Fostering Asynchronous Collaborative Visualization

Francis T. Marchese and Natasha Brajkovska
Department of Computer Science, Pace University, New York, NY 10038
fmarchese@pace.edu

Abstract

A novel asynchronous collaborative visualization system for the chemical sciences has been created as a mash-up between an interactive visualization program and a wiki. The system supports features such as annotation, information foraging, and visualization session playback. Building the system from predefined disparate components was a simple process. Yet, as an integrated whole, the system displays an unexpected synergy beyond the sum of its parts.

1. Introduction

Computer-supported collaborative visualization typically employs a synchronous collaboration paradigm in which multiple users meet during a predetermined agreed upon time to work in concert within a WYSIWIS (What You See Is What I See) environment. Because synchronous collaboration is such a powerful idiom for bringing groups of individuals together over a distance, nearly all of the research in this field has focused on building and characterizing synchronous collaborative visualization systems [1]. However, asynchrony is an essential part of group process. Edwards and coworkers [2] have pointed out that, in an asynchronous setting, collaboration occurs at different times because users neither require interactive coordination nor real-time notification of updates to shared artifacts. Many work practices favor asynchrony, where individuals exploit time and space to work at their own convenience. In addition, time and scheduling constraints may conspire against synchrony, making it difficult to collaborate, particularly where time zones must be spanned. Technology may constrain synchronous collaboration because of limited access to resources such as network bandwidth or videoconferencing systems. As a result, and out of necessity, much collaborative scientific research is asynchronous, with data sharing and discussion following the path of least resistance - email.

The ubiquity of email and the reliance placed on it for communication have made it the *de facto* asynchronous collaborative medium. Yet, email systems are inadequate for sharing artifacts that include text, complex data, charts, imagery, screen-captures, and video – all of which are part-and-parcel of the research process, This has lead Viégas and Wattenberg [3] to argue for what they call communication-minded visualization, visualization systems designed to support communication and collaborative analysis. To demonstrate their thesis and to investigate the social mechanisms that support asynchronous collaboration, Heer, Viégas and Wattenberg created *sense.us*, a prototype web application for social visual data analysis that supports view sharing, discussion, graphical annotation, and social navigation [4]. They found that users not only explored the data directly but also used the comments, exploratory trails, and graphical annotations inserted by other users to investigate the data in different ways. What is clear from their results is that despite the fact that asynchronous collaboration decouples the research artifacts from the discussion, an asynchronous visualization system that persists-in-time can maintain the coupling between commentary and artifact to a sufficient degree that it can be an extremely useful collaborative tool.

Viégas and Wattenberg contend that asynchronous communication of visualization discoveries and processes is an area where there is an opportunity to make important contributions to visualization research [3]. We agree! We know of only two systems that have been built specifically to support asynchronous collaborative visualization. The first is DecisionSite Posters by Spotfire, Inc. [5]. This system is a web-based client that resides within an information visualization package. It was designed specifically to support asynchronous sharing of visualizations so that interactive snapshots of analyses could be captured and sent to collaborators for further analysis. The second system is *sense.us*; as we noted above,

11th International Conference Information Visualization (IV'07)
0-7695-2900-3/07 $20.00 © 2007 IEEE

sense.us is a demonstration. Therefore, we have built an asynchronous collaborative visualization system for chemical applications. It was created as a mash-up, if you will, by integrating two technologies, - an interactive three dimensional visualization program and a wiki [6]. The interactive visualization program provides the means for interrogating data in real-time while the wiki supports asynchronous sharing and commentary. Besides building a useful asynchronous collaboratory for molecular visualization, the purpose of this research was two-fold: to investigate how well such a system meets the requirements for an asynchronous visualization system and to scope out the design issues for implementing such a system.

In the following section we lay out the background to the problem. Section 3 contains the system design with a scenario of use presented in section 4. Section 5 includes a discussion and conclusions.

2. Background

Independence may be the most important issue underlying asynchronous work [2] in which collaborators function independently while working on shared artifacts. In these situations communication about the collaboration is less frequent than it is in synchronous work. For example, when authors collaborate to create a manuscript, coordination is infrequent, only synchronizing for integration of parts or reorganizing the context in which the manuscript is being created. Moreover, when a shared artifact is manipulated, it is not necessary for each collaborator to instantly know any changes that have been made by others. And it could actually be disruptive to the work process if an author must continually address each updated artifact.

With these issues in mind, Edwards and coworkers [2] suggest the following general characteristics of a software system that supports asynchronous collaboration while maintaining work independence. First, it should insulate collaborators so they can continue working regardless of the actions taken by coworkers. Second, it should support replication of data in order to separate the actions of users from their colleagues in such a way as to provide performance, fault tolerance, and the ability to locally integrate changes before releasing them to the world at large. Third, it should support disconnected use, so users are able to view, update, and add to their own private replicas of data even when they are not on the network. Fourth, it should support automatic resolution of conflicts to help reduce the need for coordination.

In addition, Viégas and Wattenberg propose that asynchronous visualization systems should support features such as annotation, information foraging, and visualization session playback [4]. Annotation should allow users to select objects and add some kind of information. Information foraging should allow users to study the data in their own way, in total or in part, and provide a means to communicate discoveries quickly and easily. A playback feature should allow users to edit a session sequence (e.g. a sequence of snapshots), picking out only a few key frames and discarding any false starts or superfluous navigation. Alternatively, a more robust playback mechanism could be employed that follows Manohar and Prakash's *replay by re-execution* paradigm [7]. In their approach, a series of input events is recorded that shows the exact sequence of steps necessary to create the effect of WYSNIWIST (What You See Now, Is What I Saw Then). When it is time to replay a session, each input event is faithfully re-executed. Their method has two benefits. Because only input events are recorded, recorded sessions are small in size and thus easier to exchange among collaborators; and because input events are re-executed, the re-execution approach allows for interactive replay.

Another important issue that should be considered is version control. It has been noted that for hypermedia systems the advantages of versioning include: historical revisions, security and exploratory productions, distributed and asynchronous collaborations, emergent forms of collaborations, work flow support, efficiency and scalability [8]. In an open editability model, such as that supported by a wiki, documents can be modified at will. This has lead Di Iorio and Vitali to argue that versioning becomes necessity in order "to trace the contributions of the different authors of different sources, to outline differences between documents, to restore old changes, to revise modifications, and to differentiate all the personal interventions." [9]

Mindful of these issues, we have built our asynchronous collaborative visualization system. In the following section we will discuss its design.

3. System Design

The asynchronous collaborative visualization system that we built is composed of only two parts: an interactive molecular visualization program and a wiki.

Molecular visualization was selected as the application domain because molecular visualization environments typically integrate data generation, visualization, and analysis [10]. These systems must render a vast array of graphical representations not found in mainstream visualization systems to support visual analysis of data generated for molecular structure and dynamics by both theory and experiment. In addition, the builders of a number of molecular visualization systems have considered issues of annotation and recording. In the Kinemage (*Kine*tic I*mage*) molecular visualization system [11], an ASCII input file contains source data, annotated display lists for visualization scenarios, and accompanying descriptive comments. These files can be shared, edited, appended to, and evolve over time. And Chime™ , a browser plug-in by Elsevier MDL [12] can read script files containing a sequence of transformations that change molecular representations, styles, and execute geometric transformations. Hence, Chime scripts can be used in a *replay by re-execution* mode.

Jmol was selected for the program to embed within the wiki [13-14]. Jmol is a free, open-source molecule viewer for chemistry and biochemistry that runs on multiple platforms, including Windows, Mac OS X, and Linux/Unix systems. The software consists of three parts, all written in Java: the Jmol application that runs on the desktop; the Java development toolkit (JmolViewer) - a set of Java "classes" that can be integrated into other Java applications to provide molecular visualization and analysis of chemical structure; and the Jmol applet that can be integrated into web pages. This is the visualization component of our system.

Besides its extensive visualization capabilities, Jmol has two attributes that are essential to our collaborative system. Like Chime, it can read scripts, thus providing a means for session feedback. Also, it possesses a Javascript interface so it may be controlled by means of user interface objects (e.g. buttons, check boxes, etc.) that are embedded within the descriptive text. With a button click, a script command is sent to the applet that in turn updates the display. The power of such an interface is that it allows for the embedding of *replay by re-execution* scripts within the narrative commentary to tightly bind descriptive text and image. We have discussed the importance of this issue previously [15]. Examples of how Jmol may be used in this way is found in the work of the biology students at Kenyon College [16].

Wiki selection can be daunting. There exist literally hundreds of wiki engines that have been written in a wide variety of programming and scripting languages [17]. However, JSPWiki was chosen for the wiki component of our system [18]. It is a WikiWiki engine built around standard J2EE components (Java, servlets, JSP). The reasons for selection are that it is a mature and stable Wiki with simple formating rules, is easy to install, provides a simple authentication mechanism, supports RCS-based version control, allows file attachments to a page, and can be extended using plug-ins.

There are a number of ways of inserting a Java applet into a wiki. Creating a plugin is one possibility. Another is to allow the wiki to execute HTML code including Javascript. This method is suggested for intranet installations only. That said, we will use the latter method for our demonstration.

Figure 1 displays a snapshot from our wiki that contains a molecular visualization and associated text. A two column HTML table has been used to arrange Jmol with the body text embedded within an HTML frame. Beyond technology, the essential issue is that narrative and visualization should coexist within close proximity so as the text is read, the graphical representations may be manipulated.

4. Sample Scenario

Figure 1 depicts the starting point for asynchronous collaboration. A scientist has created a wiki entry that contains a molecule and descriptive text within which is embedded Jmol commands linked to interface buttons. The scientist began with an HTML file incorporating Javascript code. This is inserted into a new wiki page by a copy and paste. Then a file is uploaded as an attachment to the wiki page that contained the coordinates of the molecule to be displayed.

With the document in place, Jmol may be used to directly interact with the data by moving the mouse over the image. A right-click of the mouse opens a pop-up menu, providing access to all Jmol's features. Alternatively, the scientist can read the descriptive text and click on the embedded buttons to load Jmol scripts that focus attention on the data components that the narrative addresses. In so doing, the commentary and visualization maintain a tight binding (c.f. reference 15 for a discussion). In addition, at any time in the viewing process, it remains possible for the scientist to manipulate the graphical representation.

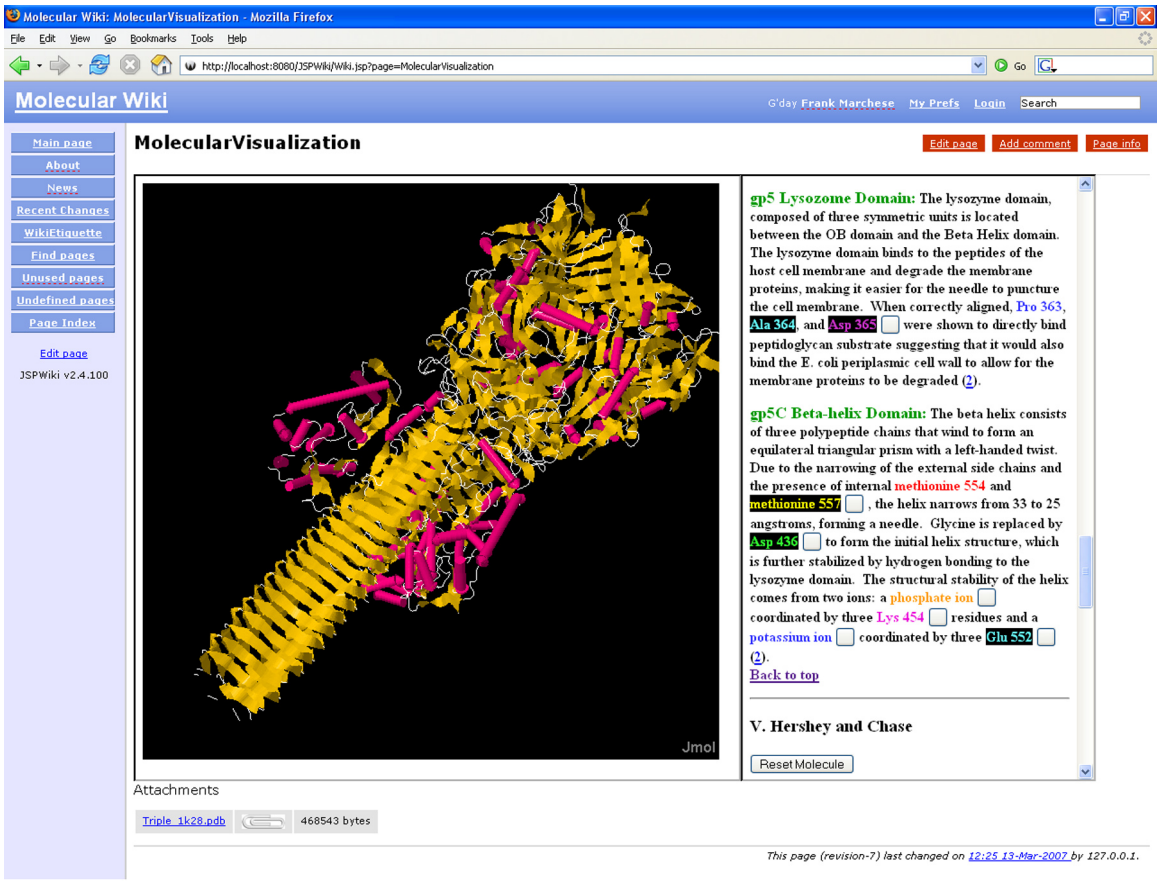

Figure 1. View of the molecular visualization wiki with Jmol (left) and embedded UI buttons (right) that launch Jmol scripts from the commentary.

Later, a second scientist explores the wiki page. She right-clicks on Jmol opening a console to input script commands on-the-fly. The console allows her to view the command history as well. She can copy-and-paste between the history and script input fields, working with different combinations of script commands. When she is done, she can either enter a comment, edit the document to include her new scripts, or download the molecule structure file to refine the data analysis on her local machine that is disconnected form the network. Here she can use her version of the Jmol Java application for visualization. Finally, she can upload her new commentary to the wiki.

Over time each scientist can review the document's history by using the wiki's version control system. Specifically, it can display how the document has evolved as each successive change and addition has been made. Each version in the history can be loaded and explored at any time.

4. Discussion and Conclusions

In Section 2 we set two goals for this project: to investigate how well a wiki-visualization program mash-up meets the requirements for an asynchronous collaborative visualization system, and to scope out the design issues for implementing such a system. We discussed nine general characteristics that an asynchronous collaborative visualization system should possess. They include:

1. Isolation of work practices.
2. Support for replication of data.
3. Automatic resolution of conflicts.
4. Version control.
5. Interactive real-time visualization.
6. Playback by means of *replay by re-execution.*

7. Support for disconnected use.
8. Information foraging.
9. Annotation.

Our system meets all these goals. The first four goals are met by JSPWiki; goals five through seven by the Jmol applet and application. Both JSPWiki and Jmol support information foraging: the wiki by means of text, and the visualization program by means of graphics. Although annotations may appear to fall entirely under the wiki's purview, the embedding of Jmol scripting commands within the document's body creates a tight dynamic annotation in which a sequence of Jmol commands can be used to display the data in such a way as to focus attention on the data attributes that the commentary addresses. Even if Jmol did not support LiveConnect [19], the API that furnishes JavaScript with the ability to call methods of Java classes using the existing Java infrastructure, the requirements for annotation are met either by attaching a Jmol script to the wiki page or embedding them as a comment. This script can be cut and pasted into Jmol's console and subsequently executed. Ideally, direct annotation within the graphical window should be supported, something that is found in the *sense.us* system. But few systems do this. And those that do, typically do it as image captures (for example, see the CAV system [20]). Thus, this area is ripe for exploration.

From a design perspective, the ability to easily insert a graphical application into a wiki to create instantly an asynchronous collaborative visualization system is an empowering notion. It means that the need for specialized software designed to support collaborative visualization is mitigated because the wiki becomes the foundation for sharing. More complex asynchronous collaborative visualization systems may be constructed by embedding additional graphical modules. For example, our system could have been extended by adding an interactive charting applet to display *xy* graphs and statistical plots, complementary data representations that support scientific research. It means as well that any visualization applet employing today's specialized hardware such as 3D stereographic displays, videowalls, or other multi-display technology can become integrated into an asynchronous collaboratory. This is particularly important, because visualization systems are not able to display, annotate, and format the large body of descriptive text required to thoroughly document research. What this research demonstrates is that these systems can be integrated into the scientific communication and collaboration process by embedding them within a wiki.

The quality of the synergy between wiki and visualization software depends on how well the visualization software handles annotation, playback, and most importantly LiveConnect. Indeed, what we have demonstrated by using Jmol is that the Javascript to Java communication strongly couples the text-based wiki with graphical application to provide a level of synchronized, dynamical text-image collaborative communication not seen in most other systems. We see this as an opportunity for further development of asynchronous collaborative visualization systems - specifically, the enhancement of wikis to better accommodate applets and the extension of applets to accommodate LiveConnect or similar technology.

Finally, our collaborative visualization software is constructed from two modules that have been used extensively as independent systems (*vide supra*). However, the usability of the system as a whole remains to be tested. Since the goals of this research project were to scope out the issues in designing and building such a system, and those goals essentially has been met, we will now move onto this next segment of the research project.

References

[1] I.J. Grimstead, D.W. Walker, and N.J. Avis, "Collaborative visualization: a review and taxonomy," In *Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications,* (Oct. 2005), pp. 61-69.

[2] W.K. Edwards, E.D. Mynatt, K. Petersen, M.J. Spreitzer, D.B Terry, and M.M. Theimer, "Designing and implementing asynchronous collaborative applications with Bayou," In *Proceedings of the 10th Annual ACM Symposium on User interface Software and Technology* (Banff, Alberta, Canada, October 14 - 17, 1997). UIST '97. ACM Press, New York, NY, pp. 119-128.

[3] F.B. Viégas and M. Wattenberg, "Communication-minded visualization: a call to action," *IBM Systems Journal* 45, 4, 2006, pp. 801-812.

[4] J. Heer, F.B. Viégas, and M. Wattenberg, "Voyagers and voyeurs: supporting asynchronous collaborative information visualization," In *Proceedings of CHI 2007*, ACM Conference on Human Factors in Computing Systems, 2007.

[5] DecisionSite Posters, [cited Mar. 10, 2007], available from World Wide Web: < http://www.spotfire.com/products/decisionsite_posters.cfm >.

[6] B. Leuf and W. Cunningham, *The Wiki Way: Quick Collaboration on the Web*, Addison-Wesley Longman Publishing Co., Inc. 2001.

[7] N.R. Manohar and A. Prakash, "Replay by re-execution: A paradigm for asynchronous collaboration via record and replay of interactive multimedia sessions," *SIGOIS Bull.* 15, 2 (Dec. 1994), pp. 32-34.

[8] F. Vitali, "Versioning hypermedia," *ACM Comput. Surv.* 31, 42s (Dec. 1999), pp. 24-30.

[9] A. Di Iorio, and F. Vitali, "From the writable web to global editability," In *Proceedings of the Sixteenth ACM Conference on Hypertext and Hypermedia* (Salzburg, Austria, September 06 - 09, 2005). HYPERTEXT '05. ACM Press, New York, NY, pp. 35-45.

[10] F.T. Marchese, J. Mercado, and Y. Pan, "Adapting single-user visualization software for collaborative Use," In *Proceedings of the Seventh International Conference on Information Visualization: IV'03* (London, July), IEEE Press, 2003, pp. 252-257.

[11] D.C. Richardson and J.S. Richardson, "The kinemage: a tool for scientific communication," *Protein Sci.* 1, 1 (Jan. 1992), pp. 3-9.

[12] Elsevier MDL Chime, [cited Mar. 10, 2007], available from World Wide Web: <http://www.mdl.com/products/framework/chime/ >.

[13] A. Herráez, "Biomolecules in the computer: Jmol to the rescue," *Biochemistry and Molecular Biology Education* 34, 4, 2006, pp. 255-261.

[14] Jmol, [cited Mar. 10, 2007], available from World Wide Web: < http://jmol.sourceforge.net/>.

[15] F.T. Marchese, "Dynamically binding image to text for information communication," In P*roceedings of the Eighth International Conference on Information Visualization: IV'04* (London, July 2004), IEEE Press, pp. 707-712.

[16] Biomolecules at Kenyon, [cited Mar. 10, 2007], available from World Wide Web: <http://biology.kenyon.edu/BMB/chime.htm>.

[17] Wiki Engines, [cited Mar. 10, 2007], available from World Wide Web: <http://c2.com/cgi/wiki?WikiEngines/ > .

[18] JSPWiki, [cited Mar. 10, 2007], available from World Wide Web: < http://jspwiki.org/wiki/>.

[19] LiveConnect, [cited Mar. 10, 2007], available from World Wide Web: < http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.3/guide/lc.html >.

[20] S.E. Ellis and D.P. Groth, "A collaborative annotation system for data visualization," In *Proceedings of the Working Conference on Advanced Visual interfaces* (Gallipoli, Italy, May 25 - 28, 2004). AVI '04, ACM Press, New York, NY, pp. 411-414.