The BAsAS Architecture For Semantic Web Annotations

Valentin Zacharias
ontoprise GmbH
Amalienabdstr. 36 (Raumfabrik 29)
76227 Karlsruhe, GERMANY
zach@ontoprise.de

ABSTRACT

We describe a generic architecture for the (semi-automatic) creation, storage and querying for annotations of web resources. Our BAsAS architecture uses recent advances from the Semantic Web and Web 2.0 communities to make Semantic Web annotations a reality. The BAsAS architecture makes it easy for users to start to annotate and easy for developer to use the annotations that get created.

Besides describing the general architecture we will also detail an implementation of this architecture build for a Semantic Web community portal.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

1. INTRODUCTION

The Semantic Web is the vision of having data on the web defined and linked in such a way, that it can be used by machines not just for display purposes, but for integration, automation and reuse of data across various applications.
An Annotation is a piece of information or knowledge entity that is associated with a document or part therof. The combination of these two research strands is often understood as *Semantic Annotation* - annotating documents using semantic technologies. Usually this means creating links between the content of the document and entities in an ontology. In this paper, however, we focus on *Semantic Web Annotation*, understood as annotations of web resources with the goal of lifting them into the semantic web. Unstructured content outside of the semantic web is annotated and these annotations become part of the Semantic Web. In this way (a part of) the content of the document becomes part of the Semantic Web, the document gets woven into the Semantic Web. Research that understands Semantic Web Annotations in this way must discuss the question of how the data created in the annotation process becomes part of the Semantic Web, how it is retrievable and usable for computer agents.
A third (but very uncommen) way to understand the combination of these two research threads is Semantic Web Annotation as the annotation of Semantic Web resources. Even in a future where the web consists only of structured, semantic content there is still room for annotations. For example a rule set on the web may be annotated with a remark that it is outdated or an music ontology with links to other ontologies that extends it with more detail for parts of the musical spectrum. We do not specifically discuss this interpretation, but many of the techniques described here could be applicable there as well.

2. EARLIER WORK

Of direct interest to this paper are only those approaches that take the same holistic view of Semantic Web Annotations. Approaches that deal with the question of Semantic Annotation as well as how this annotation then becomes part of the Semantic Web. We will not further discuss those approaches that focusses on the details of automatic and semi-automatic Semantic Annotations (e.g. [9][16][4][11][22][5]).

We also omit the discussion of all approaches that use embedded annotation. While embedded annotations may play some role in the Semantic Web, they cannot form the foundation for lifting large amounts of already existing documents into the Semantic Web. Embedded annotations always require that the person or system doing the annotation has some means of changing the original document - this is just not the case for all but a tiny part of the internet. The web as it exists today usually reserves write privileges to a very small group.

2.1 The KIM- Semantic Annotation Platform

The KIM-Semantic Annotation Platform[15][14] is a system for the automatic semantic annotation of web documents, the indexing of these and their retrieval. The creators of KIM understand Semantic Annotations as links from entities in a text to their semantic descriptions. The semantic descriptions are organized with respect to a lightweight upper level ontology - the KIM Ontology KIMO. This ontology is prepopulated with entities extracted using an adabted version of the GATE[7] toolkit from large publicly available data sources. All annotations are stored on the KIM server. The primary interface to the KIM server is a browser plugin for the Internet Explorer. A user who has installed this

plugin can request an annotated version of the page she is currently visiting. The page is send to the server, automatically annotated and displayed with the all entities as links. The user can then use the plugin to learn more about the entities, browse the populated ontology and explore documents that where indexed earlier.

2.2 The Annotea RDF Infrastructure

Annotea[13][17] is a web-based system for shared annotations. Annotations are understood as RDF statements made by an author about a web page. These annotations are not embedded in the pages but stored in one or more external annotation servers. The authors of the system tried to make it as open as possible by relying on standards such as XPointer, RDF and HTTP for all data and interfaces of the system. The annotation servers accept new data that is submitted using a simple HTTP POST call to an annotation server, the new annotation data is submitted as RDF data at the same time. Annotea also defines a simple format for retrieving annotation data from an annotation server. The protocol for doing this is only partially defined, everything beyond requesting all annotations for a specific url is left to the creators of individual annotation servers.

Recently[17] the Annotea infrastructure has been adapted for the management of shared bookmarks. The main change is the extension of the annotation schema, in particular the inclusion of the notion of a *topic hierarchy*. The interface for this infrastructure allows to define a semi-formal hierarchy of the topics of interest and to associate bookmarks with these topics.

Quite a few applications that use the annotatea infrastructure have been created. The best known is propably the Annotea implementation included in the w3c Amaya[1] web browser / editor that allows to create and view annotations. Annozilla[3] and Annotea Ubimarks[2] are plugins for the Mozilla /Firefox browsers that work with the Annotea RDF infrastructure. Of these the Ubimarks plugin ist more capable, allowing to create a topic hierarchy and to annotate web documents with respect to it.

2.3 Contribution

What we are trying to do is similar to the Annotea infrastructure: define an architecture that can form the backbone for Semantic Web annotations. We, however, improve on the well known Annotea idea in three crucial points: we show how the SPARQL language solves the problem of queries for annotations. We demonstrate how AJAX can be employed to build an annotation tool that is more lightweight and less browser dependent than anything that exists for Annotea. Finally we show a simple architecture that allows semi-automatic annotation while still retaining the extrem lighweight property of the annotation tools. To a large part this paper can be understood as an attempt to update the ideas pioneered by annotea using technology that was not available at the time it was initially conceived.

3. ARCHITECTURE

The BAsAS architecture is so named for its main characteristics: **B**rowser, **A**nnotation **s**erver, **A**JAX and **S**PARQL. These parts and their interaction are shown in Figure 1. At the core of the architecture is a Semantic Web data store that holds the annotations and that offers a SPARQL interface to access them. It should also offer an HTML interface for browser based access.
Annotations are done by the user with the help of an AJAX interface to an annotation server. This annotation server helps to keep the client side of the annotation as lighweight as possible, in particular any complex algorithms for automatic annotation are run on this server. The annotation server is also responsible for receiving the information from the annotation interface and translating it into update requests for the data store. Only with the help of such a server component is it possible to build an extrem lightweight annotation component.
The annotation component needs some way to establish the context it is called in for this gives the information on what the user wants to annotate. This context can be established by a lighweight browser "plugin". As we will see this needs not be an actual browser plugin. A short snippet of JavaScript that browsers treat like a bookmark is sufficient in many cases. We will also show that sometimes not even that is necessary.

3.1 SPARQL

We understand Semantic Web Annotation as lifting documents into the Semantic Web by adding semantic data about them to the Semantic Web. As we argued in the introduction this means that we have to find a way to make the annotated data available to agents using the Semantic Web. Both Annotea and KIM address this question by offering interfaces to the server storing the annotations. At the same time, however, both approaches use non-standard query interfaces. Annotea does define a very simple query syntax that relies mostly on RDF, but the possible queries are extremly limited. For example it does not allow to ask the data store for all topics used in the annotations. Vendors of Annotea data stores *may* offer more sophistictad query functionality but this is not standartized. All in all this means that agents utilizing the Semantic Web will need some kind of custom build wrappers to access the annotation data in these systems.

SPARQL, the SPARQL Protocol[6] And RDF Query Language[20], defines how clients can pose queries to an RDF store. In the short time since its inception it has already garnered a great deal of attention and has been implemented for a large number of RDF stores. Even though still only a W3C candidate recommendation SPARQL looks set to become the Semantic Web query language for the foreseeable future. With this SPARQL is also the best candidate for a query language for Semantic Web annotation stores: almost all agents using Semantic Web data will be able to use SPARQL and can then automatically access the annotated data. SPARQL is also powerful enough to support all realistic queries for annotations.

3.2 AJAX

AJAX, shorthand for Asynchronous JavaScript and XML, is a technique for the development of web applications. Through the use of AJAX it is possible to create web applications that "feel" like applications that run directly on the client computer. AJAX web applications are web pages that heavily utilize JavaScript to exchange small amounts of data with

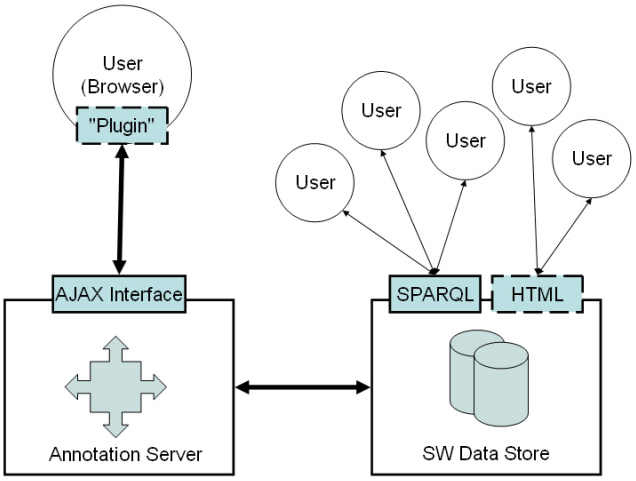

Figure 1: High level view of the BAsAS architecture

the server and to change the page structure. AJAX pages can respond to a user action, update data on the server and change the display based on the servers respone all without requiring the user to reload the page. Compared to other technologies for the creation of rich clients - like Flash or Java Applets - AJAX has the advantage that JavaScript is supported by virtually all browsers on all platforms. It also works without the installation of any plugins, has a short startup time and integrates seamlessly with HTML. With the recent development of AJAX libraries such as the GoogleWebToolkit[10], Rico[21] or the Yahoo UI Library[23] it is now easily possible to create AJAX applications that work accross a large number of browsers.

Figure 2: A simple AJAX interface used for web annotation (from the social bookmarking site del.ico.us)

By using the AJAX technology it is now possible to create productive annotation interfaces that the user can use directly in her browser and that do not require any installation. Unlike normal webpages AJAX interfaces can support the user for example with auto completetion of topics or efficient ways to add multiple topics. The usefullness of AJAX for annotation is also evident from the AJAX interfaces used by many of the successfull tagging and social bookmarking pages (see Figure 2 for an example).

3.3 Browser Integration

The recent success of web based bookmarking and social bookmarking sites such as del.ico.us shows that a large number of web users is willing to annotate if its just simple enough. All of these sites integrate the annotation into the browser. As the main tool that is used to interact with the web, the browser is the logical choice for the integration of an web annotation program. Most sites also try to make starting to annotate as simple as possible many not even requiring the installation of a browser plugin.

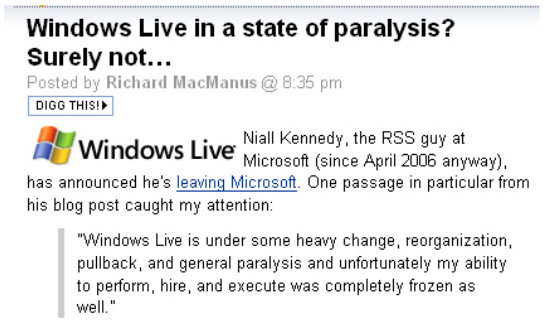

Figure 3: The "digg this" button at this webpage allows a simple annotation of the webpage without any installation of annotation tools in the browser

Is is obvious that an annotation program needs some information about the document the user wants to annotate. Its possible but inconvenient to ask the user to supply this information in a web form, hence necessary to retrieve it automatically. To extract this context from the user the annotation infrastructure needs some integration with the browser. A lightweight method to achieve this without requiring the installation of a plugin are *bookmarklets*. A bookmarklet is a short snippet of JavaScript that is handled by the browser like a normal html link. It can be saved just like other links into the favorites or bookmarks. When selected the browser executes the JavaScript that then extracts some information about the page currently viewed and calls the annotation interface with this information. The bookmarklet can retrieve for example the url of the website the user looks at and the text that is currently selected. At least for simple annotation schemes this is enough to characterize what the user wants to annotate.

The bookmarklet passes the information to the annotation interface by appending http get parameters to the url of the annotation page. This means that all the context information needed for a simple annotation can be encoded in a HTML link. A creator of a webpage that is interested in having it annotated by its visitors can therefore include such a link that already defines the current page as the annotated document. Figure 3 shows an example of such a link to the annotation page of the social bookmarking site digg[8].

3.4 Annotation Server

The exclusive reliance on very thin clients for annotation means that more sophisticated functionality - such as semi-automatic annotation or interfaces for arbitary rdf stores - is harder to realize. While it is possible to realize this kind of functionality in JavaScript directly embedded in an AJAX page, it is costly and would result in bad usability. It would be costly because many of the libraries that could aid the creation of such functionality are not available in JavaScript. Usability problems arise because all the code needed for the functionality must be tranferred to the browser and be parsed there, resulting in loading times that are too long for a "lightweight" annotation tool.

To combine sophisticated annotation functionality with a very lightweight annotation client we propose an *annotation server* that supports the AJAX interface during the annotation process. This annotation server can fetch the site that is currently annotated, run algorithms to automatically perform an automatic annotation that is shown to the user in the client interface. In the end the annotation server accepts the completed annotation and handles the communication with the data store.

3.5 Annotation Process

To illustrate how the pieces of the BAsAS architecture fit together, we detail what happens during one annotation transaction.

1. With her browser the user discovers a webpage that she wants to annotate. She clicks on the bookmarklet that she bookmarked earlier.

2. The JavaScript of the bookmarklet is interpreted by the browser. It extracts the url of the current page and the text that is currently selected. It opens the annotation interface in a new browser window. The url and the selected text are passed as parameters.

3. The annotation server receives the request for the annotation transaction. It creates a new server side annotation task and serves the files for the annotation interface. In a seperate thread it starts to fetch the website that should be annotated and then runs any available algorithms for the automatic annotation.

4. The browser receives the annotation client and starts interpreting its JavaScript code. The annotation client immediatly ask the annotation server for data that could help the user perform the annotation. This could be data about other annotations by the same user or annotations of other users for the same resource. The client also request the result from the automatic annotation. These request are done asynchronous, meaning the interface stays responsive and the user can start annotating.

5. The annotation server accesses the data store to fetch the requested data about earlier annotation. This is send to the client. As soon as the current document is fetched and annotation proposals have been created, these too are send to the client.

6. The client receives the information about the earlier annotations and updates the interface. The results from the automatic annotation are also received and displayed as proposals.

7. The user adds any information she wants, accepts or declines the annotation proposals and presses "save".

8. The annotation server receives the annotation data from the client and contacts the data store to save them.

After the annotation data has been saved in the data store in can be accessed by any agents capable of SPARQL or via a web interface.

4. UNITRACC COMMUNITY PORTAL

We are currently implementing the architecture described in the preceeding chapter for the use in the system unitracc.

Unitracc, the "Underground Infrastructure Training and Competence Center" is a internet based e-learning system for the area of canalization. Unitracc also contains a collection of web based tools that help public authorities manage and monitor underground infrastructure. The system already contains a large number of information units, especially enhanced digital versions of two standard textbooks about canalization. Unitracc is developed by the company Prof. Dr. Ing. Stein & Partner GmbH, a leading engineering firm whose founder is also the author of many standard works of technical literature. The development has been funded in part by the German Federal Ministry of Education and Research. Access to unitracc is available on a subscription basis, the target audience ranges from beginning trainees and their teachers to architects.

We are currently in the process of extending this platform in the direction of a community portal. In addition to the core content supplied by the creators of unitracc there should be an outer layer of user created content, such as comments, technical manuals uploaded by tool vendors or annotations of web sites. We expect this layer of content to be less reliable but also to be more current and diverse. We chose to build the this part of the platform as open as possible: we believe that the openness increases the motivation of people to contribute. We hope that people will be less reluctant to put effort into a commercial site when they know that this content can be used by everyone.

4.1 Unitracc Metadata Structure

The metadata structure was developed based on ideas from Metadatastandards like LOM[18], or IMS[12] and strived

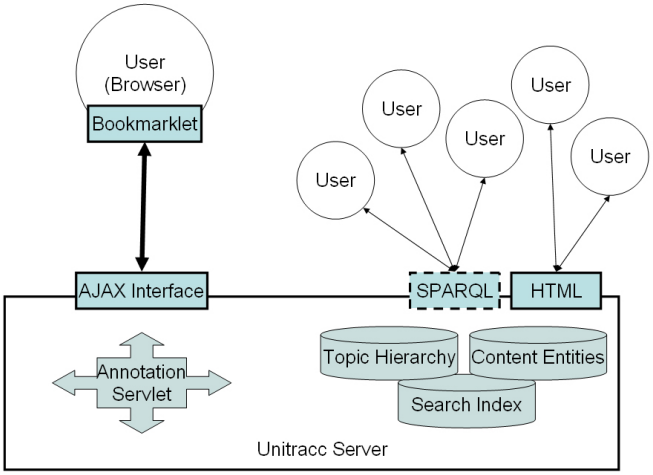

Figure 4: Overview of the Unitracc annotation architecture

to be compatible to the LOM/SCORM[19] standard which wasn't finalized at the time the unitracc architecture was developed. The metadata for content entities fall into three different categories:

1. Metadata for internal administration, like versioning information or size and format of multimedia data.
2. Didactic metadata allows to associate information with areas of interest, occupation groups and the learning level of the user. For example didactic metadata classifies an infobit according to its difficulty, area of interest (occupational safety, law, environment) and kind of information (calculation, history, construction site picture..).
3. Content metadata describe the actual content of a content entity. Content metadata consist of a topic from a topic hierarchy contain roughly one hundred topics.

During the annotation of web resources only the content part of the metadata is assigned by the user - everything else is trivially determined from the fact that it is an user generated annotation of a web resource. The topic hierarchy can be changed using the AXAJ interface shown in Figure 5. Changes to the topic hierarchy are currently meant to be done only be the administrators of unitracc.

4.2 Unitracc Annotation Architecture

Figure 4 shows a high level overview of the Unitracc annotation architecture. We will quickly discuss the points in which it differs from the architecture shown in Figure 1. The most obvious change is that the annotation server and the data store are now part of the same server. This is owed to the fact that we are not dealing with a generic Semantic Web data store but a domain specific one for this application. Since the annotation server is also not very generic but tailored for this application it was just simpler to combine the two. Another difference is that for historical reasons the data store is not one homogeneous database but that it consists of three major parts:

1. A database for all content entities of unitracc. This is a MySQL database, storage and retrieval of the entities is managed by Hibernate.
2. A Lucene index of most content entities for quick retrieval.
3. A database that stores the topic hierarchy. This is the newest addition to the system. Currently Hibernate is used to store the taxonomy in a HSQLDB database.

Because neither of the unitracc data stores is a RDF store we have no automatic support for SPARQL and have currently only implement a small subset - hence the dashed lines around the SPARQL interface in Figure 4.

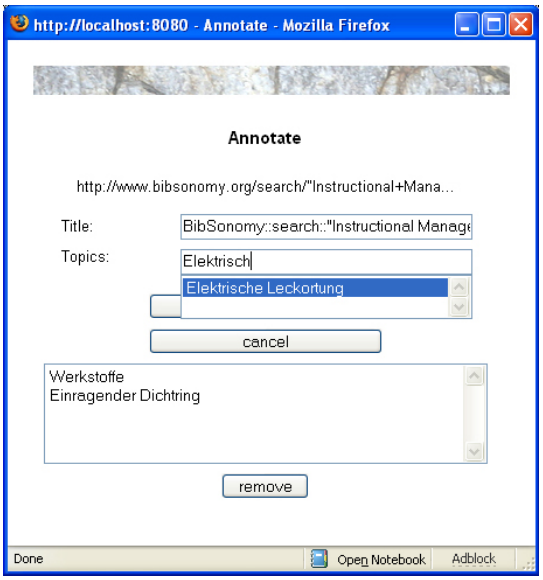

Figure 6: The unitracc annotation interface

The annotation servlet is the server side component for the annotation interface. The annotation servlet fetches all sites

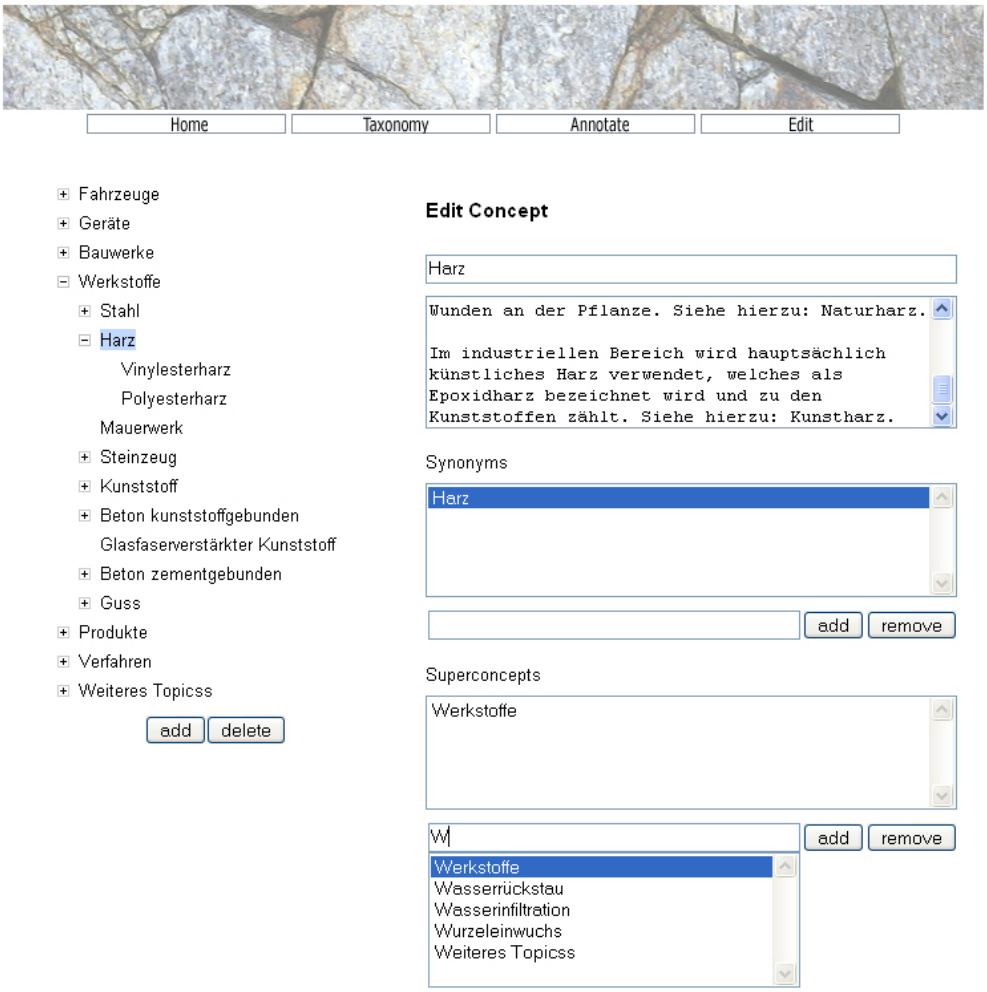

Figure 5: AJAX editor for the unitracc topic hierarchy

that are currently annotated by the user and uses a simple automatic categorization to identify suitable topics.

4.3 Annotation

Users of the site will be encouraged to annotate the web resources relavant to unitracc using a simple annotation tool. For the time beeing we have settled on a very simple annotation format: the annotation is always for the whole web page and it only consists of "has topic" relations to the topics from the unitracc topic hierarchy. The integration of the annotation tool with the browser is done with a bookmarklet as described earlier. The annotation interface is realized with AJAX, it supports the user mainly by offering auto complete for the topics. It also receives the proposals from the automatic categorization performed by the annotation servlet and displays them to the user. A screenshot of the annotation interface is shown in Figure 6. The annotation interface, like the topic hierarchy editor, is created using the Google Web Toolkit[10].

5. CONCLUSIONS AND FUTURE WORK

We have presented the architecture of a Semantic Web annotation architecture. We belive that systems build in this image could form the foundation for Semantic Web annotations. These systems could be lightweight, easy to use and powerfull at the same time.

We have given a short overview our implementation in progress of such a system for a domain specific community portal. The system shows great promise but is still only a prototype. Next steps for this system are the implementation of better SPARQL support, access controls and general stability and scalability issues.

One serious issue that has already emerged in our work with this system is that in some cases the annotation server cannot retrieve that page the user tries to annotate. This may be because the page requires some kind of login, needs values in a cookie that have been set before or is based on frames. The user can still annotate such an url, but she will not get any annotation proposals. It is propably not possible to completely solve this issue without recourse to browser plugins. Creating these, however, is not an option in this context since unitracc just lacks the resources to build plugins for many different browsers.

6. ACKNOWLEDGMENTS

This work was supported in part by the German Federal Minsitry of Education and Resarch under the ksi_underground project.

7. REFERENCES

[1] Amaya home page, http://www.w3.org/amaya.

[2] Annotea ubimarks homepage, http://www.annotea.org/mozilla/ubi.html.

[3] Anozilla home page, http://annozilla.mozdev.org/.

[4] S. Chapman, A. Dingli, and F. Ciravegna. Armadillo: Harvesting information for the semantic web. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[5] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 462–471, New York, NY, USA, 2004. ACM Press.

[6] K. G. Clark. Sparql protocol for rdf. Technical report, W3C, 2006.

[7] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of the 40th Annual Meeting of the ACL*, 2002.

[8] Digg, http://digg.com/.

[9] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 178–186, New York, NY, USA, 2003. ACM Press.

[10] Google web toolkit, http://code.google.com/webtoolkit/.

[11] S. Handschuh, S. Staab, and F. Ciravegna. S-cream - semi-automatic creation of metadata. In *Proceedings of the EKAW 2002*, pages 358–372, 2002.

[12] Ims (instructional management systems) project from educause. http://www.imsproject.org/.

[13] J. Kahan and M.-R. Koivunen. Annotea: an open rdf infrastructure for shared web annotations. In *WWW*, pages 623–632, 2001.

[14] A. Kiryakov, B. Popov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Semantic annotation, indexing, and retrieval. In *International Semantic Web Conference*, pages 484–499, 2003.

[15] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *J. Web Sem.*, 2(1):49–79, 2004.

[16] P. Kogut and W. Holmes. Aerodaml: Applying information extraction to generate daml annotations from web pages. In *Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001)*, 2001.

[17] M.-R. Koivunena, R. Swick, and E. Prud'hommeaux. Annotea shared bookmarks. In *Proceedings of the KCAP03 workshop*, 2003.

[18] Ieee learning technology standardization committee,draft standard for learning object metadata, 2001.

[19] Adl sharable courseware object reference model, scorm http://www.adlnet.org/.

[20] E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf. Technical report, W3C, 2006.

[21] Rico javascript for rich internet applications, http://openrico.org/.

[22] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. Mnm: Ontology driven semi-automatic and automatic support for semantic markup. In *EKAW*, pages 379–391, 2002.

[23] Yahoo ui library, http://developer.yahoo.com/yui/.