WEB-BASED ANNOTATION AND COLLABORATION
Electronic Document Annotation Using a Standards-compliant Web Browser

Trev Harmon
School of Technology, Brigham Young University, 265 CTB, Provo, Utah, USA

Keywords: Annotation, collaboration, web-based, e-learning.

Abstract: The Internet provides a powerful medium for communication and collaboration through web-based applications. However, most web-based annotation and collaboration applications require additional software, such as applets, plug-ins, and extensions, in order to work correctly with the web browsers typically found on today's computers. This in combination with the ever-growing number of file formats poses an obstacle to the wide-scale deployment of annotation and collaboration systems across the heterogeneous networks common in the academic and corporate worlds. In order to address these issues, a web-based system was developed that allows for freeform (handwritten) and typed annotation of over twenty common file formats via a standards-compliant web browser without the need of additional software. The system also provides a multi-tiered security architecture that allows authors control over who has access to read and annotate their documents. While initially designed for use in academia, flexibility within the system allows it to be used for many annotation and collaborative tasks such as distance-learning, collaborative projects, online discussion and bulletin boards, graphical wikis, and electronic grading. The open-source nature of the system provides the opportunity for its continued development and extension.

1 INTRODUCTION

While the telegraph and telephone may have cracked open the door of instantaneous, worldwide communication, the Internet has flung it wide open. No longer is it necessary for collaborators to be situated physically close to one another. With its ubiquitous nature, the Internet allows for space- and time-shifting in many collaborative projects and breaks down the barriers of distance and time zones.

Due to the *laissez-faire* nature of the Internet, the divergent approaches and capabilities of web browsers plagued early web developers. Cross-browser, let alone cross-platform, web development could be quite difficult. However, with the advent of modern web browsers that adhere more strictly to a standard Document Object Model (*DOM*) than earlier browsers, web application development has grown drastically forming the movement known colloquially as *Web 2.0*.

Mainstay technologies in the web application arena include applets, plug-ins and extensions. These serve the important purpose of extending web browser functionality, as the original designers of many of the technologies utilized by web browsers did not foresee the broad spectrum of uses expected of their technologies by today's users. While serving this useful purpose, such add-on software can become problematic in some circumstances because specialized versions must be made for each web browser on each operating system. While one could argue the majority of computer systems are made by a handful of hardware and software companies, one would be ignoring the many niche user bases in the Internet world culture.

There are many users using different web browsers with varying goals. Just as with other web development projects, electronic annotation and collaboration systems face the following problem:

> "Even when the interface to the server is public, the small installed base of a single system does not encourage external development of clients." (Kim, Slater, and Whitehead, 2004)

This often leads to such systems becoming obsolete. (Olsen, Taufer, and Fails, 2004). However, the advances in web browser technology and their adherence to standards provide a basis for the development of promising new web applications to aid in annotation and collaboration. If a web browser

Harmon T. (2007).
WEB-BASED ANNOTATION AND COLLABORATION - Electronic Document Annotation Using a Standards-compliant Web Browser.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Society, e-Business and e-Government / e-Learning*, pages 321-329
DOI: 10.5220/0001277803210329

is *DOM*-compliant and supports other standards such as *CSS*, *XHTML*, *DHMTL*, *JavaScript*, and *XML*, it should be able to use web applications using only those standards, regardless of the client's underlying system specifications. To show this in practical terms, the On-line Annotation System (*OAS*) was developed.

1.1 The OAS System

OAS is a web application providing freeform (i.e., handwritten/drawn) and typed annotation functionality for a range of file formats while only requiring a standards-compliant web browser for a client.

While developed mainly for the world of academia, *OAS* is flexible enough to address the annotation and collaboration needs of other process domains. For example, it allows students to submit work to professors using alternate office software, consultants to easily make presentations using only a web browser, engineers to remotely make notes on designs in the office using a kiosk at a conference, and collaborators to have "graphical" conversations.

OAS was designed and implemented in order to demonstrate the technical feasability of such a system. Additional testing, especially in the realm of usability and other human factors, is needed in order to develop a refined application.

1.2 Usage Scenarios

The ability to annotate a large number of document formats using both freeform and typed annotations in a standards-compliant web browser offers a wide range of usage possibilities. These possibilities are even more exciting when considered in light of the collaborative functionality provided. This section will describe several hypothetical scenarios highlighting this functionality.

1.2.1 Document Collaboration

An engineer working for a design firm in California has just finished the design of a new cog for an important client in Europe. He posts the design to *OAS* from his workstation. The lead engineer, who is at a conference in New York securely logs into the system via one of the conference kiosks. She makes some handdrawn annotations as suggestions on the document. The client, using one of their workstations, reviews the annotated document, adding their own comments to the design. All the comments are then reviewed by the engineer when making the final changes to the design.

1.2.2 Online Grading

A journalism student uses a word processor that her professor does not own. When she finishes typing her paper, she uploads it to *OAS*, which converts it to a generic image file format. The professor is able to read and annotate the paper from home on his PC using his web browser. Later, the student accesses *OAS* from her dorm room and is able to review the professor's comments online.

1.2.3 Discussion Boards

A person is struggling with a certain software package. He accesses *OAS* and posts a screenshot, annotating it with his question. One helpful user answers the question and draws a circle on the screenshot to indicate the problem area.

1.2.4 Field Presentations

A sales representative is at a client's site discussing the proposed design of a new product. The client's computers, however, do not have the software needed to display the design. So, the sales representative logs into *OAS* using the client's web browser and is able to continue with the presentation. During the presentatin both the client and the sales representative are simultaneously making notes on the design, which can be reviewed at a future time by either party.

1.2.5 Annotating for Collaboration

As can be seen, online anotation provides a great tool for collaboration. Through the use of Internet standards, *OAS* provides this functionality, which can be easily extended to satisfy the needs of many different types of users. Only with the support of robust annotation functionality can true online collaboration flourish.

2 RELATED WORK

Humans have been attempting to relate to their world through "annotations" since the earliest cave paintings (Hansen, 2006). It is part of the knowledge acquisition process – a fact not lost on researchers who have shown annotations do, in fact, support a number of objectives in the learning process and can affect a reader's response to a document on both the cognitive and emotional levels (Wolfe, 2000). Indeed, it is usually the case that "personal

annotations reflect unselfconscious reactions to reading material" (Marshall and Brush, 2004).

In today's technology-driven world, an increasing number of people consume the majority of their reading material in a digital format (Olsen, Taufer, and Fails, 2004). With this shift in human reading habits, efforts have been made to accommodate the related human urge to create a "personal geography" of their digital reading material through the use of the underlining, asterisks, and notes often seen adorning used college textbooks (Marshall and Bush, 2002).

2.1 Annotation Process Model

A large number of annotation system frameworks and implementations have been developed by both commercial and academic research to address the needs of different user bases. However, while each annotation system implementation uses a unique approach to the annotation model, the underlying meta-process is actually the same. This meta-process has four basic parts:

- Creation of content
- Retrieval of content
- Annotation of content
- Archiving of content

The relationship between these parts is shown in Figure 1. As the user supplies content to the system, it flows through the different cyclic stages shown.

In actual implementations, credential verification methods vary widely. They may simply be using a user-supplied alias to tag the annotation or full-blown security systems. Also, all content must be stored in some type of data archive (e.g., file system, database, etc.).

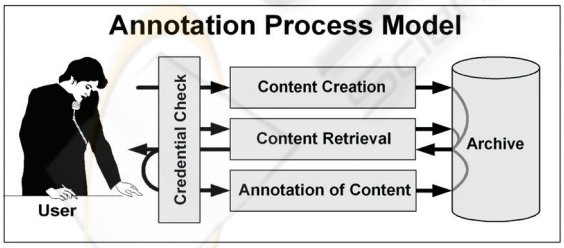

Figure 1: Annotation Process Model used in *OAS*.

2.2 Historical Approaches

Several annotation system implementations have described or addressed key issues that affected the design of *OAS*. These are discussed below.

Xlibris, a hardware device roughly the size of a large book, was originally presented in 1998 as a device allowing users to not only read electronic documents, but also to annotate them in order "to organize their reading for later review and retrieval" through the use of "different colors of highlighters and pens to increase users' flexibility of expression" (Schilit, Golovchinsky, and Price, 1998). While providing a theoretical basis for many of the annotation systems that followed, *Xlibirs* suffers from the fact that it is a dedicated hardware device, thereby limiting widespread acceptance.

Following on the paper metaphor used in *Xlibris*, *ALT* is an A4-size device that enables "users to annotate and sketch on paper in collaboration with a remote peer" (Gabrielli and Law, 2003). Like *Xlibris*, it has the drawbacks of a hardware device.

ScreenCrayons attempted to address the wide range of file formats that exist. By using screen captures, it makes the bold claim of being able to "[collect] annotations on any type of document or visual information from any application." However, it has the problem that when content is "currently scrolled out of sight" the "non-visible context is lost." (Olsen, Taufer, and Fails, 2004)

The *Annotea* project, along with the related web browser *Amaya*, shows the benefits of text annotation functionality as a native browser feature. Both are under the auspice of the W3C and are based entirely on their published standards (Kahan and Koivunen, 2001 and Koivunen, 2005). While this project works well for text, it does not provide support for freeform annotation.

The *Digital Graffiti* project allowed users to use a variety of portable devices, such as PDAs and cell phones, to annotate content on *Plasma Posters*, "large-screen, interactive, digital community bulletin boards [located] in public spaces" (Carter et al., 2004). In this case, the user-centric design essentially allows users to choose their own tools. However, it does require the devices to have special software in order to operate.

While annotation is not its main purpose, the Tablet PC supports annotating digital content when used in conjunction with applications such as *Microsoft Journal* and *OneNote* that provide freeform ink capabilities (Mock, 2004). Through the use of a virtual printer driver, any printable content can be imported into *Microsoft Journal* for annotation (Willis and Miertschin, 2004). As with *Digital Graffiti*, the requirement for special software is the major drawback to this approach.

2.3 Other Considerations

People have a propensity to mimic the real world inside the computer's digital realm. One needs look no farther than the desktop metaphor used by many GUI-based operating systems. Unfortunately, digital documents do not always behave the same as their physical counterparts. Therefore, a number of issues related to this must be addressed by any digital annotation system for electronic documents.

2.3.1 Freeform Annotation

When working with a digital representation of a paper document, users would like to interact with the digital version the same way as they do with the hardcopy. Basically, they want to be able to doodle.

> "[Freeform ink annotation] allows the reader to mark anywhere on the document, does not constrain the shape of the marks, and does not impose any structure on them." (Golovchinsky and Denoue, 2002)

The user needs the freedom to annotate anywhere on digital content (Plimmer and Mason, 2006). These annotations, while bearing no special meaning to the annotation device, are rich in meaning to the user (Schilit, Golovchinsky, and Price, 1998). Restricting this freedom only limits the device's effectiveness, as it does not allow the user to take advantage of the full descriptive power and meaning a few small marks can easily portray.

2.3.2 Anchoring

From the user's perspective, annotations are attached to a certain place within the document, known as an anchor (Golovchinsky and Denoue, 2002). The anchoring of annotations to digital content is a difficult problem, especially when the document's content is allowed to change (Plimmer and Mason, 2006). This is because, if not handled correctly, annotations can lose "the links to their proper positions within the document" (Brush, 2002). Known as orphaning, this loss of position is a problem is unique to annotation for digital content.

> Each time the content of a digital document reflows to a new layout, any digital ink annotations must also reflow to keep up with it. (Bargeron and Moscovich, 2003)

Annotations must be controllable and anchored appropriately. Without this, annotations easily loose their contextual relevance. When the content can be changed to a static form, such as a graphic, this problem is avoided.

2.3.3 Layering of Annotations

Often annotations are stacked on top of one another in the order they were created, thereby forming a relationship between the annotations – a simple timeline. This technique was described as glosses in the *Fluid Document* system (Zellweger et al., 2001). While this is a common approach, careful design of the layering procedure is necessary to avoid "Z-fighting, a classical 3D graphics problem" (Hong, Chi, and Card, 2005). Otherwise, it is possible for annotations to lose their position in the stack, effectively destroying these temporal relationships.

3 OAS SYSTEM DESIGN

OAS is a multi-faceted system that needed to be generic while still providing a simple process for adding future updates and extensions. Consequently, the bulk of the system was designed as an API using the Model-View-Controller paradigm.

3.1 Design Goals

With many example systems to look to for ideas, the design of *OAS* attempted to pull the best practices and design principles together into a single system. The following were the stated design goals for *OAS*:

- No specialized or proprietary software should be required to access or use any functionality.
- The user should be able to submit documents to the system in a wide variety of file formats.
- The user should be able to annotate the documents using freeform drawing or text entry via a keyboard.
- The user should be able to take advantage of time-shifting when interacting with *OAS*.
- The system should be available at any given time of day or night via the Internet.
- The system should maintain copies of both the original and annotated document.
- The system should support contextual annotations, regardless of the document's original file format. However, the layout of the original document should not be altered by the addition of annotations.
- Constraints should be designed into the system to control access.
- The design and implementation of new system extensions to support additional process domains should be simple.
- In order to encourage continued and future the development, it is desirable that the system

and API be built with open-source tools and technologies as much as possible.

3.2 Development

OAS consists of three physical parts: the Linux server, *Win32* server, and client(s).

The *Linux* server consisted of an Intel Pentium 4 CPU (3.20GHz) with 1GB of RAM running the *Fedora Core 5* distribution, *Apache 2.2.2*, and *MySQL v5 Community Edition*. The majority of the server-side programming was done with *Perl 5*. The *Image::Magick* library was used for rendering freeform annotations.

The *Win32* server, used for rendering *Microsoft Office* formats, used an Intel Pentium 4 Mobile CPU (1.60GHz) with 1GB of RAM running *Windows XP Home Service Pack 2*. *ActiveState Perl* provided the network communications framework and access to the *Microsoft Office* application objects via the *Win32::COM* module.

Clients consisted of several web browser and operating system pairs. In all cases, the web browsers are commonly considered to be standards-compliant. Client-side programming used *DHTML* (i.e., *JavaScript*, *CSS*, and *HTML*/*XHTML*).

3.3 Testing Procedures

A series of tests were needed to ascertain web browser compliance and system performance.

3.3.1 Compliance Testing

OAS was tested to insure support for the most popular web browsers on the most popular operating systems. In order to ascertain whether or not *OAS* supported a certain web browser, the web browser had to be capable of completing the following tasks:

- Log into *OAS*
- Open and view documents
- Create new documents
- Create free-form annotations
- Create text annotations
- View annotations
- Edit text annotations
- Delete annotations
- Delete documents

Each of these tasks was considered to be atomic, with only two considered states: successful or unsuccessful. As each of these tasks deals either with capturing and submitting data or rendering the resulting *HTML*, the success or failure of the test was easy to ascertain visually.

3.3.2 Performance Testing

The scalability performance of *OAS* was tested. A test script was developed using *Perl*'s *LWP* modules, which created virtual web users. Each virtual web user completed the following three steps, which mimic the actual chain of actions that occur with a real user using a web browser:

1. Download a random page in the document.
2. Add a random sample annotation to the page.
3. Re-download the page.

This script was run as multiple instances on multiple computers to test *OAS*'s ability to handle concurrent connections. This was done in concert with the continual testing of *OAS* in a single-user development environment.

4 RESULTS

In general, *OAS* was successful in attaining its design goals and objectives. However, there were a few areas that did not go as well as planned.

4.1 Document Acceptance

Table 1 shows the list of document file formats *OAS* recognizes (via file extension) and is able to convert. Source code is listed separately because *OAS* applies context-sensitive highlighting and alternate layouts to these formats. As can be seen, *OAS* was successful in handling many of the common file formats used in academia, as well as a few less common formats.

The process used to handle the different file formats varies depending on the nature of the file. Specifics for some of the file format families will be discussed individually. All file formats are rendered to one or more image files for display (one image per page). Consequently, documents become static once submitted to *OAS*, which addresses the reflow problem inherent in many annotation systems. The document in its original file format is also archived and available to the users.

4.1.1 PS/PDF Strategy

With the exception of images, all file formats are first rendered to either a *PostScript* (*PS*) or *Portable Document Format* (*PDF*) file. This intermediate file is then processed by *GhostScript*, which handles pagination. This strategy works well because many applications can either create a *PDF* directly or print to a file via a *PostScript* printer driver.

Table 1: File formats accepted by *OAS*.

File Extension	File Format Name
C/c++/h	C/C++ Source Code
css	CSS Source Code
csv	Comma-delimited Spreadsheet
doc	Microsoft Word Document
gif	GIF Image
htm/html	HTML Source Code
jpg/jpeg	JPEG Image
js	JavaScript Source Code
odg	OpenOffice.org Draw Drawing
odp	OpenOffice.org Impress Presentation
ods	OpenOffice.org Calc Spreadsheet
odt	OpenOffice.org Writer Document
pdf	Portable Document Format
pl/pm	Perl Source Code
png	PNG Image
ppt	Microsoft PowerPoint Presentation
ps	Postscript File
rtf	Rich-Text Format Document
sh	Shell Programming Source Code
txt	Plain Text
wpd	Word Perfect Document
wrl	VRML Source Code
xls	Microsoft Excel Spreadsheet

4.1.2 OpenOffice.org

OpenOffice.org documents proved to be a special challenge, as a bug in the version used did not allow for headless *OpenOffice.org* servers to directly create *PDF* files. Consequently, *vncserver* was used to create a virtual *X server* where a full version of the *OpenOffice.org* server could be instantiated. This poses some security issues, as the virtual *X server* created with vncserver is owned by the *OAS* user id but needs to be accessible by the *Apache* user. However, locking inbound connections to the virtual *X server* to *localhost* minimizes the risk.

With the virtual *X server* in place, the converter software can load *OpenOffice.org* into a full GUI environment. From the shell command line, an *OpenOffice.org BASIC* macro is called, which handles the conversion of the document to *PDF*. This is then converted to the final format using the *PS*/*PDF* strategy previously described.

As an additional note, *OpenOffice.org* can convert all of the *Microsoft Office* formats. However, it was found these conversions were generally not as good as those produced using native *Microsoft Office* programs due to missing fonts, layout differences, etc. The only exception to this is *PowerPoint* presentations, which are rendered using *OpenOffice.org*. Therefore, if need be, *OAS* can run without the *Win32* server.

4.1.3 Microsoft Office

With the exception of *PowerPoint* presentations, the *Win32* server handles all *Microsoft Office* file formats by default. The server software receives the original file from the Linux server. It then creates an application object for the appropriate *Office* software product via *ActiveState Perl*'s *Win32::COM* module. Once the object is created, it is used to open the file in the appropriate application and print it to a file using a *PostScript* printer driver. The resulting *PostScript* file is returned to the Linux server where it is rendered to an image format using the *PS*/*PDF* strategy described previously.

4.2 Annotation

Annotation creation is a multistage process dependent on the type of annotation being created. In either case, the user begins the process by tracing the area where the annotation is to appear using the cursor controlled by the mouse, stylus, etc. During tracing, the web browser captures the cursor coordinates using *JavaScript*. *OAS* uses this captured information to create the actual annotation.

Figure 2 shows examples of both freeform and typed annotations on part of a document in *OAS*. The rough circle is an example of a freeform annotation. The shaded box is a typed annotation. Both of these methods will be discussed below.

4.2.1 Freeform Annotation Mode

In the freeform mode, the trace path becomes the actual annotation. The user is able to change pen size and colour mid-annotation – the new size and colour being effective for all subsequent tracings.

Once the user has completed tracing the new annotation, they select the *Create Annotation* button that sends the captured coordinates, along with pen size and colour data to the server. The server then uses this data to render an image file. Through a

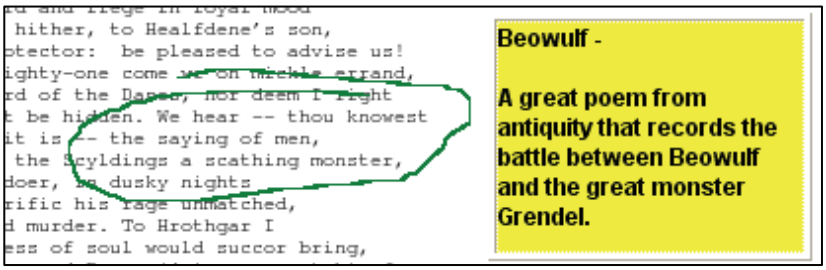

Figure 2: Example of Annotations in *OAS*.

forced reload of the web page, this new image is layered on top of the document in the correct location using *DHTML*. As the background of the image is transparent, it appears on the page just as an ink annotation would on a piece of paper.

4.2.2 Typed Annotation Mode

While in typed annotation mode, the trace path determines the boundaries of the rectangular box that will hold the typed annotation.

Once the user has mapped out the location for the new annotation, they press the *Create Annotation* button, which brings up the dialog for text entry. In addition to allowing any length of text to be typed or copied into the dialog, the user has control of font size, colour, and type.

When the user is satisfied with the text of the new annotation, the information is submitted to the server, which builds the necessary *HTML* to add the new annotation to the document. A forced reload of the web page is then used to have the new annotation appear in its correct location. A scrollbar is added to all text annotations that are too large to fit in the annotation area defined by the user.

Once created, text annotations may be edited or deleted by users with appropriate user rights. Freeform annotations may only be deleted. The user may also temporarily hide any given annotation.

4.2.3 The Tracer

One of the largest challenges with this system was the providing of proper feedback to the user during the annotation creation process (i.e., a visual representation of the annotation being created in real time). Several approaches were explored. For example, on-the-fly creation of text-based vector graphic formats such as *VRML* or *SVG* did not have enough native support within the different web browsers. Also, preliminary testing of the rendering capabilities of the server suggested that it could not keep up with an *AJAX* solution.

The approach used for *OAS* was to use a small (10x10) image of a ball to trace the path created with the cursor. So, as the user creates a new annotation, the tracer image is continuously moved along the trace path using *DHTML*. Once it reaches the end of the trace path, it starts over. The user controls the speed of the tracing action. Consequently, the annotation line cannot be seen in its entirety while it is being drawn, only when the annotation has been completed and rendered by the server. If the completed annotation is unsatisfactory, it must be deleted and redrawn by the user.

While this method did provide some feedback to the user, it is still inadequate. This is especially true for Tablet PC users, who want to use the stylus to write comments directly onto the document. Not having the lines that have been drawn show up immediately can be rather disconcerting in this case. Consequently, this is an area that needs further research in order to refine the user interface.

4.3 Web Browser Support

OAS supports the web browsers shown in Table 2, per compliance requirements discussed previously.

As can be seen, support is strong through the major web browsers across multiple platforms. The only issue is the tracer must be turned off in Safari in order for it to function correctly while doing freeform annotations. This is due to event ordering in *Safari*'s event structure.

4.4 Results of Performance Testing

While web browser testing went well, performance testing did not. Five *Ubuntu Linux* systems were used as clients to perform load testing on *OAS*. Each created five simultaneous instances of the test script, which connected to the *OAS* server over one of the university's networks. This created a continuous load of twenty-five concurrent virtual users on *OAS*. Because these were virtual users being controlled by scripts, the natural delay associated with human interaction was not present, resulting in an effective

Table 2: Web browsers supported by *OAS*.

Operating System	Web Browser	Supported?
Windows XP Home	Internet Explorer 6.0	Yes
	Firefox 1.0	Yes
	Opera 9.0	Yes
Windows XP Tablet PC Edition 2005	Internet Explorer 7.0 RC1	Yes
	Firefox 1.5	Yes
	Mozilla 1.7.3	Yes
Mac OS X Jaguar	Safari 1.3	Yes (no tracer)
Linux Fedora Core 5	Firefox 1.5	Yes
	Opera 9.00	Yes
	Mozilla 1.7.13	Yes
	Konquerer 3.5.3	Yes
	"Epiphany" GNOME Web Browser 2.14.2.1	Yes

load much higher than what would be produced by twenty-five concurrent human users.

While the test was running, a large number of entries began appearing in *Apache*'s error log. This was unexpected, as *OAS* had been continually monitored during development in a single-user environment. Most errors were caused by undefined variable values generally populated either by database calls or by *Apache* when it created the environment for the *Perl* handlers. The population of these fields, however, occur at different stages of the *Apache* request cycle. Errors were occurring in stages where no custom *Perl* handlers were being used. This suggested the errors are occurring as a result of the corruption of the *Apache* child process being used to handle the requests. This is supported by the fact that restarting *Apache* would temporarily correct the problem.

Attempts were made to tweak the settings in the *httpd.conf* file to force *Apache* to recycle its child processes at a faster rate. However, this had no noticeable effect on the problem. Eventually, it was determined the maximum number of concurrent users was three. Occasionally, a fourth could be temporarily added.

The evidence suggests this problem is probably not hardware related, but is caused instead by memory corruption in one of the *C* libraries called by one the *Perl* handlers. Identifying and moving the offending routines out of the *Perl* handler and into a *CGI* script to localize the damage could temporarily fix the problem. However, this does not address the underlying problem. This is an area of future research and development.

4.5 Usability Testing

Official usability testing was not conducted with human participants because of the persistence of the noted problems with *OAS*, as they would have artificially skewed the results. Once these problems have been properly addressed, meaningful usability testing can be performed.

5 CONCLUSIONS

OAS was designed to address some of the issues with earlier annotation systems, as described in Section 2.2. Many of these shortcomings centred on the need for specialized hardware and/or software. *OAS* addressed this by using only a standards-compliant web browser for the client. Overall, *OAS* was able to meet its design goals. However, there were still problems with the implementation.

OAS is able to accept over twenty different file formats for both freeform and typed annotations. Through the use of web standards, support for a number of web browsers on several different operating systems was achieved. This means greater flexibility for the user, who is able to access their information from an employer's workstation, school lab machine, friend's personal system, or even an airport kiosk just as easily as they can from their own laptop or home computer. As *OAS* renders all documents to images, client systems do not need to have the software required by the original file format. So a student may submit a paper written using *OpenOffice.org* to a professor who only uses *Microsoft Office*, and the professor will still be able to read and annotate the document.

By utilizing the multi-tier security and document permission model, *OAS* can easily support any number of annotative or collaborative tasks through user and group permissions.

While the design goals were reached, there are still areas of *OAS* that need improvement. One area that needs work is the providing of proper feedback to the user during the annotation process. The tracer method, while functional, does not provide the desired nor needed level of feedback. The other area needing more research is the correction of the memory corruption issues when multiple users are accessing the system at a single time. These represent areas of future research and development, which should be followed with full user testing to

address any additional user experience and interface issues that may arise.

However, even with the problems faced in the current iteration of *OAS*, it provides insights into an exciting area of future web applications. Additional research and development could make *OAS* into a practical and powerful tool for online annotation and collaboration in both academia and industry.

ACKNOWLEDGEMENTS

I would like to thank the following faculty members of the School of Technology at Brigham Young University, without whose help this project would not have been possible: C. Richard Helps, Joseph J. Ekstrom, and Michael G. Bailey.

REFERENCES

Bargeron, D., Moscovich, T., 2003. Reflowing digital ink annotations [Electronic version]. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 385-393. ACM Press.

Brush, A., 2002. Annotating digital documents: Anchoring, educational use, and notification [Electronic version]. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, 542-543. ACM Press.

Carter, S., Churchill, E., Denoue, L., Helfman, J., Nelson, L., 2004. Digital Graffiti: public annotation of multimedia content [Electronic version]. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, 1207-1210. ACM Press.

Gabrielli, S., Law, A., 2003. Annotation in the wild: Benefits of linking paper to digital media [Electronic version]. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, 890-891. ACM Press.

Golovchinsky, G., Denoue, L., 2002. Moving markup: Repositioning freeform annotations [Electronic version]. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, 21-30. ACM Press.

Hansen, F., 2006. Ubiquitous annotation systems: Technologies and challenges [Electronic version]. In *Proceedings of the Seventeenth Conference on Hypertext and Hypermedia*, 121-132. ACM Press.

Hong, L., Chi, E., Card, S., 2005. Annotating 3D electronic books [Electronic version]. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, 1463-1466. ACM Press.

Kahan, J., Koivunen, M., 2001. Annotea: An open RDF infrastructure for shared Web annotations [Electronic version]. In *Proceedings of the 10th International Conference on World Wide Web*, 623-632.
ACM Press.

Kim, S., Slater, M., Whitehead, E., 2004. WebDAV-based hypertext annotation and trail system [Electronic version]. In *Proceedings of the Fifteenth ACM Conference on Hypertext and Hypermedia*, 87-88. ACM Press.

Koivunen, M., 2005. Annotea project. W3C. Available from http://www.w3.org/2001/Annotea/; Accessed September 20, 2006.

Marshall, C., Brush, A., 2002. From personal to shared annotations [Electronic version]. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, 812-813. ACM Press.

Marshall, C., Brush, A., 2004. Exploring the relationship between personal and public annotations [Electronic version]. In *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries*, 349-357.
ACM Press.

Mock, K., 2004. Teaching with Tablet PC's [Electronic version]. In *Journal of Computing Sciences in Colleges*, 20, no. 2 (Dec. 2004):17-27. Consortium for Computing Sciences in Colleges.

Olsen, D., Taufer, T., Fails, J., 2004. ScreenCrayons: Annotating anything [Electronic version]. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, 165-174. ACM Press.

Plimmer, B., Mason, P., 2006. A pen-based paperless environment for annotating and marking student assignments [Electronic version]. In *Proceedings of the 50th Conference on User Interfaces 2006*, 50:37-44. Australian Computer Society.

Schilit, B., Golovchinsky, G., Price, M., 1998. Beyond paper: Supporting active reading with free form digital ink annotations [Electronic version]. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 249-256. ACM Press/Addison-Wesley Publishing Co.

Willis, C., Miertschin, L., 2004. Tablet PC's as instructional tools or the pen is mightier than the 'board! [Electronic version]. In *Proceedings of the 5th Conference on information Technology Education*, 153-159. ACM Press.

Wolfe, J., 2000. Effects of annotations on student readers and writers [Electronic version]. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, 19-26. ACM Press.

Zellweger, P., Bouvin, N., Jehøj, H., Mackinlay, J., 2001. Fluid annotations in an open world [Electronic version]. In *Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia*, 9-18. ACM Press.