Annotea: An Open RDF Infrastructure for Shared Web Annotations

José Kahan
W3C INRIA Rhône-Alpes
ZIRST 655, av. de l'Europe
38330 Montbonnot Saint-Martin, France
kahan@w3.org

Marja-Riitta Koivunen *,
Eric Prud'Hommeaux, Ralph R. Swick
W3C MIT/LCS
200 Tech. Square
Cambridge, MA 02139
marja,eric,swick@w3.org

ABSTRACT

Annotea is a Web-based shared annotation system based on a general-purpose open RDF infrastructure, where annotations are modeled as a class of *metadata*. Annotations are viewed as statements made by an author about a Web document. Annotations are external to the documents and can be stored in one or more *annotation servers*. One of the goals of this project has been to re-use as much existing W3C technology as possible. We have reached it mostly by combining RDF with XPointer, XLink, and HTTP. We have also implemented an instance of our system using the Amaya editor/browser and a generic RDF database, accessible through an Apache HTTP server. In this implementation, the merging of annotations with documents takes place within the client. The paper presents the overall design of Annotea and describes some of the issues we have faced and how we have solved them.

Keywords

Annotations, Metadata, RDF, Semantic Web, RDF, XPointer, XML, World-Wide Web

1. INTRODUCTION

One of the basic milestones in the road to a Semantic Web [22] is the association of metadata to content. Metadata allows the Web to describe properties about some given content, even if the medium of this content does not directly provide the necessary means to do so. For example, a metadata schema for digital photos [15] allows the Web to describe, among other properties, the camera model used to take a photo, shutter speed, date, and location. An interesting side effect, is that a same piece of metadata can be used not only for describing content, but also to organize and classify it, thus setting up other properties we had not thought about at first. For example, we can use it to search for photos of a given location taken at a given time.

A first step towards building a Semantic Web is to have the infrastructure needed to handle and associate metadata with content. In order to reach this goal, we have been developing Annotea, a shared Web annotation system. In its simplest form, a Web annotation [26] can be seen as a remark about a document identified by a URI, made by the author of the document or by a third party, with or without author knowledge. In a shared Web annotation system, annotations are stored in specialized servers. Annotations are shared in that everyone having access to an annotation server should be able to consult the annotations associated with a given document and add their own annotations.

From a general viewpoint, annotations can be considered as metadata: they associate remarks to existing documents. We chose to use the annotation scenario to drive our initial metadata infrastructure development as it is a relatively simple metadata application and it would allow us to concentrate on the general details of the infrastructure without getting lost with the more specific details of the application. The most important goal of this project has been to use as much existing W3C specifications as possible. This paper describes how we have reached this goal by combining RDF with XPointer, XLink, and HTTP

The paper concentrates on describing the Annotea RDF infrastructure and its implementation in Amaya. Section 2 gives the overall design of Annotea. Section 3 describes the client implementation. Section 4 briefly discusses differences with related work by others. Section 5 concludes the paper and presents our perspectives for future work on Annotea.

2. DESIGN

In this section we describe the architecture of the Annotea system and the RDF annotation schema. We start with a discussion of the requirements that motivate some of the aspects of our design.

2.1 Requirements

Since the early design of Annotea, we decided to build an infrastructure that was based on generic RDF, with annotations being one possible instantiation of the infrastructure. This choice has allowed us to concentrate more on the infrastructure than on the application itself. We now list the principal requirements that have shaped Annotea (given in no particular order):

- Open technologies. Many of the existing annotation systems are based in proprietary schemes or protocols. This makes it hard to extend them. Annotea is built on top of open standards to simplify the interoperabil-

*W3C Fellow from Elisa Communications (Finland).

Copyright is held by the author/owner.
WWW10, May 1-5, 2001 Hong Kong
ACM 1-58113-348-0/01/0005.

ity of Annotea with other annotation systems and to maximize the extensibility of the data this system can carry.

- Annotated documents are well-formed, structured documents. Many Web annotation systems allow users to annotate any kind of resource that has a URI. To simplify our design, we decided to limit annotated resources to those that have a structure, that is, any HTML or XML-based document, including other annotations.

- Annotations are first class Web resources. As any other Web resource, each annotation should be associated with a URI.

- Annotations are typed. At the same time that an annotation can be seen as metadata related to an annotated document, annotations themselves can have distinct properties. The type of an annotation is metadata about the annotation itself. It allows users to classify the annotations as they are creating them (for example, saying this annotation is a comment or an erratum about a given document).

- Annotation types can be defined by users. Different users have different views and needs. Annotea should make it possible for any user group to define their own annotation types.

- Annotation properties must be described with an RDF schema [20, 5].

- Annotations are stored in generic RDF databases. Rather than making a specialized annotation server, we decided to view the servers as generic RDF databases. This is important as it will allow users, in the general Semantic Web picture, to reuse the information stored in such databases without having to change them.

- No assumptions on User Interface. Annotea describes how metadata can be associated with documents and how to query the RDF databases. It does not specify how a user agent must present the metadata to the user. We do predict, though, that some user interface consistency is needed. However, with our approach it is easy to provide additional views of the metadata.

- Local (private) and remote (shared) annotations. Annotations can be stored either locally in the user's host computer or in an annotation server. We assume that local annotations are private and remote ones shared. An annotation server is responsible for controlling access to the annotations that it stores.

- Multiple annotation servers. Having a centralized server may present both scalability [16] and privacy problems. User groups must be able to easily set up an annotation server and define who can consult it. Thus, in an Annotea instantiation, there may be a number of annotation servers. We do not attempt to solve the general scalability problem at this time.

2.2 Annotea and its operation

In Annotea, annotations are described with a dedicated RDF schema and are stored in annotation servers (Fig. 1). The annotation server stores the annotations in an RDF database. Users can query a server to either retrieve an existing annotation, post a new annotation, modify an annotation, or delete an annotation. All communication between a client and an annotation server uses the standard HTTP methods [21].

The annotations that we handle are collections of various statements about a document. They may be comments, typographical corrections, hypothesis or ratings, but there is always an author that makes a statement about the document or some part of it at a certain time. This is illustrated in Figure 2, where an author makes a statement about a document named XDoc. An annotation is represented as a set of metadata and an annotation body.

The metadata of an annotation is modeled according to an RDF schema and gives information such as the date of creation of the annotation, name of the author, the annotation type (e.g., comment, query, correction...) the URI of the annotated document, and an XPointer that specifies what part of the document was annotated. The metadata also includes a URI to the body of the annotation, which we assume to be an XHTML document. The annotation metadata does not say how the annotations must be presented to the user. This choice is left open to the developer of the client. Section 2.3 describes the annotation schema further in detail.

Annotations are stored in generic RDF databases, which are accessible through an HTTP server. The following scenario explains the interaction between these components when a user creates a new annotation document. For simplicity, we will suppose that annotations are displayed by highlighting the annotated text in the document.

- The user browses a document.

- The user selects some text on the document and tells its browser that he wants to annotate this text.

- The browser pops up a new window, where the user can type the text of his annotation and choose the type of the annotation.

- The user then publishes the annotation to a given annotation server. To do this, the browser generates an RDF description of the annotation that includes the metadata and the body and sends it to the server, using the HTTP POST method. The server assigns a URI to the annotation and to the body and replies with an RDF statement that includes these URIs.

- If the user further modifies the annotation, he will publish it directly to the URI that was assigned.

Note that the first time that a user publishes an annotation, this annotation does not have any URI. It is the server that assigns the URI. When the user requests the URI from the server later, the server will reply with the annotation metadata.

We now describe the scenario where the user browses an annotated document. We suppose this user has previously configured his browser with the list of annotation servers that he wants to query.

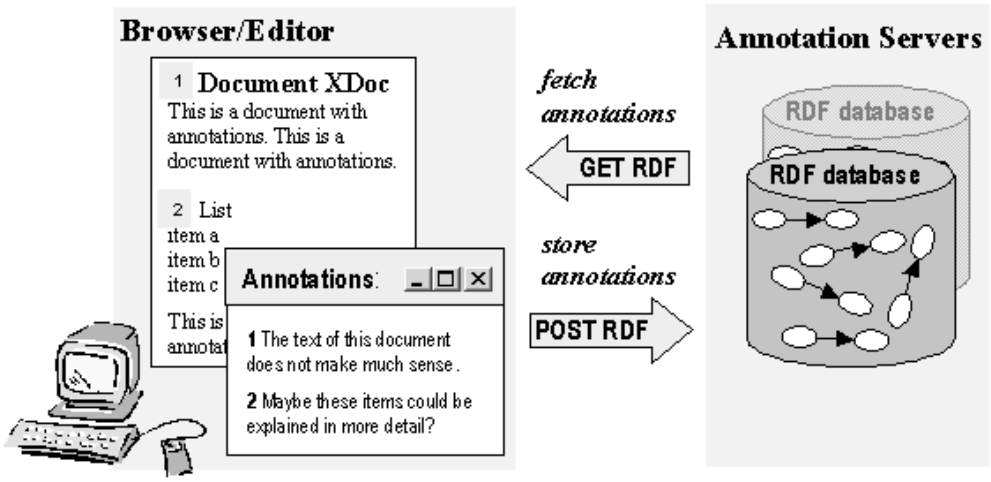

Figure 1: The basic architecture of Annotea.

- The user browses a document
- The browser queries each of the annotation servers, requesting via an HTTP GET method the annotation metadata that is associated with the document's URI.
- Each server replies with an RDF list of the annotation metadata. If the server is not storing any related annotations, it replies with an HTTP 404 Not Found message.
- For each list of annotations that it receives, the browser parses the metadata of each annotation, resolves the XPointer of the annotation and, if successful, highlights the annotated text.
- If the user clicks on the highlighted text, the browser will use an HTTP GET method to fetch the body of the annotation from the URI specified in the metadata.
- Finally, the browser will open up a window showing the metadata and the body.

In the above scenario, we divided the downloading of annotations into two stages. First, the browser downloads the metadata of an annotation. Next, and only if the user requests it explicitly, the browser downloads the body of the annotation. The motivation for this choice is to reduce the amount of data that is being sent back to the browser. In a heavily annotated document, sending the complete annotations will consume resources and the user may not actually be interested in seeing the body of all the annotations.

Note that once that an annotation is published to a server, it becomes a shared annotation. That is, any user with the correct access rights may retrieve the annotations from the server. For the moment, we expect that the HTTP server will enforce the access control to the annotations, using the standard HTTP authentication mechanisms.

It is also possible to store annotations locally in the host computer of the user, provided that the client simulates the reply to the first query of the server. Our Amaya prototype, that we describe later in Section 3, implements such a feature.

Figure 2: A basic annotation model with an author making a statement about a document.

The next section presents the Annotation RDF schema. Appendix A contains a more thorough presentation of the Annotea protocols.

2.3 RDF Schema for Annotations

The most important feature of an annotation is that it supports the evolving needs of the collaborating groups. For instance, an annotation system for classifying new technologies will need to expand their annotation types to classify specific characteristics of the technologies they are reviewing. Another working group may start with a set of annotation types and then modify this set according to the evolution of their work. Annotea users may wish to define new types of annotations as they use the system more. The group may also add relationships to other objects that are specific to the group's work. RDF provides support for these needs, e.g., by allowing the expression of new relationships, by allowing new annotation types, and by supporting the transformations from one annotation type to another.

RDF provides a simple yet very flexible framework for describing properties of any Web resources. In its most simple level, RDF provides (resource, property, value) triples (Fig. 3). A single triple is a statement that indicates that a *resource* has a given *property* with a given *value*. The *resource* can be any Web resource identified by a URI. The *value* may be a literal string or may be the URI of another Web resource. Literal strings may contain XML markup. By

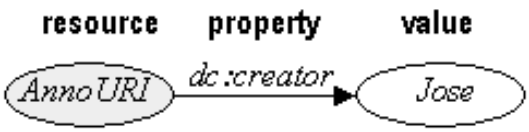

Figure 3: RDF triple model.

Annotation A super class describing the common features of annotations.

Advice A subclass of *Annotation* representing advice to the reader.

Change A subclass of *Annotation* describing *annotations* that document or propose a change to the source document.

Comment A subclass of *Annotation* describing annotations that are comments.

Example A subclass of *Annotation* representing examples.

Explanation A subclass of *Annotation* representing explanations of content.

Question A subclass of *Annotation* representing questions about the content.

SeeAlso A subclass of *Annotation* representing a reference to another resource.

Figure 4: Basic annotation classes.

design, RDF permits separate communities to develop independent metadata vocabularies and then freely mix statements using those vocabularies in a single database of triples. In RDF, the property names themselves are Web resources, and applications can use the URIs of those properties to make other statements about the properties themselves, such as their meaning and their relationship to other properties.

The type of an annotation is defined by the user or the group by declaring additional annotation classes. These classes are a part of the RDF model and may be described on the Web in an RDF Schema [5]. The general annotation super class is called *Annotation* (more precisely, its name is `http://www.w3.org/2000/10/annotation-ns#Annotation` – a URI about which an application can expect to ask the Web for more information) and we have defined several sample subclasses based on it (Fig. 4). These subclasses are defined in a separate RDF Schema whose namespace is `http://www.w3.org/2000/10/annotationTypes#`. Likewise, other user groups can easily create new subclasses. We can also easily add new properties to the annotation classes, for instance, we could add a property that defines an annotation set. This property can then be queried with general RDF mechanisms and also presented as text. However, to do more advanced presentations with the basic RDF mechanisms we would need to develop presentation schemas for RDF.

Annotations are user made statements that consist of these main parts: the body of the annotation, which contains the textual or graphical content of the annotation, the link to the annotated document with a location within the document, an identification of the person making the annotation and additional metadata related to the annotation. By using RDF we can take advantage of other work on Web metadata vocabularies wherever possible. Specifically, we use the Dublin Core [9] element set to describe some of the properties of annotations. The annotation properties are illustrated in the RDF model presented in Figure 5 and the

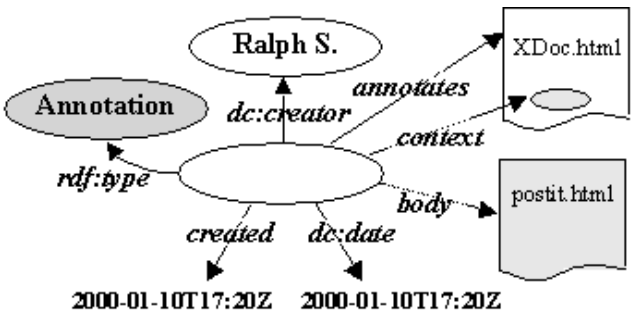

Figure 5: The RDF model of an annotation.

corresponding schema definitions for properties are defined in Figure 6.

The RDF schema that defines the annotation properties consists of the property name and the natural language explanation. The type is one of the basic classes in Figure 4 or some other type of annotation defined elsewhere. The *annotates* property stores the link to the annotated document, *body* is a link to the content of the annotation, and *dc:creator* to the author making the annotation.

The *context* defines where exactly inside the document the annotation is attached. We use XPointer [7] for defining positions within XML documents. This works well for static (unchanging) documents, but with documents that go through revision, such as working group drafts, we may end up with orphan annotations or annotations pointing to wrong places. To prevent unnecessary loss of pointers we can search for the nearest ID to a parent of the object use it as the starting point for the XPointer path. Fortunately, many documents usually have IDs at least at their main levels. Pointing to finer details after the ID can be done by other XPointer means, such as using text matching.

The additional annotation metadata includes date for the creation and last modified time, and *related* for adding relationships to other objects. Other metadata can be added to the annotation when the working group needs that. For instance, the working group will probably add their own properties directly and not specialize the *related* property.

Sample annotations utilizing this schema definition are presented in Appendix A while discussing the protocols.

3. ANNOTATIONS IN AMAYA

One of the goals of Annotea is to help us gain experience on building an RDF infrastructure. Since the beginning of the project, we have been implementing both a client and a server prototype. For the client, we have been using Amaya, W3C's testbed editor browser. For the server, we have been using Apache, a MYSQL database running on top of it and some Perl scripts. The rest of this section describes the implementation choices we have made in Amaya 4.0.

Amaya [1] is a full-featured web browser and editor developed by W3C for experimenting and validating web specifications at an early stage of their development. Amaya supports CSS, MathML, XHTML, HTML, and also provides a basic implementation of XLink and XPointer. Libwww [17] is linked to Amaya and provides HTTP/1.1 support and an RDF parser. Amaya can also show different views of a document. In particular, we have a Formatted view, which shows the interpreted document, and a Links view, which gives a

rdf:type An indication of the creator's intention when making an annotation; the value should be of rdf:type *Annotation* or any of its subclasses.

annotates The relation between an annotation resource and the resource to which the annotation applies.

body The content of the annotation.

context Context within the resource named in annotates to which the annotation most directly applies. Eventually this will be an XPointer. It may include a location range too. First locations will points to XML IDs.

dc:creator The creator of the annotation.

created The date and time on which the annotation was created.

dc:date The date and time on which the annotation was last modified.

related A relation between an annotation and a (collection of) resource(s) that augment the resource that is the *body* of the annotation. This may point to related issues, discussion threads, etc.

Figure 6: The basic annotation properties.

list of all the links in the document.

Our prototype implementation is able to interpret the complete Annotation RDF schema and supports all of the Annotea protocols as described in Appendix A. It is also possible to specify additional annotation types (subclasses) as an RDF schema that can be can be downloaded at runtime. The namespaces for these additional types are specified to Amaya in a local configuration file that is read at startup. Amaya will use the namespace name to try to retrieve an RDF schema from the Web or the schema content can be cached in a local file and specified with the same startup configuration. The prototype is not yet able to recognize the need to download schemas dynamically from the information given in annotations metadata.

We will now describe the most important features of our implementation: creating an annotation, browsing annotations, and filtering annotations.

3.1 Creating an annotation

The user has three choices for creating an annotation: annotate a whole document, annotate the position where the caret is, annotate the current selection. After making this choice, a popup annotation window appears. The annotation window shows the metadata of the annotation, as defined in Section 2.3, inside a box and the body of the annotation. Figure 7 shows a screen capture of Amaya when creating an annotation on a selection.

Three of the metadata items are active. If the user clicks on the *Source document* field, Amaya will scroll to the annotated text and highlight it if it is a selection. Clicking on the *Annotation type* field allows the user to change the type of annotation. Finally, each time that the user saves the annotation Amaya updates the value of the *Last modified* field. Note that we do not show the value of the XPointer (context), but rather use it to select the source document highlighting.

The body of the annotation can be edited as any other XHTML document. Users can cut and paste fragments from other documents, add links to other documents, and so on.

Amaya support both local (private) and remote (shared) annotations. When a user creates an annotation, it is considered a local one and will be stored in the user's Amaya directory. When the user decides to post it to an annotation server, the local annotation will be deleted and subsequent saves will be sent to the server. Both local and remote annotations are represented with the same schema format. The only difference is that for local ones, we emulate the annotation server's query response by using an index file that associates URIs with annotation metadata.

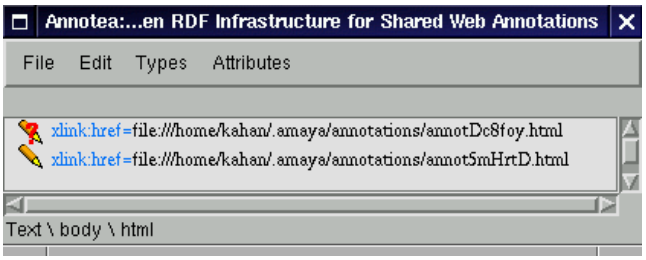

Figure 8: The Links View showing an orphan annotation and a normal one.

3.2 Browsing annotations

By means of a setup menu, the user can specify the URIs of the annotation servers he wants to query, as well as the local annotation repository. The user can also say if he wants annotations to download automatically or only on-demand. In order to avoid hampering performance, we separated the downloading process in two steps. Once a document is downloaded, the annotations metadata is downloaded asynchronously, just like images, and merged into the document. The body of an annotation is only downloaded when the user opens an annotation. The motivation for this choice is that metadata may be relatively smaller than the body of an annotation. Moreover, if the user does not open all the annotations, we save time by not downloading the body.

For showing annotations, we defined an active XML element, that we will call A-element, that has an XLink pointing to the body of the annotation and a special icon (currently, a pencil). This is similar to the X element that was used in the Annotated XML specification [3], with the difference that in Amaya, it is an active element. When the user clicks once on the A-element, Amaya highlights the target of the annotation. Clicking on it twice will open the annotation window and show both the metadata and the body of the annotation. The A-element is visible in both the Formatted Document and Links views and it is ignored when saving or printing an annotated document. Clicking on the A-element on any view has the same effect.

In the Formatted view, we position the A-element to the location to which the XPointer of the annotation resolves. We made an exception for MathML documents, as it would be disturbing to add it anywhere in the expression. Instead, we place it as the the beginning of the Math expression. Clicking on the A-element will highlight the target of the annotation, even if this target is not close to the A-Element.

If an annotated document is edited, some of its annotations may become orphan. That is, the XPointer will not resolve anymore to a point in the document. In this case, Amaya will warn the user and make the orphan annotation visible from the Links view. Figure 8 shows this view in a document that has an orphan and a valid annotation. The

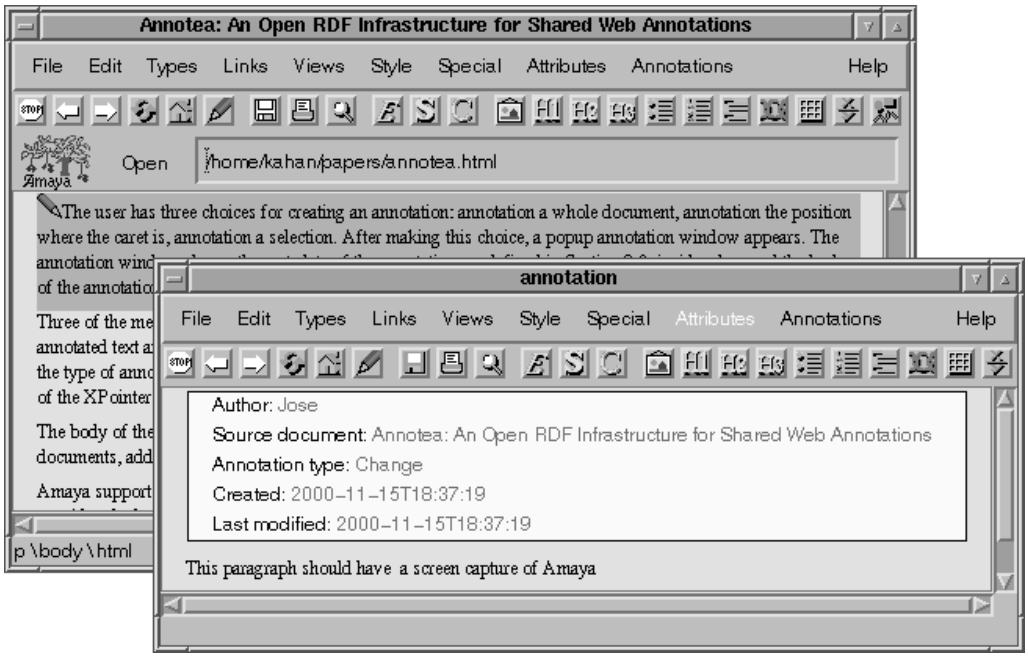

Figure 7: Annotating a paragraph with Amaya.

user may then open the orphan annotation and reposition its XPointer or delete it.

3.3 Filtering annotations

For a heavily annotated document, seeing the A-element icon can make reading the document bothersome. To avoid this problem, we defined a local filter that allows the user to hide the annotations according to one of three criterion: by author name, by annotation type, and by annotation server. It is also possible to hide all the annotations in the Formatted view. Using this menu, the user can hide all but the annotations that interest him. This filter menu does not have any effect on the Links view.

As an alternative to hiding annotations, the user can also temporarily disable some annotation servers using the configuration menu. We also have an experimental customized query feature, where the user can describe his own query, using a language we have named "Algae". The Algae language is derived from Algernon [4]. This customized query interface makes it possible to start filtering the annotations on the server side, for example, by only requesting those done in the past week by a given author and belonging to a given annotation type. Appendix B[1] gives a brief description of Algae.

4. RELATED WORK

This section discusses some previous annotation approaches. We concentrate on document-centered approaches where users are browsing documents and examining annotations related to them. There are also discussion-centered approaches to annotations, such as HyperNews [12], where users browse discussion messages and threads and follow a link to a document that these messages annotate.

Web annotations first appeared in version 1.2 of Mosaic [18, 19], almost ten years ago, and many other web annotation aware tools or servers have seen the light since then, such as CritLink [25] and ThirdVoice [23]. [10, 11] list other existing annotation technologies. Due to the lack of existing annotation standards, most of these proposals are proprietary or closed.

The two main categories to Web annotation systems are *proxy-based* approaches and *browser-based* approaches. In a *proxy-based* approach, annotations are stored and merged with a Web document by a proxy server. The browser user agent only sees the result of the merge, typically with some semantic content removed. In a *browser-based* approach the browser is enhanced (either by an external application or by a plugin) to merge the document and the annotation data just prior to presenting the content to the user. The annotation data is stored in the proxy or a separate annotation server. It is also possible to store annotations locally or provide site specific annotations, but these are less interesting to us because of their limitations.

The CritLink [25] annotation tool uses the proxy approach where a Web page and its annotations are served through a different URI address than the original page. This approach works with any existing browser. However, the user must use different addresses for the document depending on which annotation proxy server is used. This is a limitation when a user wants to use more than one annotation server. The proxy approach also inherently restricts the types of content that can be annotated and the presentation styles that can be used for the annotations. Typically, presentation of the annotations is limited to the presentation styles available through HTML. Finally, as the browser does not have any knowledge about annotations, it makes it harder to filter the annotations locally, without having to send a new request to the proxy server.

ThirdVoice [23] uses plugins to enhance web browsers so that they understand annotations. The users can annotate

[1] Available only in the HTML version of this paper.

the page or some text on the page with discussions on selected topics. The discussions can be closed to a group of participants or open to anybody. Unfortunately, users cannot host their own servers.

IMarkup [13] is an Internet Explorer annotation tool that has an interesting user interface. Users have a wide variety of palettes for annotation markers and can even circle parts of the text with something akin to a real marker. Annotations can be placed anywhere on the browser's document window, without taking into account the markup of the document itself. All the annotations are local. A menu entry allows to mail annotations to other users and to import them. The format used for describing annotations is proprietary and too related to the browser's API, making their use with other tools practically impossible.

An interesting possibility for presenting the annotations on a Web page is to use internal DOM [14] events without actually changing the mark-up of the page. Yawas [6] is an annotation tool that uses this approach. It codes the annotations into an extended URI format and uses local files similar to bookmark files to store and retrieve the annotations. A modified browser can transform the URI format into DOM events. The local annotation files can be sent to other users only by mail or copied by other means. There is no provision for having active links or filtering options. This kind of approach is limited by the API provided by the browser.

XLink [8], an XML linking technology currently under development in W3C, has some built in features in the mark-up for creating annotations. For instance, it is possible to store XLink arcs in an external document that can be loaded with another document. The content defined by the end locator of an XLink arc may be embedded to the location in a document defined by a starting locator of the arc. Using XLink provides the means to easily present the annotations in predefined ways in any browser implementing XLink. However, the metadata properties that can be expressed with XLink are limited.

5. CONCLUSIONS AND FUTURE PLANS

Being able to associate metadata with Web resources is an important milestone for building a Semantic Web. Annotea provides a simple infrastructure for associating annotations with Web documents, without having to modify these documents. The principal contributions of Annotea are as follows:

- Annotations are metadata. Annotea is not designed as a specific annotation system, but rather as a general application of a generic RDF infrastructure. This allows a variety of applications to reuse the information that is stored in an Annotea system with other RDF tools that are not necessarily specific to annotations.

- Open infrastructure. Annotea is built on top of W3C specifications. We use an RDF schema for describing the properties of annotations, XPointer for associating annotations to documents, and HTTP for the client/server interactions.

- Use of RDF databases. By storing annotations inside RDF databases, it is possible to make customized queries and limit the amount of data returned by the servers.

- Extensible RDF schema. Our annotation RDF schema defines general properties about annotations. Users can extend it by defining their own annotation types or by adding other annotation properties.

- Client-less. Annotea defines an infrastructure for associating metadata with documents and for storing and retrieving this metadata. In principle, it is possible to build an Annotea client on top of any browser that handles DOM, XPointer, XLink and RDF.

In November 2000, we made the first public release of the Annotea prototypes. The client is included as a built-in feature of the Amaya 4.0 release. We have also set up a public annotation server [2]. All the source code is freely available too. The public server is not intended to be a permanent service, but rather one that will be purged periodically. Its goal is to let people experiment with annotations and motivate them to set up their own servers.

Our wish list for future work on Annotea includes:

- Shared bookmarks. Shared bookmarks are quite similar to annotations. The annotation schema provides a set of fixed annotation types. The user is expected to classify his annotation by selecting one of these types. With shared bookmarks, the user should be able to define his own types on-the-fly, for example, by highlighting keywords on the annotated document. These types can then be used to automatically classify the bookmarks.

- User interface. Currently, a user can see annotations either as pencil icons next to the fragment that was annotated or in a special Links view. We would like to experiment with other ways for displaying annotations. For example, by embedding the body of the annotations in the annotated document.

- Author metadata. The Annotation schema defines the author as a string. We plan to expand the schema so that the author is defined by another RDF schema and use this property in the Annotation schema. It will then be easy to search for the metadata of the author and, for example, substitute the pencil icon with the photo of the author.

- Robust XPointers. Currently, we are able to detect orphan annotations. However, our XPointer expressions are very simple. If a user edits an annotated document, in some cases, the XPointer of an annotation may point to the wrong place and thus become a misleading annotation. We have made some provisions for this case (use of the ID attribute), but this is not enough. A better XPointer expression would be one that is more tolerant of document changes, but robust enough to prevent misleading annotations.

- Discussion threads. When a user wants to reply to an annotation, s/he can either modify the body of the annotation or make a new annotation. This can become cumbersome as we would need to browse each annotation in order to follow the discussion. We can improve this situation by adding new RDF properties for distinguishing such discussions and by showing all the replies to a given annotation in a specialized view.

6. ACKNOWLEDGEMENTS

The authors would like to thank Art Barstow, Tim Berners-Lee, Dan Brickley, Daniel LaLiberte, and Charles McCathie-Nevile for their useful feedback and suggestions concerning Annotea. Irène Vatton and Vincent Quint from the Amaya team have given irreplaceable help to us. Eric Miller gave us very enthusiastic encouragement to pursue this application as a testbed for RDF infrastructure.

7. REFERENCES

[1] Amaya Home Page. http://www.w3.org/Amaya/.

[2] Annotest (Annotea Test Server) Home Page. http://annotest.w3.org/.

[3] T. Bray. Using XML to Build the Annotated XML Specification, Sep. 1998. http://www,xml.com/xml/pub/98/09/exexegesis-0.html.

[4] J. Crawford. *Access-Limited Logic: A Language for Knowledge Representation.* UT Artificial Intelligence TR AI90-141, Department of Computer Sciences, University of Texas at Austin, Austin, Texas., Oct. 1990.

[5] D. Brickley and R.V. Guha (eds.). Resource Description Framework (RDF) Schema Specification 1.0. CR, W3C, Mar. 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327.

[6] L. Denoue and L. Vignollet. An annotation tool for Web browsers and its applications to information retrieval. In *Proceedings of RIAO200*, Apr. 2000. http://www.univ-savoie.fr/labos/syscom/Laurent.Denoue/riao2000.doc.

[7] S. DeRose, R. D. Jr., and E. M. (eds.). XML Pointer Language (XPointer). WD, W3C, Dec. 1999. http://www.w3.org/TR/1999/WD-xptr-19991206.

[8] S. DeRose, E. Maler, D. Orchard, and B. T. (eds.). XML Linking Language (XLink). WD, W3C, Feb. 2000. http://www.w3.org/TR/2000/WD-xlink-20000221.

[9] Dublin Core Metadata Element Set, Version 1.1: Reference Description. Technical report, Dublin Core Metadata Initiative, Jul. 1999. http://purl.org/DC/documents/rec-dces-19990702.htm.

[10] J. Garfunkel. Web Annotation Technologies, 1999. http://ps.pageseeder.com/ps/ps/papers/annot/jongar/jongar.pshtml.

[11] R. M. Heck, S. M. Luebke, and C. H. Obermark. A Survey of Web Annotation Systems, 1999. http://www.math.grin.edu/~luebke/Research/Summer1999/survey_paper.htm%l.

[12] HyperNews Home Page. http://www.hypernews.org/.

[13] IMarkup Home Page. http://www.imarkup.com.

[14] L. Wood et al. (eds.). Document Object Model (DOM) Level 2 Specification Ver. 1.0. CR, W3C, Mar. 2000. http://www.w3.org/TR/2000/CR-DOM-Level-2-20000307.

[15] Y. Lafon and B. Bos. Describing and Retrieving Photos Using RDF and HTTP. Note, W3C, Sep. 2000. http://www.w3.org/TR/photo-rdf/.

[16] D. LaLiberte and A. Braverman. A Protocol for Scalable Group and Public Annotations, 1996. http://www.hypernews.org/~liberte/www/scalable-annotations.html.

[17] libwww Home Page. http://www.w3.org/Library/.

[18] *NCSA Mosaic Documentation: Group Annotations in NCSA Mosaic*, 1993. http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/group-annotations.htm%l.

[19] *Mosaic Users's Guide:Annotations*, 1998. http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-on-annotate-win.%html.

[20] O. Lassila and R. R. Swick (eds.). Resource Description Framework (RDF) Model and Syntax Specification. Recommendation, W3C, Feb. 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[21] R. Fielding et al. (eds.). Hypertext Transfer Protocol – HTTP/1.1. RFC RFC2616, IETF, Jun. 1999. http://www.ietf.org/rfc/rfc2616.txt.

[22] Semantic Web Development. http://www.w3.org/2000/01/sw/.

[23] ThirdVoice Home Page. http://www.thirdvoice.com/.

[24] XML Query Working Group. http://www.w3.org/XML/Group/Query.

[25] K.-P. Yee. CritLink: Better Hyperlinks for the WWW. Submitted to Hypertext '98, Apr. 1998. http://crit.org/http://crit.org/~ping/ht98.html.

[26] R. Zohar. Web Annotation - an Overview, Feb. 1999. http://www-ee.technion.ac.il/~ronz/annotation/.

APPENDIX

A. ANNOTEA PROTOCOLS

We distinguish five types of client-server interactions in Annotea:

- **Posting** a new annotation: the client publishes a new annotation
- **Querying** the annotation server: the client sends a query to the server and gets backs the annotation
- **Downloading** the body of an annotation
- **Updating** an annotation: the client modifies an annotation and publishes these modifications
- **Deleting** an annotation: the client deletes an annotation from the server

For all of these cases, we use the standard HTTP protocol methods. We use HTTP POST for uploading a new annotation to a server, HTTP PUT to update an annotation, HTTP GET to query and download an annotation, and HTTP DELETE to delete an annotation. We will now describe each of these operations in detail.

We use the standard HTTP POST protocol for storing a new annotation to the annotation server and HTTP GET protocol for fetching the annotations and returning the result to the client. POST provides the necessary interface for the server to construct a URI for the new annotation and return that URI to the client. When the client has the URI for a previously created annotation, it can (with the proper permissions) use HTTP PUT to modify the annotation. In all the examples, we use the Apache shorthand CGI convention, using the string `annotations` to refer to the actual CGI script. This makes it easier to refer to an existing annotation.

A.1 Posting a new annotation

To create a new annotation, the client posts some RDF describing the annotation to a selected annotation server. Both the annotation and its body are specified as anonymous RDF resources in the POST message. The server is responsible for allocating the URIs for them. If the body already exists, as will happen if the annotation body is another document that the user wants to use as an annotation, the URI of that existing document can be specified in the RDF when the annotation is posted.

In Figure 9 we illustrate a request to create a simple annotation using an existing document as the body of the annotation. Note that the resource `http://www.example.com/mycomment.html` is presumed to exist independently of this annotation.

```
POST /annotations HTTP/1.1
Host: www.example.org
Content-Type: application/xml
Content-Length: 636

<r:RDF xmlns:r="http://www.w3.org/1999/02/
22-rdf-syntax-ns#"
       xmlns:a="http://www.w3.org/2000/10/
annotation-ns#"
       xmlns:d="http://purl.org/dc/elements/1.1/">
 <r:Description>
  <r:type resource="http://www.w3.org/2000/10/
annotation-ns#Annotation"/>
  <r:type resource="http://www.w3.org/2000/10/
annotationType#Comment"/>
  <a:annotates r:resource="http://example.com/some/page.
html"/>
  <a:context>#xpointer(id("Main")/p[2])</a:context>
  <d:creator>Ralph Swick</d:creator>
  <a:created>1999-10-14T12:10Z</a:created>
  <d:date>1999-10-14T12:10Z</d:date>
  <a:body r:resource="http://www.example.com/mycomment.
html"/>
 </r:Description>
</r:RDF>
```

Figure 9: Creating an annotation with POST, using an existing document as the body.

A design issue we encountered is that we wanted to be able to use XML for describing the body of an annotation, and at the same time we wanted to be able to publish the complete annotation in a single HTTP transaction. In order to use XML in the body, the correct architectural approach is to store the body as a separate resource with its own content type. We therefore designed a simple packaging protocol that permits both the client and server to specify embedded HTTP message bodies. To do this, we declare an RDF namespace for describing certain HTTP headers and we specify those HTTP headers as normal RDF properties, as shown in Figure 10.

In Figure 10, we show the metadata that specifies an annotation of the page whose URI is `http://example.com/some/page.html`. The creator of this annotation is identified as `Ralph Swick`. The text of the annotation body is `This is an important concept`.

As specified by the RDF model, the data we pass to the server in the POST is a set of statements describing properties of the new (and unnamed) annotation resource that we would like the server to create. In response to the POST (Fig. 11), a new annotation is created and the server assigns

```
POST /annotations HTTP/1.1
Host: www.example.org
Content-Type: application/xml
Content-Length: 1082

<r:RDF xmlns:r="http://www.w3.org/1999/02/
22-rdf-syntax-ns#"
       xmlns:a="http://www.w3.org/2000/10/
annotation-ns#"
       xmlns:d="http://purl.org/dc/elements/1.1/"
       xmlns:h="http://www.w3.org/1999/xx/http#">
 <r:Description>
  <r:type resource="http://www.w3.org/2000/10/
annotation-ns#Annotation"/>
  <r:type resource="http://www.w3.org/2000/10/
annotationType#Comment"/>
  <a:annotates r:resource="http://example.com/some/page.
html"/>
  <a:context>#xpointer(id("Main")/p[2])</a:context>
  <d:creator>Ralph Swick</d:creator>
  <a:created>1999-10-14T12:10Z</a:created>
  <d:date>1999-10-14T12:10Z</d:date>
   <a:body>
    <r:Description>
     <h:ContentType>text/html</http:ContentType>
     <h:ContentLength>250</http:ContentLength>
     <h:Body r:parseType="Literal">
  <html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Ralph's Annotation</title>
  </head>
  <body>
  <p>This is an <em>important</em> concept; see
    <a href="http://example.com/other/page.
html">other page</a>.</p>
  </body>
  </html>
    </h:Body>
   </r:Description>
  </a:body>
 </r:Description>
</r:RDF>
```

Figure 10: Creating an annotation with POST.

URIs. Now the server has created the URIs for the anonymous resources and they can be used by the browser. The value of the `a:body` property is a URI of the content of the annotation; in this case the server implementation chose to store the text in a separate location and give it its own URI.

With this little bit of ad hoc packaging we can have a POST method that explicitly creates two resources at the same message and a GET method that returns these same resources in one message. This packaging protocol has the additional advantage that it makes POST and GET of multiple resources an atomic operation; there is no window in which another client might modify the annotation body after the annotation properties have been returned but before the body is returned.

A.2 Querying an annotation server

An annotation server is queried for the URIs of annotations it may hold using the GET method. Since the client will most commonly wish to query for annotations that have an `annotates` property naming a specific page that the user may currently be viewing, a particular query parameter is designated to pass the URI of that page, as shown in Figure 12.

The query parameter `w3c_annotates` may be best thought

```
HTTP/1.1 201 Created
Location: http://www.example.org/Annotation/3ACF6D754
Content-Type: application/xml
Content-Length: 404

<r:RDF xmlns:r="http://www.w3.org/1999/02/
22-rdf-syntax-ns#"
       xmlns:a="http://www.w3.org/2000/10/
annotation-ns#"
       xmlns:d="http://purl.org/dc/elements/1.1/">
 <r:Description about="http://www.example.org/
Annotation/3ACF6D754">
  <a:annotates r:resource="http://example.com/some/page.
html"/>
  <a:body resource="http://www.example.org/Annotation/
3ACF6D754text"/>
 </r:Description>
</r:RDF>
```

Figure 11: Sample response when creating a new annotation.

```
GET /annotations?w3c_annotates=http://example.com/some/
page.html HTTP/1.1
Host: www.example.org
Accept: application/xml
```

Figure 12: A query for annotations related to http://example.com/some/page.html.

of as an abbreviation for the longer property name `http://www.w3.org/2000/10/annotation-ns#annotates`; that is, this GET is a short-hand for a query that says "return the names of resources that are the subjects of RDF statements in which the predicate is `http://www.w3.org/2000/10/annotation-ns#annotates` and the object is `http://example.com/some/page.html`". The server responds to this GET request by returning RDF/XML describing the properties of each annotation that has an annotates relationship to the given URI. In the first release of our server implementation, we return all the properties of each annotation including the URI of the body resource. Figure 13 illustrates a typical response; in this case there is only one annotation for the specified page.

A.3 Downloading an annotation

An annotation is downloaded from an annotation server using the GET method and specifying the annotation URI, as returned in a query response (Fig. 14).

The response to this GET will be as in Figure 13.

A.4 Updating an annotation

An existing annotation is updated using the PUT method, specifying the URI of the annotation we wish to update. For example, to update the annotation created in the messages illustrated in Figures 10 and 11 above, we might specify the message in Figure 15.

A.5 Deleting an annotation

An annotation is deleted using the DELETE method, specifying the URI of the annotation we wish to remove. For example, to delete the annotation created in the messages illustrated in Figures 10 and 11 above, we might specify the message in Figure 16.

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 689

<r:RDF xmlns:r="http://www.w3.org/1999/02/
22-rdf-syntax-ns#"
       xmlns:a="http://www.w3.org/2000/10/
annotation-ns#"
       xmlns:d="http://purl.org/dc/elements/1.1/">
 <r:Description about="http://www.example.org/
Annotation/3ACF6D754">
  <r:type resource="http://www.w3.org/2000/10/
annotation-ns#Annotation"/>
  <r:type resource="http://www.w3.org/2000/10/
annotationType#Comment"/>
  <a:annotates r:resource="http://example.com/some/page.
html"/>
  <a:context>#xpointer(id("Main")/p[2])</a:context>
  <d:creator>Ralph Swick</d:creator>
  <a:created>1999-10-14T12:10Z</a:created>
  <d:date>1999-10-14T12:10Z</d:date>
  <a:body r:resource="http://www.example.com/mycomment.
html"/>
 </r:Description>
</r:RDF>
```

Figure 13: A typical response to the query in Figure 12.

```
GET /annotations/3ACF6D754 HTTP/1.1
Host: www.example.org
Accept: application/xml
```

Figure 14: Downloading a specific annotation.

```
PUT /annotations/3ACF6D754 HTTP/1.1
Host: www.example.org
Content-Type: application/xml
Content-Length: 657

<r:RDF xmlns:r="http://www.w3.org/1999/02/
22-rdf-syntax-ns#"
       xmlns:a="http://www.w3.org/2000/10/
annotation-ns#"
       xmlns:d="http://purl.org/dc/elements/1.1/">
 <r:Description about="http://www.example.org/
Annotation/3ACF6D754">
  <r:type resource="http://www.w3.org/2000/10/
annotation-ns#Annotation"/>
  <r:type resource="http://www.w3.org/2000/10/
annotationType#Example"/>
  <a:annotates r:resource="http://example.com/some/page.
html"/>
  <a:context>#xpointer(id("Main")/p[2])</a:context>
  <d:creator>Ralph Swick</d:creator>
  <a:created>1999-10-14T12:10Z</a:created>
  <d:date>1999-10-14T13:14Z</d:date>
  <a:body>
    ...
  </a:body>
 </r:Description>
</r:RDF>
```

Figure 15: Updating an annotation using PUT.

```
DELETE /annotations/3ACF6D754 HTTP/1.1
Host: www.example.org

HTTP/1.1 200 OK
```

Figure 16: Deleting an annotation using DELETE.