Sieve: A Java-Based Collaborative Visualization Environment

Philip L. Isenhour James "Bo" Begole Winfield S. Heagy Clifford A. Shaffer *

Virginia Tech

Abstract

We describe **Sieve**, a prototype Java-based collaborative environment for constructing visualizations interactively. **Sieve** allows collaborative construction of data-flow networks from an extensible set of modules. Modules may read data from a variety of sources, filter and transform the data in various ways, and generate visualizations. Annotation tools are also provided for mark-up and documentation of the environment's shared workspace.

Sieve supports collaboration through a replicated architecture and provides real-time information about participants' actions and locations in the workspace. **Sieve** exploits JavaBeans functionality both to provide dynamic, interactive modules and also to support module state consistency among distributed replicas.

CR Categories and Subject Descriptors: H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces – Synchronous interaction; I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction Techniques

Additional Keywords: Collaborative Modular Visualization Environment, Computer-Supported Cooperative Work, Java, JavaBeans

1 Introduction

We describe **Sieve**, a prototype Java-based collaborative environment for constructing visualizations interactively. **Sieve** allows multiple users to simultaneously construct and manipulate a data-flow network in real-time. The users can then collaboratively analyze the resulting visualizations. Visualization state can also be stored, to be regenerated later.

Construction of visualizations by building data-flow networks is common to modular visualization environments (MVEs) such as AVS [6] and Data Explorer [1], which provide powerful tools for performing sophisticated analyses. **Sieve** provides many of the benefits of analysis through data-flow network creation. **Sieve**'s primary innovation is its combination of features: It supports data analysis over the Web by multiple users collaborating in real-time. Furthermore, **Sieve** demonstrates several experimental techniques for using the JavaBeans [5] component architecture to build collaborative interactive systems. One possible characterization of **Sieve** is that it is a collaborative BeanBox.

2 The Sieve Environment

Sieve presents the user with a large, scrollable workspace onto which data sources, processing modules, and visualization components may be dropped, linked, and edited. Figure 1 shows a **Sieve** workspace containing a simple data-flow network.

Modules representing data sources may be written for a wide range of raw data formats and sources. These may include objects which parse files retrieved over the Web, objects which access SQL or other databases, and objects which retrieve data from remote servers using CORBA, Java's Remote Method Invocation library, or proprietary protocols.

Our design allows processing and visualization modules to be generic, with all data-source specific details hidden by the source modules. All data-flow modules implement an API which allows data to be viewed by adjacent modules in the network as a two-dimensional table containing objects of any type supported by the Java language. Source modules simply convert raw data into out table representation. Processing modules can manipulate this data and present an altered or extended table. Visualization modules can then produce visual representations of the data in a table. Visualization modules can serve as an interface for data selection, in which case they may also present an altered or extended table to adjacent modules.

The resulting data-flow network is fully interactive, using an event mechanism to notify interested modules of changes to the data or to the configuration of the network. These modules can then retrieve new or modified data from their source. Users modify the network directly on the workspace, with changes to both the structure of the network and the modules themselves reflected to all collaborators as quickly as processing power and network speed permit.

2.1 Collaboration Support

Sieve supports flexible collaborations and provides real-time information about participants' actions and locations in the workspace.

A range of collaboration styles are supported by providing location-relaxed WYSIWIS (What You See Is What I See) [4], where collaborators may view and manipulate the same or different parts of the shared data simultaneously. Hence while changes made to the workspace are propagated to all remote collaborators, all collaborators need not be working in the same part of the workspace simultaneously.

Sieve provides workspace awareness (continuous knowledge of remote participants' interactions and locations [3]) through two interface elements: telepointers and a multi-user overview of the workspace. A telepointer represents a remote user's mouse pointer position and thus provides location awareness. Collaborators can also use telepointers to gesture at items on the workspace to augment communication. To provide additional workspace awareness information, **Sieve** uses a multi-user overview, or **radar view** [3], of the workspace. The radar view displays a rectangle for each user representing that user's viewport into the workspace, providing additional location information. Remote users' mouse positions are also indicated on the radar view. Figure 2 shows **Sieve**'s telepointer and radar view mechanisms from one collaborator's point of view.

The state of each **Sieve** session is stored on the server, allowing "late-joiners" to be brought up to date. This persistence mechanism also allows **Sieve** to be used for *asynchronous* collaboration. Collaborators may work at different times, leaving their modifications for coworkers to manipulate later.

Sieve additionally supports asynchronous collaboration by providing a set of whiteboard-style tools for annotating the workspace. Lines, arrows, text, images, and even arbitrary Java objects can

* {isenhour, begolej, heagy, shaffer}@cs.vt.edu
Department of Computer Science
660 McBryde Hall
Virginia Tech
Blacksburg, VA 24061

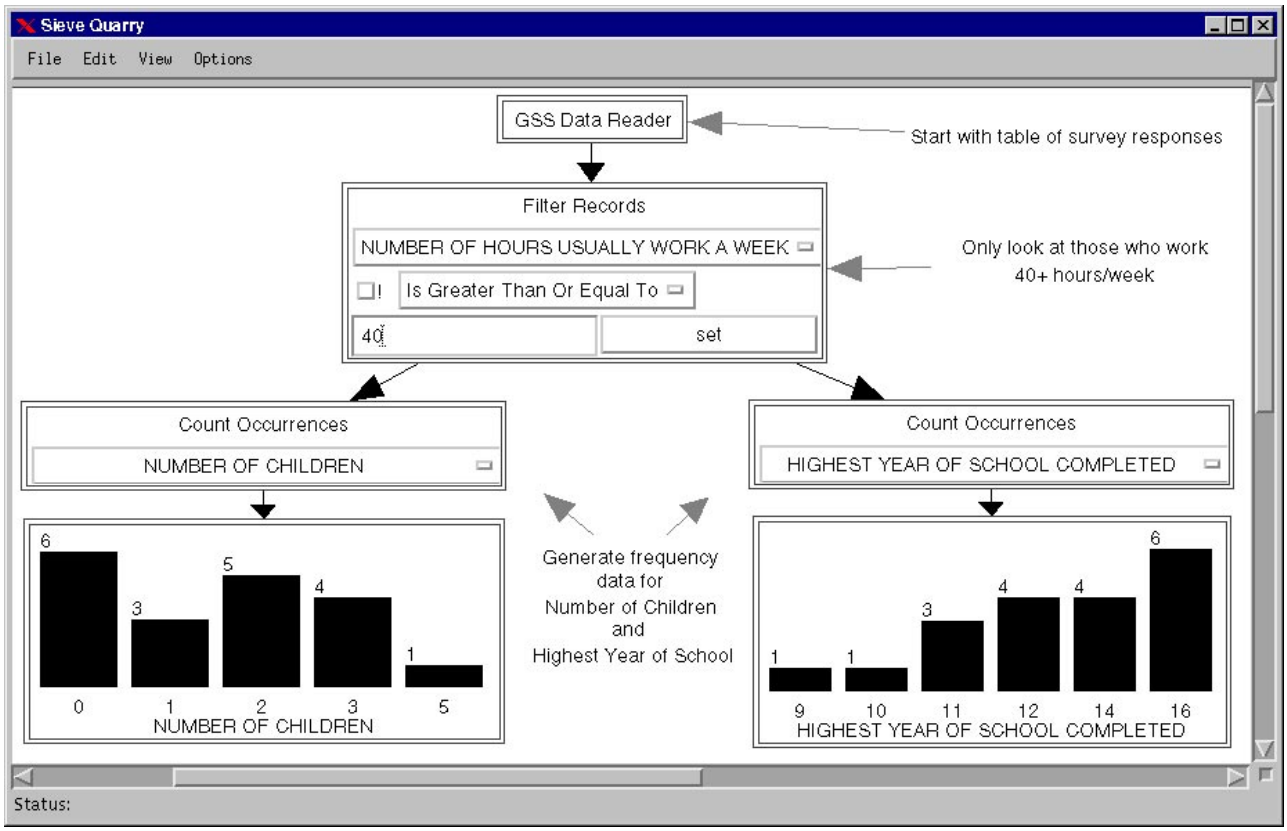

Figure 1: A view of the **Sieve** workspace in single-user mode. Here the user is exploring a subset of the 1993 General Social Survey (GSS) data by selecting only those responses from people who worked 40 or more hours per week, counting the occurrences of values for the "Number of Children" and "Highest Year of School Completed" variables, and then generating bar charts. The steps have been labeled on the workspace using **Sieve**'s annotation tools.

share the workspace with data-flow networks. As with data-flow components, whiteboard components are shared across all collaborating sessions. These tools enhance **Sieve**'s support for asynchronous collaboration, since one collaborator can leave notes on the workspace for later review by other collaborators.

2.2 Use of Java and JavaBeans

Because it is implemented in Java, the **Sieve** workspace and components can be delivered over the Web and can run on a range of hardware platforms and operating systems. These advantages of the Java language make it a natural choice for implementing many kinds of visualization tools. Applets for generating and delivering visualizations over the Web were among the first tools written in Java, and numerous packages are available that allow software developers to generate charts and graphs within Java applets and applications. **Sieve**, however, does not simply provide a set of visualization tools for use by developers, but also supports open-ended collaborative creation of visualizations by multiple end-users.

Beyond simply using Java to create visualization modules, we are exploring new ways to leverage the functionality of the JavaBeans component framework in constructing collaborative and interactive environments. The **Sieve** workspace implementation borrows a number of concepts from Sun's BeanBox builder tool and uses JavaBeans functionality in the following ways:

- **Interaction.** The JavaBeans event model provides a natural communication mechanism among interactive components. This is particularly useful for data-flow visualization systems, as it provides a means for handling both dynamic data sources and interactive filter modules.

- **Module Configuration.** JavaBeans contains mechanisms for generating property editors and customizers for use at design-time by application-builder tools. **Sieve** uses these mechanisms to construct lightweight user interfaces for run-time configuration of modules.

- **Collaboration.** The JavaBeans concept of "bound properties" (binding attributes of one object to compatible attributes of another object) is extended by **Sieve** to support collaboration. JavaBeans provides mechanisms for detecting and propagating changes to bound properties of local objects. In **Sieve**, we extend this to support propagation of these changes to each replica of an object in the collaborating sessions. This allows many kinds of components to be written without knowledge that they are being used collaboratively.

- **Persistence and Publication.** By preserving the JavaBeans features intended for use by application builder tools, we can turn all or part of the contents of a **Sieve** workspace into an applet (or other appropriate Java object) for publication in a Web page.

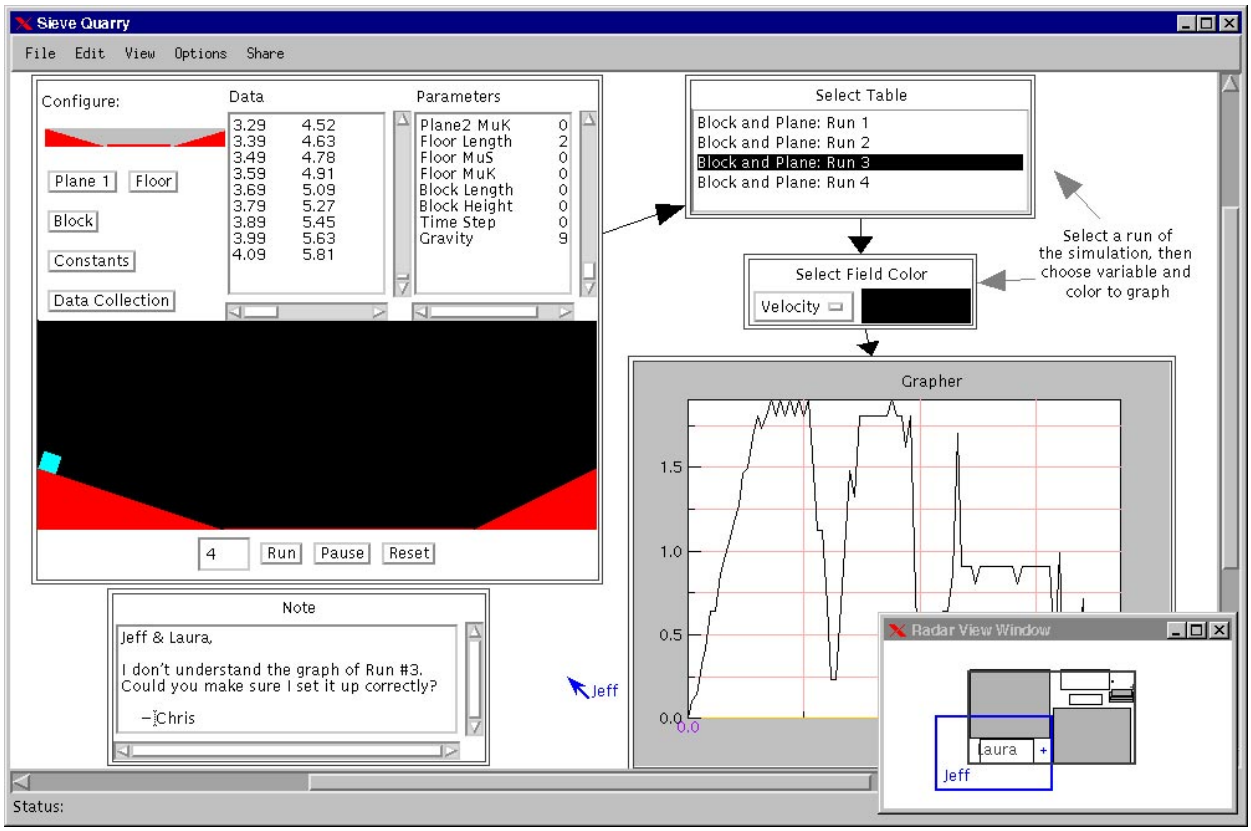

Figure 2: Two users (Jeff and Laura) collaborating in a **Sieve** virtual physics laboratory. The users have constructed a simple network that allows them to run a simulation of a block and plane experiment. Variables from different simulation executions can then be graphed. Laura's view is shown. Jeff's viewport is shown in the radar view window (lower right corner), and his present mouse pointer position is shown by a telepointer. A note left by a third collaborator (who is not currently present in the workspace) is shown at the bottom left of the screen.

3 Applications

Although its capabilities extend beyond education, **Sieve** is particularly targeted at real-time collaboration support of education. Teachers often demonstrate previous findings to students. Extending presentations to remote students gives teachers and students more flexibility, because they do not have to be physically co-located. A more exciting aspect to real-time collaborative visualization is the potential for groups of students to work through an analysis together.

Sieve was originally developed primarily as a platform for two educational software systems: a tool for interactively exploring social science data (Figure 1) and a virtual physics laboratory (Figure 2). For exploring social science data, **Sieve** includes modules for reading databases such as the U.S. Census and General Social Survey, modules for querying and filtering these data, modules for performing statistical analysis, and basic charting tools. In the virtual physics laboratory, students produce visualizations of data collected from sensors or generated by simulation components. To demonstrate the range of applications in which **Sieve** can be used, we have also written source modules for reading access logs generated by Web servers and financial data such as Consumer Price Index statistics.

4 Future Work

By basing much of **Sieve**'s functionality on standard JavaBeans mechanisms, it is possible to use a wide variety of arbitrary Java objects within **Sieve**. This flexibility makes the system useful not only for collaborative analysis tasks, but also for a range of collaborative design tasks.

We are currently extending the set of modules to include more types of interactive components. In particular, we are interested in providing components which support design and construction of simulations, especially of systems for which the concept of linked components is natural. One such simulation currently being developed uses **Sieve** for collaborative design, construction, and analysis of simple electrical circuits.

Other extensions include improved support for aggregate components. This functionality will allow pieces of a component to be individually edited and linked to other components.

Because **Sieve** uses the JavaBeans' property change listener mechanism to update each collaborator's copy of the workspace, developers of **Sieve** components are not concerned with whether the component is used by a single person or collaboratively. In this way, **Sieve** components are **collaboration unaware**. However, developers do need to conform to the JavaBeans specification. JAMM [2] is a system we are developing that allows legacy single-user applications, which are inherently collaboration unaware, to be used collaboratively. We plan to include sharing of legacy applications via JAMM as part of **Sieve**.

5 Conclusions

We have presented **Sieve**, a prototype collaborative interactive visualization environment. The environment allows multiple users to analyze and visualize data through the creation of data-flow networks. These networks are created from an extensible set of modules that represent sources of data, data manipulation processes, and visualizations. The modules can be linked by users to perform a variety of analysis and visualization tasks. In addition to support for data-flow modules, simple modules are provided for workspace annotation.

Sieve supports flexible collaborations through location-relaxed WYSIWIS. Real-time information about participants' actions and locations in the workspace is provided through the use of telepointers and a multi-user radar view of the workspace. A persistence mechanism supports late-joiners to **Sieve** sessions, and also allows **Sieve** to be used for asynchronous collaboration.

Sieve makes use of the JavaBeans component architecture in several ways. In particular, the JavaBeans concept of a bound property is extended to support module state replication across collaborating sessions.

More information about **Sieve** is available at the following URL: http://simon.cs.vt.edu/Sieve.

6 ACKNOWLEDGMENTS

We gratefully acknowledge the support of the National Science Foundation under grant REC-9554206, and the Department of Education's FIPSE program under grant P116B60201.

References

[1] G. Abram and L. A. Treinish. An Extended Data-Flow Architecture for Data Analysis and Visualization. In *IEEE Visualization '95*, pages 263–270. IEEE, October 1995.

[2] James 'Bo' Begole, Craig A. Struble, Clifford A. Shaffer, and Randall B. Smith. Transparent Sharing of Java Applets: A Replicated Approach. In *1997 Conference on User Interface Software and Technology (UIST'97)*, October 1997.

[3] C. Gutwin, S. Greenberg, and M. Roseman. A Usability Study of Awareness Widgets in a Shared Workspace Groupware System. In *Computer-Supported Cooperative Work*, pages 258–67. ACM Press, 1996.

[4] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar. WYSIWIS Revised: Early Experiences with Multiuser Interfaces. In *ACM Transactions on Office Information Systems*, pages 147–167. ACM Press, April 1987.

[5] JavaBeans (tm). URL: http://splash.javasoft.com/ beans/-beans.100A.pdf, December 1996.

[6] Craig Upson, Thomas A. Faulhaber, Jr., David Kamins, David Laidlaw, David Schlegel, Jeffrey Vroom, Robert Gurwitz, and Andries van Dam. The Application Visualization System: a Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, July 1989.