The Modeler's Workspace: Making model-based studies of the nervous system more accessible

Michael Hucka
mhucka@caltech.edu
Division of Biology 216-76
California Institute of Technology
Pasadena, CA 91125

Kavita Shankar
kshankar@bbb.caltech.edu
Division of Biology 216-76
California Institute of Technology
Pasadena, CA 91125

David Beeman
dbeeman@dogstar.colorado.edu
Dept. of Electrical and Computer Engineering
University of Colorado
Boulder, CO 80309

James M. Bower
jbower@bbb.caltech.edu
Division of Biology 216-76
California Institute of Technology
Pasadena, CA 91125

Running head: **The Modeler's Workspace**

Contact: James M. Bower
Phone: (626) 395-6820
Fax: (626) 795-2088

ABSTRACT

A realistic neuronal model represents a modeler's understanding of the structure and function of a part of the nervous system. The increasing number of such models represents a significant accumulation of knowledge about the structural and functional organization of nervous systems. However, locating appropriate models and interpreting them becomes increasingly more difficult as the number of online model and experimental databases grows. The central motivation for the Modeler's Workspace project is to address these problems.

The Modeler's Workspace is a collection of software tools being created to enable users to interact with databases of models and data. It will provide facilities for: searching multiple remote databases for model components based on various criteria; visualizing the characteristics of the components retrieved; creating new components, either from scratch or derived from existing models; combining components into new models; linking models to experimental data as well as online publications; and interacting with simulation packages such as GENESIS to simulate the new constructs. It is being written in Java for portability and extensibility. It is modular in design and uses pluggable components. To increase the probability that the Modeler's Workspace will be compatible with future databases and tools, we are using the XML, the eXtensible Markup Language, as the interchange format for communicating with databases.

1 INTRODUCTION

A structurally 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 more difficult as the number of online model and experimental databases grows. This is exacerbated by the fact that computational models developed by different researchers are often implemented using different software tools.

The central motivation for the Modeler's Workspace project is to address these problems [1]. Our goal is to develop a free, open software environment that provides a variety of capabilities, including: searching multiple remote databases for model components based on various criteria; creating new components; combining model components together and translating them into formats suitable for simulation systems such as GENESIS [2], NEURON [3], XPP [4], or NEOSIM [5]; managing personal databases of models and other information; and collaborating interactively with other researchers to work with models and simulations. The Modeler's Workspace is being written in Java [6] for portability and extensibility. It is modular in design and uses pluggable components. To increase the probability that the Modeler's Workspace will be compatible with future databases and tools, we are using the XML, the eXtensible Markup Language [7] as the interchange format for communicating with databases.

In this chapter, we describe a design for the Modeler's Workspace user interface, overerall architecture, model representation scheme, and the motivations behind the various design decisions. We believe that the Modeler's Workspace can provide integrated access to a wide variety of model databases and simulation tools. The system's modular and extensible design will make it possible for others to write new components, for example for handling new forms

of model representations or interfacing to new databases. In fact, it is our hope that by providing a sound, open framework and a collection of elements demonstrating its capabilities, others will be encouraged to contribute to its implementation and add new functionality to the environment, rather than creating entirely new software tools or simulator-dependent model representations.

2 THE NEED FOR MODEL-BASED APPROACHES

The field of biology has made great advances over the last several decades in our ability to describe experimentally the organization of biological systems. However, with these technical advances has come an enormous increase in the amount of available data, in many cases already exceeding the ability of single investigators to keep up with what is known. For this reason neuroscience, and biology as a whole, is running the risk of becoming increasingly fragmented and disorganized.

The stress of information overload and the lack of appropriate conduits for information have lead to several national and multinational efforts to develop electronic databases [8]. Most of these projects are based on the idea that information in electronic form is more readily accessible and manipulated than traditional forms of publication. However, it is our view that in order for these efforts to be successful they must provide enhanced functionality over traditional means of distributing information. To take one example, efforts at providing electronic forms of data delivery need to address the difficult issue of data validation. Data validation involves not only assessing the likely accuracy of the information stored, but also providing some measure of the relevance of the data to the current state of our understanding. In traditional forms of scientific data reporting, the peer review process provides the means of validating accuracy, while the discussion sections of published papers provide the opportunity to speculate on the functional significance of the results. It is not clear that electronic databases in their existing forms will provide these functions. Creators of electronic databases

seem to assume that if the databases are available, interested parties will figure out how to use them appropriately.

2.1 The Role of Structurally Realistic Models

We believe that computational models provide a much more sophisticated, flexible and powerful base for electronic storage and retrieval of information than do traditional on-line databases. Our work has therefore focused on developing tools for working with *structurally realistic* biological models. These are models whose first objective is to capture what is known about the anatomical structure and physiological characteristics of a neural system of interest.

In the case of a model of part of the brain, for example, this modeling approach typically starts with as detailed a description as possible of the relevant neuroanatomy. At the single cell level, this usually means a description of the three-dimensional structure of the neuron and its dendritic tree [9]. Most contemporary modeling approaches for realistic models of neural cells are based on compartmental modeling [10]. In each case, the overall compartmental structure of the model is first established on the basis of neuroanatomical details. Information about neuronal morphology used in computational models typically includes such parameters as soma size, length of interbranch segments, diameter of branches, bifurcation probabilities, and density and size of dendritic spines. At the neuronal network level, parameters include some description of the cell types found in the network and their connectivity [11]. The next stage of structurally realistic model building involves establishing the basic physiological behavior of the modeled structure, for instance by tuning the model to replicate neuronal responses to experimentally-derived data [12, 9]. The final stage of model construction involves exploring the model's behavior to novel inputs, using it to generate new ideas about the neural system's function as well as a guide for new experimental investigations.

As a result of their faithfulness to biological anatomy and physiology, structurally realistic

models can be a means of *storing* anatomical and physiological experimental information. As model sophistication grows, structural models themselves become a form of information storage about the biological systems being studied.

2.2 Data Evaluation and Functional Assessment

Because of their nature, structurally realistic models also contain precise information about the relationships between known facts. For example, neurons display a wide range of dendritic morphologies, ranging from a single simple dendrite in retinal bipolar cells to compact but highly branched spiny and smooth arborizations of the cerebellar Purkinje cell. Detailed compartmental models of reconstructed neurons have become important tools for investigating how dendritic morphology and membrane biophysics interact [13] in integrating synaptic inputs [14] and propagation of action potentials [15, 16], with further implications for development and plasticity [16].

Another issue related to the quality of stored electronic data involves the question of which data are most relevant and therefore useful to our current knowledge of a particular system. Just because data can be collected does not necessarily mean that this data will help expand our current understanding of a particular system. Modeling can reveal the importance of a particular data point as well as provide an immediate context for the data once obtained. To take one example, in one effort at modeling Purkinje cells [9, 17], the morphology of the models was based on a detailed light microscopic reconstruction of horseradish peroxidase-filled guinea pig Purkinje cells [17, 18], and all simulations were usually performed with a model based on the morphology of cell 1. However, when identical channel equations and densities were placed onto two additional Purkinje cells with morphologies labelled as cell 2 and cell 3 [19], the details of the firing patterns were quite different for these three morphologies. The firing pattern of cell 3 with current injection was more similar to cell 1, but with a shift in the frequency-current curve. This turned out to be because of the small soma and short thin main dendrite, which caused smaller total potassium delayed rectifier

and muscarinic conductances in the model of this cell [9, 17].

Finally, perhaps the most valuable contribution of a structural model is the way in which it can help to organize our understanding of the system being studied. In the large and growing electronic databases of the most common type in use today, it is not at all clear how the data will contribute to our understanding of function. Models, however, are specifically constructed to explore the functional implications of the data on which they are built. Models thus can serve not only as the point of entry for data; they can also serve as dynamic tools that can be used to understand its significance. As models become more sophisticated, so does the representation of the data. As models become more capable, they extend our ability to explore the functional significance of the structure and organization of biological systems. Thus, there is a direct link between the ultimate objective of acquiring the data and the data acquisition process itself.

3 OVERVIEW OF THE MODELER'S WORKSPACE

The Modeler's Workspace project is an effort to create a software environment that will make it easier for biologists to develop, use, and share structurally realistic models and thereby gain some of the benefits discussed above. As mentioned in the introduction, our goal is to provide the following kinds of facilities in the system:

- Search and retrieval facilities for interacting with multiple databases of models and other information;
- Facilities for creating, editing and visualizing models and their components;
- Facilities for combining model components together and translating them into formats suitable for simulation systems such as GENESIS and NEURON;
- Facilities for linking models to experimental data as well as online publications;

- 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 chat facilities.

There are three main elements in the system: a user interface, a database server, and a global registry and repository called the Modeler's Workspace Directory. In this section, we summarize the overall organization and intended operation of the Modeler's Workspace from a user's perspective. In Section 4, we turn to the software architecture of the system, describing in more detail its extensible, modular structure.

3.1 The Modeler's Workspace User Interface and Workspace Database

The Modeler's Workspace *User Interface* is being implemented as 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 (where models, notes and other objects are stored) and as a rendezvous point for collaborative activities. The User Interface communicates with the Workspace Database using network protocols.

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. Model attributes potentially can be of any type, including other models. 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". We discuss templates and representation issues in more depth in Section 5.

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 will be available for download from the Modeler's Workspace web site and will run on any Java-enabled computer 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.

3.2 Elements of 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 prototype 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.

Inspectors are incorporated into the Modeler's Workspace through a pluggable component architecture (with "plugin" software modules), 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 would 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 (see Section 6).

The **Search** pane will provide the ability to search multiple databases for objects having specific characteristics. As shown in the prototype in Figure 2, 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, in order to indicate the type of object desired, 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 **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.

3.3 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*—will be 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 publicly 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 3 shows a prototype of the Site Browser.

3.4 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 plugin architecture, so that any simulation package can potentially be interfaced to the Modeler's Workspace once someone writes an appropriate plugin module and it is made generally available. Although this interface will be most useful for simulators such as GENESIS or NEURON that construct structurally realistic compartmental models and can make use of the model representations that we provide, the programming interface is available for use by any simulator, including more general purpose ones such as Yorick or MATLAB. 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, or another). 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 plugin interface for the simulator translates objects from the representation used in the Modeler's Workspace to a format understood by the simulator.

3.5 An Example Usage Scenario

As an example of how one might use the Modeler's Workspace, consider the case of a neuroanatomist who is interested in the effects of dendritic morphology on the behavior of Purkinje cells. For example, this researcher may be wonder whether a model with a sim-

plified morphology might have sufficiently realistic behavior for use in a network model of part of the cerebellum. (We note that in the case of the Purkinje cell, the answer is likely to be "no" [20].) The user might begin by using the **Search** pane in the Modeler's Workspace to search for various Purkinje cell models in databases on the Internet. Figure 2 shows a possible search result that includes a hypothetical model based on that created by De Schutter and Bower [9], but using a variant morphology that was generated by the L-Neuron program [21].

The next step might be to use an inspector to examine the particular cell model in more detail. The user could do this by double-clicking the line containing the model of interest in the search results. The CDROM for this book contains additional documents and figures that could not be included in this chapter due to space limitations. There, the figure for the "Purkinje Cell Inspector" illustrates a mock-up of an inspector view with information about the geometrical and passive properties of the Purkinje cell model soma. In order to examine the specific channels used in the model in more detail, the user would employ a Channel Inspector, as illustrated in the figure "Purkinje Channel Inspector", which shows details of the channel dynamics used for the fast sodium channels in the model. Note that, as with the case of most of the inspectors, it is possible to see the actual equations that are used when the model is simulated. As well as providing further details of the model, the availability of these equations may give reassurance to modelers who distrust software that they did not write themselves.

As discussed in Section 4, the various templates used in the Modeler's Workspace contain a great deal of descriptive information about the model and the experimental data on which it is based. However, the most information may be obtained if the model is actually used in a simulation, so that the behavior of different models may be compared under the same simulation conditions. In this case, users might want to import the model into the local Workspace Database so that they can run it in a simulator. Importing a model can be done from the table of search results in the **Search** pane by highlighting the line containing the

particular model desired and then clicking on the "import" button. Once the model is copied into a user's Workspace Database, the user may switch to the **Build** pane to send the model to a simulator and compare current clamp simulations under the conditions described in [9]. The **Build** pane and the inspectors could also be used to create additional models, either by starting from and modifying an existing model, or by importing passive morphologies from elsewhere and populating them with channels taken from a Purkinje cell model database.

4 THE UNDERLYING ARCHITECTURE

The previous section makes clear that the Modeler's Workspace User Interface is the most essential part of the system. It must provide interfaces not only for interacting with model components, but also with databases and simulation tools. Because model representations, databases and simulation/analysis tools will all change and evolve over time, the User Interface must itself be easily adapted and extended as the needs of users change.

We knew from the outset that the success of the project would be contingent on providing an open framework that could be taken and extended by other developers. In designing the system, we identified the following characteristics as being crucial to meeting our needs:

- *Simplicity*: The framework must be simple enough that interested developers can use it in their projects with a minimum amount of effort. The framework should *not mandate* the use of complex technologies such as CORBA [22], although it should *not prevent* developers from using any particular technology in implementing an extension if they so desire.

- *Extensibility via plugin components*: As new tools and methods are developed, it must be possible to implement them as pluggable modules that can be added to the existing framework without having to modify the framework itself. A plugin may either reimplement existing capabilities in new ways (for example, if someone develops an improved version of an existing module), or implement an entirely new capability.

- *Free distribution.* All interested users must be able to obtain the system for free. Any software that is incorporated into the system and distributed with it, such as GUI widgets or object libraries, must itself be free of licensing fees or restrictions on redistribution.

- *Portability.* Except for the Modeler's Workspace Directory server, the framework must be portable to at least Microsoft Windows (Win32) and Linux, with support for other varieties of Unix and MacOS X preferable as well. (The directory server exists only as a global server or multiple replicated servers, and runs separately from any user environments; therefore, it does not have the same restrictions on portability.)

We believe we have met these objectives by creating a layered, highly modular architecture. In the rest of this section, we discuss the high-level design of this architecture.

4.1 Layered Framework

We sought to maximize the reusability of the software that we developed for the Modeler's Workspace by dividing the Workbench infrastructure into two layers: the Modeler's Workspace itself, and a lower-level substrate called the *Biological Modeling Framework* (BMF). The latter is a general software framework that provides basic scaffolding supporting a modular, extensible application architecture, as well as a set of useful software components (such as GUI tools) that can be used as black boxes in constructing a system. In fact, BMF is already being used to implement another tool, the Systems Biology Workbench [23]. Further information about the Systems Biology Workbench may be obtained from `http://www.cds.caltech.edu/erato/`. Figure 4 depicts the core of BMF as a gray substrate holding a number of shaded blocks representing the basic plugins provided with BMF.

The Modeler's Workspace is a particular collection of application-specific components layered on top of BMF. These collectively implement what users experience as the "Modeler's

Workspace." Some of these components add functionality that is needed for supporting the overall operation of the Workspace, such as the main screen of the User Interface described in Section 3; other components implement the interfaces to databases and simulators. Figure 5 illustrates the overall organization.

As mentioned previously, the Workspace Database is a separate server program that can act as both a private repository for a user's work (where models, notebooks and other items are stored) and as a rendezvous point for collaborative activities. The Modeler's Workspace system uses an XML-based model description language (discussed in Section 5). The Workspace Database will accept data directly in XML format [7], making it generic and capable of storing any information that can be encoded in XML.

4.2 Highly Modular, Extensible Architecture

The Biological Modeling Framework layer that underlies the Modeler's Workspace is implemented in Java and provides the scaffolding that (1) supports pluggable components and (2) provides a collection of basic components (such as an XML Schema-aware parser, file utilities, basic plotting and graphing utilities, etc.) that are useful when implementing the rest of the Modeler's Workspace system.

Compared to many other frameworks, the Biological Modeling Framework is conceptually quite simple. The BMF Core consists of only one primary component, the *Plugin Manager*, and its operation is straightforward. Few requirements are placed on plugins themselves. For example, a plugin merely needs to be packaged in a Java JAR file, implement one specific interface (though it may implement others in addition), and obey a few rules governing behavior. There can be any number of plugins in the system, subject to the usual limitations on computer resources such as memory.

By virtue of the software environment provided by the Java 2 Platform, plugins can be loaded dynamically, without recompiling or even necessarily restarting a running application. This can be used to great advantage for a flexible and powerful environment. For example,

the Modeler's Workspace can be smart about how it handles data types, loading specialized plugins to allow a user to interact with particular data objects on an as-needed basis. If the user does not already have a copy of a certain plugin stored on the local disk, the plugin could be obtained over the Internet, much like current-generation Web browsers can load plugins on demand. In this manner, plugins for tasks such as displaying specific data types or accessing third-party remote databases could be easily distributed to users.

5 REPRESENTATION OF MODELS AND DATA

One of the most difficult conceptual issues has been developing a strategy for describing models and their components. The Modeler's Workspace requires a representation language that abstracts away specifics of particular simulators such as GENESIS, and also provides ways of interacting with existing neuroscience databases on the Internet.

Devising such a representation is difficult. The thorniest issue has been balancing the need for specificity in the representation (so that we can develop useful software tools for manipulating models) against the need for extensibility (so that as people's conceptualizations of neuronal characteristics change, the software does not need to be rewritten). It is not sensible to try to dictate every detail of how models and data are to be stored in databases. At the same time, some definitions of permissible data structures and formats must exist, so that we can proceed to develop software.

To begin addressing this problem, we first distinguish between a *Modeler's Workspace Database*, which is the database component of the Modeler's Workspace system, and a *foreign database*, meaning any other kind of database. In order to support some level of interoperability with foreign databases as well as neural simulators, and allow users and software developers to evolve new representations and tools, we use a multifaceted approach having the following key aspects:

1. Model templates are organized following a simple object-oriented metaphor, with a base

template serving as the root of all representations and new templates being derived from either the base or another existing template. We give users the ability to define new templates, but only through the addition of attributes; deletions are not allowed. This ensures that all models have at least a minimum set of common attributes that the Modeler's Workspace software can count on.

2. The Modeler's Workspace comes with a collection of default templates that can be used to describe the most common types of neuronal structures. These templates are part of a recently-established effort to produce *NeuroML*, a common exchange language for computational models in neuroscience [24]. We hope that these templates will act as de facto standards that will coalesce users around them and prevent the proliferation of many similar-yet-different representations for the most commonly-used types of models.

3. Borrowing ideas from Burns (personal communication) and Gardner [25], we require that the model templates used in a database be explicitly described and communicated by the database server when a client first contacts it. We also require that the templates used in publicly-accessible databases be made available separately through the Modeler's Workspace Directory. This permits the User Interface component to determine the structure of models in any given database by contacting either the MWD or the database in question.

4. Interfaces to databases as well as simulation packages are mediated through software plugins. A database plugin for a particular foreign database provides the network interface and translation needed to interact with the database; a simulator plugin for a particular simulator handles translating commands and representations between the simulator and the Modeler's Workspace system.

5.1 Template Hierarchy

As mentioned above, models in NeuroML and the Modeler's Workspace are represented using a limited form of object-oriented description, in which each object in a database is defined according to either a root template called Base, or a template derived from it. Figure 6 shows the current hierarchy of templates used in the system.

The Base template contains a core set of attributes that are common to all main objects in the database. The first level of templates derived from Base consists of templates named Author, Reference, Method, Model, Data, and Site. The particular choice of first-level templates was taken from the work of Gardner et al. [25]. They are generic and do not contain any attributes that are specific to particular kinds of biological structures. The additional templates derived from the first-level ones then add specific templates for representing models of neurons and related structures. The following list briefly summarizes the templates defined at the time of this writing; the full definitions are available from the NeuroML web site (`http://www.neuroml.org`).

Base: This is the root of the template hierarchy. This template has only two attributes, `id` and `version`. The former places a unique identifier on every object; the latter allows the system to track the evolution of data objects.

Author: This template inherits attributes from Base and adds attributes for identifying a person by name, address, web home page and other characteristics. Using separate objects for author information allows users to enter into their databases the information about a given author once, then link to the author information from other objects (such as models and article references).

Reference: This is used to represent literature references. The attributes provided by this template are based on BibTeX records [26]. Author and editor information is represented in terms of links to Author objects.

Data: This is used for storing data in a Workspace Database or pointing to data stored in a remote database. It can record information describing the data, links to authors and references, and data (or pointers to data) grouped into data sets.

Method: This template is intended to capture information about experimental methodologies.

Site: This template is intended to capture information about such things as neuronal recording sites, brain regions, etc.

Model: This is intended to serve as a common starting point for all model template definitions. It is a generic structure, not specific to any particular kind of modeling. Specific kinds of models, such as for neuronal cells and intracellular and transmembrane mechanisms, are derived by starting from Model and adding new attributes.

Neuronal Anatomy: This template specializes the basic Data template to provide a container for anatomical information about neurons. It is primarily intended to be used for storing information about cell morphologies.

Imaging/Histoloy: This is a specialization of the Method template, used to describe methods of imaging and histology used in neuronal anatomy work.

Neuron: This is the main template for representing models of neurons. It extends the basic Model template with additional attributes for anatomical information, experimental information, the segmented cable structure of the model neuron, and other characteristics. It includes pointers to objects based on several other templates, in particular Neuronal Anatomy, Neuron Part, Transmembrane Mechanism and Intracellular Mechanism.

Neuron Part: This exists to provide a way to construct reusable part models for Neuron objects. Portions of neuron models can be recorded in Neuron Part objects, allowing those portions to be reused in models by linking to them from within Neuron objects.

Transmembrane Mechanism: This template is intended to serve as a starting point for definitions of models of cell mechanisms such as ion channels, calcium concentration pools, etc.

Intracellular Mechanism: This is intended to serve as a common starting point for defining items such as calcium concentration pools.

Voltage-Gated Channel: As its name implies, this template can be used to represent models of voltage-gated ion channels. The representation is based in large part on that used by GENESIS, and can handle channels not only of the common Hodgkin-Huxley variety, but also a number of variants.

Ligand-Activated Channel: This template can be used to represent models of channels that are activated by neurotransmitters.

Ionic Pump: This is used to represent various mechanisms for removing ions from a cell.

Concentration Pool: This template is used for models of intracellular ionic concentrations.

Nernst: This is one of several possible mechanisms for calculating changes in ionic equilibrium potentials.

The Modeler's Workspace uses XML Schemas to define templates. *XML Schema* is a recently introduced standard [27, 28, 29] for specifying the tags allowed in an XML data stream, how the tags can be organized into a hierarchy, and the data types of attributes delineated by the tags. The definition of the Base is expressed as a single XML Schema file. Each derived template (e.g., Model, Neuron, etc.) is similarly expressed using a separate XML Schema definition, making the organization very modular and easily extensible.

5.2 Advantages of the Approach

The object-oriented style of representation is useful for a variety of reasons. First, the existence of categorical templates allows the Modeler's Workspace User Interface to present the user with intelligent search forms. Specifically, the Modeler's Workspace search interface prompts the user to specify the type of object to search for (which is equivalent to specifying the template), and based on the user's choice, the system constructs a form using knowledge of the attributes defined by the template. The search form may include graphical elements specialized for the particular category of object involved. This allows the system to go beyond the usual fill-in-the-blanks search form and provide something more powerful and user-friendly.

A second reason is that, by choosing the search category appropriately, database searches can be made more or less specific. Because of the hierarchical relationships, a user can select a template in the middle levels of the hierarchy, and search operations can be designed to encompass all objects that are below it in the hierarchy. This means, for example, that a search using Model will encompass objects created from templates derived from it, such as Neuron class objects, TransmembraneMechanism class objects, etc.

A final reason for the utility of the representational framework presented here is that software can be made modular and extensible. New software modules can be developed for the Modeler's Workspace alongside new templates, customizing the system to interact with new types of objects without redesigning or restructuring the whole system. For each representation derived from an existing template, all the software elements that worked with the parent template will also work with the derived templates. This is because the derived template can only add attributes, and while the existing tools will ignore the new attributes, they will continue to work with the attributes that were inherited from the parent template. Developers can write new software modules that interact with the additional fields in the new templates and these software modules can be loaded into the Modeler's Workspace on demand, extending the software's functionality.

6 INTERACTING WITH DATABASES

The Modeler's Workspace design supports the ability for users to interact not only with theirs and other users' Workspace Databases, but with databases that were not designed specifically for the Workspace. In this section, we describe how the User Interface component of the system interacts with Workspace Databases and foreign databases.

6.1 Workspace Databases

Separating the Workspace Database from the User Interface, and making the former be a stand-alone server, is essential for providing the desired functionality in the Modeler's Workspace system. Not only does this approach support different access scenarios (described next), but it also has the advantage that when the user "starts running the Modeler's Workspace", they usually only need to start the User Interface—the Workspace Database typically will already be running, usually having been started up at computer boot time and waiting for connections over a network.

6.1.1 Individual Access to Workspace Databases

One of the access scenarios involves a one-to-one mapping between User Interface processes and Workspace Database processes. A certain user of a Workspace Database (typically, the user who establishes and configures it) is designated as its *owner*. In many cases, the owner may be an individual who simply wishes to use the Modeler's Workspace in private, and may remain the sole user of the database. Whenever the user starts a copy of the Workspace User Interface, she or he only needs to supply the address of their Workspace Database server, and the User Interface connects over the network to this database process. The network-based nature of the database allows a user to roam anywhere on the Internet and still access their home Workspace Database. A user can connect, disconnect, and reconnect to their database from different computers.

6.1.2 Multiuser Access to Workspace Databases

Another access scenario involves multiple users connecting to a single Workspace Database server and simultaneously interacting with its contents. The owner of a Workspace Database has the ability to add other users to a list of designated permitted users—collaborators (possibly running at remote sites) who are allowed to connect to the Workspace Database and view and (possibly) edit its contents. This is the basis of the *shared workspace* facility of the Modeler's Workspace. The motivation for this facility is to permit various collaborative activities in the context of developing models. By allowing users to "meet" online, with the ability to see each other's Modeler's Workspace environments, we hope to encourage users to interact with each other and with their ongoing modeling work.

6.2 Foreign Databases

Foreign databases are those that were created by other groups prior to or without concern for the Modeler's Workspace. Because of the differences in interfaces and model representations, it is generally impossible to provide full interoperability between the Workspace and foreign databases. Models in such databases may be organized along different lines than those in Workspace Databases, and consequently, it may be impossible to provide more than the ability to search and retrieve models on the most basic characteristics (perhaps limited only to title, author, and similar fields). The degree to which a foreign database's representation can be mapped onto a Workspace representation must be determined on a case-by-case basis, and the interface must be implemented in the form of a plugin for each specific database.

Interaction with foreign databases occurs when the user performs search operations in the **Search** pane of the Modeler's Workspace User Interface. When the user chooses to search in selected foreign databases, each database plugin must do the work of translating the search request into the appropriate forms for the individual foreigns databases. Specifically, the database plugin performs the following functions: (1) engage the network communica-

tions protocol required by a particular foreign database (e.g., CORBA/IIOP [22], HTTP, Z39.50 [30]; and (2) translate back and forth between the Modeler's Workspace representation and search language and the corresponding elements of a foreign database.

A mediator must perform its translations by using the closest appropriate Modeler's Workspace templates. For example, a foreign database that stores models of ion channels would presumably be mapped to a representation based on the Transmembrane Mechanism template or perhaps even the more specific Voltage-Gated Channel template. In some cases, mapping foreign representations may only be possible in a partial way, with many attributes in the equivalent Workspace representation left blank. The database plugin must note which fields are missing in a particular model, so that the search process can distinguish between *missing* values and *blank* values. The treatment of missing attribute values during search can be handled specially and placed under control of a user preference setting; the user can elect to have the system either perform partial-matching (where missing fields are treated as "don't-care"), always-succeed-matching (where missing fields are treated as if they matched), or always-fail-matching (where missing fields are treated as if they failed to match).

Due to the lack of standardization in existing neuroscience databases, we expect that a custom mediator will need to be hand-written for each foreign database for which we want to provide interoperability. Initially we expect that database plugins will be written by the Modeler's Workspace developers, and distributed through the Modeler's Workspace Directory. In time we hope that other developers will also write database plugins for the system.

6.3 Template-Driven Search Interface

The existence of templates and the rules for exchanging them serve an important purpose in addition to structuring model representations: the definitions are used to construct search forms and inspector interfaces.

The search form in the **Search** pane (see Figure 2) is constructed dynamically using the

following approach. First, the names of all known model templates are collected together and presented in a pull-down list at the top of the **Search** pane (in the box in the line "Search for ▭ models having the following characteristics"). The user is required to first select one of the template names, or else to select **Any**. The requirement to select a template is necessary to enable the Modeler's Workspace User Interface to present a search form that lists the attributes appropriate to that specific model category. The User Interface creates the search form by reading that class's XML Schema definition file.

6.4 The Modeler's Workspace Directory Server

The *Modeler's Workspace Directory* (MWD) is a global server located at a particular network URL. It acts as a global registry and repository supporting the community of users, making it possible for Workspace users all over the Internet to be able to learn about the databases that are available for public access. The MWD server will fulfill several roles:

- It will maintain a list of all databases that are known to be available on the net and with which the Modeler's Workspace can interact. For each database, it will list information describing the contents of the database in terms of the corresponding Modeler's Workspace templates. If it is a foreign database, the MWD will include the appropriate mediator plugin which Modeler's Workspace clients can download if necessary.

- It will maintain a list of all users of the Modeler's Workspace, for those users who elect to be listed in the Directory. The list will be updated in real-time: whenever users start up a Modeler's Workspace process, they will be greeted with a request to allow the process to contact the MWD and register itself in the list of users. The users' locations can be viewed on the world map displayed in the **MW Users** pane of the Site Browser (see Section 3).

The existence of a Modeler's Database Directory is important not just to enable the Workspace to find out about available databases. It also means that the Workspace User Interface component does not need to contact all databases every time it is started. The alternative, having the Workspace query each known database directly at start-up time, would introduce a long startup delay. Centralizing the list of databases on a directory server will shorten startup time.

The MWD will run a simple program to update its contents daily by polling every database on its list and testing whether it is accessible. The MWD will record the last time that each database was known to be available, as well as download any new or changed template definitions from that database.

7 CONCLUSION

Structurally realistic neuronal models can serve as devices to collect, evaluate and distribute information concerning the functional organization of nervous systems. As we have described in this chapter, the central goal of the Modeler's Workspace project is to provide the neuroscience community with a modular, extensible, and open software environment enabling neuroscientists to develop, use, and share structurally realistic models. The purpose of this chapter is to describe a design that we are just beginning to implement, and to encourage others to participate in this effort. We hope that by providing common infrastructure for interfacing to databases and simulation packages, other software authors will gravitate towards this framework rather than developing entirely new software tools from scratch. Ultimately, we hope that others will be encouraged to contribute to the community new representations for models and new functionality in the form of plugins.

For the latest information on the status of this project, and the latest design documents for the the Modeler's Workspace, please visit the web sites at `http://www.bbb.caltech.edu/hbp/` and `http://www.modelersworkspace.org`. When working prototypes of the Modeler's

Workspace become available for download, they will be announced on these sites. Further information about GENESIS may be obtained from `http://www.bbb.caltech.edu/GENESIS`.

REFERENCES

1. Forss, J., Beeman, D., Eichler-West, R., and Bower, J. M. (1999). The modeler's workspace: A distributed digital library for neuroscience. *Future Generation Computer Systems*, **16**, 111–121.

2. Bower, J. M. and Beeman, D. (1998). *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* Springer-Verlag, New York, second edition.

3. Hines, M. and Carnevale, N. T. (1997). The NEURON simulation environment. *Neural Computation.* **9**, 1179–1209.

4. Ermentrout, G. B. (2000). XPP-Aut:X-windows PhasePlane plus Auto. http://www.math.pitt.edu/ bard/xpp/xpp.html.

5. Goddard, N., Hood, G., Howell, F., Hines, M., and De Schutter, E. (2001). NEOSIM: Portable plug and play neuronal modelling. *Neurocomputing.* In press.

6. Arnold, K. and Gosling, J. (1997). *The Java Programming Language.* Addison-Wesley, Reading, MA.

7. Bosak, J. and Bray, T. (1999). XML and the second-generation web. *Scientific American* (May).

8. Koslow, S. H. and Huerta, M. F., editors (1997). *Neuroinformatics: An Overview of the Human Brain Project.* Lawrence Erlbaum Associates, Mahwah, NJ.

9. De Schutter, E. and Bower, J. M. (1994). An active membrane model of the cerebellar Purkinje cell I. Simulation of current clamps in slice. *J. Neurophysiol.* **71**, 375–400.

10. Segev, I., Fleshman, J. W., and Burke, R. E. (1989). Compartmental models of complex neurons. In Koch, C. and Segev, I., editors, *Methods in Neuronal Modeling*, chapter 3, pages 63–96. MIT Press, Cambridge, MA.

11. Wilson, M. and Bower, J. M. (1992). Cortical oscillations and temporal interactions in a computer simulation of piriform cortex. *J. Neurophysiol.* **67**, 981–995.

12. Bhalla, U. S. and Bower, J. M. (1993). Exploring parameter space in detailed single neuron models: Simulations of the mitral and granule cells of the olfactory bulb. *J. Neurophysiol.* **69**, 1948–1965.

13. Segev, I. (1992). Single neurone models: oversimple, complex and reduced. *Trends Neurosci.*, **15**, 414–421.

14. Johnston, D., Magee, J. C., Colbert, C. M., and Cristie, B. R. (1996). Active properties of neuronal dendrites. *Ann. Rev. Neurosci.*, **19**, 165–186.

15. Stuart, G., Spruston, N., Sakmann, B., and Hausser, M. (1997). Action potential initiation and backpropagation in neurons of the mammalian CNS. *Trends Neurosci.*, **20**, 125–131.

16. Vetter, P., Roth, A., and Hausser, M. (2001). Propagation of action potentials in dendrites depends on dendritic morphology. *J. Neurophysiol*, **85**, 926–937.

17. Rapp, M., Yarom, Y., and Segev, I. (1992). The impact of parallel fiber background activity on the cable properties of cerebellar Purkinje cells. *Neural Computation*, **4**, 518–533.

18. Rapp, M., Segev, I., and Yarom, Y. (1994). Physiology, morphology and detailed passive models of cerebellar Purkinje cells. *J. Physiol. (Lond.)*, **474**, 87–99.

19. Jaslove, S. W. (1992). The integrative properties of spiny distal dendrites. *Neurosci.*, **47**, 495–519.

20. Howell, F. W., Dyrhfjeld-Johnsen, J., Maex, R., Goddard, N., and De Schutter, E. (2000). A large scale model of the cerebellar cortex using PGENESIS. *Neurocomputing*, **32**, 1041–1036.

21. Ascoli, G. A. and Krichmar, J. L. (2000). L-Neuron: A modeling tool for the efficient generation and parsimonious description of dendritic morphology. *Neurocomputing*, **32**, 1003–1011.

22. Vinoski, S. (1997). CORBA: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, pages 46–55. Specification available at http://www.omg.org/.

23. Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J., and Kitano, H. (2001). The ERATO systems biology workbench. In *Foundations of Systems Biology* (H. Kitano, ed.). MIT Press, Cambridge, MA. In press.

24. Goddard, N., Hucka, M., Howell, F., Cornelis, H., Shankar, K., and Beeman, D. (2001).

Towards NeuroML: Model description methods for collaborative modelling in neuroscience. *Phil. Trans. Roy. Soc. B.* In press.

25. Gardner, D., Knuth, K. H., Abato, M., Erde, S. M., White, T., DeBellis, R., and Gardner, E. (2001). Common data model for neuroscience data and data model interchange. *J. Am. Med. Informatics Assoc.* **8**, 17–33.

26. Lamport, L. (1994). *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, MA.

27. Biron, P. V. and Malhotra, A. (2000). XML Schema part 2: Datatypes. W3C Working Draft at http://www.w3.org/TR/xmlschema-2.

28. Fallside, D. C. (2000). XML Schema part 0: Primer. http://www.w3.org/TR/xmlschema-0/.

29. Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. (2000). XML Schema part 1: Structures. W3C Working Draft at http://www.w3.org/TR/xmlschema-1/.

30. ANSI/NISO Z39.50-1992 (1992). American national standard information retrieval application service definition and protocol specification for open systems interconnection. NISO Press, Bethesda, MD.

Figure Captions

Figure 1 (Fig1-build.eps)

A prototype 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 prototype shows the **Build** pane.

Figure 2 (Fig2-search.eps)

A prototype 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.

Figure 3 (Fig3-map.eps)

Prototype 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 4 (Fig4-bmfcore.eps)

The BMF Core and Core Plugins together constitute the foundations upon which applications are constructed. The *BMF Core Plugins*, shown here as shaded rectangles plugged into the BMF Core, are a set of essential modules provided with BMF and available to all applications built with BMF.

Figure 5 (Fig5-bmfcore+app.eps)

Application specific plugins are added to the BMF Core and Core Plugins.

Figure 6 (Fig6-levels.eps)

All templates are derived from Base or another existing template. Open arrows indicate inheritance; for example, template Model inherits attributes from Base and adds its own new attributes. Additional tmeplates derived from the six top-level tempates are shown with darker shading.

NOTE: This page and the next should be discarded.

References

[1] J. Forss, D. Beeman, R. Eichler-West, and J. M. Bower. The modeler's workspace: A distributed digital library for neuroscience. *Future Generation Computer Systems*, 16:111–121, 1999.

[2] J. M. Bower and D. Beeman. *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System*. Springer-Verlag, New York, second edition, 1998.

[3] M. Hines and N. T. Carnevale. The NEURON simulation environment. *Neural Computation*, 9:1179–1209, 1997.

[4] G. B. Ermentrout. XPP-Aut:X-windows PhasePlane plus Auto, 2000. http://www.math.pitt.edu/ bard/xpp/xpp.html.

[5] N. Goddard, G. Hood, F. Howell, M. Hines, and E. De Schutter. NEOSIM: Portable plug and play neuronal modelling. *Neurocomputing*, 2001. In press.

[6] K. Arnold and J. Gosling. *The Java Programming Language*. Addison-Wesley, Reading, MA, 1997.

[7] J. Bosak and T. Bray. XML and the second-generation web. *Scientific American*, May 1999.

[8] S. H. Koslow and M. F. Huerta, editors. *Neuroinformatics: An Overview of the Human Brain Project.* Lawrence Erlbaum Associates, Mahwah, NJ, 1997.

[9] E. De Schutter and J. M. Bower. An active membrane model of the cerebellar Purkinje cell I. Simulation of current clamps in slice. *J. Neurophysiol.*, 71:375–400, 1994.

[10] I. Segev, J. W. Fleshman, and R. E. Burke. Compartmental models of complex neurons. In C. Koch and I. Segev, editors, *Methods in Neuronal Modeling*, chapter 3, pages 63–96. MIT Press, Cambridge, MA, 1989.

[11] M. Wilson and J. M. Bower. Cortical oscillations and temporal interactions in a computer simulation of piriform cortex. *J. Neurophysiol.*, 67:981–995, 1992.

[12] U. S. Bhalla and J. M. Bower. Exploring parameter space in detailed single neuron models: Simulations of the mitral and granule cells of the olfactory bulb. *J. Neurophysiol.*, 69:1948–1965, 1993.

[13] I. Segev. Single neurone models: oversimple, complex and reduced. *Trends Neurosci.*, 15:414–421, 1992.

[14] D. Johnston, J. C. Magee, C. M. Colbert, and B. R. Cristie. Active properties of neuronal dendrites. *Ann. Rev. Neurosci.*, 19:165–186, 1996.

[15] G. Stuart, N. Spruston, B. Sakmann, and M. Hausser. Action potential initiation and backpropagation in neurons of the mammalian CNS. *Trends Neurosci.*, 20:125–131, 1997.

[16] P. Vetter, A. Roth, and M. Hausser. Propagation of action potentials in dendrites depends on dendritic morphology. *J. Neurophysiol*, 85:926–937, 2001.

[17] M. Rapp, Y. Yarom, and I. Segev. The impact of parallel fiber background activity on the cable properties of cerebellar Purkinje cells. *Neural Computation*, 4:518–533, 1992.

[18] M. Rapp, I. Segev, and Y. Yarom. Physiology, morphology and detailed passive models of cerebellar Purkinje cells. *J. Physiol. (Lond.)*, 474:87–99, 1994.

[19] S. W. Jaslove. The integrative properties of spiny distal dendrites. *Neurosci.*, 47:495–519, 1992.

[20] F. W. Howell, J. Dyrhfjeld-Johnsen, R. Maex, N. Goddard, and E. De Schutter. A large scale model of the cerebellar cortex using PGENESIS. *Neurocomputing*, 32:1041–1036, 2000.

[21] Giorgio A. Ascoli and Jeffrey L. Krichmar. L-Neuron: A modeling tool for the efficient generation and parsimonious description of dendritic morphology. *Neurocomputing*, 32:1003–1011, 2000.

[22] S. Vinoski. CORBA: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, pages 46–55, February 1997. Specification available at http://www.omg.org/.

[23] Michael Hucka, Andrew Finney, Herbert Sauro, Hamid Bolouri, John Doyle, and Hiroaki Kitano. The ERATO systems biology workbench. In H. Kitano, editor, *Foundations of Systems Biology*. MIT Press, Cambridge, MA, 2001. In press.

[24] N. Goddard, M. Hucka, F. Howell, H. Cornelis, K. Shankar, and D. Beeman. Towards NeuroML: Model description methods for collaborative modelling in neuroscience. *Phil. Trans. Roy. Soc. B*, 2001. In press.

[25] D. Gardner, K. H. Knuth, M. Abato, S. M. Erde, T. White, R. DeBellis, and E.P. Gardner. Common data model for neuroscience data and data model interchange. *J. Am. Med. Informatics Assoc.*, 8:17–33, 2001.

[26] L. Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, MA, 1994.

[27] P. V. Biron and A. Malhotra. XML Schema part 2: Datatypes, April 2000. W3C Working Draft at http://www.w3.org/TR/xmlschema-2.

[28] D. C. Fallside. XML Schema part 0: Primer, 2000. http://www.w3.org/TR/xmlschema-0/.

[29] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML Schema part 1: Structures, April 2000. W3C Working Draft at http://www.w3.org/TR/xmlschema-1/.

[30] American national standard information retrieval application service definition and protocol specification for open systems interconnection. NISO Press, Bethesda, MD, 1992.

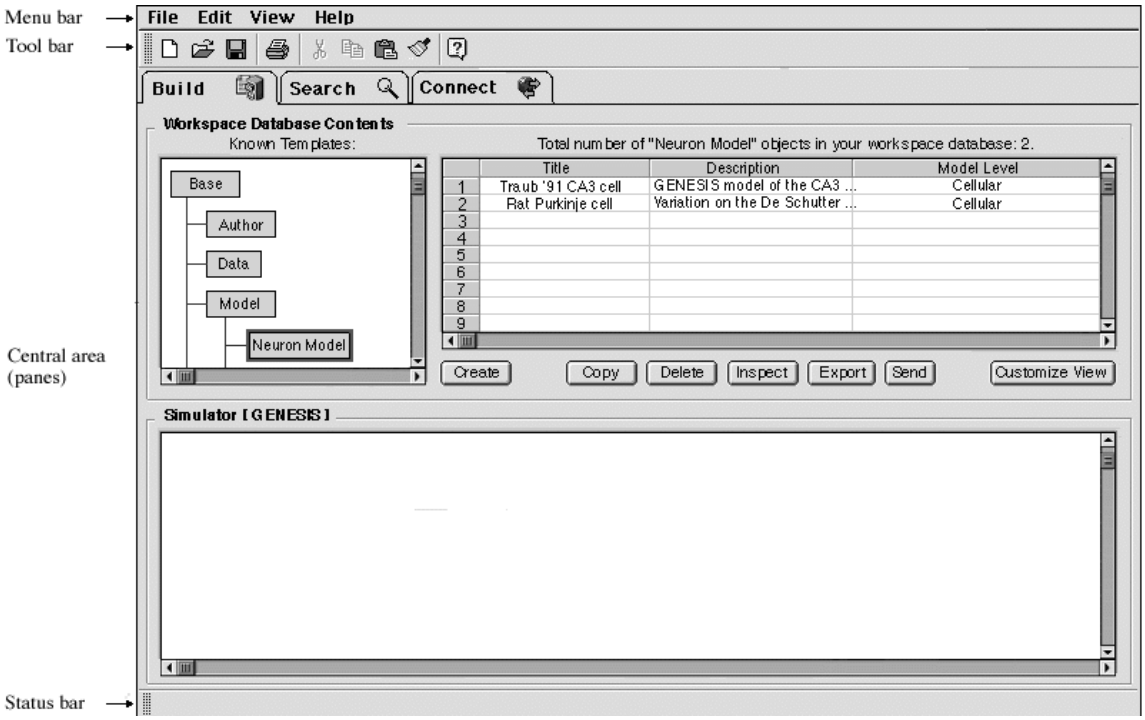

Figure 1: Fig1-build.eps

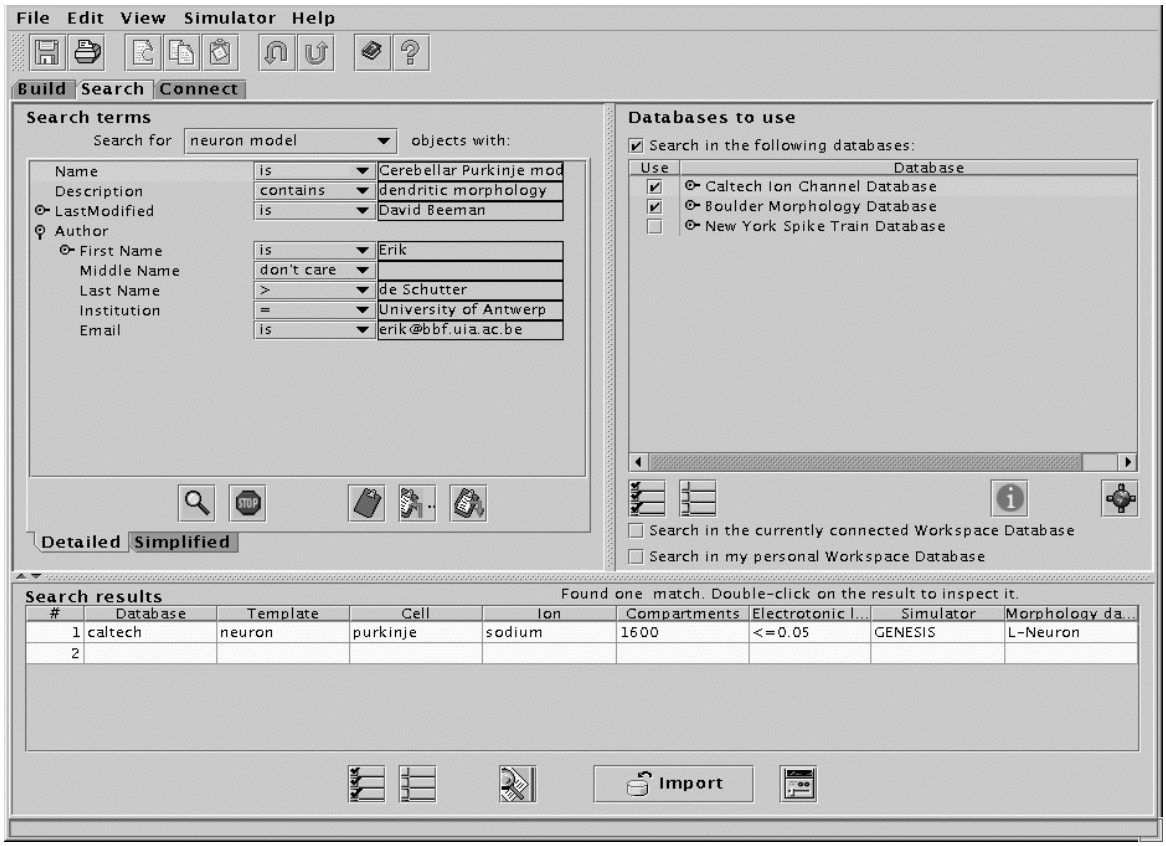

Figure 2: Fig2-search.eps

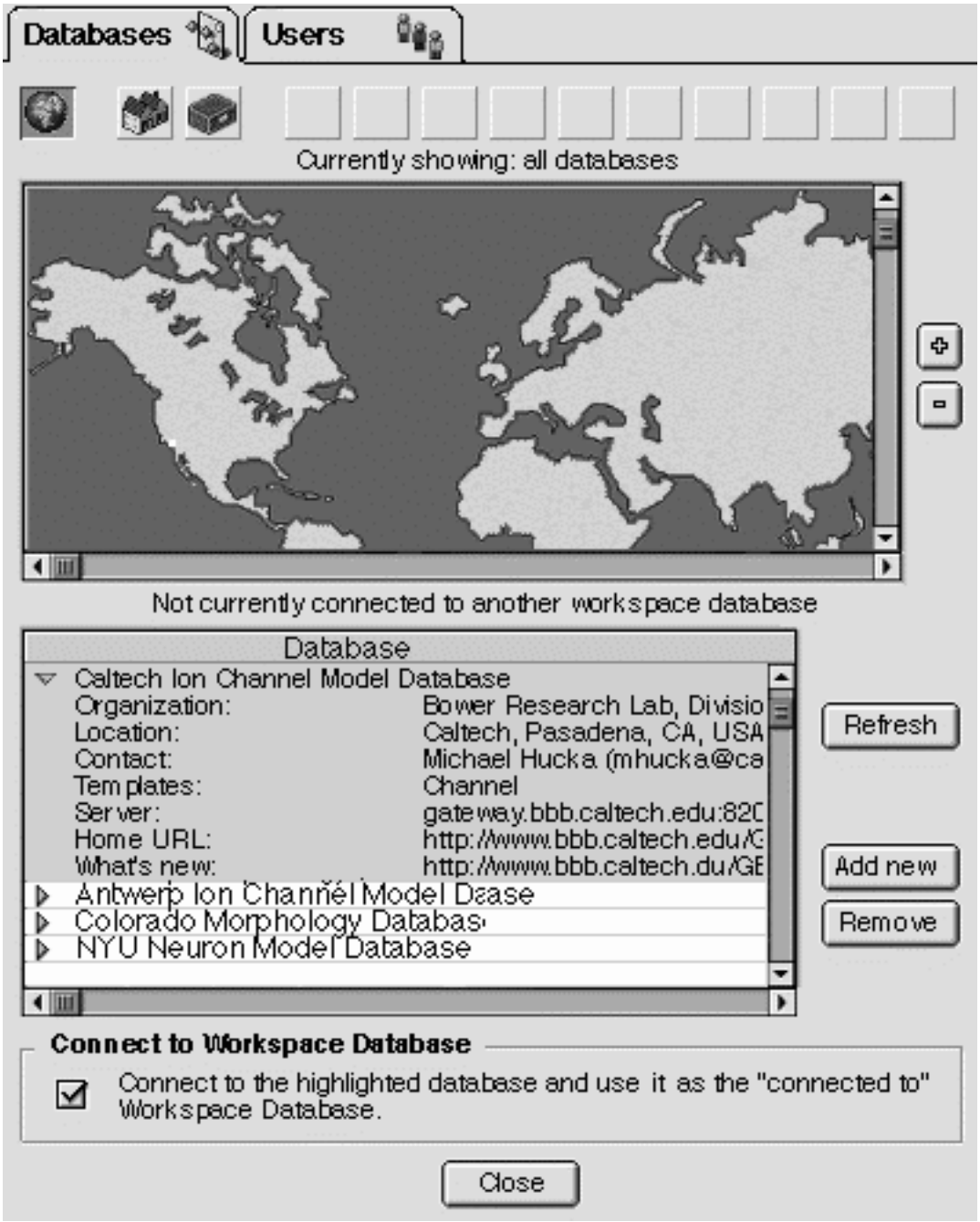

Figure 3: Fig3-map.eps

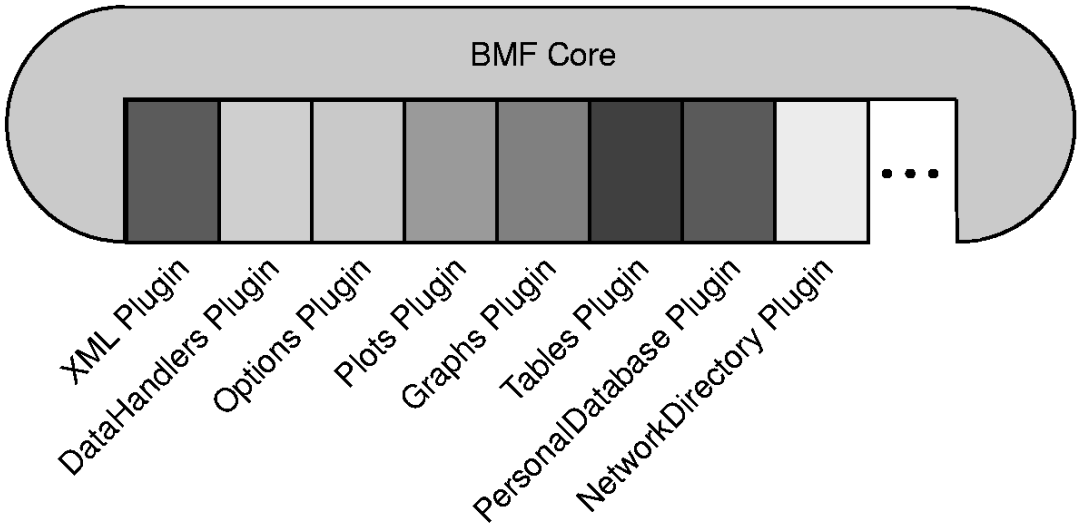

Figure 4: Fig4-bmfcore.eps

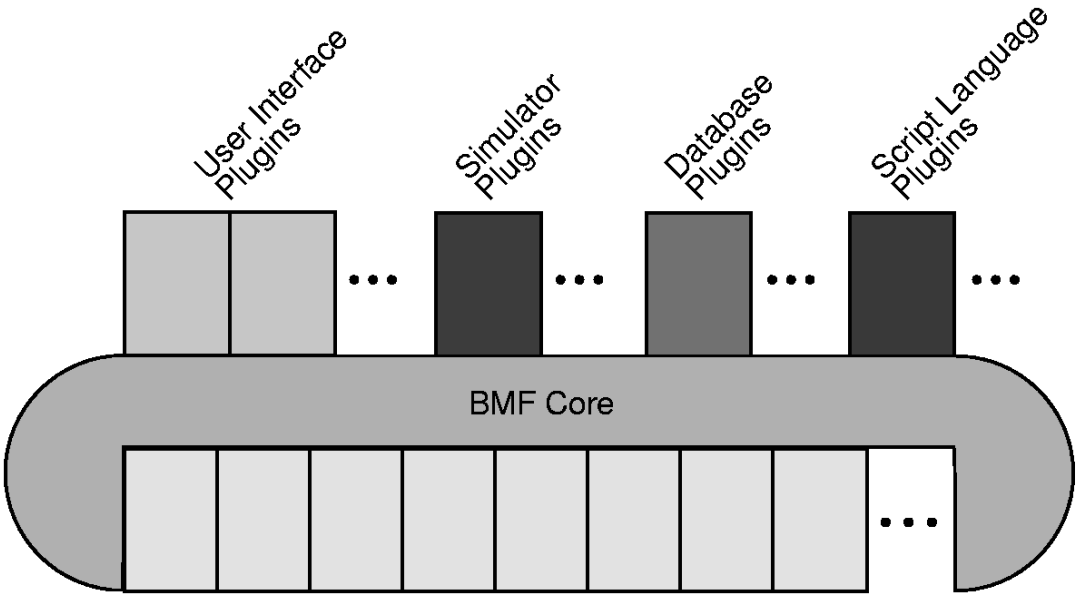

Figure 5: Fig5-bmfcore+app.eps

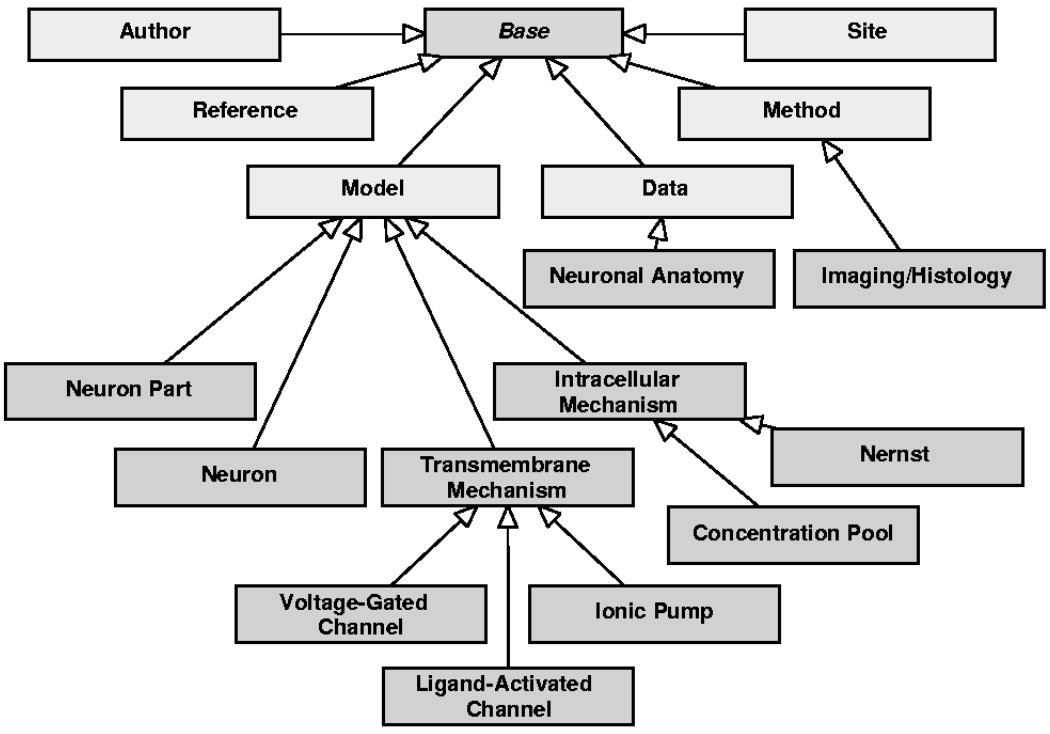

Figure 6: Fig6-levels.eps