An Overview of the Modeler's Workspace

Michael Hucka, Kavita Shankar, Sara Emardson
{mhucka,kshankar,semard}@bbb.caltech.edu
Bower Research Laboratory, Division of Biology 216-76
California Institute of Technology, Pasadena, CA 91125

27 April 2000

1 Introduction

A realistic neuronal model represents a modeler's understanding of the structure and function of a part of the nervous system. As the number of neurobiologists constructing realistic models continues to grow, and as the models become ever more sophisticated, they collectively represent a significant accumulation of knowledge about the structural and functional organization of nervous systems. But at the same time, locating appropriate models and interpreting them becomes increasingly difficult.

The central motivation for the Modeler's Workspace project is to address these problems of locating and interpreting models by creating software for assisting computational neuroscientists. The system is intended to help users create and organize models and to interact with databases of models and neuronal simulation software. More specifically, the Modeler's Workspace is designed to provide the following:

- Search and retrieval facilities for interacting with databases of models and other information;
- Facilities for creating, editing and visualizing the characteristics of models;
- Facilities for combining model components together and translating them into formats suitable for simulation systems such as GENESIS and NEURON;
- Facilities for managing a personal database where a user can collect models and other objects; and
- Collaboration facilities for connecting one or more users together, to allow them to simultaneously edit objects in a shared database and communicate with each other using real-time chat.

In this document, we provide an overview of the Modeler's Workspace system, focusing on its overall architecture. There are three components in the system: a user interface, a database server, and a global registry and repository called the Modeler's Workspace Directory. We summarize each of these components below and discuss key points about the model representation and communication mechanisms.

2 The Modeler's Workspace User Interface and Database

The Modeler's Workspace *User Interface* is a stand-alone program written in Java that can run either as a separate application, or as an applet from within a web browser. All user interactions with the Modeler's Workspace take place through the User Interface component. The *Workspace Database*, the database component of the system, is a separate server program that can act as both a private repository for a user's work

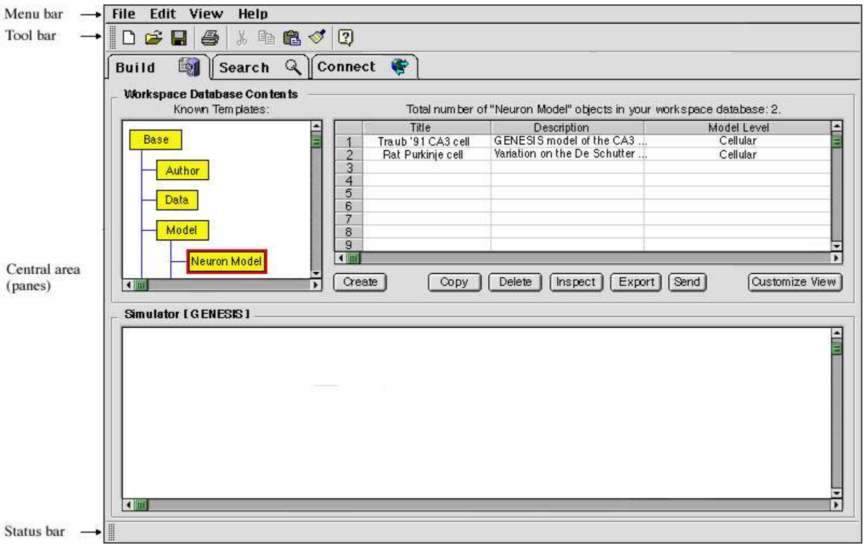

Figure 1: *A mock-up of the Modeler's Workspace User Interface. It consists of a menu bar, a tool bar, a large central area, and a status bar at the bottom. This particular mock-up shows the* **Build** *pane.*

(where models, notebooks and other items are stored) and as a rendezvous point for collaborative activities. The User Interface communicates with the Workspace Database using network protocols.

The separation of the two components allows the flexibility of connecting to a Workspace Database from different computers. Because the User Interface is a portable Java program, it can be downloaded from the Modeler's Workspace web site (`www.modelersworkspace.org`) and run on any Java-enabled device connected to the Internet. When the User Interface is started, it prompts for the network address of the Workspace Database as well as a login name and password for accessing the database. This approach allows a roaming user to access his or her models and notes from anywhere on the network, using almost any kind of computer—from a public-access computer in a library to a laptop connected to the Internet via a modem.

2.1 Objects in Workspace Databases

A Workspace Database contains objects that represent different types of entities such as models or bibliographic references. Each database object is structured according to a particular template. A *template* defines the format of an object, including its attributes and subattributes and the permitted types of data that can be stored in each attribute. Templates have names, and different templates are used to define different types of objects. Some of the basic types of objects predefined by the Modeler's Workspace are "Author", "Reference", "Data", "Neuron Model" and "Ion Channel Model".

A template named *Base Template* is the lowest common denominator for all representations. All other templates are defined by starting either with Base Template or an existing template, then extending it to add new attributes. Only the addition of attributes is permitted; deletions are not allowed in order to ensure that attribute meanings stay the same when a specialized template is derived from a more general one.

2.2 The Main Work Areas in the User Interface

There are three central regions in the User Interface: the **Build** pane, the **Search** pane, and the **Connect** pane. The first pane provides an interface for managing one's personal database of objects; the second, for searching other databases; and the third, for connecting to a specific workspace database for the purpose of browsing its contents and (optionally) engaging in collaborative activities with other connected users.

Figure 1 shows a mock-up of the **Build** pane in the User Interface. The upper left region of the **Build** pane contains a list of the template types known to the system; the upper right region provides a listing of the objects in the user's workspace database that are based on a selected template; and the bottom half of the pane contains a window to the command line interpreter of a neuronal simulation package (in this example, GENESIS). The user may select objects in the workspace database list in the upper right and perform actions on them, such as editing them in an inspector window (described next) or sending them to the running simulation program. The columns displayed in the table can be changed through a dialog box accessed through the "Customize View" button.

Editing and viewing of objects in the **Build** pane takes place using graphical interface tools called *inspectors*. An inspector is simply a user interface module designed to let a user interact with information in a certain way. The default inspector in the Modeler's Workspace is called the *Generic Inspector*; it displays an object using a tree-structured table of attribute-value pairs, and is used as the default editor/viewer for those types of objects that do not have their own specialized inspector. Figure 2 shows a mock-up of the Generic Inspector. The contents of the window are constructed dynamically by interpreting the definition of a given template to produce the form in the window.

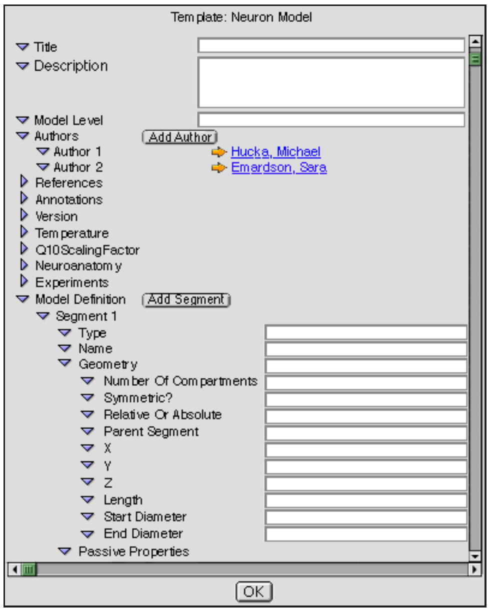

Figure 2: *A mock-up of the Generic Inspector interface. Clicking on the arrow to the left of an attribute toggles between an expanded and contracted view. Some attributes contain a simple value such as a string (e.g., "Title"), whereas other attributes contain more structure (e.g., "Authors", which contains a list of links to other database objects). Nesting of attributes is unlimited; for example, "Model Definition" contains a number of attributes, each of which itself is a container for more attributes.*

Inspectors are incorporated into the Modeler's Workspace through a plug-in architecture, so that new ones can be added dynamically to support new templates. Specialized inspectors may provide other modes of interaction besides the form-based approach of the Generic Inspector; for example, a single-cell inspector could provide a graphical, three-dimensional tree viewer/editor for working with cell morphologies. A few different inspectors will be provided with the Modeler's Workspace; as others become available, they will be made available for downloading through the Modeler's Workspace Directory.

The **Search** pane provides the ability to search multiple databases for objects having specific characteristics. As shown in the mock-up in Figure 3 on the following page, the pane contains three main regions. The first region provides a pull-down list of known templates and a form. Once the user selects an object type from the list of templates, the form is filled with attribute name-value slots corresponding to the chosen template. This allows the user to specify the attribute values on which to search. The form is similar to that presented by the Generic Inspector mentioned above. The second region in the **Search** pane displays a list of databases. This allows the user to select which databases should be used for a search. The third region in the **Search** pane contains a table listing the results of the search. The columns in this table summarize the different attributes of the objects matched by the search. To obtain more detailed information about a given object, the user can double-click on a line in the table to view the object in an inspector window. More than one object can be inspected simultaneously, allowing, for example, multiple models to be examined side-by-side in separate windows. The user can also import objects from the search results into their personal Workspace Database.

In addition to searching multiple databases using the **Search** pane, a user may want to connect to a single Workspace Database and browse its contents. The **Connect** pane provides this one-database-at-a-time connection capability. It resembles the **Build** pane in organization, with a line at the top naming the currently connected database, and a central region containing a table listing the contents of the database. As in the

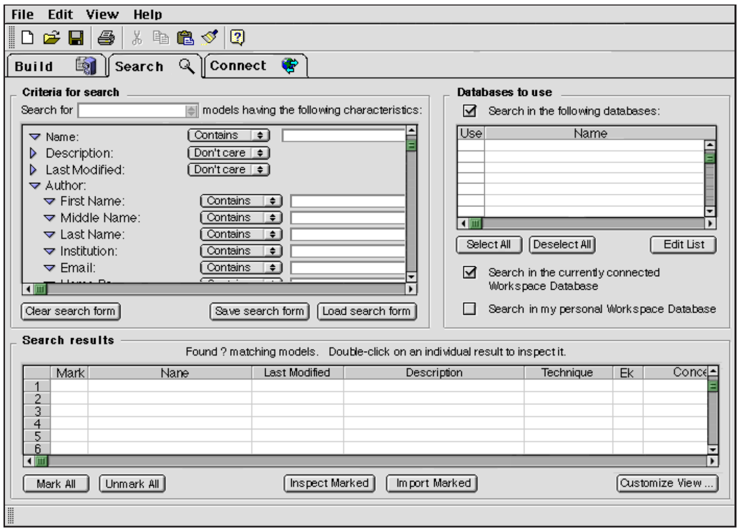

Figure 3: *A mock-up of the* **Search** *pane. The upper left area allows the user to specify the object characteristics to search for; the upper right area allows the user to specify which databases should be used in a given search; and the bottom half provides a summary of the search results.*

Build pane, the user can double-click on an entry to examine it in detail in an inspector window. If the user has write access to the Workspace Database, she or he can also edit objects in this way. The **Connect** pane also provides the basis for engaging in collaborative activities with other Workspace users. Connecting to another user's Workspace Database (assuming that the owner has set appropriate permissions) allows one to view and possibly edit the objects in that database. The Workspace Database server is designed to allow multiple such simultaneous users to be connected. The User Interface implements a chat facility and a shared desktop viewing capability that allows all users connected to a particular Workspace Database to communicate with each other and see what each other sees on her or his computer screen. This is intended to make it easy for users to interact and work together on developing models.

2.3 *The Site Browser*

The need to select from a collection of databases or a collection of users occurs in several contexts in the Modeler's Workspace. Since databases and users are geographically scattered, a map-based interface—the *Site Browser*—is presented to the user in these situations in order to provide cues that may help the user locate and remember the items involved. The Site Browser contains two tabbed panes, **Databases** and **MW Users**. The first pane displays a map of the world and a table of databases known to be publically accessible. The map displays the geographic location of each database. In the **MW Users** pane, the same map-and-table interface is used, but here, the table contains a list of the users known to be actively using the Modeler's Workspace. The map in this case displays the geographic location of each user. The information in all cases is obtained by contacting the Modeler's Workspace Directory. Figure 4 shows a mock-up of the Site Browser.

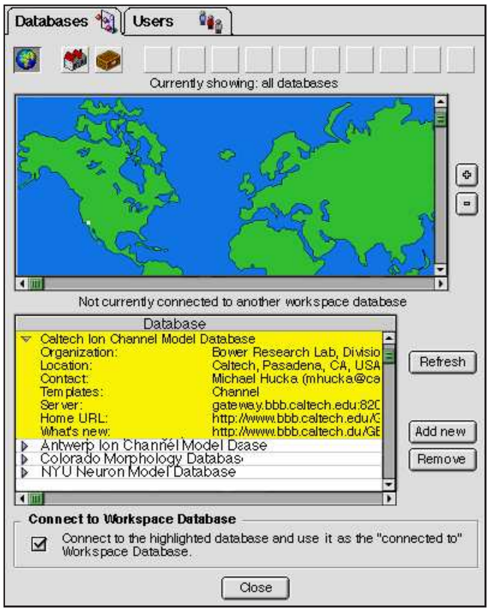

Figure 4: *Mock-up of the Site Browser. This particular example screen displays the Databases pane, when the user is being asked to select a database for the* **Connect** *pane.*

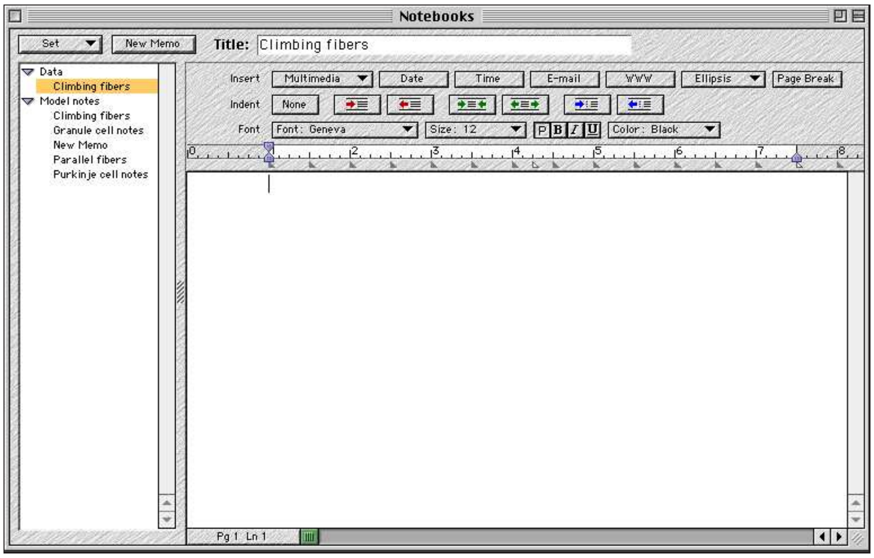

Figure 5: *A mock-up of the Notebook Editor. Each notebook is a single document of unlimited length. Notebooks can be given titles and grouped hierarchically if desired. The Notebook Editor provides typical WYSIWYG-style editing and formatting facilities, and the ability to paste text and graphical clippings into notebooks.*

2.4 Notebooks

The Modeler's Workspace User Interface provides an electronic notebook editing and management facility called the *Notebook Editor*. This is a WYSIWYG document editor resembling a straightforward word processor, with the addition of a simple tree-structured management facility for organizing notebooks into folders and subfolders. Each notebook is essentially a free-form document and can contain text, image clippings, equations and other elements. A mock-up of the Notebook Editor interface is shown in Figure 5.

The Notebook Editor can be used to write simulation scripts for GENESIS, NEURON, or other simulators. Portions of notebooks can be highlighted and sent to a running simulator process; the simulator's output can be inserted back into the notebook. Notebooks (or portions thereof) can be also exported to files in a variety of formats, including plain text and HTML. The contents of notebooks, in part or in whole, can be copied and pasted into the annotations attribute of a model; likewise, the contents of the annotations attribute of a model can be pasted into a notebook.

2.5 Access to Neural Simulation Packages

The final component of the Modeler's Workspace User Interface is the facility for communicating with simulation software. Interaction with simulators is supported through a plug-in architecture, so that any simulation package (GENESIS, NEURON, or even a general-purpose mathematical package such as Yorick or MATLAB) can potentially be interfaced to the Modeler's Workspace once someone writes an appropriate plug-in module and it is made generally available. Interaction with the package takes place in the **Build** pane, which provides a window connected to the command-line interface of the simulation program (whether the simulator is GENESIS, NEURON, or other). A user can type commands directly to the simulator, and the output appears in the **Build** pane's interface window or in separate windows created by the simulator itself. The User Interface can also send model representations to the running simulation program; the plug-in interface for the simulator translates objects from the representation used in the Modeler's Workspace to a format understood by the simulator.

3 The Modeler's Workspace Directory

The *Modeler's Workspace Directory* (MWD) is a server located at `www.modelersworkspace.org` that acts as a global registry and repository supporting the community of users. It performs several key functions: (1) it maintains a real-time-updated list of who is running the Modeler's Workspace around the world, to help users in setting up collaboration meetings; (2) the MWD serves as the place where stand-alone database servers are listed, so that the Workspace can find databases for search operations; and (3) the MWD records the latest definitions of templates used by the different databases, so that the Workspace can determine which types of objects are defined in each database.

If a particular database is a foreign database (meaning, not a Workspace Database), the MWD entry for that database also includes the appropriate plug-in for allowing the User Interface to communicate with the foreign database. Such database interoperability plug-ins in the Modeler's Workspace are called *mediators*. A mediator is a program module that extends the functionality of the Modeler's Workspace to implement two functions: (1) interface to the network communications protocol required by a particular foreign database (e.g., CORBA/IIOP, HTTP, Z39.50, etc.); and (2) translate back and forth between the Modeler's Workspace representation and search language and the corresponding elements of a foreign database.

The Modeler's Workspace User Interface by default contacts the MWD whenever the User Interface is started or stopped. This allows the MWD to take note of users that are running the Modeler's Workspace.

4 Creating Publically-Accessible Database Servers

By introducing an easy-to-use tool like the Modeler's Workspace, we hope to encourage users to share their models with each other. To enable users to set up publically-accessible databases, the Modeler's Workspace includes a facility for creating stand-alone databases. The facility automates the steps of copying the user's personal Workspace Database, setting it up as a server process, and setting public access permissions. As part of this process, the Workspace contacts the Modeler's Workspace Directory, registers the new stand-alone database, and uploads the definitions of any new templates defined in the new database. This not only promotes the evolution and sharing of templates, it also enables the Modeler's Workspaces of other users to learn about the templates and to conduct searches using them.

5 Conclusions

The design of Modeler's Workspace system attempts to satisfy many goals, yet at the same time be as simple as possible from the user's point of view. The details of the system are described in several companion documents: *The Modeler's Workspace Database Organization and Interaction*, *The Modeler's Workspace Model Representation Schemas*, and *The Modeler's Workspace User Interface*.

6 Acknowledgements

Many goals of the design presented here were set forth by James M. Bower. The design is based on earlier versions of the Modeler's Workspace developed and implemented by Jenny Forss and David Beeman. The design presented here has also benefited from discussions with the members of the Bower Lab at Caltech and David Beeman at the University of Colorado. We are grateful for everyone's help.

This research has been funded by a grant from the NIMH Human Brain Project, and an Academic Equipment Grant (#EDUD-7824-000127-US) from Sun Microsystems, Inc.