# Maskkot – An Entity-centric Annotation Platform

**Armando Stellato[1], Heiko Stoermer[2], Stefano Bortoli[2], Noemi Scarpato[1], Andrea Turbati[1], Paolo Bouquet[2], Maria Teresa Pazienza[1]**

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

[2] University of Trento
Department of Engineering and Information Science
Trento, Italy
{bortoli, bouquet, stoermer}@disi.unitn.it

**Abstract**

The Semantic Web is facing the important challenge to maintain its promise of a real world-wide graph of interconnected resources. Unfortunately, while URIs almost guarantee a direct reference to entities, the relation between the two is not bijective. Many different URI references to same concepts and entities can arise when -- in such a heterogeneous setting as the WWW -- people independently build new ontologies, or populate shared ones with new arbitrarily identified individuals.
The proliferation of URIs is an unwanted, though natural effect strictly bound to the same principles which characterize the Semantic Web; reducing this phenomenon will improve the recall of Semantic Search engines, which could rely on explicit links between heterogeneous information sources.
To address this problem, in this paper we present an integrated environment combining the semantic annotation and ontology building features available in the Semantic Turkey web browser extension, with globally unique identifiers for entities provided by the okkam Entity Name System, thus realizing a valuable resource for preventing diffusion of multiple URIs on the (Semantic) Web.

## 1. Introduction

Whichever name it will assume, be it Web 3.0, Giant Global Graph, or any other, the Semantic Web will have to face an important challenge to maintain its promise of a real world-wide graph of interconnected resources: their identity and retrieval. If entities are uniquely identified in the Web, then anyone can make statements about them, thus incrementing their intensional description and contributing to their success and retrievability.
Unfortunately, while URIs almost guarantee a direct reference to entities, the relation between the two is not bijective. Many different URI references to same concepts and entities can easily arise when, in such a heterogeneous setting as the WWW, people independently build new ontologies, or populate shared ones with new arbitrarily named individuals.
In this work we present an integrated framework combining the semantic annotation and ontology building features provided by the Semantic Turkey web browser extension, with global, unique identifiers for entities provided by the OKKAM Entity Name System (ENS). The integration has been carried out through the development of a dedicated ENS extension for Semantic Turkey – maskkot -- which extends its ordinary ontology building functionalities with entity search over the okkam service. Through maskkot, users can create, extend and/or populate ontologies with individuals, while maintaining reference to their okkam unique identifiers, giving life to a virtuos cycle in which they may contribute and/or get additional information from the ENS-empowered semantic search engine, and at the same time fighting a proliferation of identifiers for the same entity.

It is a commonly accepted fact that whenever a computer system needs to describe an object (or ``entity'' as we call it), it needs to create some kind of identifier in the system which is then regarded as the placeholder or proxy for this object. This holds true for e.g. database systems, but is of special significance for the Semantic Web, where the notion of the Uniform Resource Identifier (URI) fulfills exactly this task, but has the additional goal of linking information about entities in a distributed but global fashion. In this way, distributed information sources are supposed to become integrateable on the fly, to create links between pieces of information that were previously not linked before, enabling systems to answer queries that were previously impossible to answer.
The Semantic Web in its current state suffers from several weaknesses which we are trying to address in this paper:

- A lack for convenient, user-friendly tools for semantic annotation of Web content. While solutions exist and are described in more detail in section 2, we believe that means for semantic annotation should be given as pervasively as possible, and should be (almost) as easy as creating a bookmark.
- A proliferation of URIs for entities. As we have argued in (Bouquet, Stoermer, & Bazzanella, 2008), to date no scalable and open service is available to make possible and to support a *consistent reuse* of identifiers for entities, and this undermines the practical possibility of a seamless integration of distributed knowledge into a global knowledge space.

The work presented in this paper attempts to contribute to the state of the art in the Semantic Web on several levels: firstly, by providing an intuitive tool for the creation of semantic content; secondly, by making sure that the semantic annotations created are also globally aligned on

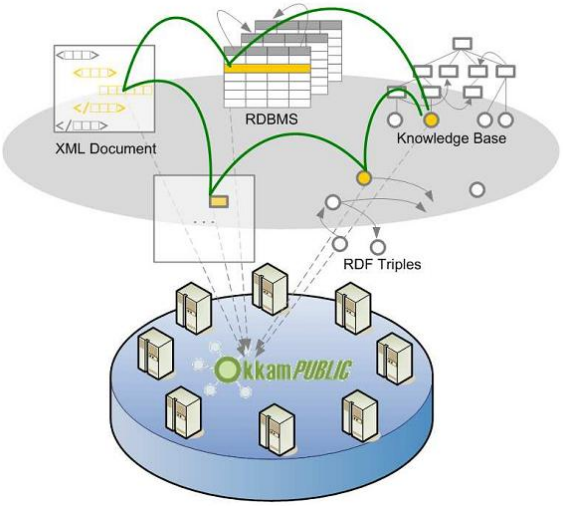


Figure 1: Entities in different information sources and formats, annotated with unique identifiers issued by the okkam ENS

the identifiers for entities, enabling seamless, syntactical integration of data without the need for complex ex-post alignment mechanisms; and thirdly, by contributing to an ever-expanding public space of entity identifiers which offers significant positive network externality effects[1]

## 2. Related Work

The maskkot integrated platform is rather original in its combination of ontology editing/annotation/semantic browsing functionalities supported by an entity identification service. We therefore report here relevant past works related to the most relevant features characterizing the presented tool.

### 2.1. Semantic Browsing and Semantic/Social Bookmarking/Annotation

One of the first examples of Semantic Browser can be probably traced back to the Haystack (Quan & Karger, May, 2004) web client. Developed at the MIT laboratories, Haystack was conceived as an application that could be used to browse arbitrary Semantic Web information in much the same fashion as a Web browser can be used to navigate the Web. Standard point-and-click semantics let the user navigate over aggregation of RDF repositories from different arbitrary locations. The application had been built as an extension for the popular Integrated Development Environment Eclipse[2] this choice facilitated extension of the tool thanks to Eclipse flexible plug-in mechanism, but required the user to adopt Eclipse as a platform for browsing the web and collecting data from it: a strong requirement for the user, who would just prefer to rely on his trusted personal web browser and try out other features which are not too invasive for his usual way of working.

Such a less invasive approach was followed by Magpie (Dzbor, Domingue, & Motta, 2003), that was deployed as a plug-in for the Microsoft Internet Explorer Web Browser. Magpie allowed for semantic browsing, and perceived it as a parallel navigational style to complement the "exposed" web content (i.e. free text) by an associated, dynamic semantic layer (which was derived from one or more ontologies semantically describing typical content in a particular domain). Magpie also allows for collaborative semantic web browsing, in that different persons may gather information from the same web resource and exchange it on the basis of a common ontology. Subsequent work on Magpie (Dzbor, Motta, & Domingue, 2004) extended the platform more and more towards the vision of the Semantic Web as "an open web of interoperable applications" (Berners-Lee, Hendler, & Lassila, 2001), by allowing bi-directional exchange of information among users and services, which can be opportunistically located and composed, both manually (web services) or automatically (semantic web services).

From (part of) the same authors of Haystack, comes Piggy-Bank (Huynh, Mazzocchi, & Karger, 2005), an extension for the Firefox web browser that lets Web users extract individual information items from within web pages and save them in RDF, replete with metadata. Piggy Bank then lets users make use of these items right inside the same web browser. These items, collected from different sites, can then be browsed, searched, sorted, and organized, regardless of their origins and types. Piggy-Bank users may also rely on Semantic Bank, a web server application that lets them share the Semantic Web information they have collected, enabling, as for Magpie, collaborative efforts to build sophisticated Semantic Web information repositories from daily navigation through their enhanced web browser.

### 2.2. Identity and Reference

When attempting to give an identity to ``things" in a way that makes them describable in the Semantic Web (i.e. choosing or creating a URI as a placeholder for them), we can encounter three different approaches:

#### 2.2.1. Local Identification

This is unfortunately -- as mentioned in the introduction -- the common practice at the moment: new URIs for entities are created on the fly, because they are regarded as a mere technical necessity to be able to make RDF statements. Such identifiers do not consider a scope that goes beyond the local knowledge base, and even the Semantic Web community itself has been following this practice, e.g. in the case of authors of Semantic Web conferences as we have shown in (Bouquet, Stoermer, & Bazzanella, 2008).

#### 2.2.2. Vertical Identification

Vertical approaches usually refer to a certain domain of interest, for which an organization is issueing identifiers for entities. Examples include publications (DOI[3]), geographical locations (Geonames[4] or Yahoo! Internet Locations[5]), life science entities(LSID[6]), and many more. The issue with these vertical approaches is the findability of the identifiers: if we are creating RDF statements about entities from many different domains, how do we find out

[1] See (Liebowitz & Margolis, 1998) or http://en.wikipedia.org/wiki/Network_effect for an introduction.
[2] http://www.eclipse.org/
[3] http://www.doi.org
[4] http://www.geonames.org
[5] http://developer.yahoo.com/geo/
[6] http://lsids.sourceforge.net/

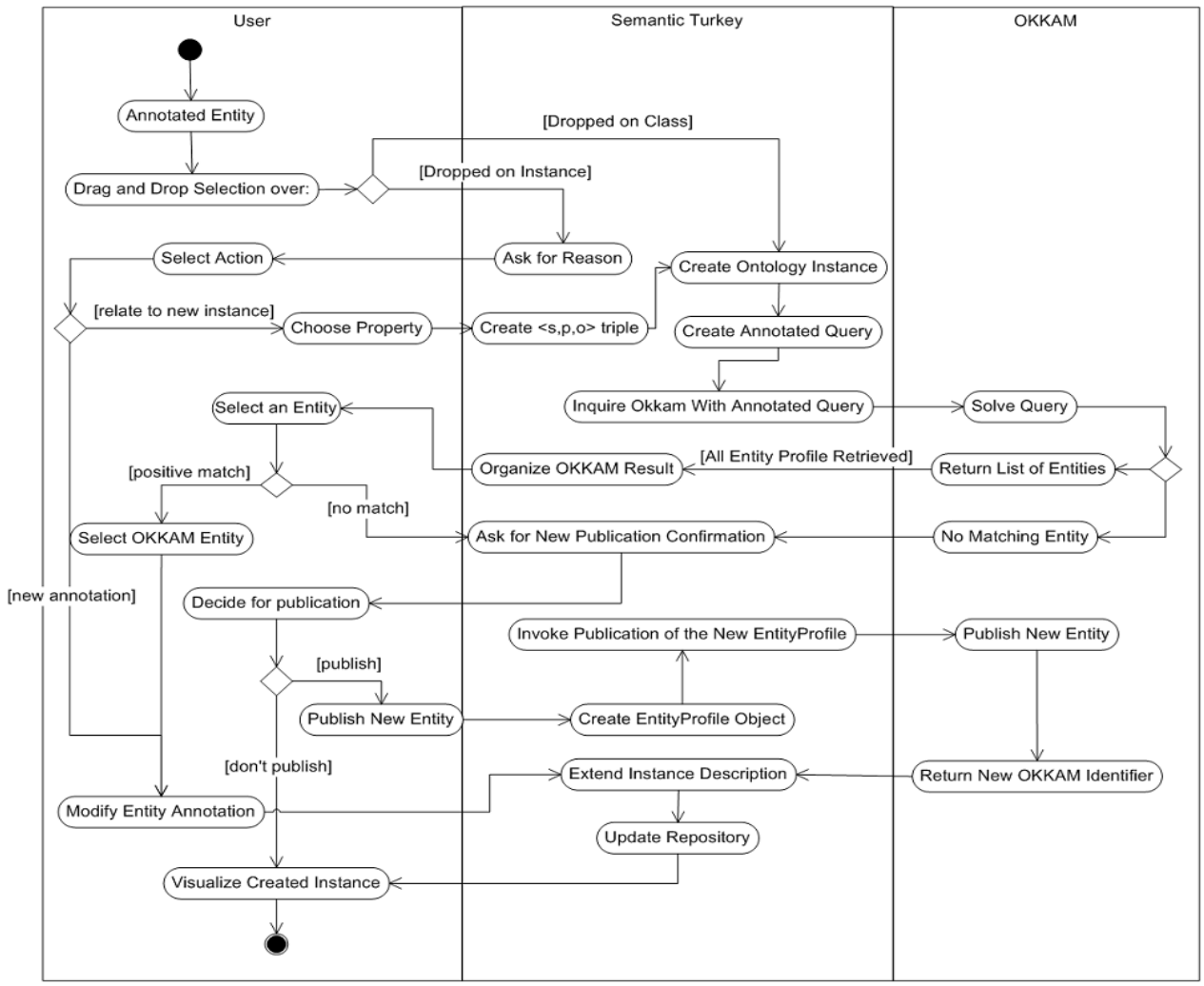


Figure 2: Entity-centric Annotation Activity Diagram

which is the source of an identifier for an entity, and how do we make sure that we chose ``the right'' (i.e. authoritative) one?

#### 2.2.3. Global Identification

Global identification in our sense is a horizontal approach as an attempt to overcome the issues of both the local and the vertical approach. Currently, there are two main streams of activity in the Semantic Web which can be considered relevant in this respect. The first one is the Linking Open Data initiative[7], which pursues an ex-post approach trying basically to discover identity relations between entities that have been given different identifiers in different knowledge bases, but are actually (believed to be) ``the same''. As we have discussed in (Bouquet, Stoermer, Cordioli, & Tummarello, 2008), this approach is viable due to the simple fact that such un-aligned data sources exist, but it has the obvious downside that it does not provide a solution for *avoiding* such a proliferation of identifiers. An orthogonal approach to address all these issues are the efforts around the okkam Entity Name System (ENS), which is described in further detail in section 3.1.

## 3. Towards an Entity-centric Semantic Annotation Platform

### 3.1. The Entity Name System (ENS)

The key idea behind the proposal of an ENS is that the Semantic Web can become an open and scalable space for publishing knowledge (in the form of RDF data) only if there will be a reliable (and trustworthy) support for the reuse of URIs. Therefore, at a very general level, the core functionality of the ENS can be characterized as follows: given any representation of an entity (e.g. a bag of keywords, a paragraph of text, a collection of key-value pairs, a graphical depiction, and so on), decide if a URI for this entity is already available in an entity repository (using some method(s) for *entity matching*); if it is, then the ENS will return its URI (or at least a ranked list of candidates), otherwise it will issue a new URI which will be stored in the ENS repository.

As we have argued in (Bouquet, Stoermer, & Bazzanella, An Entity Naming System for the Semantic Web, 2008), issues of entity identification are optimally solved a-priori, across data sources and formats. Instead of creating

[7] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

RDF repositories in which the same real-world entity is denoted by two or more different URIs, and then trying to reconcile these URIs, we should aim at enabling any application which produces RDF content to reuse a globally unique URI for that resource from the outset, possibly even beyond the Semantic Web data space, as illustrated in Figure 1.

The positive effects are evident. Instead of using one of the many possible *names* for an entity[8] a uniform electronic *surrogate* is used. The local effect within a single system is that ambiguities of references to entities in metadata can be eliminated to the largest part already at creation time. The global effect is that:

i. information integration is largely reduced to schema level integration, as entity identifiers provide large parts of data-level integration for free (besides dealing with conflicting and redundant data in different collections), and
ii. completely new hyper-structures are possible that link between different entities and between artifacts and entities via the shared entity identifiers.

Optimally, such a global identifier for every entity referenced in a data source is used throughout *all* records/terms/statements that refer to this entity, in every data source referring to this entity, and in (external) content such as websites or other documents (Figure 1). This leads to the possibility to relate and integrate -- without additional efforts -- textual and multimedial content referring to a specific entity. This becomes more and more relevant taking into consideration the fast pace of development in multimedia libraries, as can be seen in current services such as YouTube[9] or Flickr[10].

Okkam is an implementation of an Entity Name System (ENS), which is currently under development in a large-scale European project. The aim of this system is to provide a more complete set of distributed ENS functionality, an adaptive matching layer, and vastly improved storage.

The standard use-case -- further illustrated in Figure 2 -- for assigning an okkam identifier to an entity that is being annotated in any kind of content, such as an OWL/RDF ontology, an XML file, or a database, to make the entity globally identifiable: querying okkam for the existance of the entity at hand, and re-using the global identifier for this entity. This is usually achieved through functionality provided by a client application -- e.g. the one presented in this paper, or others like Foaf-O-Matic (Bortoli, Stoermer, & Bouquet, 2007) or okkam4p (Bouquet, Stoermer, & Xin, Okkam4P - A Protégé Plugin for Supporting the Re-use of Globally Unique Identifiers for Individuals in OWL/RDF Knowledge Bases., 2007). Such a client application accesses the okkam API, and presents (if available) a list of top candidates which match the description for the entity provided within the client application. If the entity is among these candidates, the client agent (human or software) uses the associated okkam identifier in the respective information object(s) *instead* of a local identifier. If the entity cannot be found, the client application can create a new entry for this entity in okkam and thus cause an identifier for the entity to be issued and used as described before. The okkam ENS implements a variety of methods for entity matching, typically achieving very good recall and precision values( (Stoermer, Rassadko, & Vaidya, 2010), (Ioannou, et al., 2009).

## 3.2. Semantic Turkey

Semantic Turkey (ST), in its original version (Griesi, Pazienza, & Stellato, 2007), is a "Semantic Bookmarking" platform: an hybrid between a Web Browser, an Annotation tool and an Ontology Editor. The expression Semantic Bookmarking was coined to indicate the process of annotating information from (web) documents, to acquire new knowledge and represent it through representation models. Its basic functionalities allow for:

1. capturing information from web pages - both by considering the page as a whole, as well as by selecting portions of its text - and annotating them with respect to adopted referenced ontologies
2. editing the above ontologies for classifying the annotated information and for better characterizing their relevance to the interests of the user
3. navigating the structured information as an underlying semantic net which, populated with the many relationships which bind the annotated objects between them, eases the process of retrieving the knowledge (and associated web pages) which was buried by the past of time, by means of associative search (e.g. i remember there were a guy - who? - who worked in that project - which? - which were led by that university - again, which? - where that other person - yes, i have that name! - has a position) rather than traditional target-focused search&retrieval solutions

Its architectural and functional design make Semantic Turkey differentiate from similar, existing semantic browsers and annotation tools, as it offers a lightweight structure, which completely exploits the infrastructure of the hosting web browser (with respect to, for example, the complex completely-web based interface of Piggy-Bank (Huynh, Mazzocchi, & Karger, 2005) and which grants the user a good control over its personal domain representation (while traditional semantic annotation/browsing tools like Magpie (Dzbor, Domingue, & Motta, 2003) and Melita (Ciravegna, Dingli, Petrelli, & Wilks, 2002), are only able to import and adopt ontologies without any editing capability).

## 3.3. maskkot: Entity-centric Annotation

This section aims at introducing the concept of entity-centric annotation and describing its realization by means of the integration of ENS services in the Semantic Turkey Firefox extension, named maskkot. As described in previous section, Semantic Turkey allows for ontology-based annotation of web pages. In simple words, users annotate web pages linking web documents to an instance of a concept in a domain ontology, creating some kind of ``surfing experience" knowledge base. Successively, users of Semantic Turkey can make use of the semantic structure of the defined ontology to discover further annotations related with other instances, improving the capability of reusing the already performed surfing/annotation experience.

[8] The interested reader is referred to the seminal philosophical discourse about naming by Saul Kripke
[9] http://www.youtube.com
[10] http://www.flickr.com

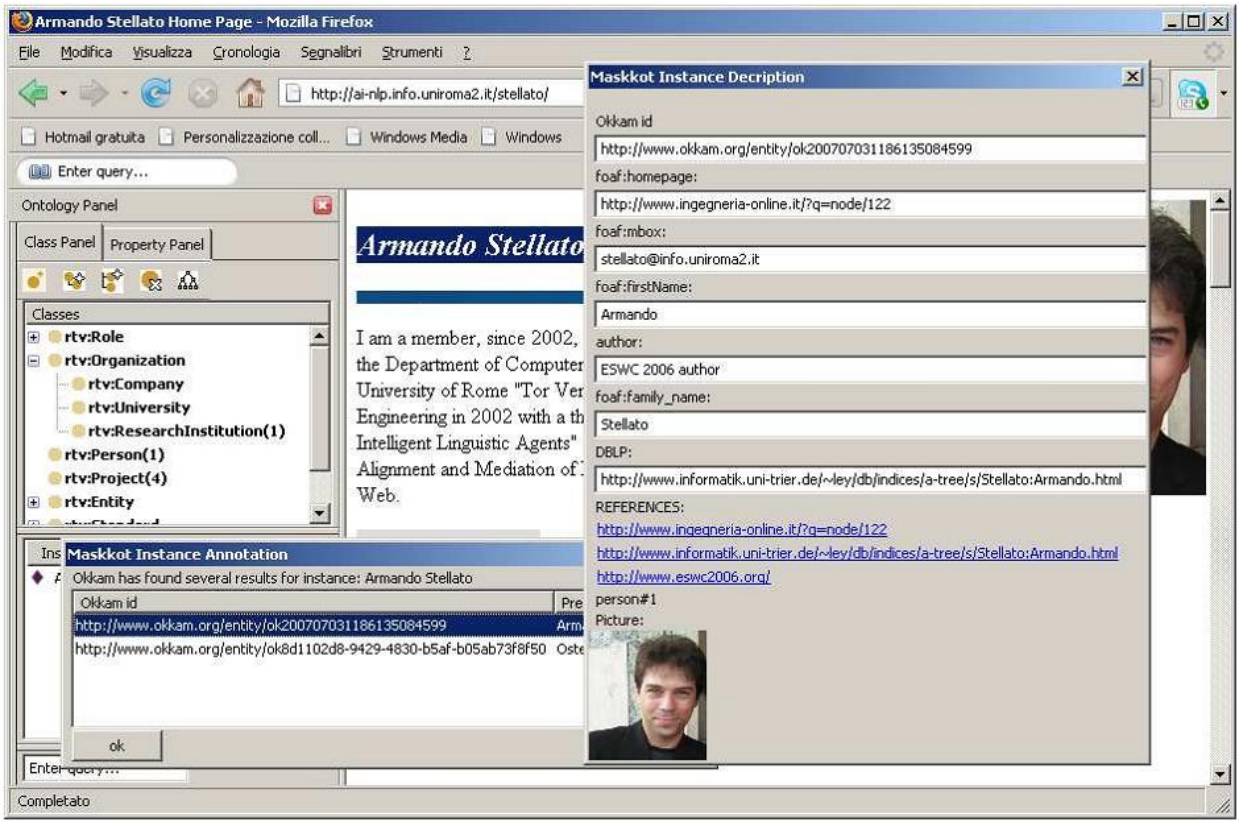


Figure 3: Maskkot inaction

The current Semantic Turkey annotation process consists in highlighting one or more keywords, thus giving a hint about the definition of an instance of a concept in the selected domain ontology. The highlighted keywords are used as a name for the annotated instance, and a web-link to the web document is added.

The keyword-based instance identification, combined with an analysis of the related semantic structure defined in the ontology, generally allows human users to recognize the entity described by the annotated keywords. Unfortunately, this kind of instance naming leads to a proliferation of the identifiers used for the annotation of the same entity in different knowledge bases created by means of Semantic Turkey. As a result, this fact prevents a precise and easy information integration of such knowledge bases. In order to solve this integration problem we decided to apply an apriori approach, evolving the concept of local keyword-based annotation to a more global entity-centric annotation approach. The goal of this approach is that, in the act of annotating, a user is enabled to discern to which real world entity the annotation refers, allowing the reuse of the globally unique related entity identifier. In this way the annotation explicitly refers to a globally recognizable entity, creating a knowledge base which is, in principle, integrable with others knowledge bases created using the same approach.

In order to realize the entity-centric annotation process we need to modify the standard Semantic Turkey annotation work-flow, integrating in the process the functionalities provided by the okkam ENS.

Without diving into technical details, the current Semantic Turkey annotation system can be described as this sequence of steps:

1. the user highlights a set of keywords
2. the keywords are dragged and dropped on a class in the *ontology panel* triggering the creation of a new instance in the ontology
3. a link to the web document annotated is added to the instance description;

The entity-centric annotation process based on okkam is described in the activity diagram depicted in figure Figure 2. The steps of the entity-centric annotation process implemented in maskkot and described in the activity diagram are:

1. the user highlights a set of keywords
2. the keywords are dragged and dropped on a class in the *ontology panel* triggering the creation of a new instance in the ontology
3. the keywords are used to create an annotated query and inquire okkam
4. okkam solves the query and returns a list of candidate entities matching the provided description, if any
5. the user selects one among the returned entities or triggers the publication of a new entity when none of the returned entities represent the annotation
6. the identifier of the selected/created entity is integrated in the knowledge base as URI of the annotated ontology instance
7. a link to the web document annotated is added to the instance description;

The screenshot in Figure 3 shows the retrieval of potential matching entities for the annotated text ``Armando Stellato" from the okkam service

## 4. maskkot Architecture

### 4.1. Semantic Turkey Architecture

Semantic Turkey integrates different technologies which are in part dictated by its hosting web browser (Mozilla Firefox) and in part expressly chosen (Java) due to the large availability of libraries for managing Semantic Web data.

Though the implemented system can easily be deployed as a single XPI (Cross Platform Installer) and installed as a Firefox extension application, with no further configuration requirement, the architecture of Semantic Turkey consists of a rather complex framework, designed as a web application, using a three layered approach.

The first layer, the *Presentation layer*, is the real extension for the Firefox web browser. This layer has been developed through a combined use of the XML User Interface Language XUL[11], the eXtensible Binding Language XBL[12] and a version of the Javascript language comprising the standard ECMA-262 ECMAScript and dedicated extensions for interacting with the Mozilla technology. Everything related to user interaction is directly managed by this layer: user operations trigger client requests which are forwarded according to the AJAX paradigm as http requests to the service layer; the returned XML-formatted data is then parsed to dynamically modify/populate the XUL user interface.

The second layer, the *Service layer*, is realized through a collection of Java Web Services, published through the Web Server "Jetty"[13]. This java web server is embedded in a java application which is activated during initialization of the Semantic Turkey extension, through an XPCOM (Cross Platform Component Object Model) bridge[14] implemented in javascript. Once the server is started, any further interaction between the first and second layer is handled via client-server AJAX communication. This solution also allows for a flexible deployment of the tool, since it can both be adopted as a completely autonomous web browser extension, as well as a personal access point for collaborative web exploration and annotation: in the latter case, a centralized solution should be adopted, in which distributed Firefox clients communicate with the same centralized server, requiring extra-effort due to the technologies and policies to adopt for managing the collaborative environment, but no substantial modifications to the architecture as a whole.

Finally, the *Data layer* provides access to a RDF/OWL semantic repository of data. RDF triples are made accessible through the ART Ontology API, an abstract set of API developed at the University of Tor Vergata, which actually serve as an interface for the several available semantic technologies.

### 4.2. Semantic Turkey Extension Framework

Though Semantic Turkey has been developed as a Firefox extension, it can in turn be extended by dedicated components, to add new functionalities, extend existing ones, or to host entire new applications which may sit on top of ST and benefit of its working environment and facilities. Being the result of two different technologies - those associated to the Mozilla framework on the client side, and Java regarding the service and data layers, Semantic Turkey required proper integration of two different extension frameworks to produce guidelines and a coordinated environment for the production of dedicated extensions.

Each ST plug-in can be developed by extending either one of or both the presentation and service layers. Another feature allows to create specific extensions for the data layer, implementing different technologies for RDF management. The presentation layer extension mechanism completely inherits the Mozilla extension framework: each ST plug-in extending the presentation layer must be declared as a new Firefox extension which has a "dependency" over Semantic Turkey. Mozilla extension mechanism has an highly unrestricting policy regarding interference with pre-existing extensions or with the hosting application (the Firefox web browser, in our case): it is possible to remove/overwrite existing content of the user interface and/or its associated events and functionalities, so that an extension could even drastically change the aspect/behavior of the hosting application or of an another extension. Though this may reveal of potential interest for high customization of the tool, it is not recommendable to do that for plug-ins which aim to coexist - and thus be compatible - with other ones in an open scenario. While this freedom of development cannot be restricted in any way, it would be important to support the ST extension developer with facilities like firing and catching of events associated to responses from the server, data changes etc… apart from a few triggers and handlers, we are still investigating on how to realize the whole framework for supporting extension development in this sense.

The services and data layers (realized in Java) required instead the development of a dedicated extension mechanism. We decided to adopt OSGi technology for java modularization[15] for fulfilling this objective. OSGi technology originally targeted embedded devices and home services gateways, but it is ideally suited for any project that is interested in principles of modularity, component-oriented, and/or service-orientation, as it is the case of Semantic Turkey. From OSGi strong impact on the open-source community, originated the Apache Felix project, a community effort to implement the OSGi R4 Service Platform, which includes the OSGi framework and standard services, as well as providing and supporting other interesting OSGi-related technologies, such as the OASIS standard Service Component Architecture SCA[16]. Semantic Turkey features the Felix component, allowing developers to extend the middle and lower layer of ST through dynamically loaded components. The extension policy is driven by the concept of "Extension Points": extension points are interfaces which specify "entry points" for external components, so that the application knows in advance where extensions will be connected to its architecture and will be able to interact with them without being previously "aware" of their existence.

Currently, two main extension points, which can be seen in figure 4, allow the development of new services to be dynamically added to the collection of servlets in the

[11] http://www.mozilla.org/projects/xul/
[12] http://www.mozilla.org/projects/xbl/xbl.html
[13] http://jetty.mortbay.org/jetty/
[14] http://www.mozilla.org/projects/xpcom/
[15] http://www.osgi.org
[16] http://www.oasis-opencsa.org/sca

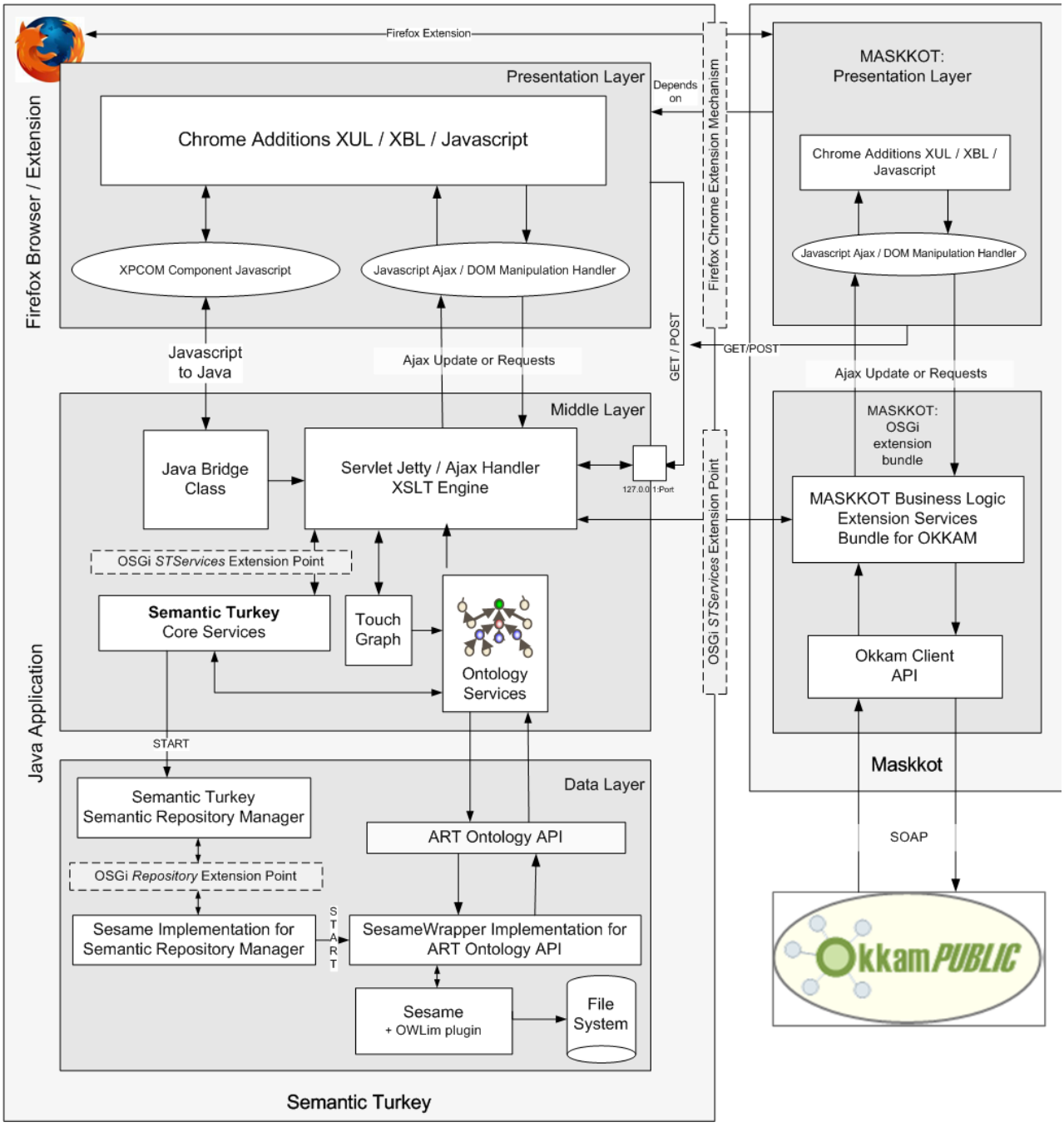


Figure 4: Architectural view of the maskkot extension and its interfacing with Semantic Turkey and okkam

middle layer, and to the adoption of wrappers for different RDF management technologies in place of the default one, which exploits the Sesame RDF API (Broekstra, Kampman, & van Harmelen, 2002) and the OWLim Semantic Repository (Kiryakov, Ognyanov, & Manov, 2005). Currently, any kind of extension, be it a complete plugin (client+service extension), a purely java service extension (e.g. a monitor reacting to events fired in the middle layer) or a RDF technology replacement, can be packed as a Firefox XPI package, thus easing installation procedures for the user. Once started, Semantic Turkey always re-scans the Firefox extension directory looking for declared ST extension bundles. When one is detected, its content is dynamically added to the core system.

### 4.3. Extending Semantic Turkey to host maskkot

The maskkot platform has been developed as a Semantic Turkey plug-in which extends already available annotation and instance creation functionalities with a lookup operation on the okkam service, exploiting its results to reuse existing individuals from the okkam repository or, conversely, contributing to the repository with new annotated entities.

The kind of interaction required with the okkam service, described in Figure 2: Entity-centric Annotation Activity DiagramFigure 2, is rather intrusive with respect to the ordinary operations of instance creation and semantic annotation which can be performed through Semantic

Turkey. An optimal approach would have required a non trivial extension mechanism in the client which should have foreseen in advance possible interruptions in the editing operations and partial rerouting of the standard workflow of operations, which unfortunately is not available at the moment in the client layer of ST. On the other hand, the open architecture of the client and the strong modularization of the servlets, which tend to separate as possible the preparation of the XML response from the internal query/update operations, still permitted the development of a dedicated extension, without any modification on the core ST system, nor rather unclean pratiques of copying&pasteing existing code from the core system to the developed extension.

When maskkot extension is loaded inside Firefox, it informs Semantic Turkey that its instance creation and semantic annotation operations are redirected on different services (i.e., the http GET request to the service layer changes using different parameters). These new services - provided by a new component dynamically loaded through Felix (maskkot *business logic* in figure 4) - largely reuse the same methods (exposed by the core Semantic Turkey API) adopted by the original services, but include the added interaction with the okkam service to get/contribute with identifiers from/to the okkam entity repository. The new services produce - at each interaction - an enriched versions of the XML responses traditionally associated to the modified operations, which includes the additional information obtained from the okkam service. When the client receives the XML, it is passed to the new handlers which have been added through the client extension.

## 5. Conclusion

Semantic Web technologies are becoming ever and ever a concrete reality which is no more limited to academical scope. However, while existing frameworks and products offer now solutions for Knowledge Management and Information Exchange which may reveal to be of interest inside confined settings, the original dream of a distributed and pervasive *Semanticized Web* has still a few steps to do before its realization. With maskkot we propose an application which is ready to introduce Semantic Web into everyday life of the average web user: the web browser-embedded bookmarking system can be utilized during usual web navigation for personal needs, as well as being adopted in collaborative environments involving social tagging of data with respect to reference ontologies, and be customized according to different scopes and users. At the same time, constant reference to the okkam ENS, which is almost transparently injected inside the annotation&build process, can avoid the proliferation of concurrent references to same entities, thus enabling content reuse and proliferation of information across distributed and independent actors.

## 6. References


Berners-Lee, T., Hendler, J. A., & 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 , 279* (5), 34-43.

Bortoli, S., Stoermer, H., & Bouquet, P. (2007). Foaf-O-Matic Solving the Identity Problem in the FOAF Network. *In Proceedings of the Fourth Italian Semantic Web Workshop (SWAP2007).* Bari, Italy.

Bouquet, P., Stoermer, H., & Bazzanella, B. (2008). An Entity Naming System for the Semantic Web. *In Proceedings of the 5th European Semantic Web Conference (ESWC 2008).* Springer Verlag.

Bouquet, P., Stoermer, H., & Xin, L. (2007). Okkam4P - A Protégé Plugin for Supporting the Re-use of Globally Unique Identifiers for Individuals in OWL/RDF Knowledge Bases. *In Proceedings of the Fourth Italian Semantic Web Workshop (SWAP2007).* Bari, Italy.

Bouquet, P., Stoermer, H., Cordioli, D., & Tummarello, G. (2008). An Entity Name System for Linking Semantic Web Data. *In Proceedings of LDOW2008.*

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.

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.

Dzbor, M., Motta, E., & Domingue, J. B. (2004). Opening Up Magpie via Semantic Services. *3rd Intl. Semantic Web Conference (ISWC04).* Hiroshima, Japan: November.

Griesi, D., Pazienza, M., & Stellato, A. (2007). Semantic Turkey - a Semantic Bookmarking tool (System Description). In E. Franconi, M. Kifer, & W. May (A cura di), *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007, Innsbruck, Austria, June 3-7, 2007, Proceedings. Lecture Notes in Computer Science. 4519*, p. 779-788. Springer.

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.

Ioannou, E., Sathe, S., Bonvin, N., Jain, A., Bondalapati, S., Skobeltsyn, G., et al. (2009). Entity Search with NECESSITY. *12th International Workshop on the Web and Databases.* Rhode Island.

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.

Liebowitz, S., & Margolis, S. E. (1998). Network Externalities. In *The New Palgrave's Dictionary of Economics and the Law.* MacMillan.

Quan, D., & Karger, D. (May, 2004). How to Make a Semantic Web Browser. *Thirteenth International World Wide Web Conference (WWW2004).* New York City, USA.

Stoermer, H., Rassadko, N., & Vaidya, N. (2010). Feature-based Entity Matching : The FBEM Model, Implementation, Evaluation. *CAISE'10, the 22nd International Conference on Advanced Information Systems Engineering.* Springer.