A Prototype Notebook-Based Environment for Computational Tools

Jenifer L. Skidmore
Department of Computer and Information Science
1202 University of Oregon
Eugene, OR 97403-1202
`skid@cs.uoregon.edu`
`http://www.cs.uoregon.edu/~skid/`

Matthew J. Sottile
Department of Computer and Information Science
1202 University of Oregon
Eugene, OR 97403-1202
`matt@cs.uoregon.edu`
`http://www.cs.uoregon.edu/~matt/`

Janice E. Cuny
Department of Computer and Information Science
1202 University of Oregon
Eugene, OR 97403-1202
`cuny@cs.uoregon.edu`
`http://www.cs.uoregon.edu/people/faculty/cuny.html`

Allen D. Malony
Department of Computer and Information Science
1202 University of Oregon
Eugene, OR 97403-1202
`malony@cs.uoregon.edu`
`http://www.cs.uoregon.edu/people/faculty/malony.html`

Abstract:

The Virtual Notebook Environment (ViNE) is a platform-independent, web-based interface designed to support a range of scientific activities across distributed, heterogeneous computing platforms. ViNE provides scientists with a web-based version of the common paper-based lab notebook, but in addition, it provides support for collaboration and management of computational experiments. Collaboration is supported with the web-based approach, which makes notebook material generally accessible and with a hierarchy of security mechanisms that screen that access. ViNE provides uniform, system-transparent access to data, tools, and programs throughout the scientist's computing infrastructure. Computational experiments can be launched from ViNE using a visual specification language. The scientist is freed from concerns about inter-tool connectivity, data distribution, or data management details. ViNE also provides support for dynamically linking analysis results back into the notebook content.

In this paper we present the ViNE system architecture and a case study of its use in neuropsychology research at the University of Oregon. Our case study with the Brain Electrophysiology Laboratory (BEL) addresses their need for data security and management, collaborative support, and distributed analysis processes. The current version of ViNE is a prototype system being tested with this and other scientific applications.

Keywords:
electronic notebook, distributed computing, computational science, heterogeneous, tools, World Wide Web, collaboration

1 Introduction

Scientists increasingly rely on computational tools in their research. These tools range from large-scale simulations to the more routine packages that perform statistical and mathematical analysis, collect and store information in databases, produce graphical visualizations of data and results, publish papers on the web, and support collaboration over distances. Coordinating the use of such disparate programs across today's distributed, heterogeneous computing platforms can be challenging. This paper introduces the Virtual Notebook Environment (ViNE): a platform-independent, web-based interface that utilizes a traditional lab notebook analogy with chapters, pages and a table of contents. ViNE extends the notebook to support a range of scientific computational activities in a uniform manner.

Our target users are typified by neurophysiology researchers at the University of Oregon. They use high density electrode arrays to collect brain voltage potentials measured on the scalp over intervals lasting serveral milliseconds. A standard (paper) lab notebook is kept for each subject, recording information about the experiment (such as time, experimental setup, and environmental distractions that were present, etc.). This information is later used during analysis to identify valid tests and subjects. The analysis itself is done with commercially available statistical and numerical packages that run on a variety of laboratory machines; data is manually moved from machine to machine when necessary. Whereas the subject notebooks are shared among all of the researchers, each individual scientist keeps his/her own analysis results, usually in folders. The neurophysiology researchers seldom formally record the "process" of analysis (that is, the sequence of computations run), but rather reconstruct the process for design walk-throughs with colleagues or for publications. Other scientists that we have worked with on large simulations [6] do keep a detailed record of their computational analysis, recording parameter modifications and intermediate simulation output. ViNE aims to support all of these activities across modern computing networks.

ViNE provides scientists with a web-based equivalent of a traditional paper notebook extended with support for notebook sharing, security, and computational tool access and management. Although other electronic notebooks provide collaboration support [8, 10, 14, 18] and features such as automatic data acquisition [20] and image annotation [10], ViNE focuses on the entire computational environment, giving the scientist complete access to laboratory notebooks, experimental data, and analysis tools in a location- and platform-independent manner. He/she can review data, launch analysis tools, visualize and save the results, add additional notes or annotations, and capture the analysis process. Furthermore, ViNE is accessible through a simple, easy-to-use visual interface, and it frees the scientist from considerating the exact locations of data, programs, and tools.

Specifically, ViNE contributes:

- a basic data and tool organization infrastructure;
- a framework for distributed, heterogeneous computing; and
- an abstract, high-level interface that supports the seamless integration of computational resources.

In the next two sections, we describe the architecture and implementation of ViNE respectively. Following that, we present a case study of its use in the dense electrode array research project.

2 Architecture

The ViNE architecture consists of one or more nodes, called *leaves*, distributed across the set of computer systems that make up the scientist's computational environment, as in Figure 1. Leaves can reside on the same or different platforms with no restrictions on their numbers or configuration. Leaves can be configured and located in a way that best fits the distributed system and the scientists' needs. Together, the collection of leaves within the computer system defines the notebook environment.

Leaves are connected together by *stems* which provide inter-leaf communication. Each stem has a local server and a library of communication methods. The server communicates with other stems by sending messages across the network. Servers are responsible for packing and unpacking messages and interacting with the local, low level communication protocols. The library of communication methods hides these low level details from the user by providing the leaves with standard, platform-independent interfaces. Thus, ViNE makes it possible to uniformly access distributed resources (notebooks, tools, data, etc.), supporting the exchange of information and access to specialized capabilities across platforms. In this manner, users are shielded from the distributed nature of ViNE.

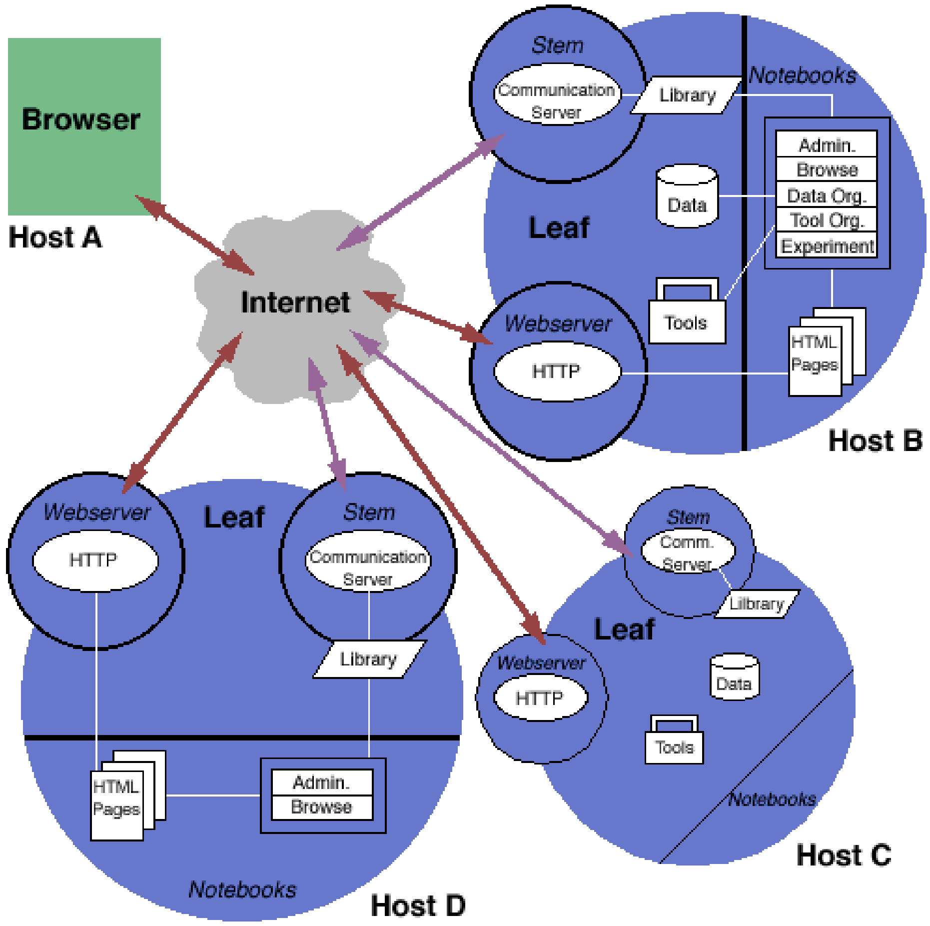

Figure 1 - ViNE architecture with example leaves.

Each leaf contains a web server, a stem, and a set of notebooks as shown in Figure 1. In addition, it may contain one or more *functional components*, depending on its resources and purposes. There are five possible components -- notebook browsing, notebook administration, data organization, tool organization, and experiment control -- each with its own function and well-defined interface. Leaves can provide complete functionality with all five components, such as Host B in the figure, or they can be more restricted. For example, a leaf limited to notebook administration and browsing components, like the leaf on Host D), can provide only the basic recording functions of a traditional (paper) notebook. Maximally restricted leaves have no components at all, just a stem and a web server, as seen in Host C. Such leaves function as data and computation servers that do not house notebooks. Instead they provide access to data stored in large data repositories or act as computation servers running specialized tasks.

The functional components are divided into two groups. The first provides basic, traditional notebook facilities, and the second provides the ability to structure, launch, and record the results of computational experiments. We describe each group in more detail.

2.1 Functional Components: Basic Notebook Functionality

The notebook browsing and notebook administration components work together to provide basic notebook functions. The *notebook browsing component* is available to all users who have minimal security clearance for a notebook. It displays the notebook in a series of formatted, easily browsed pages constructed from elements described in the data organization and notebook administration components. The *notebook administration component* provides the users with a set of controls to manage notebook structure and security parameters. Structural notebook controls allow item addition and deletion, page organization, and individual page layout.

To provide a secure environment, each notebook has a set of permissions and an owner; only the owner can grant access to a notebook. There are three levels of permissions: none, read only, and read/write. When a user wants to enter the ViNE environment, they are required to log in, after which they are provided with a list of accessible notebooks. As a user maneuvers through the environment, they are monitored to guarantee that there are no unauthorized accesses to a notebook. ViNE security supports a flexible collaborative environment while maintaining a range of data protection levels for each user.

2.2 Functional Components: Computational Experiments

The other three components -- data organization, tool organization, and experiment control -- extend the notebook functions for executing and recording computational experiments. These components allow the user to create named objects with descriptive information maintained by ViNE. They can be referenced throughout the notebook, freeing the user from considering many low level aspects of tool and data use.

The *data organization component* gives notebooks the ability to reference data throughout the ViNE system. Users must describe and name their data within their notebook before it can be referenced. Flat files, for example, are described by giving them a name, providing the name of the host it is located on, and the complete path to the data file. Data that can be interpreted and understood when viewed, such as matrices and images, can be displayed on the notebook pages. All described data is available for use in the experiment control component.

The *tool organization component* is used to describe and name tools that are available within the scientist's computational environment. In order for tools to be accessed by ViNE their interface must be "wrapped." A tool and its wrapper are located on the same leaf, where the wrapper provides an interface between ViNE and the tool. Part of the description the user gives includes information about wrappers. For example, when describing MATLAB [17] as a tool, the tool name given would be "MATLAB", and the custom wrapper name would be "matlab.cgi". Additional information is also required about tools such as location, host name, and inputs and outputs. Once a tool and wrapper are described, the tool is available for use in the experiment control component and is referenced by the given name.

The *experiment control component* provides users with a powerful method of combining data, tools, and distributed computing resources to perform a variety of analysis tasks. This component is split into an *experiment builder* and *execution controller*. The experiment builder provides an abstract visual interface for constructing experiment processes. The controller module acts as a central hub for experiment operations, handling data and tool synchronization, error control, and output management.

Once a user has properly described their data and tools to ViNE, they can be utilized in an experiment with no further concern for inter-tool communication or distributed data management. Experiments are represented at a high level of abstraction using a directed, acyclic graph with nodes representing entities (tool, data, and experiments) and edges representing data flow between them (c.f. Figure 4).

Once an experiment has been designed with the visual builder and stored in a notebook, it can be executed at any time. "Execution" of the experiment occurs at the leaf on which the user's notebook resides using its experiment controller module. During execution, the controller communicates with remote leaves to move data needed for tools, and to issue commands for tool execution. The controller ensures that data and tool operations are properly synchronized.

Final output and experiment status information are stored in the notebook by the experiment controller. Progress information can be gathered during an experiment for monitoring operations that span long periods. After an experiment has been executed and its results have been stored, the new data is available for use in subsequent experiments. As suggested above, the experiment can also be treated as a discrete entity within a larger experiment. This ability to embed experiments allows users to build libraries of common tasks.

3 Implementation

The World Wide Web provides a collaborative, distributed, heterogeneous environment. Our implementation takes advantage of web-related technologies such as Hypertext Transport Protocol (HTTP) [9], Common Gateway Interface (CGI) [5], Java [1], and Perl [23] to achieve portability and consistent interfaces across platforms. Here we discuss the implementation of stems, the data and tool organization components, and the experiment control component.

3.1 Stems

Stems use their servers to communicate across the network with each other. The stem servers are persistent and multi-threaded, allowing them to handle multiple requests at one time. Requests are sent across TCP/IP sockets in a specified format similar to Uniform Resource Locators (URLs). This format is used for all requests made to the servers. The servers are written in Java [1], exploiting its widespread availability on many commonly used platforms, and its abundant built-in routines for socket communication and threads.

In order for other programs to interact with the servers, they must use the library of communication methods provided by the stem. The library includes, and other Java programs can import, a Java package that handles socket communication. Perl [23] scripts executed within the notebook invoke a Java-based stub program that then makes the server request on its behalf. This set of library routines thus provides a consistent, portable interface with the communication servers.

3.2 Data and Tool Organization Components

The data and tool organization components provide the management facilities for a notebook's resources. The data component allows notebooks to reference data throughout the ViNE system. ViNE currently recognizes just two types of data -- *tabular* and *general* data. Tabular data is a matrix of values, fully described by its dimensions. General data is treated as a flat file. Data files are accessed either locally or remotely by giving a host name and path. The data component provides the facilities to insert and remove data into a notebook regardless of its location and format.

The tool organization component maintains descriptions of available tools and their wrappers. Tools are categorized as either *general* or *custom*, depending on whether they can be fully controlled from the command line (general) or require more complex interaction and control (custom). Wrappers are Common Gateway Interface (CGI) [5] scripts that run a tool with the given parameters and can be accessed by a leaf's web server. ViNE provides wrappers for general tools. To register a general tool with ViNE, the user provides its name, location, and a description of its input and output parameters. For a custom tool, he/she must also supply a wrapper to interface between ViNE and the tool. The wrapper is responsible for accepting a complex string containing the input and output information, translating it for the specific tool, and executing the tool correctly. In either case, once the tool is registered, it can be used in experiments.

3.3 Experiment Control Component

The experiment control component facilitates the construction and execution of distributed computational tasks. It is composed of an experiment builder and an experiment controller and is implemented in a mixture of Perl-based CGI and Java-based executables for portability.

The experiment builder is a Java applet executed within a user's web browser. It gathers information from the user's notebook to determine which objects are available. Once this list is populated, users can visually construct experiments by choosing the experiment "entities" (data, tools, other experiments, and control structures) and defining the data flow between them (c.f. Figure 4). The builder monitors the syntax of these constructed data flow models, preventing such things as models with obvious, nonterminating loops or input/output incompatibilities between entities.

Between experiment construction and execution, CGI-based scripts allow the user to store experiments within his/her notebook in two forms. A visual description of the data flow model is kept for future editing, and a textual version of the experiment is kept for execution. The language for the textual version was designed to support the easy definition of ViNE data flow models and to support the addressing of program entities as abstract methods with well-defined input/output characteristics. Figure 2 shows the language representation for a visually constructed experiment. The definition of entity behavior is established by both the data and tool organization components.

After an experiment is constructed it can be executed with the Java-based execution controller that acts as a central manager. The controller utilizes the built-in Java threading mechanisms to create and manage sequences of operations as specified in the data flow model. Each entity is assigned a single thread, and each is given references to the threads that can affect its execution state. An example of the thread structure for a visually constructed experiment graph can be seen in Figure 2. Entity interactions are supported in sequential, fork-join parallel, and disjoint parallel situations with inter-sequence synchronization.

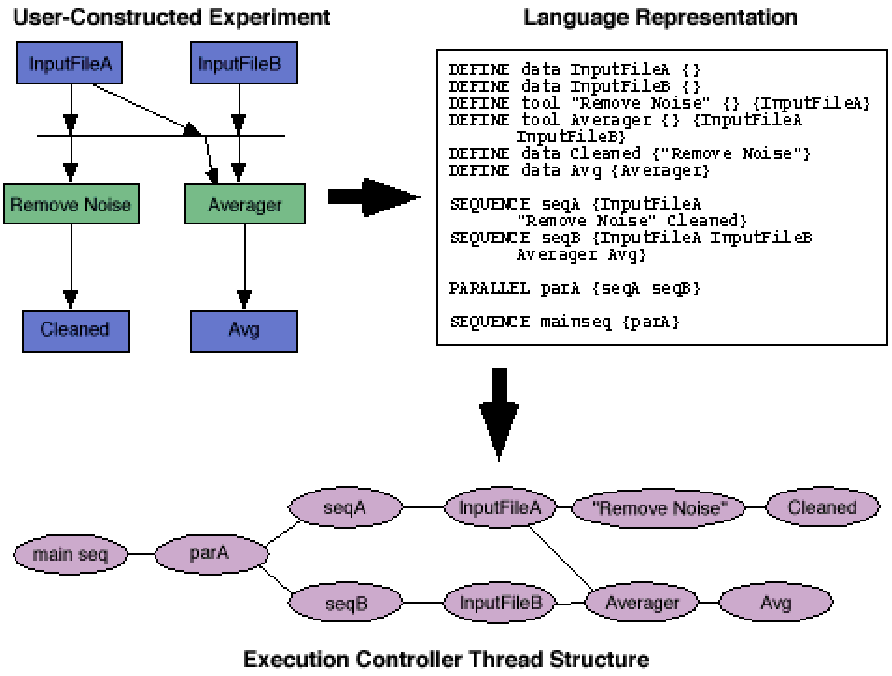

Figure 2 - Experiment Representations

The final step in any experiment executed within ViNE is the gathering of results. The data produced by an experiment can be automatically stored within the user notebook for later analysis or browsing. This ability to capture results and store them allows the users to easily keep a record of many experiments, greatly reducing the possibility of accidental information loss that often occurs with paper notebooks and manually organized files.

4 Application

The Brain Electrophysiology Laboratory (BEL) at the University of Oregon combines the studies of the brain with analytic methods found in cognitive psychology [2]. Much of our work with them has focused on the dense electrode array research project as described earlier.

After analyzing their computational environment and how they use it, we identified the following needs:

- Support for collaboration among the BEL scientists through notebook sharing;
- A secure environment where unpublished results could be protected;
- The ability to access notebooks from any platform or location; and
- Support for more fully exploiting their distributed, heterogeneous computing facilities.

To address these needs, we created a set of secure, collaborative ViNE notebooks for the BEL scientists. A common notebook is open to everyone in the lab and allows sharing of data and results; individual notebooks are private and contain recent, unpublished data and results. The scientists have full control over how their notebooks are organized and who has access to them.

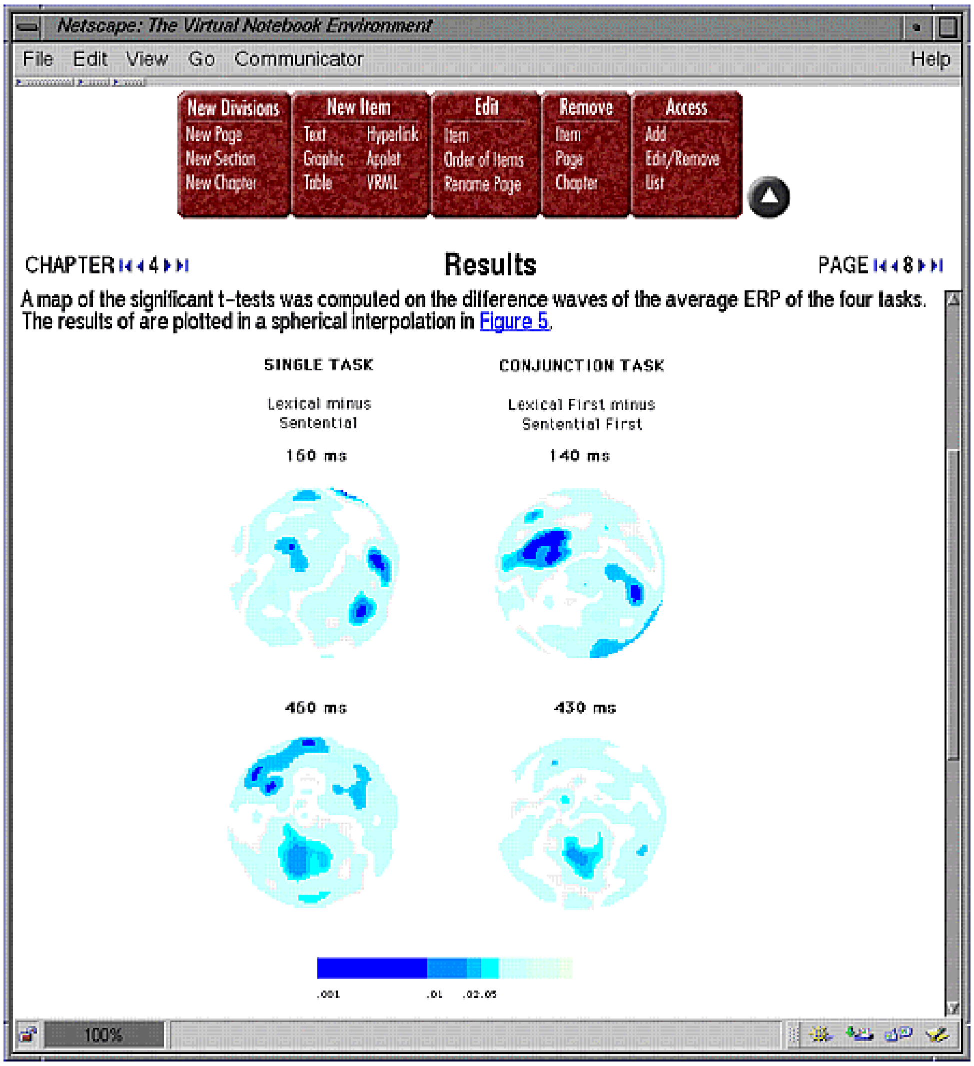

Figure 3 - Notebook page

The BEL scientists' data consists of electroencephalogram (EEG) output, numerical results, and visualizations. It is stored as it would be in a traditional notebook, on pages organized into sections and chapters. A page is shown in Figure 3. It resembles a traditional notebook page with the addition of the command palette and navigation aids across the top. As the notebook pages are turned, the command palette remains at the top for easy access.

Most of the BEL scientists' tools are command files containing functions that are interpreted in the MATLAB environment. The scientists rely heavily on MATLAB, and thus it was the first application we wrapped, making most of their existing tools available within ViNE. In addition, we added GNUplot [12], an image conversion tool, and a movie generator to enhance their capabilities.

Information regarding location and usage of data and tools is captured as they are added to the notebook and is available to the experiment builder. Most of the BEL tools were built by combining the MATLAB application with a specific command file. These MATLAB-abstracted tools allow various analysis processes to be combined in a larger experiment. One such experiment is shown in Figure 4 where tools Grand Averager and Interpolation are used consecutively on data file OAInputs to produce data files Interp_s139, Interp_s140, and Interp_s141.

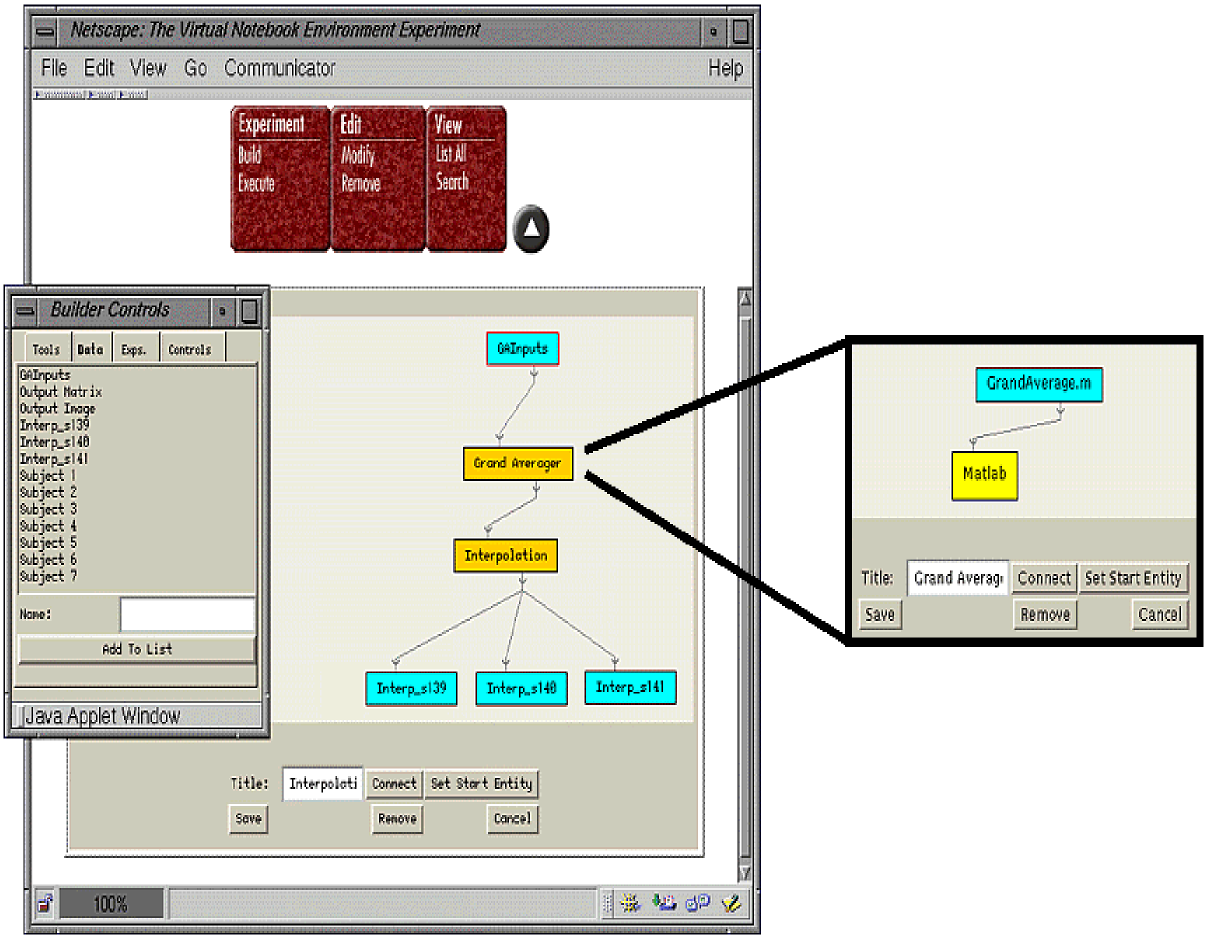

Figure 4 - Neurophysiology experiment (abstract tool detail)

Once the experiment is described, it can be executed and the results can be stored back in the notebook. This experiment was executed in conjunction with a visualization tool.

Together, they produced the image shown in Figure 3, which was automatically inserted into the notebook. In this manner, many of the tedious details of their computing environment were hidden from the BEL scientists; as they ran their experiments they did not need to be mindful of where their tools were, and they did not need to move any data manually.

5 Related Work

Related work on electronic notebooks falls into two categories: those that focus on data recording and those that provide support for running tools. In the first category, notebooks that permit data entry in the form of text and images are prevalent [13, 20, 21, 22]. In recent years, some have been enhanced to provide collaborative environments [3, 8, 10, 14, 18, 19]. In one large collaborative notebook project, part of the DOE 2000 project [7], researchers at Oak Ridge National Laboratory [10], Lawrence Berkeley National Laboratory [21], and Pacific Northwest National Laboratory [18] are working to combine their individual electronic notebook projects with a common notebook and communications mechanism.

Electronic notebooks that allow the user to run tools and store results also exist. Gene Inspector [11] combines a DNA sequence analysis package and an electronic notebook to aid in studying DNA sequences. Tektronix sells an electronic lab notebook that records Digital Storage Oscilloscope measurements [24]. These highly specialized notebooks are packaged with other software tools and are not general purpose in nature. ViNE, in contrast, focuses on incorporating existing tools without requiring the user to learn new tools. Similarly, the Northwest Parallel Architectures Center at Syracuse University provides collaborative tool environments, such as SciVis [16] and WebWisdom [25], that can be customized for the user. At Purdue, work has been done on creating a system where users can access and run existing software tools across the World Wide Web using a browser [15]. NetSolve [4] is another system that allows users to access computational resources distributed across the network. It provides an interface to scientific packages and supports interaction from within MATLAB, a command-line shell, C and Fortran programs, or the World Wide Web. Like ViNE, these systems are general frameworks that can be applied to specific groups of users and their tools. However, they lack ViNE's standard notebook and automated recording abilities.

6 Future Directions

The current version of ViNE is a prototype in use by scientists who are testing it as we make ongoing improvements. During this testing, we have identified a number of enhancements for future versions of the environment. The user interface, for example, needs some modification. The notebook administration component will be extended with a framework for integrating editors appropriate to a variety of data content, allowing users to directly manipulate images, data, and text without leaving ViNE. Currently, we use a persistent-server model for leaf process scheduling but to support systems with low resources, we will provide on-demand processing. For the scientist with very large amounts of data, we will provide more sophisticated ways to organize it, and we need to be able to link directly to existing databases. Adding support for automated maintenance of the tools' configurations described in the notebook is another area of future development. Finally, we will enhance the experiment component, adding facilities for experiment monitoring, steering, and analysis.

Acknowledgements

Thanks to Richard Holleman, Scott Herz, Craig Thornley, Kristin Kaster, Kimberly Miller, and Dustin Preuitt for their work on the project. We also appreciate the interest and involvement of the people working at BEL, especially Antonella Pavese, Michael Posner, and Don Tucker.

This project was supported by the following grants: National Science Foundation Metacenter Regional Alliance No. ASC-9523629, National Science Foundation No. ACI-9457530, and National Science Foundation No. CCR-9023256.

References

[1] K. Arnold and J. Gosling. **The Java Programming Language**. Addison-Welsey Publishing Co., 1996.

[2] **Brain Electrophysiology Laboratory**. http://hebb.uoregon.edu/brainlab/belHome.html.

[3] A. M. Burger. **The Virtual Notebook System**. in Proceedings Hypertext'91, pp. 395-401.

[4] H. Casanova and J. Dongarra. **NetSolve: A Network Server for Solving Computational Science Problems**. In Proceedings of the Supercomputing Conference, 1996.

[5] K. Coar, et. al. **The WWW Common Gateway Interface Version 1.2**. http://Web.Golux.Com/coar/cgi/cgi-120-00a.html.

[6] J. Cuny, et. al. **Building Domain-Specific Environments for Computational Science: A Case Study in Seismic Tomography**. Intl. Journal of Supercomputing

Applications and High Performance Computing, Vol. 11, No. 3, pp. 179-196, 1997.

[7] **DOE 2000 Electronic Notebook Project**. http://www.epm.ornl.gov/enote/.

[8] D. C. Edelson, R. D. Pea, and L. M. Gomez. **The Collaboratory Notebook**. Communciations of the ACM, 39(4):32-33, April 1996.

[9] R. Fielding, et al. **Hypertext Transfer Protocol - HTTP/1.1**. Internet Engineering Task Force, http://ds.internic.net/rfc/rfc2068.txt, January, 1997.

[10] A. Geist and N. Nachtigal. **Oak Ridge National Laboratory Electronic Notebook Project**. http://www.epm.ornl.gov/~geist/java/applets/enote/.

[11] **Gene Inspector**. http://www.textco.com/.

[12] GNU's Not Unix. **GNUplot**. http://www.gnu.org/software/gnuplot/gnuplot.html.

[13] J. Gwizdka, J. Louie, and M. S. Fox. **EEN: A Pen-based Electronic Notebook for Unintrusive Acquisition of Engineering Design Knowledge**. In Proceedings of the IEEE Fifth Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises, June 1996.

[14] J. Hong, et al. **Personal Electronic Notebook with Sharing**. In Proceedings of the Fourth Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises, April 1995.

[15] N. H. Kapadia and J. A. Fortes. **On the Design of a Demand-Based Network-Computing System: The Purdue University Network-Computing Hubs**. In Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 1998.

[16] B. Ki and S. Klasky. **Scivis**. In Proceedings of ACM 1998 Workshop on Java for High-Performance Network Computing, Palo Alto, CA, February 1998.

[17] Math Works, The. **MATLAB: High Performance Numeric Computation and Visualization Software Reference Guide**. Natick, MA. 1992.

[18] J. D. Myers, et al. **Electronic Laboratory Notebooks for Collaborative Research**. In Proceedings of the IEEE Fifth Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises, June 1996.

[19] D. K. O'Neill and L. M. Gomez. **The Collaboratory Notebook: a Networked Knowledge-Building Environment for Project Learning**. In Proceedings of Ed-Media '94: World Conference on Educational Multimedia and Hypermedia, Vancouver, BC, Canada, June 1994.

[20] B. Rex and D. St. Pierre. **PNNL NMR Electronic Logbook**. In Proceedings of the IEEE Fifth Workshop on Enabling Technology: Infrastructure for Collaborative Enterprises, June 1996.

[21] S. Sachs, et al. **The Spectro Microscopy Electronic Notebook**. Technical report LBNL-39886, January 1997.

[22] W. H. Uejio, et al. **An Electronic Project Notebook from the Electronic Design Notebook (EDN)**. In Proceedings of the Third National Symposium on Concurrent Engineering. Concurrent Engineering Research Center, West Virginia University, February 1991.

[23] L. Wall, T. Christiansen, and R. L. Schwartz. **Programing Perl**. O'Reilly & Associates, 2nd edition, 1996.

[24] **WaveStar**. http://www.tek.com/Measurement/Products/catalog/wavestar/.

[25] **WebWisdom**. http://www.webwisdom.com/.