# Collaborative Visualization via Application Bookmarks

Steven L. Rohall, Adam Marcus, and John F. Patterson
IBM T.J. Watson Research Center
One Rogers Street
Cambridge, MA 02142
+1 617 693 1840

{steven_rohall, john_patterson} @us.ibm.com, marcua @alum.rpi.edu

## ABSTRACT

*Collaborative visualizations* are visualizations designed to support communication and collaborative analysis. Research in information visualization has focused on improving the display of data for individual users. However, the task of understanding and analyzing data is typically not solitary—an analyst will want to share findings or consult with others. We have been implementing a collaborative visualization framework that enables visualization developers to concentrate on the visual aspects of their systems without worrying about implementing collaborative features. In particular, our framework provides support for application bookmarking, a key feature needed for both synchronous and asynchronous collaboration. In this demonstration, we will show the current state of our prototype.

## Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*patterns*. H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*asynchronous interaction, computer-supported cooperative work*, *synchronous interaction*.

**General Terms:** Design, Theory.

**Keywords:** Collaboration, Information Visualization, Bookmarking.

## 1. MOTIVATION

Today, other than a few visualization systems that have incorporated collaboration as a primary feature (e.g., CoMotion [5]), sharing happens in spite of, not because of, the underlying visualization system. Sharing is typically superficial, consisting of screen grabs that are emailed among users or navigational instructions for a remote user (e.g., "go to this URL, and then click on 'report,' and then…"). These *ad hoc* collaboration mechanisms fall short. Screen shots do not allow people to further interact with the visualization—key information, such as tooltips on data points or the ability to rotate three-dimensional objects, is lost. Additionally, written or spoken commands, guiding one user to pick up where another left off, are very cumbersome for joint exploration of the data.

What is needed is a framework for building collaborative visualizations. A collaborative visualization framework would allow visualization developers to concentrate on building visualizations without worrying about the details of collaboration, much the way that visualization toolkits relieve developers from the details of drawing curves, double buffering their output, or handling the animation of sprites. While others have developed toolkits for building collaborative applications (e.g., [4]) or information visualizations (e.g., [1]), our approach has been to look at the specific needs of collaboration within an information visualization framework.

## 2. PROTOTYPE FEATURES

Writing collaboration-aware applications from scratch is difficult while collaboration-transparent approaches (e.g., [2]) have met with limited success. Recognizing that some input from the application developer is necessary to enable collaboration, our goal has been to minimize and simplify that input through the use of a library. Our library provides support for the following features.

### 2.1 Bookmark State

An application bookmark is a pointer to the running state of an application. It must indicate the application to run (including version information) and the application state from which to resume. The idea is that a user, in the course of analysis, can create bookmark snapshots of the system. At a later point, he or she can return to a prior snapshot and continue interacting from there. More importantly, a *different* user can continue the analysis at some point in the future. While bookmarks in this sense support asynchronous collaboration, bookmarks are essential for supporting synchronous collaboration—they can be used to bring a latecomer in an online meeting up to speed.

In general, applications are not bookmarkable—critical state is buried in the code and not easily shared. The heart of bookmarking is *serialization*—the process of converting an object's state to a sequence of bytes that can be written to a file, as well as the process of rebuilding those bytes into a live object at some future time. The developer must determine what state in the application is essential and ensure that the state is consistent when a bookmark is created. Our framework provides a robust


*CSCW'06*, November 4–8, 2006, Banff, Alberta, Canada.

mechanism for saving and reinstating crucial state information in uniquely named bookmarks and for storing them in a central repository where they are available for future use. Bookmarks are specified by URLs that can easily be sent via email or pasted into a chat session.

## 2.2 Handling Externalities

Externalities exist when the application accesses the computing environment outside of the application itself [2]. Visualizations often use data files that are local to the user, which must be shared with collaborators. Local information, such as user preferences or data files, can also be an important determiner of the appearance and flow of an application, but the bookmark creator's local settings are not likely to be the same as the bookmark user's settings. To handle externalities, our framework contains methods the programmer can use for externalizing them to a server. Other users of the bookmark grab the appropriate values from the server rather than trying to resolve them on their local computer. A hash of the externality ensures that it is only stored on the server once. Externalities in the framework code itself are handled automatically.

## 2.3 State Dependencies

Our library incorporates dependency graph analysis to ensure that application state gets restored in the proper order. For example, it does not make sense to pan and zoom in a visualization until after the data has been loaded and the visualization created. As the programmer specifies what state needs to be saved, he or she can easily indicate that one piece of state depends upon another. Our framework then takes care of restoring the state items in the proper order.

# 3. IMPLEMENTATION

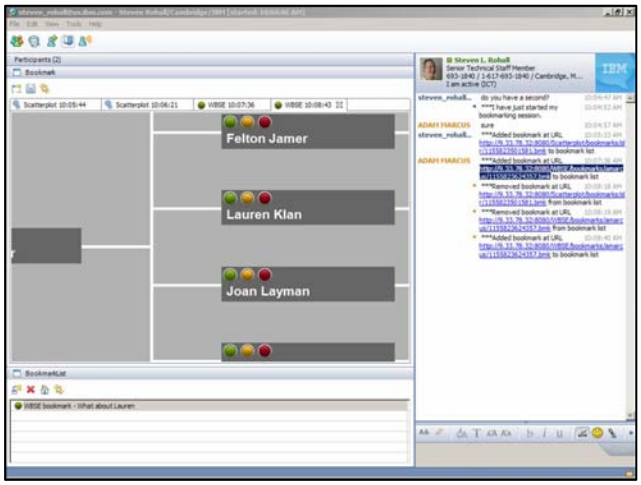


**Figure 1. Shared visualization in a Sametime chat.**

The prototype is implemented in the Java programming language. Our smart storage server, used for bookmarks and externalities is a simple web application server that uses Jetty [3] for its servlet engine. A simple shell application is used to allow a user to run the bookmarkable applications and create bookmarks that can be sent to others for asynchronous collaboration. For communication and rendezvous with collaborators in synchronous collaborations, we use the IBM Lotus Sametime [6] instant messaging client and protocol. Our application and bookmark viewers are written as a plugin to the Sametime client.

Figure 1 shows the integration of a visualization within the standard Sametime chat window. Note that bookmarking activity is included in the chat transcript for later reference and use (the blue text in the figure). In this example, fewer than 100 lines of new code (excluding error handling) were needed to add bookmarking support and event passing for synchronous collaboration to the application.

# 4. DISCUSSION

Our prototype demonstrates the addition of collaboration to an information visualization framework. In particular, we have implemented support for bookmarking, necessary for both synchronous and asynchronous collaboration. Our goal has been to ease the development burden of visualization developers so that they can concentrate on designing their visualizations without worrying about the details of collaboration.

While our investigation has focused on visualization applications, this was an expedient choice as multiple projects in our group involve visualizations and visualization frameworks. As it turns out, visualization applications have features that make them amenable to collaboration. First, they tend not to be customizable by the end user, limiting externalities that result from a user's personal preferences. Second, many visualizations only provide for viewing and analyzing data; they do not support modifying the underlying data like a traditional editing application. This means that there is less of a chance for conflicting user operations.

However, little in our prototype framework turns out to be particular to information visualization. An ongoing area of our research is to standardize our notions of bookmarks and annotations and investigate their use for providing collaboration support in other application development frameworks.

# 5. REFERENCES

[1] Bederson, B. B., J. Grosjean, and J. Meyer, "Toolkit Design for Interactive Structured Graphics," *IEEE Transactions on Software Engineering*, 30(8), August 2004, pp. 535-546.

[2] Begole, J., R.B. Smith, C.A. Struble, and C.A. Shaffer, "Resource Sharing for Replicated Synchronous Groupware," *IEEE/ACM Transactions on Networking*, 9(6), December 2001, pp. 833-843.

[3] Jetty: http://jetty.mortbay.org/jetty/index.html.

[4] Roseman, M. and S. Greenberg, "Building Real Time Groupware with GroupKit, A Groupware Toolkit," *ACM Transactions on Computer Human Interaction*, 3(1), March 1996, pp. 66-106.

[5] Roth, S., "Capstone Address: Visualization as a Medium for Capturing and Sharing Thoughts," *Proceedings of InfoVis 2004*, Austin, TX, October 10-12, pp. xiii.

[6] Sametime: http://www.ibm.com/lotus/sametime.