Design and Implementation of an XML-based Personal Web Annotation Tool

Min-Chi Tzeng, Cheng-Zen Yang and Shen-Chi Chen

Department of Computer Engineering and Science
Yuan Ze University
135 Yuantung Road
Chungli, Taiwan, R.O.C.
{angeline,czyang,shenchi}@syslab.cse.yzu.edu.tw

Please forward correspondences to Cheng-Zen Yang
e-mail: czyang@syslab.cse.yzu.edu.tw
phone:+886-3-4638800 ext 361
fax:+886-3-4638850

This paper is intended for the Workshop on Multimedia Technology.

Abstract

When people read paper documents, they often actively work with the text by highlighting and underlining passages of the text. However, the annotation activities are not considered in the native infrastructure of Web electronic publishing. Though many augmented annotation systems have been proposed, they do not simultaneously address the personalization openness, and extensibility issues that are important for user-centric annotation. In this paper, we present the design and implementation of a personal annotation tool called WebPAT. It adopts an open architecture in its native design by using JavaScript and XML. In addition, it is highly extensible because users can customize WebPAT with normal editors. Furthermore, its functionalities are all user-centric and can be performed without interaction to remote servers. To date, a preliminary prototype has been implemented on the basis of Microsoft Internet Explorer. From the working experience, though WebPAT currently lacks some fancy drawing facilities, the prototype demonstrates its open and extensible annotation architecture.

Keywords: Web Annotation, Personalization, Openness, Extensibility, XML.

1 Introduction

People often annotate paper documents as they read them by underlining text, writing comments, or highlighting text in the reading materials. These common reading activities extend the original content with personal opinions and viewpoints. However, these annotation activities are not considered in the native design of Web electronic publishing. People can hardly make annotations just with popular commercial Web

browsers such as Microsoft Internet Explorer and Netscape's Navigator. Because of the emerging need of Web annotation, many Web annotation systems are developed [1, 3, 5, 6, 8, 9, 10, 11, 13] since Mosaic 1.2 [7]. Though these systems provide tools to facilitate Web annotation, few systems consider the annotation functionality from the viewpoint of users.

We argue that Web annotation should be provided in a user-centric manner. That is, a Web annotation system should provide three functionalities from the viewpoint of users. First, an annotation tool should allow users to freely make annotations, personally design annotation types, and extend the tool at their own will. Second, the tool should be designed in an open architecture. Therefore, users have the complete authority to manage the tool and they can update any component without restrictions. Third, the tool should be highly extensible for future customized extensions. Otherwise, the tool will not keep pace with the evolution progress of computer technology. Therefore, we start to design a personal annotation tool called WebPAT (Web Personal Annotation Tool).

To provide these user-centric functionalities, WebPAT adopts JavaScript and XML as the core implementation technologies. JavaScript is chosen for three reasons. First, JavaScript is powerful to solve many frequently needed tasks such as verifying input-forms, providing multiple browser windows, and displaying warning messages. In addition, JavaScript programs can be designed in a simple manner. Furthermore, JavaScript is widely supported in commercial browsers. Similarly, some annotation systems [1, 3, 5, 6, 9] are also developed in JavaScript. However, none of them uses XML to describe the underlying annotation data.

As some annotation systems [11, 13], WebPAT uses XML (eXtensible Markup Language) in describing data and exchanging data. However, other XML-based systems do not provide an open architecture as WebPAT. Since XML is used to describe the annotation data, WebPAT is benefited from many XML advantages such as DHTML simplicity and extensibility. It also separates the content from the presentation. Therefore, WebPAT is highly extensible for future extensions.

Though CGI (Common Gateway Interface) shows its flexibility in several annotation systems [4, 5, 6, 10], it is not considered because the WebPAT design does not involve any remote server. Another popular Web technology Java is not taken into consideration because programming it needs knowledge of compilation and a complicate programming environment. Other browser-specific technologies such as are disregarded, because they do not provide an open environment.

In this paper, we will present the user-centric design and implementation of WebPAT to illustrate the design issues of a personal Web annotation tool. The rest of this paper is organized as follows. Section 2 briefly reviews different implementation systems. Section 3 describes WebPAT architecture and design issues. Section 4 presents several operational scenarios. Section 5 concludes this paper.

2 Related Work

Many previous annotation systems are designed from a viewpoint of central management [2, 4, 6, 11, 13]. In these systems, users need to login a remote server before they annotate the documents. Furthermore, annotations are usually stored at the remote server. The author has limited management control. For example, Futplex [4] is a software package that provides shared documents with read/write access on the World Wide Web. It is implemented as a collection of CGI scripts on an NCSA server. However, Futplex does not provide any annotation marks on original documents. Therefore, users cannot notice the annotations easily.

DCRS [13] is another example but it is implemented with the concept of XML-based information object. With XML-based information objects, DCRS provides a cooperative workgroup environment to exchange and share documents in an easy and effective way. The XML-based model facilitates the system design and provides good extensibility for future development. However, DCRS is designed for a specific workgroup. It does not address the openness issue.

WISPA [11] is also a server-based system that aims at mining useful information from a community by sharing personal annotations. It uses XML to structure personal annotations and XLink to integrate personal annotations. However, WISPA is not an open system because its private system interface design.

Some annotation systems [1, 3, 5, 6, 9] have paid attention to the openness issue. YAWAS [3] is such an example designed in Java and JavaScript. However, the YAWAS design does not consider extensibility. Future extensions are restricted.

In a three-tier architecture, the annotation system can be positioned at the middle proxy tier [8, 12, 14]. All annotation operations are monitored by the annotation proxy. To browse or create annotations, users need to first specify the proxy position. Annotator is such an example [8]. When a browser sends an HTTP request to the annotation proxy, the proxy forwards the request to the Web server and integrates the Web page with the related annotations. However, this approach has a severe failure problem if the proxy is out of

service due to some failures. Other proxy-based annotation systems such as GrAnT and CritLink [12, 14] have the similar disadvantage as Annotator. In addition, Annotator is not an open and user-centric design because it is based on a modified Netscape browser.

Compared with previous annotation systems, WebPAT considers the personalization, openness, and extensibility issues. There are two additional design features in WebPAT. First,WebPAT provides a mechanism to invoke external applications for handling user-defined data types. This allows WebPAT to handle user-extended annotation types. Second, WebPAT is portable to other browsers that support embedded JavaScript. Therefore, users do not need to design their own browsers to support personal annotation.

3 Architecture and Design

Annotating Web pages is an important reading activity when users browse the Web pages. From the viewpoint of personalization, WebPAT does not consider a client-server design. All annotations are maintained at the local browser end. Therefore, users need not login to a remote server to make annotations. They can freely make annotations and browse them. In addition, the WebPAT design allows users to personally design annotation types and to extend the tool's functionalities at their own will. These are achieved by the native JavaScript and XML design. In the following, we will first describe the WebPAT architecture. Then the detailed design is elaborated.

3.1 The WebPAT Architecture

WebPAT is built on the basis of Microsoft Internet Explorer to support personal Web annotations. Figure 1 depicts its architecture. Through the extended IE browser, WebPAT allows users to make annotations on remote or local Web pages. The created annotations are stored at local storage. Annotations are in XML format to facilitate future extensions.

3.2 The Design Details

The WebPAT modules include a toolbar interface (TI) and an annotation processor (AP). TI receives requests from users and invokes AP to process them. According to different operation requirements, AP retrieves the related annotations from local annotation files, processes the annotations, and presents the results in Web

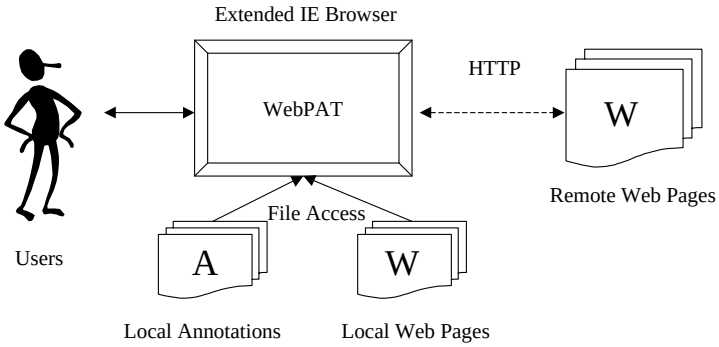

Figure 1: The WebPAT architecture.

pages. TI and AP are all implemented in JavaScript for openness purpose. The underlying annotations are structured in XML.

3.2.1 Toolbar Interface

The WebPAT modules are divided into two parts: a browser-dependent TI module and a browser-independent AP module. TI is IE-dependent and takes the responsibility for graphical interaction with users. Currently, We have embedded eight buttons in IE's toolbar area as illustrated in Figure 2.

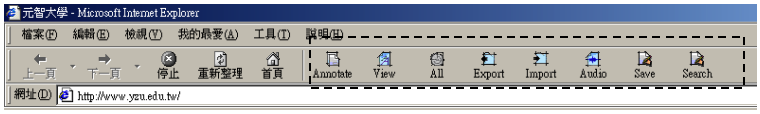

Figure 2: The toolbar interface has eight buttons.

The correspondent JavaScript file architecture is depicted in Figure 3. In the IE toolbar, TI has eight buttons. Each button is created by registering it in the Windows registry to have a link pointing to a specific HTML file for the proper annotation interface. *Regedit* is used to complete the registration and link association. To complete the registration, the following information is needed: Button Text, CLSID, Default Visible, HotIcon, Icon, and Script. Figure 4 show a screen snapshot of the registered entries. The button information is registered at the following path `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Extensions`.

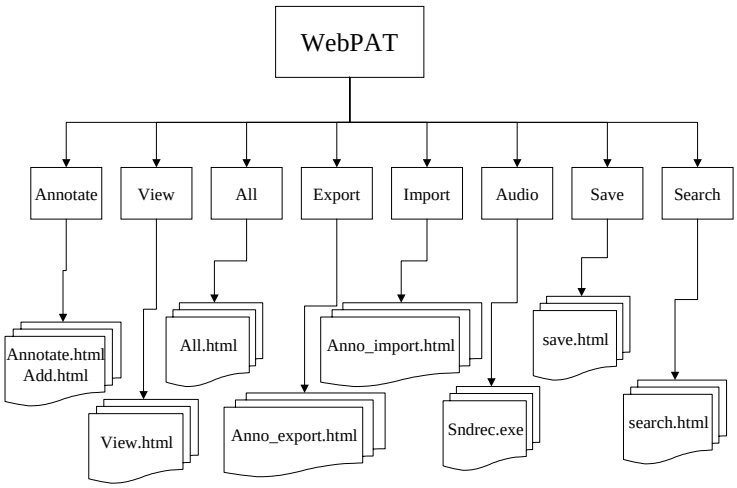

Figure 3: The HTML files in the WebPAT design.

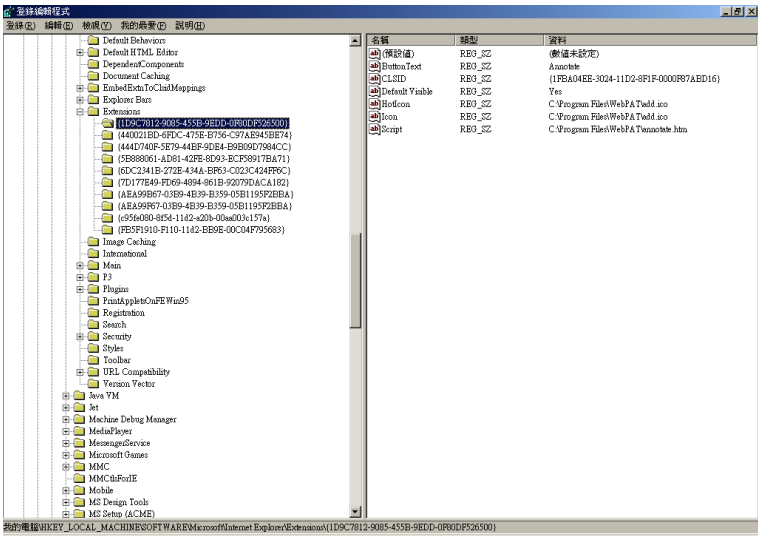

Figure 4: The registery information.

TI currently provides the following functions: annotating, browsing, annotation listing, annotation exporting/importing, making audio annotation, saving Web pages for off-line reading, and searching. Clicking each button will invoke the correspondent JavaScript function. Since these functions are written in JavaScript, users can freely adapt each function to their own needs. They can also change the button icons. However, since TI needs to interact with IE, there are a lot of hacking codes in this module.

3.2.2 Annotation Processor and XML Design

Most previous annotation systems store annotations in a separated database or a remote server. Instead, annotations are encoded in XML and stored in local XML files in WebPAT. A DTD definition file is used to define the list of legal elements for the annotation structure.

The annotation processor (AP) is responsible for reading, parsing, and saving XML annotations. It reads a DTD file to in parsing the annotations. Figure 5 shows the DTD elements. Currently we have defined seven elements. Users can extend the definition to meet their own needs.

```
<annotation>
<!ELEMENT annotation(url, title, author, date, select,
content,color)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT select (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT color (#PCDATA)>
```

Figure 5: The DTD format for personal annotation.

According to the DTD definition, WebPAT maintains the annotations in local XML files. Figure 6 shows an XML example in which a user makes an annotation on a W3C's Web page by highlighting a segment of the content. When the user selects the segment and makes the annotation, this XML document is created and saved in a local file. More operational details will be described in Section 4.

Figure 7 shows a part of JavaScript code. In this example code, the script creates an ActiveXObject in IE and an XML annotation file accordingly. Since WebPAT is all developed in JavaScript, users can adapt any functionality to their own requirements.

```
<?xml version=”1.0” encoding=”big5” ?>
<!DOCTYPE annotation (View Source for full doctype...)>
- <annotation>
<url>http://www.w3.org/</url>
<title>title</title>
<author>author</author>
<date>20020603</date>
<select>Annotea is open; it uses and helps to advance
W3C standards when possible</select>
<content>content</content>
<color>ffff00</color>
</annotation>
```

Figure 6: An XML example file.

```
1  var fso, a;
2  var ForWriting= 8;
3  fso = new ActiveXObject("Scripting.FileSystemObject");
4  var database = "C:\\Program Files\\WebPAT\\annodata.xml";
……
……..
 5  a = fso.CreateTextFile(database, true);
 6  a.WriteLine("<?xml version=\"1.0\" encoding=\"big5\"?>");
 7  a.WriteLine("<!DOCTYPE DOCUMENT SYSTEM \"annotation.dtd\">");
 8  a.WriteLine("<root>")
 9  a.WriteLine("<annotation>");
 ……
 ……..
```

Figure 7: A JavaScript code example.

3.3 User-Centric Functionalities

Besides basically making personal annotations, WebPAT provides the following functions to show its versatility. Since the WebPAT architecture is an open and highly extensible design, users can freely customize WebPAT according to practical requirements.

- **Multimedia Annotations**

  In the WebPAT design, it does not handle these multimedia data directly. It provides an interface to allow users to make multimedia annotations. When a user wants to make a multimedia annotation,

WebPAT calls an external application to process the multimedia data and makes the proper association. The user can thereafter review the multimedia annotations by invoking the associated application again. Currently, the prototype only supports audio annotations. However, this shows the possibility to make different types of annotations.

- **Recursive Annotations and Revision Control**

  Since the annotations are all in XML format, annotating any annotation can be achieved straightforward. This is described as *annotations on annotations*, or simply *recursive annotation*. To facilitate the management of the recursive annotation, WebPAT suuports revision control. Users can make revisions on a previously made annotation and save it into a new version. A historical thread is created to help users keep more information on their thinking process.

- **Annotation Sharing**

  In WebPAT, annotation sharing is achieved simply by an export/import mechanism like YAWAS [5]. From the viewpoint of users, they do not need to login to a server for sharing. Shared annotations can be transferred via different media such as email, disks, or CDs. WebPAT also improves the inconvenience incurred in YAWAS that users cannot find imported annotations without cumbersome operational procedures because YAWAS does not fully support the management of imported annotations. The WebPAT design avoids this problem.

- **Annotation Searching**

  WebPAT provides an interface for users to search annotations by keyword matching. This searching functionality helps users to find specific annotations efficiently. After the users find the matched annotations, they can further edit or delete the searched annotation results as usual.

4 Operational Examples

The installation of WebPAT is straightforward because users need only to register the correspondent machine-codes in MS Windows registry. A setup package will be provided to automatically manage the installation and de-installation.

In the following, several operational examples are presented to explain the WebPAT design. Since

WebPAT is now still under development, only a prototype of limited functionalities is provided. However, we believe that the prototype indeed illustrates how user-centric functionalities can be implemented.

4.1 Annotation Creation

To create annotations, a user needs to first select a part of Web content. Figure 8 shows the example in which a user browses the W3C homepage and selects an interested sentence. Then the user can create an annotation by clicking the **Annotate** button on the toolbar.

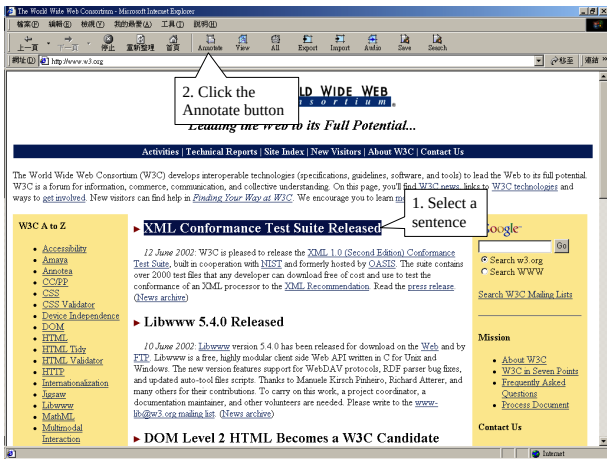

Figure 8: A sentence is selected.

After the user clicks the button, an “Add Annotation” window is prompted as shown in Figure 9. The *title*, *author*, and *content* fields are left blank for user input. After the user inputs some descriptions, the highlight color of the selected text can be changed accordingly before the annotation is saved. In the current prototype design, twelve optional colors are provided. In this example, spring green is selected.

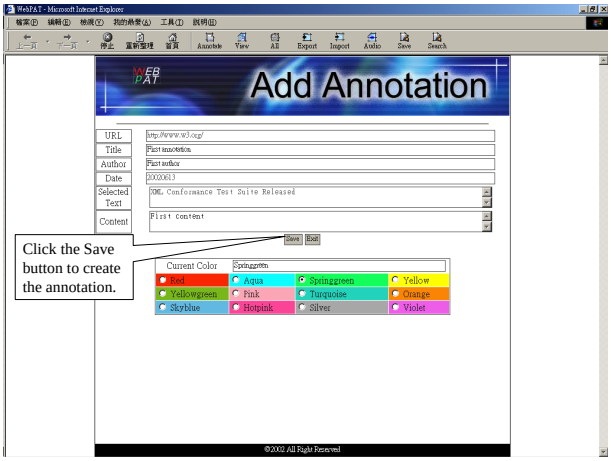

Figure 9: An annotation is created for the sentence.

After filling in three fields and deciding the highlight color, the user needs to press the **Save** button to create the annotation. The selected sentence in the W3C Web page is highlighted in spring green as shown in Figure 10.

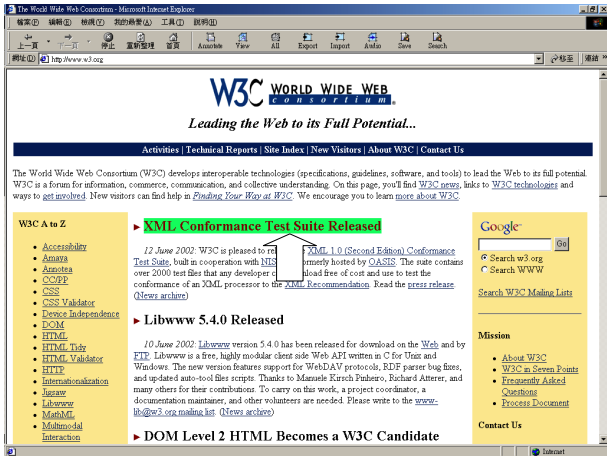

Figure 10: The sentence is highlighted.

4.2 Browsing and Editing

The **View** button on the toolbar is responsible for browsing and editing the created annotations. When the button is clicked, a new browser window is prompted to present the related annotations. Figure 11 shows a scenario of a browsing window.

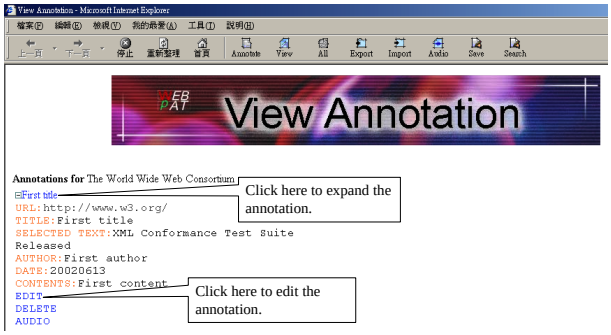

Figure 11: An annotation is browsed and the content is expanded.

In the beginning when the user browses the annotations, only the titles of the annotations are listed. To view more details, the titles can be clicked to expand the whole annotation content. Figure 11 also depicts the content expansion of the annotation.

There are three action labels at the bottom of each annotation: **EDIT**, **DELETE**, and **AUDIO**. When the user clicks the **EDIT** label, a window is prompted for modification as shown in Figure 12.

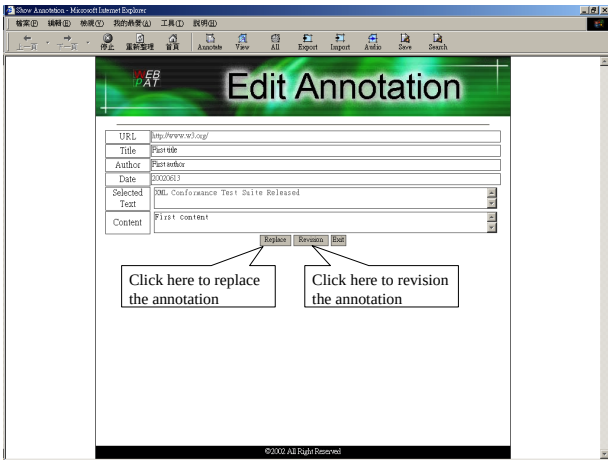

Figure 12: An annotation is under modification.

Then the annotation can be edited at the user's will. After editing the annotation, the user can select to replace the original annotation or make a new revision. Figure 13 shows the replacement example in which the original annotation is overwritten. Figure 14 shows the revision example in which the new revision of the annotation is appended in a revision thread.

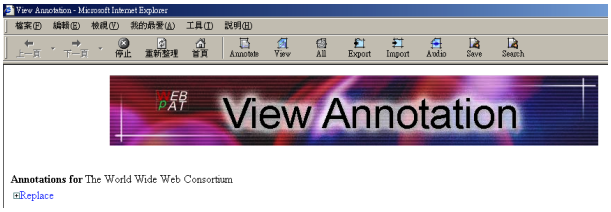

Figure 13: The original annotation is replaced.

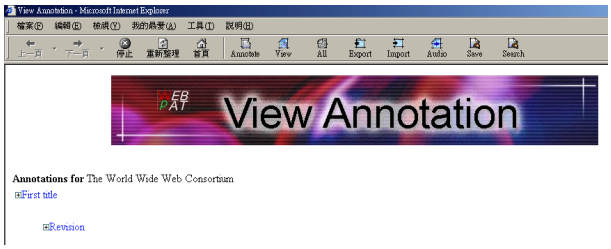

Figure 14: A new revision is made in an appended thread.

4.3 Annotation Deletion

Figure 15 shows the example in which a user clicks the **DELETE** label. In this case, the annotation and all sub-revision annotations will be deleted. In Figure 15, the fist annotation and its revision are chosen to be all deleted. Figure 16 shows the result when they are removed.

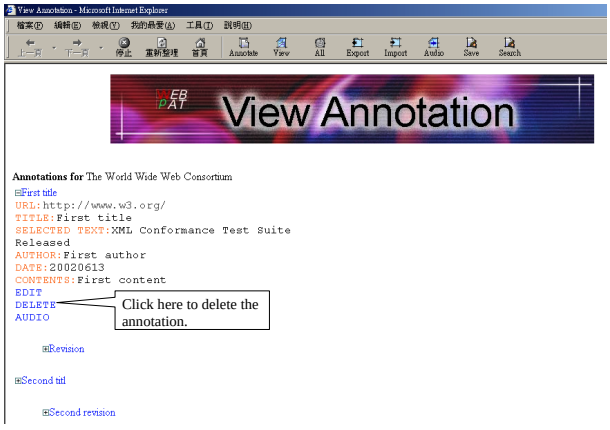

Figure 15: A user clicks the **DELETE** label.

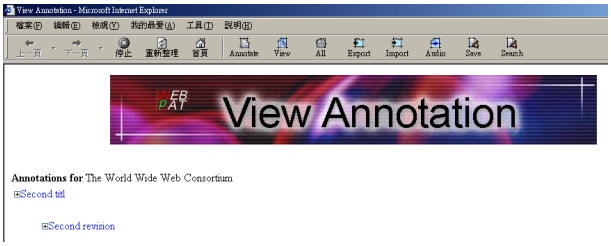

Figure 16: The annotations are deleted.

4.4 Annotation Sharing

To facilitate sharing annotations between two users, WebPAT provides the import/export functions. Figure 17 shows the example in which we assume there is a user Joe who wants to share his W3C annotations to his friend Alma.

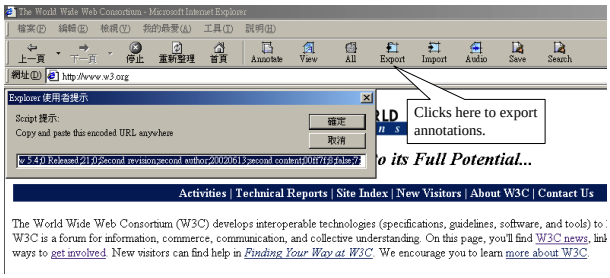

Figure 17: A user Joe clicks the **Export** button to export annotations to his friend Alma.

Joe now clicks the **Export** button on the toolbar. A text stream is produced in a text window as shown in Figure 17. Then Joe sends the data to Alma either by email or other media. After receiving the stream data, Alma clicks the **Import** button on the toolbar of her IE browser. An import window will be then prompted as shown in Figure 18. After Alma pastes the stream data in the input area, she can browse the annotations created by Joe. Figure 19 shows the imported result.

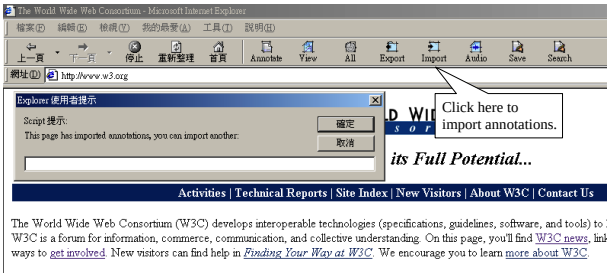

Figure 18: The **Import** button is clicked to import annotations.

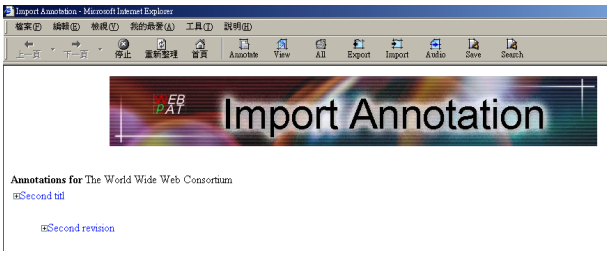

Figure 19: Browsing the imported annotations.

4.5 Multimedia Annotations

Currently, the WebPAT prototype only supports audio annotations. However, it shows the extensibility for multimedia annotations. If the **Audio** button on the toolbar is clicked, WebPAT will call a correspondent external application to make audio annotations. Figure 20 shows the example in which a recorder is prompted for recording an audio annotation.

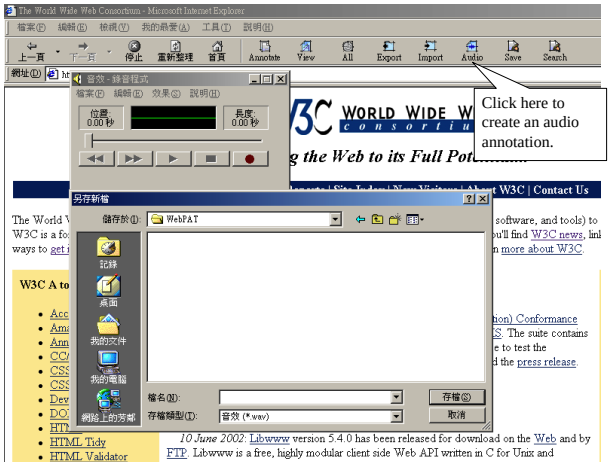

Figure 20: A user clicks the **Audio** button to create an audio annotation.

After making the audio annotation, the user can save the audio file in local storage. If the user wants to review the audio annotations after a while, the **AUDIO** label in the annotation needs to be clicked again to invoke the audio application. Figure 21 shows the example in which the user inputs the full path of the audio annotation in the prompted window, and the audio annotation is replayed.

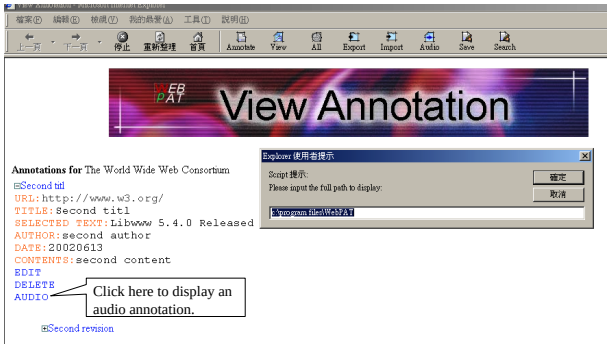

Figure 21: Replaying an audio annotation.

4.6 Annotation Search

WebPAT supports annotation search by keyword matching. Figure 22 shows the search example in which a user clicks the **Search** button on the toolbar. A search window is thus prompted, and the user can input keywords to search for matched local annotations.

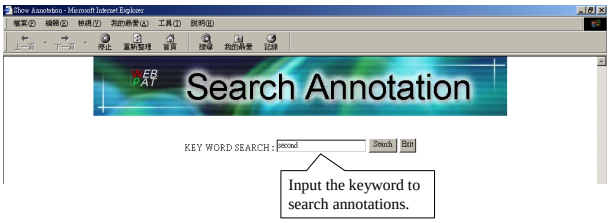

Figure 22: A user clicks the **Search** button to search annotations.

Then the search results will be listed and the user can further browse or edit the result annotations. Figure 23 shows the search results of querying “second”.

Figure 23: The search results of querying “second”.

5 Conclusion

Since 1990s, many Web annotation systems are developed for the sake of cooperation, education, teaching or document clustering. However, most of previous annotation systems do not focus on the personalization, extensibility and openness issues. In some previous systems, annotations are managed by an annotation server or a proxy. Users cannot create personal annotations. In addition, annotations are not structured in many annotation systems. They can hardly be extended in the future.

In this paper, we argue that these three issues are indeed the native spirit behind the annotation activities. Therefore, we start to design a personal Web annotation tool called WebPAT (Web Personal Annotation Tool). WebPAT is designed in JavaScript and XML to achieve the personalization, openness and extensibility goals.

Currently, though WebPAT has only a prototype providing primitive functionalities, it demonstrates the versatility. For example, It can only process text-based Web contents. Graphical contents and multimedia

contents are not taken into consideration yet. However, WebPAT supports multimedia annotations by providing an interface to invoke external multimedia applications. Though the functionalities provided in the prototype are limited, the operational experience shows that WebPAT indeed provides an open and highly extensible Web annotation environment from user-centric consideration.

Nonetheless, some useful functionalities such as annotating multimedia contents and drawing-based annotation are still absent. In addition, auxiliary tools are needed for helping users to define their own annotation structures. These are all in our future working plan. In the future, we also plan to enrich the XML definition by incorporating the RDF infrastructure. However, the prototype still demonstrates an open architecture with high extensibility.

References

[1] Ng S.T. Chong and Masao Sakauchi. Creating and Sharing Web Notes via a Standard Browser. In *Proceedings of the 16th ACM SAC 2001*, pages 99–104, March 2001.

[2] James Davis and Daniel Huttenlocher. Shared Annotation for Cooperative Learning. In *Proceedings of the Computer Support for Cooperative Learning Conference*, pages 84–88, 1995.

[3] Laurent Denoue and Laurence Vignollet. An Annotation Tool for Web Browsers and its Applications to Information Retrieval. In *Proceedings of RIAO*, 2000.

[4] Koen Holtman. The Futplex System. In *Proceedings of the ERCIM workshop on CSCW and the Web*, February 1996.

[5] Jakob Hummes, Alain Karsenty, and Bernard Merialdo. Active Annotation of Web Pages. http://www.eurecom.fr/ hummes/docs/Web4Groups/w4g.html. EURECOM, 1997.

[6] Stefan Koch and Georg Schneider. Implementation of an Annotation Service on the WWW-Virtual Notes. In *Proceedings of the 8th Euromicro Workshop on Parallel and Distributed Processing (PDP'2000)*, pages 92–98, Rhodos, Greece, January 2000. IEEE.

[7] Mosaic. http://archive.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html.

[8] Ilia Ovsiannikov, Michael Arbib, and Thomas McNeil. Annotation Technology. In *International Journal of Human-Computer Studies*, pages 329–362, 1999.

[9] Pei-Luen Rau, Shou-Hian, Yun-Ting Chin, and Kuang-Ci Liu. Developing an Online Annotation Platform for Elementary School Students. In *Proceedings of Instructional Technology and Media*, pages 73–79, June 2001.

[10] Martin Roscheisen. Beyond Browsing: Shared Comments, SOAPs, Trails, and On-Line Communities. In *Proceedings of the Third World Wide Web Conference*, April 1995.

[11] Takeshi Sannomiya, Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura. A Framework for Sharing Personal Annotations on Web Resources using XML. In *Proceedings of the Workshop on Information Technology for Virtual Enterprises (ITVE 2001)*, pages 40–48. IEEE, 2001.

[12] Matthew Schickler, Murray Mazer, and Charles Brooks. Pan-Browser Support for Annotations and Other Meta-Information on the World Wide Web. In *Proceedings of the Fifth International World Wide Web Conference*, May 1996.

[13] Shang-Rong Tsai, Jyi-Ta Chen, and Ming-Ching Kao. A Document Workspace for Collaboration and Annotation based on XML Technology. In *Proceedings of the International Symposium on Multimedia Software Engineering*, pages 165–172. IEEE, 2000.

[14] Ka-Ping Yee. CritLink: Better Hyperlinks for the WWW. In *Proceedings of Hypertext '98*, April 1998.