The use of customizable handlers, whether or not the default ones, reduces the development effort and guarantees consistent user experience despite evolving requirements. A comprehensive framework for semantic annotation of web content has been realized and will be hereafter introduced. Copyright © 2015 John Wiley & Sons, Ltd.

Keywords: Semantic Annotation; Semantic Web; Software Engineering

1. INTRODUCTION

In the Semantic Web (Berners-Lee *et al.*, 2001) the meaning of resources, possibly including services (Payne and Lassila, 2004), is captured through annotations with respect to well-defined ontologies. In fact, formalized knowledge is believed to allow software agents to better interact with web resources and perform intelligent tasks on behalf of humans, such as aggregating information from various sources and composing web services.

Beyond research on knowledge representation and automatic reasoning, the deployment of the Semantic Web required further investigation on pragmatic aspects related to the publication and the reuse of disparate knowledge on the Web. This line of development flowed into the Linked Open Data movement which elaborated a collection of best practices (Heath and Bizer, 2011) aimed at better connecting the Semantic Web to the architecture of the Web. Detractors criticized that Linked Open Data is nothing but a rebranding of the Semantic Web, perhaps aimed at revitalizing the interest on the field as a whole. We agree with the response to these concerns provided by Heath (2009), who stated: "Linked Data isn't about rebranding the Semantic Web, it's about clarifying its fundamentals".

* Correspondence to: Manuel Fiorelli, Department of Civil Engineering and Computer Science Engineering, University of Rome "Tor Vergata", Via del Politecnico 1, 00133 Rome, Italy. E-mail: fiorelli@info.uniroma2.it

The Linked Open Data principles apply uniformly to any kind of data, including statistics and spatial features, which are considered valuable on their own, regardless of their connection to a possibly unstructured resource, as in traditional metadata.

In fact, the interest on data in general produced a sound technological and methodological framework supporting the annotation of traditional information resources (documents, images, audio and video material). For example, the W3C introduced the SKOS vocabulary (Miles and Bechhofer, 2009) as a means to establish a link between the Linked Open Data cloud and the world of Knowledge Organization Systems (Hodge, 2000), historically employed by organizations and institutions to manage their assets.

So far, systems for annotating information content with respect to formal representations of knowledge have followed different (and occasionally contrasting) theories. These theories differentiated in many aspects:

- *the primary focus of the annotation* (e.g. is the traditional content that needs to be annotated with respect to a generic category, as a class in an ontology, or are specific ontological resources to be grounded on existing documentation?);
- *the granularity of the information to be reported*;
- *the nature of the annotated elements*.

Therefore, even the offer of Semantic Annotation applications is varied, and it is often difficult to see all of the requirements for a particular usage scenario satisfied by a single system.

To fulfil all previous matters, we propose a flexible approach to the development of different systems for combined management of ontologies and web content, including, but not limited to, Semantic Annotation Systems. This approach has been implemented as a subsystem of Semantic Turkey (Pazienza *et al*., 2012a), a fully fledged environment for knowledge management and acquisition based on Resource Description Framework (RDF) technologies (W3C, 2004), with a user interface deployed as a browser extension. Such an offer guarantees to end applications a high level of integration among browsing capabilities, ontology editing and cross-boundary features concerning both.

The rest of the paper is organized as follows. In Section 2 we discuss the role of annotation in different fields, and the variety of approaches to semantic annotation of web content. In Section 3 we motivate our approach to the development of semantic annotation systems. In Section 4 we describe the requirements we intend to satisfy. In Section 5 we introduce our approach, based on different levels of specification, which are detailed in Section 6 and Section 7. In Section 8 we describe the framework implementing the approach. In Section 9 we show how end-users may customize the systems. In Section 10 we draw the conclusions.

2. BACKGROUND

We can briefly state that an annotation establishes a link between two resources, by asserting that one is 'somewhat' about the other. The nature of this association is heavily domain and application dependent. For instance, informal free-text annotations are usually found as comments in a document to drive its edition, while structured annotations are the output of numerous natural language processing (NLP) tasks, including named entity recognition and relation extraction. These scenarios depend on different assumptions regarding the nature of the annotations, their granularity, their level of formality and the use, if any, of formal ontologies.

Annotation is thus a useful notion in a variety of fields, including NLP and Web Information Systems. Nonetheless, conflicting requirements demand different design decisions, which lead to completely different models and system implementations. This situation hampers the interaction between disciplines, which is in fact very important. As an example, the semantics of web resources is grounded ultimately in their natural language descriptions (Pazienza *et al*., 2007), while the Semantic Web could provide NLP algorithms with a vast background knowledge. For instance, DBpedia Spotlight (Mendes *et al*., 2011) combines language and semantic technologies in order to annotate mentions of DBpedia (Bizer *et al*., 2009) entities inside web pages, thus bridging together the Web of Documents and the Web of Data.

In the field of NLP, various analysis tasks are in fact annotation activities, in which structured annotations over the natural language prose make explicit task-related information. The understanding of natural language is a complex affair, which is usually decomposed into simpler and more manageable tasks that are performed in a pipeline, in which previous analysis results support later analysis tasks. The need of reusing and combining independently developed analysis components motivated the development of text engineering frameworks such as GATE (Cunningham, 2002) and, more recently, UIMA (Ferrucci and Lally, 2004). By adopting a data-driven architecture, these frameworks define the data structures for holding the subject of analysis and the annotations as they flow through the analysis pipeline. Furthermore, dedicated meta-modelling languages are provided to define the admitted annotation types, and dedicated tools are required to interface these frameworks with annotation formats used in publicly available corpora and shared tasks.

In fact, manual development of annotated corpora is important for the NLP community as well, since dominating empirical methods require ground-truth material for learning and evaluation. In the GATE family, Teamware (Bontcheva *et al*., 2013) satisfies this need by offering a web-based platform for creating and revising annotated corpora. It supports controlled workflows based on the separation of responsibilities among different team members.

In the field of web-based information systems, there emerged the need for a distributed approach to the annotation of web resources. Early works include Annotea (Kahan and Koivunen, 2001), which aimed at establishing a framework for the collaborative annotation of web resources. Initially thought for supporting the collaborative development of specifications within the W3C, the project aimed at establishing standards for textual annotations of marked-up documents.

Later initiatives within the bioinformatics community, Annotation Ontology (Ciccarese *et al*., 2011) and Open Annotation Collaboration (Sanderson and Van de Sompel, 2010), had a wider breadth, aimed at the annotation of any media type possibly with respect to a supplied ontology. Those projects flowed into the Open Annotation W3C community project, whose mission is to develop an RDF-based model for the annotation of digital artefacts. The annotation system Domeo (Ciccarese *et al*., 2012) supports the Annotation Ontology and it is expected to adopt the results of the novel W3C Community Group. With respect to early attempts, it is worth noticing that a shared data model is deemed sufficient, whereas dedicated protocols for querying and manipulating the annotations are no longer considered necessary, thanks to the availability of standards for performing such tasks developed in the meanwhile (e.g. SPARQL; Prud'hommeaux and Seaborne, 2008).

Despite the differences among the proposed models, the representation of an annotation in RDF minimally consists of a statement relating a resource (the target) to another (the body), which represents the desired attachment. In the case of Semantic Annotation, the attachment is part of a formally defined ontology. The choice of a domain/application ontology should reflect the particular point of view behind the annotation process. Ma *et al*. (2011) introduced a higher order semantics for capturing the meaning of semantic annotations with respect to the ontological nature of the attached resource and the property relating it to the target. They also showed how different levels of analysis (i.e. linguistic and semantic) could cooperate; for example, to suggest annotations or highlight possible errors.

Beyond the problems inherent to the representation of annotations, there is a need for a clear process to create and maintain them. According to Staab *et al*. (2000), this process should cope with the evolution of the domain ontology and the presence of mirrors or altered versions of the annotated resources.

The production of annotations by humans is often regarded as the bottleneck limiting the scale of the annotation process. Kiryakov *et al*. (2004) discussed the design issues related to a holistic system integrating semantic annotation, indexing and (semantically powered) retrieval. This vision was implemented by the platform KIM (Popov *et al*., 2003), which was heavily tested for the automatic annotation of news stories. These works re-engineered state-of-the-art NLP tools for automatically producing semantic annotations with respect to a lightweight upper ontology, called KIMO. The existence of a reference upper ontology, possibly extensible to address domain and application-specific needs, is a distinctive feature, while most works assume that semantic annotations are taken against any arbitrary domain ontology.

Finally, Uren *et al*. (2006) provided an overview of Semantic Annotation systems by comparing them based on a set of requirements that the authors consider key features for the annotation task.

3. MOTIVATION

In Section 2 we showed how different communities interested in producing and managing annotations have developed their own models and systems. In the following, we focus on semantic annotation of web content, which reveals specific features: heterogeneous content, use of formalized ontologies as providers of semantic descriptors, evolution of content and ontologies, alternative interpretations of the annotations. Even at this smaller scale we observe the same phenomenon: a varied demand produces an even more varied offer of approaches, models and systems.

We ascertain that a strong agreement on a universal data model for annotations is difficult to achieve: recent proposals cover a plethora of common usage scenarios, yet there are still corner cases—not well covered by these models—which might be very important to some communities.

Divergent methodologies have been proposed to support manual annotation rather than automatic generation of annotations. The latter can benefit, as shown by the KIM platform, from the reuse of state-of-the-art information extraction tools, which entails complex integration challenges.

These incompatible design decisions tend to cumulate, leading to very different system architectures and implementations. Therefore, pursuing the goal of realizing the ultimate annotation system appears to be fruitless. In this paper, we propose, indeed, a flexible approach that supports alternative designs as they emerge from conflicting requirements. Our approach is based on a very general model of a system for the combined management of ontologies and web content. While generality requires limited commitment on a priori decisions, a model unbound to any prior assumption makes no sense as well, because a model is always based on some grounding that characterizes its offer to the user.

As already stated, we narrow the scope of our contribution to Semantic Web annotation systems and, in general, any application combining ontological knowledge with web content. This is a general class of systems, which is not tightly bound to specific goals, interaction patterns, methodology (e.g. human work versus machine learning) or presentation mechanisms.

For what concerns our intended scope (RDF and web documents), RDF is by no means the only formalism to capture semantics; nevertheless, it is now widely spread and there are different W3C-recommended vocabularies supporting different modelling needs. The choice of supporting web documents is mostly a starting point (which does not contradict the generality of the approach), and future evolutions may foresee extensions for other kinds of sources, different in format or media type.

The development of a complete system requires filling specific extension points provided by the model, by committing to assumptions and use cases specific to the system being developed. We provide different levels of specification, which trade generality for ease of implementation. On top of a generic event-based behavioural model, we defined conventional interaction patterns and user-interface elements, as well as other reusable elements for further reduction of the development effort.

4. SYNTHESIS OF REQUIREMENTS

As previously introduced, our approach starts from the mere annotation acts, in order to support different models, methodologies and goals. Given our focus on combined management of ontologies and web content in general, in this section we state some requirements that shape more concretely the genre of systems we aim to support. To that purpose, we referred to a slightly extended version of the framework introduced by Uren *et al.* (2006) to compare different annotation systems. This framework provided us with the following features to discriminate between semantic annotation systems: standard formats, user-centred/collaborative design, ontology support, support of heterogeneous document formats, document evolution, annotation storage, automation and granularity.

For each dimension above, we positioned the class of systems we aim to support:

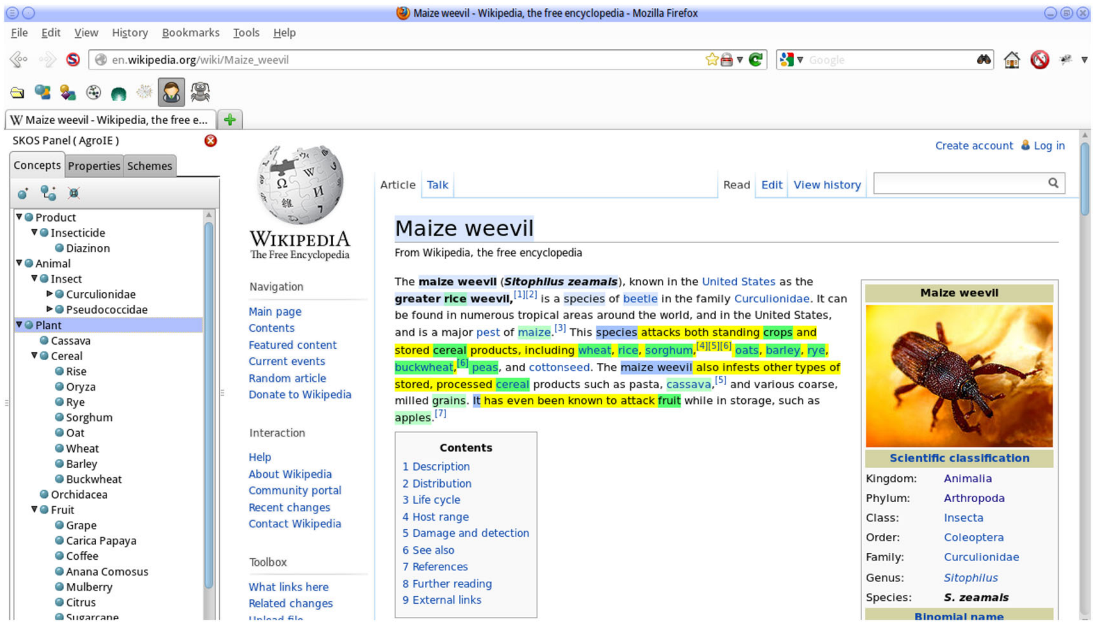
1. *Standard formats*. As we focus on semantic annotation, we depend on ontologies to provide semantic descriptors. To represent ontologies themselves, we require the use of the languages already developed inside the Semantic Web community (e.g. RDF(S), OWL and SKOS). We should support the usage of various annotation models. It should be possible to use different technologies to identify fragments of documents; for example, offset or XPointer-based (DeRose *et al.*, 2002) ranges.
2. *User-centred/collaborative design*. The user interface for ontology editing/annotation should be deployed as a web browser extension, while the browser itself hosts the web content. This combines the best of both web and desktop solutions, by providing at the same time an environment the user is well acquainted with (the browser) and extending it with annotation capabilities.
3. *Ontology support*. It should be possible to refer to any ontology, rather than committing to any given ontology (in contrast to KIM, which depends on the KIMO ontology).
4. *Support of heterogeneous document formats*. It is indeed a desirable feature, although the current implementation of our approach is tailored to web documents; however, this is a technological limitation of the current implementation and not a theoretical choice.
5. *Document evolution*. Different choices in the annotation format and in data preservation may be more or less prone to degradation with respect to the evolution of the annotated content. It should be possible to retain metadata about the target document to be able to detect changes. Option for XPointers guarantees better resilience to changes than plain offsets.
6. *Annotation storage*. As noted in Uren *et al.* (2006), there is no universally winning choice for storing the annotation content; our approach should thus allow annotations to be stored separately from the annotated resources (stand-off annotations), or to be embedded into them (in-line annotations).

7. *Automation*. Hosting of components for automatic annotation of content should be supported, as well as productive exploitation of their results and suitable interaction with the user for validating and refining these results.
8. *Granularity*. Annotations might refer both to entire documents or their fragments.

5. THE APPROACH

We distinguish two strategies for designing annotation systems, as in Kahan and Koivunen (2001): whether dedicated capabilities are injected into the browser or into the content provided by a proxy. Our research effort focuses on the first strategy, by relying on the extensibility of modern web browsers to develop the additional capabilities. In fact, we rely on a dedicated extension to enable ontology management within the browser.

In our usage scenario (see Figure 1), the traditional browser frame for visualizing the web content is complemented with a visual representation of the reference domain model (e.g. an OWL ontology or a SKOS concept scheme). In the specific case, owing to the web document content, the reference model is the AGROVOC (Caracciolo *et al*., 2013) thesaurus providing concepts about agriculture, nutrition and other areas of interest to the Food and Agriculture Organization (FAO) of the United Nations. Different colours are used to highlight mentions of insects, plants and the linguistic expression of the semantic relationship between an insect and a plant, such that the former is a pest of the latter.



- Product
  - Insecticide
    - Diazinon
- Animal
  - Insect
    - Curculionidae
    - Pseudococcidae
- Plant
  - Cassava
  - Cereal
    - Rise
    - Oryza
    - Rye
    - Sorghum
    - Oat
    - Wheat
    - Barley
    - Buckwheat
  - Orchidacea
  - Fruit
    - Grape
    - Carica Papaya
    - Coffee
    - Anana Comosus
    - Mulberry
    - Citrus
    - Sugarcane

Maize weevil

From Wikipedia, the free encyclopedia

The **maize weevil** (***Sitophilus zeamais***), known in the United States as the **greater rice weevil**,[1][2] is a species of beetle in the family Curculionidae. It can be found in numerous tropical areas around the world, and in the United States, and is a major pest of maize.[3] This species attacks both standing crops and stored cereal products, including wheat, rice, sorghum,[4][5][6] oats, barley, rye, buckwheat,[6] peas, and cottonseed. The maize weevil also infests other types of stored, processed cereal products such as pasta, cassava,[5] and various coarse, milled grains. It has even been known to attack fruit while in storage, such as apples.[7]

Contents

1 Description
2 Distribution
3 Life cycle
4 Host range
5 Damage and detection
6 See also
7 References
8 Further reading
9 External links

Maize weevil	
Scientific classification	
Kingdom:	Animalia
Phylum:	Arthropoda
Class:	Insecta
Order:	Coleoptera
Family:	Curculionidae
Genus:	*Sitophilus*
Species:	***S. zeamais***
Binomial name	

Figure 1. Overview of the Annotation Framework. The web page is annotated with concepts (insects, plants and pesticides) and relations (isPestOf) from the thesaurus AGROVOC

In this scenario, user interactions with the system fall into three main categories, with respect to the resources they affect:

1. only web content;
2. only ontological knowledge;
3. both.

The first category involves the interactions devoted to the navigation of the Web; for instance, activating a hyperlink to reach another web page. These interactions are completely managed by the hosting browser. The user might as well modify the domain model through interactions falling into the second category. Finally, there are interactions that encompass both realms; for instance, when the user drags a selection of text from a web page and drops it onto a resource, as common in most annotation systems.

Since the first two categories are in the scope of the hosting environment (i.e. the browser with additional ontology editing capabilities), we focused on supporting the interactions belonging to the third category.

This motivating scenario is, in fact, an instance of the more general class of systems for the combined management of ontological resources and web content, and not necessarily limited to semantic annotation. We propose a layered approach for the development of such systems, by providing different levels of specifications.

The proposed approach has been experimented with in its evolution, through the development of several concrete applications for semantic annotation (Fallucchi *et al*., 2008; Pazienza *et al*., 2009, 2012b). In Fiorelli *et al*. (2012) we dealt with the annotation of web content through concepts found in SKOS thesauri. These experiences helped us in understanding the features that a core approach for semantic annotation should exhibit and the right trade-off in flexibility, which should be granted to system developers, while still benefiting them with concrete support from the software.

We envision unlimited binding possibilities between annotated content fragments, their originating sources and the resources belonging to the domain model. This should allow, for instance, to generate new ontology individuals *while* annotating their occurrences within web pages, to create and annotate relationships between individuals, and so on.

Actually, all these possibilities can be understood in terms of a common behavioural model, based on the idea of assigning handlers to a predefined collection of events. These events are, to an extent, independent from the underlying presentation mechanism and the supporting technology, while representing what happens in a system at a more abstract level. Accordingly, when a bunch of text is dropped onto the hierarchical representation of the (ontological) domain model, we translate the low-level event generated by the user interface to some higher level event such as 'some text has been dropped onto a resource'. The assignment of suitable handlers to these events allows the implementation of an annotation system.

The generality of this model derives from the fact that it commits to a reduced number of assumptions about the structure and behaviour of the systems being developed. However, applications are rarely completely orthogonal, and in most cases they share part of their user interface, behaviour and data management. Our solution to this problem is twofold. First, we provide a library of reusable components for easing the development of handlers, such as user-interface elements and various content-handling functions. Moreover, we defined strategies to handle events in a conventional manner, according to use cases recurring in many scenarios. These default strategies are customized for specific application requirements, by filling their extension points. This further level of specification trades generality for ease of implementation and consistent user experience despite evolving requirements.

6. EVENT-BASED BEHAVIOURAL MODEL

We hereafter describe our general behavioural model (see Figure 2), based on the assignment of handlers to predefined events. In the forthcoming, we refer to a combination of an annotation model, events and related handlers as an *annotation family*, and by a slight abuse of language we will identify possible applications with distinct annotation families. In Section 8 we introduce our annotation families, which share most of the user interface and behaviour, although they differ in the mechanism for connecting the ontology and the web content: from pointers to specific fragments of the content, to a mere indication of the textual realization of the ontological element (regardless of its precise position).

Currently, we consider the following events:

- selectionOverResource
  fired when a selection from a web page is dropped onto an ontological resource;
- resourceOverContent
  fired upon gestures for the association of web content with an ontological resource regardless of its occurrence in the text;
- contentLoaded
  triggered when web content is loaded, in order to execute presentation-related activities (e.g. highlighting the annotated fragments).

This particular choice of events is motivated by its sufficiency and efficacy for implementing a semantic annotation system. The handlers for the events selectionOverResource and resourceOverContent encapsulate the logic for the creation of new annotations. Handling the event contentLoaded offers the opportunity to retrieve and properly visualize annotations for a web content (and to inject the code to manage them). For instance, operations such as the deletion of annotations can actually be invoked by code that is injected into the content by handlers intercepting the contentLoaded event, thus leaving the specification of these functions opaque to the framework.

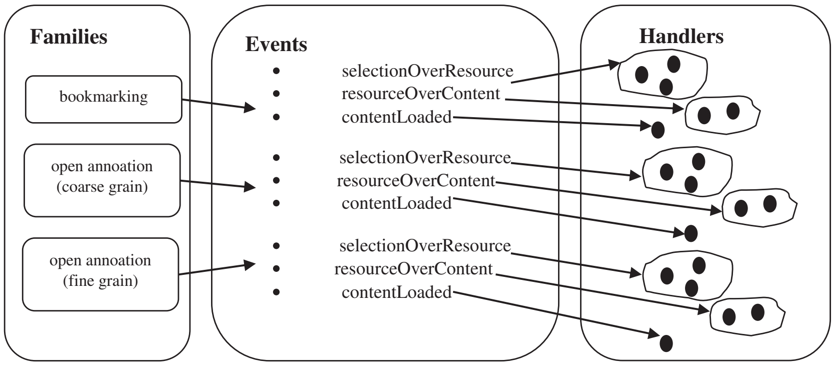


Figure 2. Event-based architecture

The model treats different genera of RDF resources (e.g. classes, individuals and properties) in a uniform manner, by declaring events concerned only with generic resources. *The uniform treatment of resources entails that the same event might be handled differently based on the target resource*. Moreover, applications might foresee the binding of multiple distinct handlers (see Figure 2) to an event related to a single resource, each handler implementing a distinct way for consuming that event. A handler can then be guarded by a *precondition*: a predicate over the event, such that the handler is executed, only if the predicate holds for the triggering event. More generally, preconditions allow filtering candidate handlers for an event based on any combination of contextual information; for example, the nature of the selected RDF resource, the selected content and the content source. Preconditions are actually redefinable functions, which can be plugged by extensions (see Section 8). The reference implementation of the model also provides some preconditions expressing basic checks; for example, with respect to the target resource type. Complex preconditions can be formed by composing existing ones through the usual logical connectives.

The preceding discussion might be more accessible through an example concerning the event selectionOverResource. As previously stated, this event is fired when a selection from a web page is dropped onto a resource. We can easily foresee multiple handlers for such an event, depending on specific application requirements. Most applications will need the ability to simply annotate an occurrence of that resource within the web page. This behaviour is well defined regardless of the specific resource type, although some applications may focus on the annotation of general categories (i.e. classes) rather than individuals. Other activities are valid, by definition, only on a subset of the events. For instance, when the target is a class, a handler might create and annotate an instance of that class based on the selected content; otherwise, if the target is an individual, we might want to set the value of one of its properties. Preconditions (e.g. based on the target resource type) allow filtering out irrelevant handlers, while preventing the user from being exposed to interactions that are undefined or otherwise violate some constraints of the given application. When contextual evidence is not sufficient to realize the user intent, there may be multiple candidate handlers for the same event. In such cases, the user is in charge of the ultimate choice.

7. CONVENTION OVER CONFIGURATION

On top of the event-based model, we provided a further level of specification, which consists of a collection of default handlers associated with any annotation family. This approach reduces the need for implementing completely new event handlers from scratch, while promoting the reuse of conventional interaction patterns. The binding of these default handlers to a specific application occurs by filling some extension points:

- checkAnnotationsForContent(contentID)
  Checks whether a given content source has been annotated. By default, this function is invoked by a predefined handler for the event contentLoaded.
- getAnnotationsForContent(contentID)
  Returns the annotations taken over a specific content source. Actually, it returns proxies for the annotations (which depend on the model) exposing some mandatory fields, such as the id and range of the annotations. The implementation/serialization of these annotation elements is left to the specific family, and must be *consistent* with the other services implemented in the family. This function is automatically invoked after a positive (returned value = true) check performed by the previous function (in the context of a contentLoaded event).

- getAnnotationsForResource(RDFResource)
  Analogous to the previous one, this function retrieves all annotations associated with a given RDF resource. This function allows including in the description of a resource a list of actionable links to annotated content sources.
- decorateContent(annotations)
  This is a client function for injecting elements inside the content, usually to show the annotations that have been previously taken over it. A standard text highlighting mechanism for web documents is provided by the system and invoked on the result of a getAnnotationsForContent(), in the context of a contentLoaded event. This mechanism can be overridden by implementing this function with custom content decorators.
- deleteAnnotation(annotID)
  This function takes care of removing all the information related to a given annotation. The *standard highlighter* injects calls to this function for each annotation shown on the web document.

8. IMPLEMENTATION OF THE APPROACH

The approach we present has been implemented as a software framework embedded in the knowledge management and acquisition platform Semantic Turkey (Pazienza *et al.*, 2012a). The fact that Semantic Turkey can be deployed as a browser extension matches our expectation of an environment that supports both ontology management and web browsing. Moreover, Semantic Turkey supports the management of a large amount of RDF data, and offers various mechanisms for loading extensions. These mechanisms will support the deployment of concrete applications built on top of our framework. In the following paragraphs we provide the necessary information on Semantic Turkey to understand how we implemented our approach. However, further details can be found in the reference paper on Semantic Turkey.

Semantic Turkey has a layered architecture (see Figure 3), composed of a presentation layer, a service layer and a data management layer. The upper layer is developed as an extension of the Mozilla Firefox web browser, while the layers beneath are developed as a stand-alone server deployed in Karaf (Edstrom *et al.*, 2013) OSGi runtime. The communication between the presentation and service layers is carried through lightweight HTTP interactions.

The framework implementing our approach is mostly located in the presentation layer, since its main responsibility is the translation of concrete user-interface events to the abstract events defined by our behavioural model. In fact, this translation represents the grounding of the behavioural model in the hosting environment. For the rest, the logic framework is mostly independent from the hosting environment, in which the model has been instantiated.

The extensibility of Semantic Turkey allows the deployment of new annotation families, the enrichment of existing ones by the addition of further handlers and the provision of new preconditions. At the very minimum, a further browser extension is required, depending on Semantic Turkey, which interacts with appropriate registries provided by the annotation framework. However, in most cases an extension to the service layer is required as well.

The browser and the server beneath offer to the implementers a wide choice of reusable capabilities for different aspects of an annotation family. The browser provides technologies for the definition of user interfaces, the manipulation of information resources and the interaction with the Web. An annotation family might exploit them to embed the annotations into a web page, which can then be saved in an updated copy. An annotation family may depend on core services provided by Semantic Turkey as

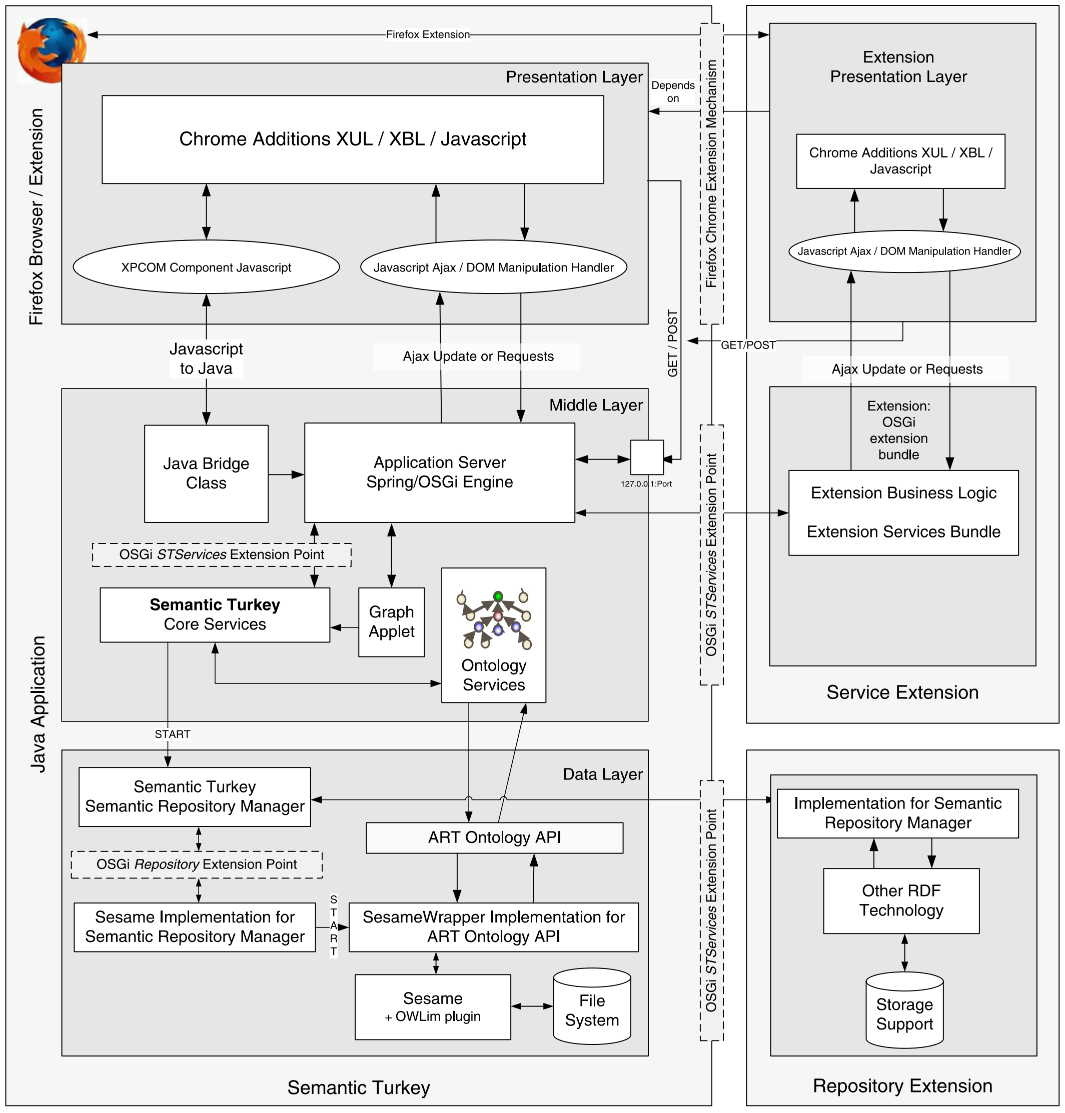

Firefox Extension
Firefox Browser / Extension
Presentation Layer
Chrome Additions XUL / XBL / Javascript
XPCOM Component Javascript
Javascript Ajax / DOM Manipulation Handler
Javascript to Java
Ajax Update or Requests
Depends on
GET / POST
Firefox Chrome Extension Mechanism
Extension Presentation Layer
Chrome Additions XUL / XBL / Javascript
Javascript Ajax / DOM Manipulation Handler
GET/POST
Ajax Update or Requests
Middle Layer
Java Bridge Class
Application Server Spring/OSGi Engine
127.0.0.1:Port
OSGi STServices Extension Point
Semantic Turkey Core Services
Graph Applet
Ontology Services
Java Application
Extension: OSGi extension bundle
Extension Business Logic
Extension Services Bundle
Service Extension
START
Data Layer
Semantic Turkey Semantic Repository Manager
ART Ontology API
OSGi Repository Extension Point
Sesame Implementation for Semantic Repository Manager
START
SesameWrapper Implementation for ART Ontology API
Sesame + OWLim plugin
File System
Semantic Turkey
Implementation for Semantic Repository Manager
Other RDF Technology
Storage Support
Repository Extension

Figure 3. The architecture of Semantic Turkey

well as define new ones for dealing with the specifics of its annotation mechanism. In fact, additional capabilities for the lower layers may be provided by deploying additional OSGi bundles in the Karaf container hosting Semantic Turkey. This extension mechanism offers the unlimited possibility of providing additional services for a given annotation family, for meeting requirements such as dedicated export mechanisms and ontology evolution management. Actually, such extended functionalities would grow on top of other services already provided by Semantic Turkey as a platform for the management of semantic content.

Our implementation comes out-the-box with a few annotation families, which differ in the underlying annotation model and, notably, in the tasks they support. The default families take into consideration the *annotation of atomic ontological resources* and complex activities that are provided as *macros*:

- the creation of new instances;
- the definition of new subclasses in OWL;
- the definition of narrower concepts in SKOS;
- set the value of a property.

The description of resources in the domain model is enriched with references to the annotated content, while suitable icons and highlighted spans show existing annotations on the content of the browser frame. This infrastructure supports concept-based browsing of the Web as well as understanding how the domain model is grounded in specific web content.

The default behaviour of the annotation framework is to store annotations as further RDF metadata inside the RDF repository of the active project. These stand-off annotations retain a connection to the source content in the form of some location metadata, comprising the URL of the source document and (if necessary) a reference to the specific annotated fragment. Existing functionalities of Semantic Turkey support advanced interaction with the annotations, since they are part of the data managed by Semantic Turkey. As an example, the embedded SPARQL client allows complex analysis and manipulation of the annotations.

Semantic Turkey supports different strategies for deploying the RDF triple store: from a locally managed store, to a remote store exposed through some remote API. The proposed annotation framework is agnostic with respect to the deployment of the RDF repository; still, in a certain sense, a shared remote repository might support a limited form of collaborative annotation. Additionally, an extension might provide further services related to collaboration, such as task assignment and computation of inter-annotator agreement.

9. END-USER CUSTOMIZABILITY

In Section 8 we introduced a software framework implementing our approach to the design of semantic annotation systems. Developers might construct a new application on the framework either as an additional annotation family or as a customization of existing ones with new event handlers.

We suggest developing annotation families that are as general as possible, by avoiding early commitment to assumptions that are not strictly necessary. As an example, an annotation family should support the annotation within a web page of any ontological resource, without distinguishing among individuals, classes and properties. The resulting annotation families are more flexible, as they can handle different application scenarios.

However, most usage scenarios are more constrained than these general-purpose annotation families. Consequently, end-users should be careful not to select options that violate the constraints on the specific task they are involved in.

We solve this issue by allowing end-users (i.e. human annotators) to tailor existing annotation families to their needs. Concretely, a user can customize a family (see Figure 4) by enabling only a portion of the annotation functionalities associated with each event, or by refining the preconditions of its associated handlers. Users are normally prompted with the list of suitable handlers (obviously, well presented through appropriated descriptors) after they trigger an event as a consequence of performing an action; as an automatic shortcut, when such a list reduces to a single handler, it is executed without prompting the user.

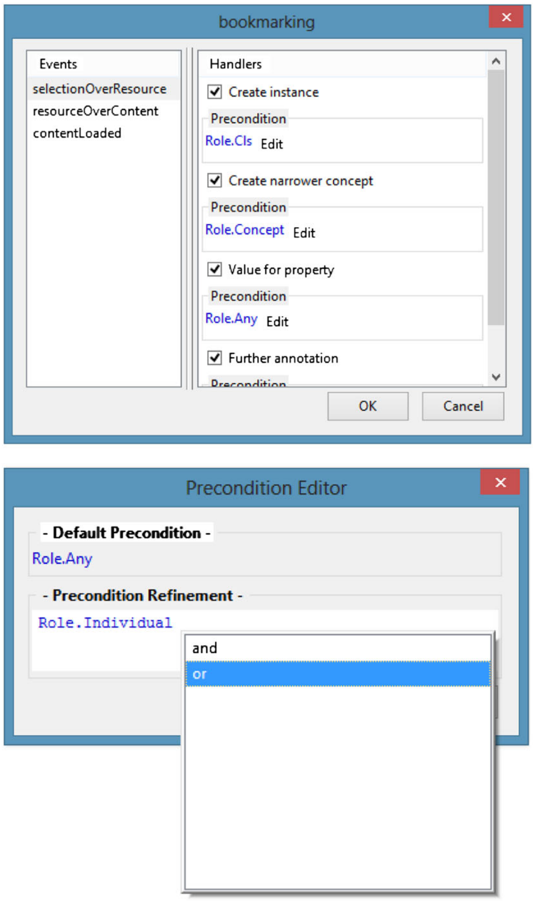


Figure 4. End-user customization: handlers are enabled for a given event and can then be filtered—by editing their preconditions—when that event is fired

This customization of an existing annotation family benefits the end-user with a less cluttered user interface, since irrelevant operations are hidden. Moreover, this refinement of an annotation family allows precisely meeting the constraints of each specific task. For instance, while there is nothing in principle preventing us from annotating any ontological element, it may be the case that a specific scenario only allows the annotations of classes (e.g. occurrences of locations, organizations, people) rather than individuals (e.g. a specific location, a specific organization, a specific person). Although our general-purpose annotation families are far more liberal, it is possible to handle this stricter requirement by strengthening the precondition on the appropriate event handler (i.e. limiting its applicability only to classes).

10. CONCLUSIONS

In this paper we propose an approach to the semantic annotation of web content that attempts to balance effective support to developers with generality. We achieve this goal by providing different levels of specification, which trade generality for convenience. The more comprehensive level consists of a behavioural model based on event handling, which is mostly independent from specific interaction patterns, goals or models. To reduce development effort and promote a consistent user experience, we provide default handlers, which support recurring use cases. Finer-grain extension points allow binding these handlers to the specific application requirements. Furthermore, we encourage the development of general event handlers that commit to as little as possible prior assumptions. Indeed, end-users may lately customize these general-purpose handlers to meet specific application requirements.

While the approach seems to us general enough in its basic assumptions, we want to increment the set of available conventions and create template libraries for recurring annotation patterns. In Pazienza *et al*. (2012b), we developed an acquisition workflow for the enrichment of the AGROVOC (Caracciolo *et al*., 2013) thesaurus, which combines text analytics and proper human interaction. We plan to generalize this workflow by combining the proposed framework with our platform for knowledge acquisition (Fiorelli *et al*., 2014).

ACKNOWLEDGEMENTS

This research has been partially supported by the EU-funded project SemaGrow (grant agreement no.: 318497).

REFERENCES

Berners-Lee T, Hendler J, Lassila O. 2001. The Semantic Web: a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American* **284**(5): 28–37.

Bizer C, Lehmann J, Kobilarov G, Auer S, Becker C, Cyganiak R, Hellmann S. 2009. DBpedia—a crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web* **7**(3): 154–165. doi: 10.1016/j.websem.2009.07.002

Bontcheva K, Cunningham H, Roberts I, Roberts A, Tablan V, Aswani N, Gorrell G. 2013. GATE Teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation* **47**(4): 1007–1029. doi: 10.1007/s10579-013-9215-6

Caracciolo C, Stellato A, Morshed A, Johannsen G, Rajbhandari S, Jaques Y, Keizer J. 2013. The AGROVOC linked dataset. *Semantic Web Journal* **4**(3): 341–348. doi: 10.3233/SW-130106

Ciccarese P, Ocana M, Garcia Castro LJ, Das S, Clark T. 2011. An open annotation ontology for science on Web 3.0. *Journal of Biomedical Semantics* **2**(Suppl 2): S4. doi: 10.1186/2041-1480-2-S2-S4

Ciccarese P, Ocana M, Clark T. 2012. Open semantic annotation of scientific publications using DOMEO. *Journal of Biomedical Semantics* **3**(Suppl 1): S1. doi: 10.1186/2041-1480-3-S1-S1

Cunningham H. 2002. GATE, a General Architecture for Text Engineering. *Computers and the Humanities* **36**(2): 223–254. doi: 10.1023/A:1014348124664

DeRose S, Daniel RJ, Grosso P, Maler E, Marsh J, Walsh N. 2002. XML Pointer Language (XPointer). http://www.w3.org/TR/xptr/ (accessed 22 December 2014).

Edstrom J, Goodyear J, Kesler H. 2013. *Learning Apache Karaf*. Packt Publishing: Birmingham, UK.

Fallucchi F, Pazienza MT, Scarpato N, Stellato A, Fusco L, Guidetti V. 2008. Semantic bookmarking and search in the Earth observation. In *Knowledge-Based Intelligent Information and Engineering Systems. 12th International Conference, KES 2008, Zagreb, Croatia, September 3–5, 2008, Proceedings, Part III*, Lovrek I, Howlett RJ, Jain LC (eds). Lecture Notes in Computer Science, vol. **5179**. Springer: Berlin Heidelberg. doi: 10.1007/978-3-540-85567-5_33

Ferrucci D, Lally A. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering* **10**(3–4): 327–348. doi: 10.1017/S1351324904003523

Fiorelli M, Pazienza MT, Stellato A. 2012. Semantic Turkey goes SKOS managing knowledge organization systems. In *I-SEMANTICS '12 Proceedings of the 8th International Conference on Semantic Systems*. ACM: New York; 64–71. doi: 10.1145/2362499.2362509

Fiorelli M, Pazienza MT, Stellato A, Turbati A. 2014. CODA: Computer-aided Ontology Development Architecture. *IBM Journal of Research and Development* **58**(2–3): 14:1–14:12. doi: 10.1147/JRD.2014.2307518

Heath T. 2009. Linked Data? Web of Data? Semantic Web? WTF? http://tomheath.com/blog/2009/03/linked-data-web-of-data-semantic-web-wtf/ (accessed 22 December 2014).

Heath T, Bizer C. 2011. Linked data: evolving the Web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology* **1**(1): 1–136. doi: 10.2200/S00334ED1V01Y201102WBE001

Hodge G. 2000. *Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files*. Council on Library and Information Resources: Washington, DC.

Kahan J, Koivunen M-R. 2001. Annotea: an open RDF infrastructure for shared web annotations. In *WWW '01: Proceedings of the 10th International Conference on World Wide Web*. ACM: New York; 623–632. doi: 10.1145/371920.372166

Kiryakov A, Popov B, Terziev I, Manov D, Ognyanoff D. 2004. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web* **2**(1): 49–79. doi: 10.1016/j.websem.2004.07.005

Ma Y, Lévy F, Ghimire S. 2011. Reasoning with annotations of texts. *In 24th International FLAIRS Conference* (*FLAIRS'11*): *Track AI*, *Cognitive Semantics*, *Computational Linguistics and Logics*. AAAI Press: Menlo Park, California; 192–197.

Mendes PN, Jakob M, García-Silva A, Bizer C. 2011. DBpedia spotlight: shedding light on the Web of Documents. In I-Semantics '11 *Proceedings of the 7th International Conference on Semantic Systems*. ACM: New York. doi: 10.1145/2063518.2063519

Miles A, Bechhofer S. 2009. SKOS Simple Knowledge Organization System Reference. *W3C Recommendation*. W3C. http://www.w3.org/TR/2009/REC-skos-reference-20090818/ (accessed 22 December 2014).

Payne TR, Lassila O. 2004. Semantic web services. *IEEE Intelligent Systems* **19**(4): 14–15. doi: 10.1109/MIS.2004.29

Pazienza MT, Sguera S, Stellato A. 2007. Let's talk about our 'being': a linguistic-based ontology framework for coordinating agents. *Applied Ontology* **2**(3–4): 305–332.

Pazienza MT, Scarpato N, Stellato A. 2009. STIA: experience of semantic annotation in jurisprudence domain. In *Proceeding of the 2009 Conference on Legal Knowledge and Information Systems*. IOS Press: Amsterdam, The Netherlands.

Pazienza MT, Scarpato N, Stellato A, Turbati A. 2012a. Semantic Turkey: a browser-integrated environment for knowledge acquisition and management. *Semantic Web Journal* **3**(3): 279–292. doi: 10.3233/SW-2011-0033.

Pazienza MT, Stellato A, Tudorache AG, Turbati A, Vagnoni F. 2012b. An architecture for data and knowledge acquisition for the Semantic Web: the AGROVOC use case. In *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, Herrero P, Panetto H, Meersman R, Dillon T (eds). Lecture Notes in Computer Science vol. **7567**. Springer: Berlin Heidelberg; 426–433. doi: 10.1007/978-3-642-33618-8_58

Popov B, Kiryakov A, Kirilov A, Manov D, Ognyanoff D, Goranov M. 2003. KIM—semantic annotation platform. In *Proceedings of the Second International Semantic Web Conference (ISWC2003)*, Fensel D, Sycara K, Mylopoulos J (eds). Lecture Notes in Computer Science, vol. **2870**. Springer: Berlin Heidelberg; 834–849.

Prud'hommeaux E, Seaborne A, 2008. *SPARQL Query Language for RDF.* W3C. http://www.w3.org/TR/rdf-sparql-query/ (accessed 22 December 2014).

Sanderson R, Van de Sompel H. 2010. Making web annotations persistent over time. In *Proceedings of the 10th Annual Joint Conference on Digital Libraries*, ACM: New York; 1–10. doi: 10.1145/1816123.1816125

Staab S, Maedche A, Handschuh S, 2000. Creating metadata for the Semantic Web—an annotation environment and the human factor. Tech. Rep. Institute AIFB.

Uren V, Cimiano P, Iria J, Hanndschuh S, Vargas-Vera M, Motta E, Ciravegna F. 2006. Semantic annotation for knowledge management: requirements and a survey of the state of the art. *Journal of Web Semantics* **4**(1): 14–28. doi: 10.1016/j.websem.2005.10.002

W3C. 2004. Resource Description Framework (RDF). http://www.w3.org/RDF/ (accessed 22 December 2014).