The Computational Biology Initiative

The Genesis 3.0 Project.

A universal graphical user interface and database for research, collaboration, and education in computational neuroscience.

David Beeman, Allan D. Coop, Zhiwei Wang, Michael Edwards, Upinder Bhalla, Hugo Cornelis, James M. Bower. The Computational Biology Initiative, UTHSCSA/UTSA, San Antonio, Texas, USA

Background

The General Neural Simulation System (GENESIS), and its parallel version (PGENESIS) was the first broad scale modeling system in computational biology to encourage modelers to develop and share model features and components. From the outset, it was developed to support the biologically realistic simulation of neural systems, ranging from subcellular components and biochemical reactions to complex models of single neurons, simulations of large networks, and systems-level models. Since its release for general use in 1988, GENESIS has provided one of the foundations for the ongoing course in Methods in Computational Neuroscience at Woods Hole MA, as well as courses offered in the European Union, Mexico, Brazil, India, and in more than 50 universities around the world, where it has been used both as an instructional tool in realistic modeling of the nervous system, and as a simulation based tool for neurobiological education in general. The most recent release of GENESIS, version 2.3, became available in March 2006, and runs under most UNIX-based systems with the X Window System, including Linux, OS/X and Windows with Cygwin.

The release of the free internet edition of The Book of GENESIS (Bower & Beeman, 2003) and the browser-based self-paced GENESIS Modeling Tutorials (Beeman, 2005), which may be downloaded from the GENESIS web site, coupled with an active users group (BABEL), have also made it easy for individuals to learn GENESIS modeling without the necessity of attending courses or workshops. This substantial support for the use of GENESIS has also provided the base for an extensive and growing use of the software system in biological research, as evidenced by the rate of growth in the number of peer reviewed scientific papers using GENESIS from outside of the Bower laboratory (Figure 1).

GENESIS 1 and 2

Most current neuronal simulators are typically non-scalable in design, the result of development by either a single person or a restricted group. In contrast, the architecture of GENESIS has always facilitated user customization and user contributions and recognized that computational neuroscience and neuroscience in general has a broad scope and many branches.

The object-oriented approach taken by GENESIS and its high-level simulation language allows modelers to easily extend the capabilities of the simulator, and to exchange, modify, and reuse models or model components. Simulations are constructed from libraries of pre-compiled objects including:

1) Neural components, such as compartments, ionic conductances, axons, and synapses,
2) A kinetics library and its GUI Kinetikit for modeling biochemical reactions,
3) Objects for computing intracellular ionic concentrations from channel currents, for modeling the diffusion of ions within cells, e.g., concentration pools, ionic pumps, and buffers,
4) Devices for providing stimulation (such as pulse generators, spike train generators, voltage clamp circuitry) and analysis (e.g. spike train analysis objects),
5) Graphical objects used for creating custom displays, and
6) The hsolve library with elements and functions for the efficient implicit solution of the systems of differential equations that describe dendritic trees, as well as routines that maximize speed with faster, more stable numerical integration.

The scripting language and the modules are powerful enough that often only a few lines of script are needed to specify a sophisticated simulation. This approach allows one to create a new GENESIS simulation by modifying one of the many example and tutorial simulations that are provided with GENESIS, the Modeling Tutorials, or that are available from the GENESIS web site.

Similarly, scripting with GENESIS GUI libraries (XODUS), allows easy creation of displays, as in the visualization of a spreading wave of activation generated with GENESIS 2 in the Nenadic et al. (2003) large scale model of turtle visual cortex (Figure 2). Davis (Data Viewing System), is a general-purpose data viewer designed for the simultaneous display and analysis of a large number of dynamic data sets. It has been used for post-run analysis of this model using files generated by GENESIS 2. The GENESIS 3 design will allow for direct interfacing of such tools.

These features have resulted in GENESIS being one of the few simulators used for a broad range of activities from education to research, and from subcellular biochemical pathways to neuronal networks. However, the recent history of GENESIS has shown it to be increasingly difficult for developers from a wide range of scientific disciplines to contribute to simulator design and development. Although the GENESIS scripting language interface produces highly efficient modular object-oriented simulations that are easy to modify and extend, this is not the case with the source code which does not cleanly separate into its underlying components. It is difficult to add more modern Java-based graphical interfaces, alternate script parsers, and interfaces via the WWW. Such non-scalable architecture is not unique to GENESIS. The result is that experimentalists are often denied the opportunity to more easily ground empirical observation within the formalism of computational neuroscience.

The Computational Biology Initiative Architecture

The CBI (Computational Biology Initiative) architecture provides a modular paradigm that places stand-alone software components into logical relationships. In this it shares a number of ideas with the well-known model-view-controller (MVC) paradigm. The distinguishing feature of the CBI architecture is that the back end comprises numerical solvers rather than relational databases. The data layers in the CBI architecture correspond to high-level data associated with biological concepts and extend to low level data such as numerical values (Figure 3). The benefit of this layering of data is that it allows the mathematical and biological aspects of a model to be distinguished and separated.

Clear delineation of the modules in the CBI architecture allows both developers and users to choose to contribute to a single component with limited complexity, instead of being forced to contribute to the whole simulator and be exposed to tremendous complexity. Within the CBI paradigm each software component becomes self contained in the sense that it can be run independently. This has important advantages as it facilitates the interoperability of software obtained from different sources by:

1) **Reduced complexity** of software modules compared to a unitary system.
2) **Simplified documentation** of modules in terms of inputs and outputs.
3) **Easy incorporation** or removal of individual modules as required.
4) **Simplified development** and testing of components as stand alone modules.
5) **Clear delineation of scope** for new module development.

The CBI architecture provides three significant advantages for software development:

1) Modules can be run separately on different machines. For example, the GUI and modeling environment might run locally, while the simulator is run elsewhere either serially or in parallel on more powerful machines.
2) Decomposition of an application into multiple software components allows reuse and extension of individual modules, whether stand alone or otherwise, clearly facilitating model development and research progress.
3) Individual components can be independently updated, enhanced, or replaced when needed, thus the life cycle of a modular architecture is smoother than that of a non-scalable application.

The GENESIS 3.0 Project

With the growing interest and involvement of both neurobiologists and technologists in computational neuroscience, it has become increasingly clear that a more sophisticated approach to both simulation environments and documentation of modeling efforts is required. While GENESIS 1 and the upgraded version 2 were both self contained modeling systems, GENESIS 3.0 (G3) is being developed within the paradigm of the CBI architecture. Importantly, an enhanced interfacing capability with other neuroscience software tools and databases is now being provided (Figure 4).

The adoption of the CBI paradigm in the development of G3 has been prompted by recent technical advances in gluing (e.g. Swig and Python) and interfacing (e.g. SOAP), and an increased maturity in model exchange languages (e.g. NeuroML) and meta data exchange formats (e.g. BrainML). Specifically, the recently developed CBI simulator architecture is an open framework that provides the general context for G3 development. The CBI architecture focuses users on the need to conceive, organize, execute and evaluate simulations, while allowing the development of new tools to support simulation based education, collaboration, and publication. Consequently, G3 no longer includes parsers, script interpreters, run time schedulers, numerical solver engines, or the other components actually required to run simulations. Instead, G3 is being developed with the necessary interfaces that will, in principle, allow any simulation system to use its features.

Figure 1: Publications

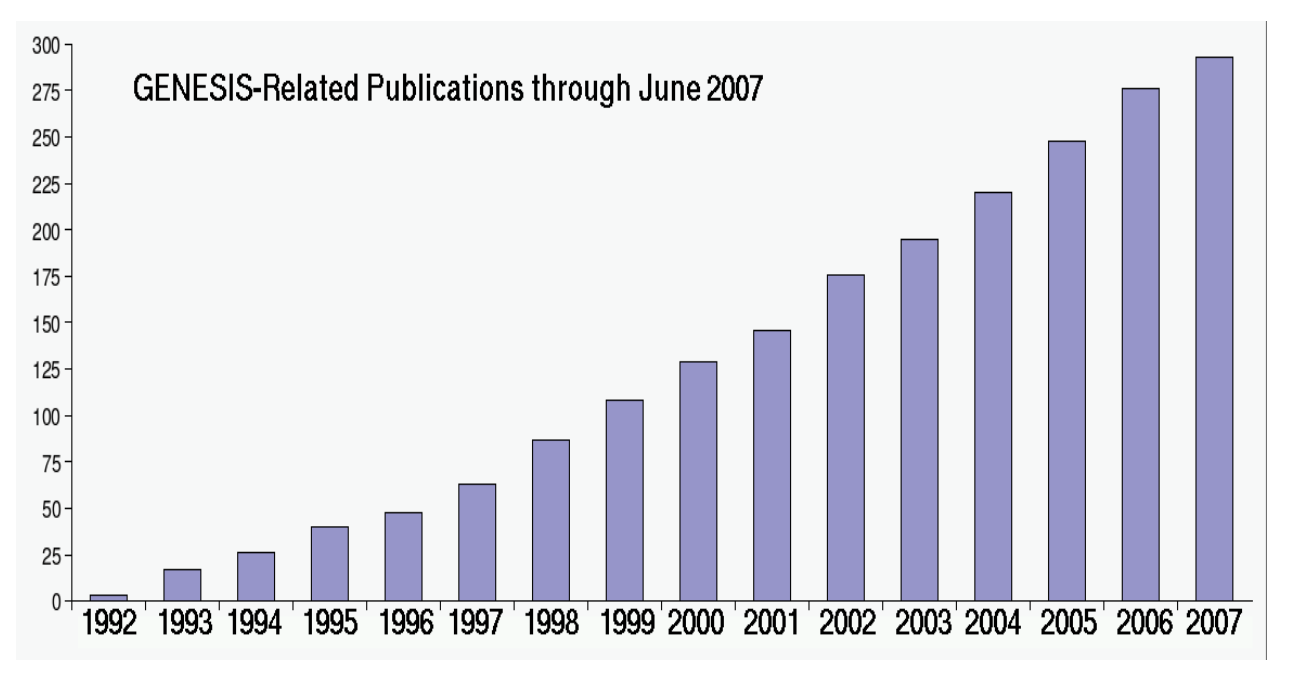


Figure 2: G2 GUI

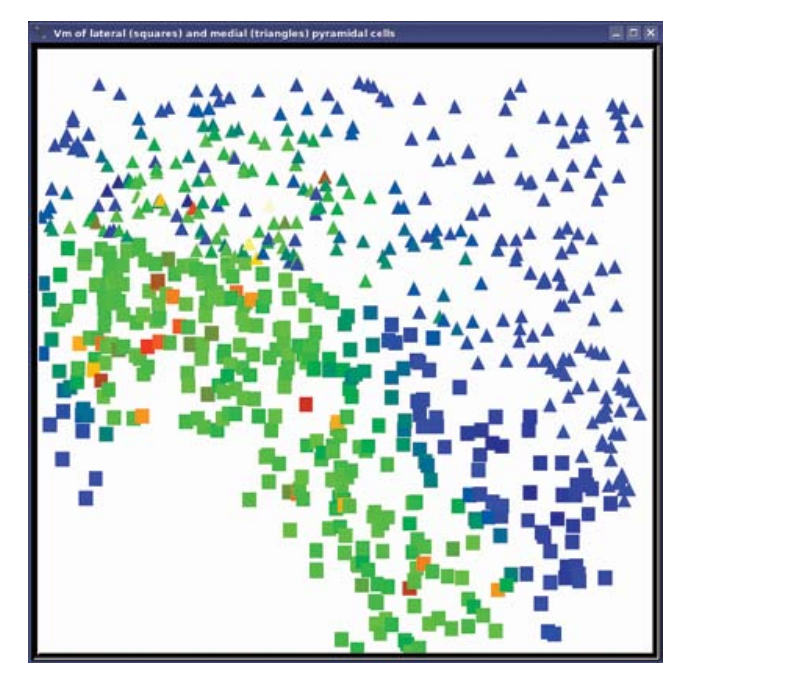
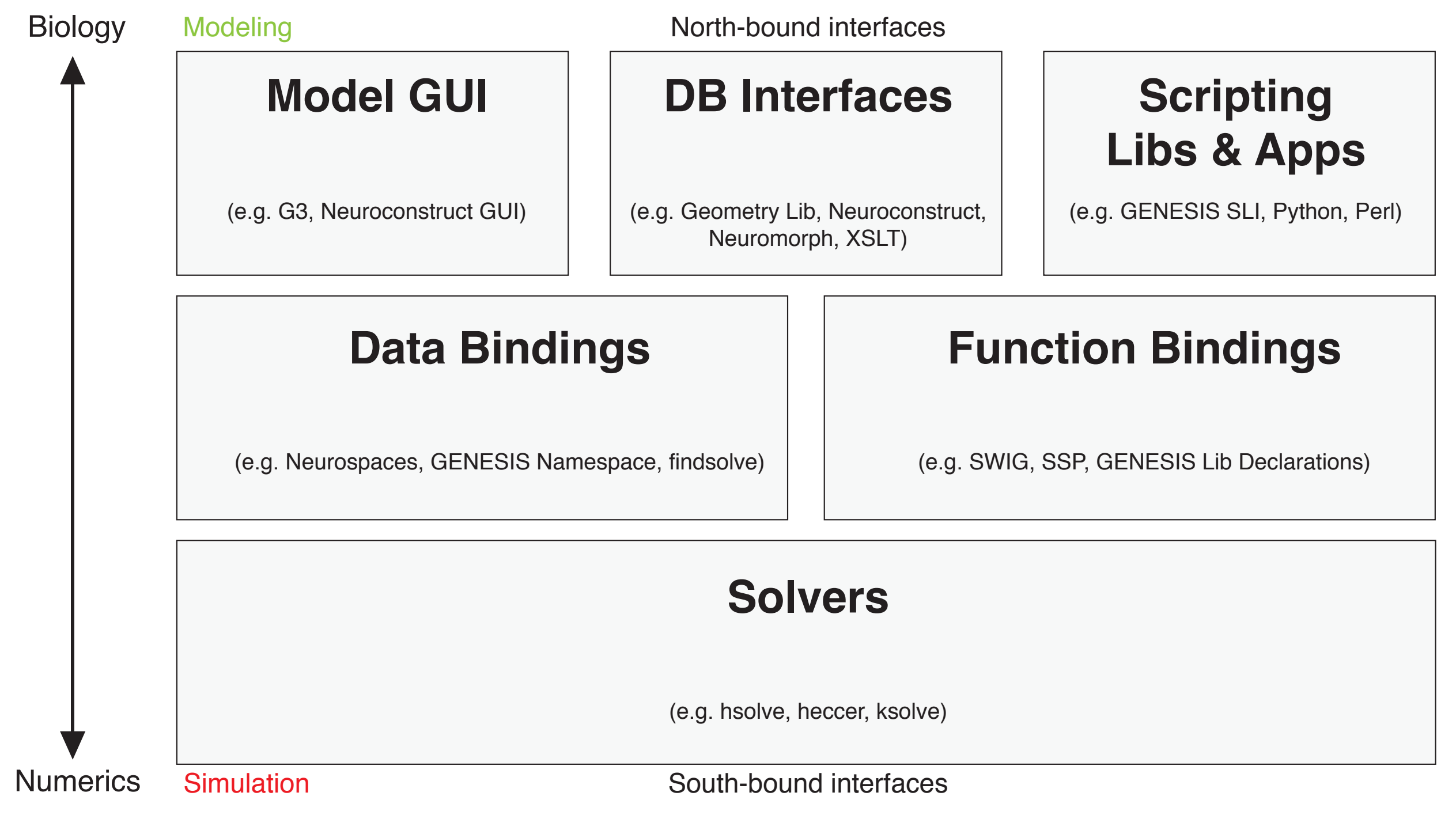

Vm of lateral (squares) and medial (triangles) pyramidal cells

Figure 3: CBI simulator functional architecture


Biology
Numerics
Modeling
Simulation
North-bound interfaces
South-bound interfaces
Model GUI
(e.g. G3, Neuroconstruct GUI)
DB Interfaces
(e.g. Geometry Lib, Neuroconstruct, Neuromorph, XSLT)
Scripting Libs & Apps
(e.g. GENESIS SLI, Python, Perl)
Data Bindings
(e.g. Neurospaces, GENESIS Namespace, findsolve)
Function Bindings
(e.g. SWIG, SSP, GENESIS Lib Declarations)
Solvers
(e.g. hsolve, heccer, ksolve)

A. Graphical User Interface

G3 is being developed with an open architecture that will allow different simulation systems to use its features. GENESIS, MOOSE, Neurospaces, NeuroML representations, and neuron imaging files (e.g. EM and confocal) will initially be supported. Here, simulator specific formats will be converted to the G3 internal representation which will then be available for manipulation by the GUI for viewing and editing.

The majority of simulators including GENESIS 1 and 2 are C/C++ programs. G3 is being developed as a Java based client/server application that communicates via sockets. One advantage of this approach is that Java Web Start technology will allow the most current version of the GUI to be deployed over the Internet with a single button click. The client and server can either be local on the same machine or distributed and run over the Internet via a communication socket. G3 compatibility is achieved by running an application as a daemon with interactions between numerical solvers and the console redirected to the socket. For example, in GENESIS, commands will no longer be input from the console, but rather solvers will accept commands from the GUI via a communication socket. This requires solution of the real time communication problem as the GUI and the numerical solvers are modules in the CBI architecture and are no longer in the same program as with a traditional unitary simulator. Real time communication between modules is achieved via the creation of dummy objects that communicate between the client and the server. For example, GENESIS 2 XODUS widgets are replaced by dummy objects prior to being forwarded to the server. The dummy objects then act as proxies on the server for the real widgets on the client side.

B. Messaging Object Oriented Simulation Environment

MOOSE reimplements the core GENESIS simulator code with much faster and cleaner messaging. It provides a general framework for making large, complex models, typically of biological and neuronal networks (Figure 5). It spans the range from single molecules to subcellular networks, single cells to neuronal networks, and extends to still larger systems. MOOSE is backwards-compatible with GENESIS, and forward compatible with Python and XML-based model definition standards such as SBML and MorphML. Features of MOOSE for kinetic modeling and simulation include:

1) Much faster stochastic solvers and very fast ODE solvers.
2) Access to nine different GNU Scientific Library integration methods.
3) Backward compatibility with G2. SLI is compatible. Currently refining *readcell*, the *kkit* reader, and implementing G2 objects.
4) Python scripting option.
5) Faster and more powerful messaging than G2.
6) General solver interface for plug-in numerical modules.
7) Unified reference scheme across nodes for transparent parallelization.

Figure 4: GENESIS 3.0

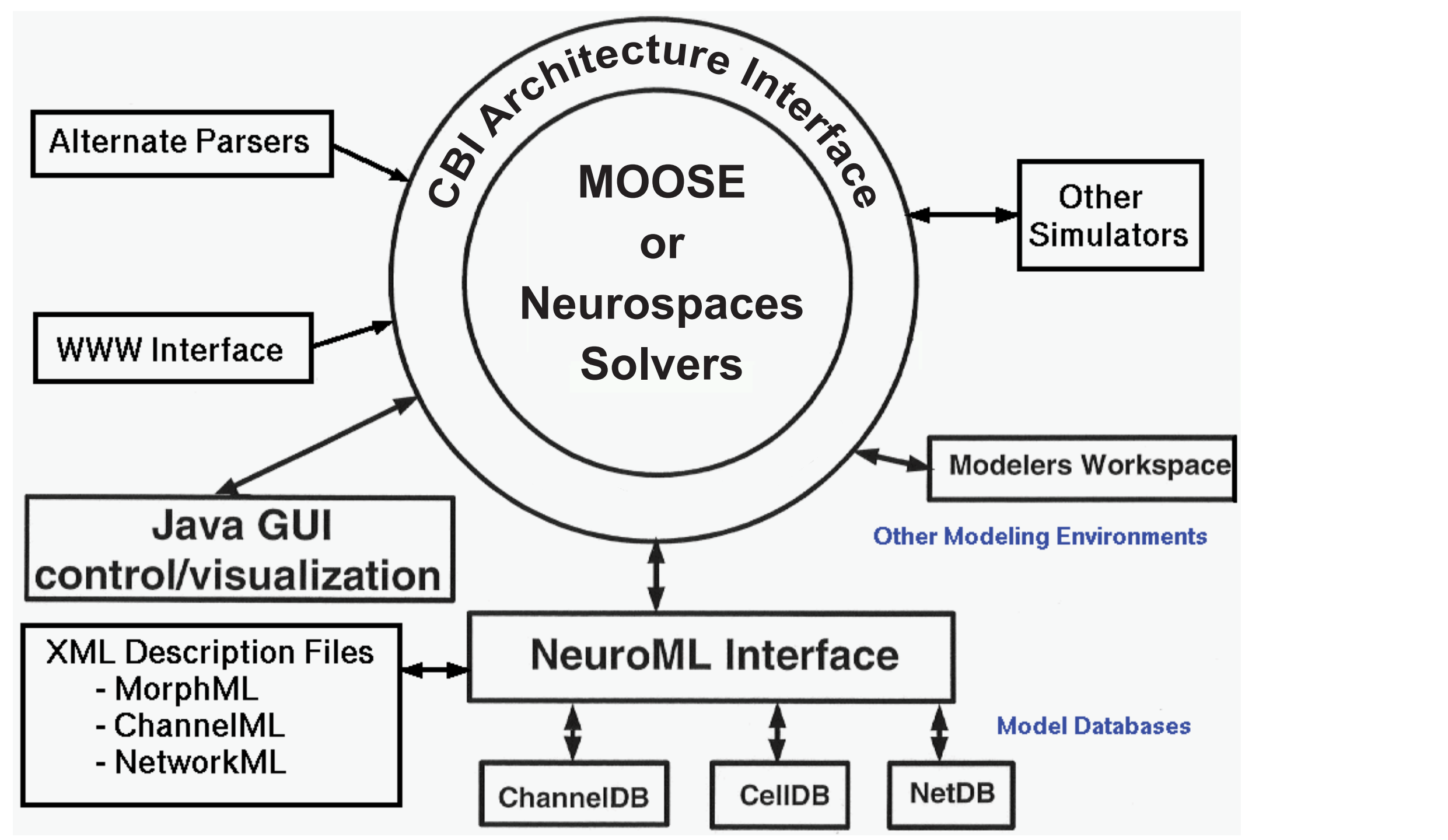


Figure 5: MOOSE

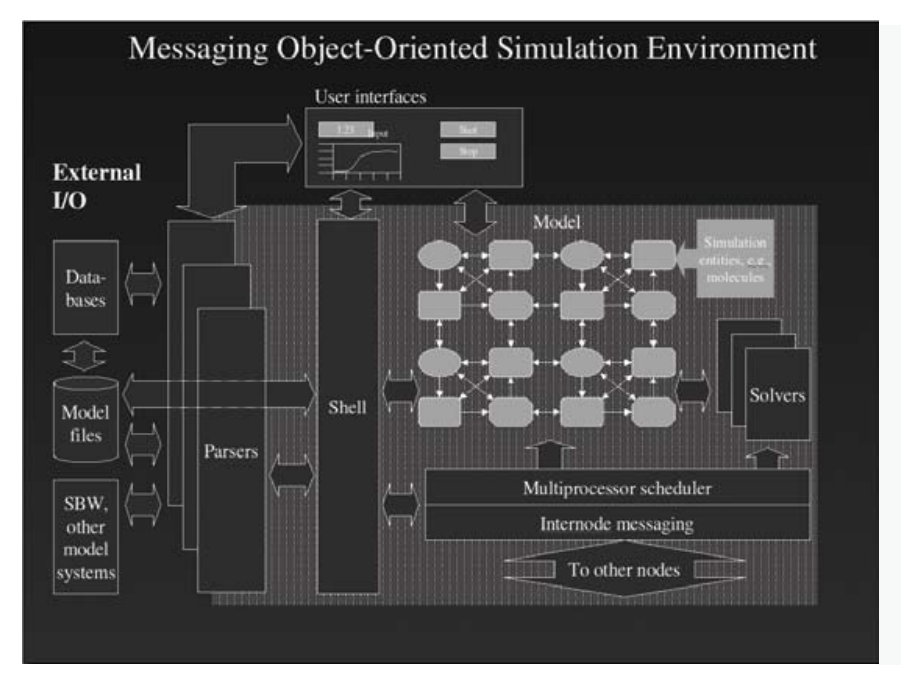
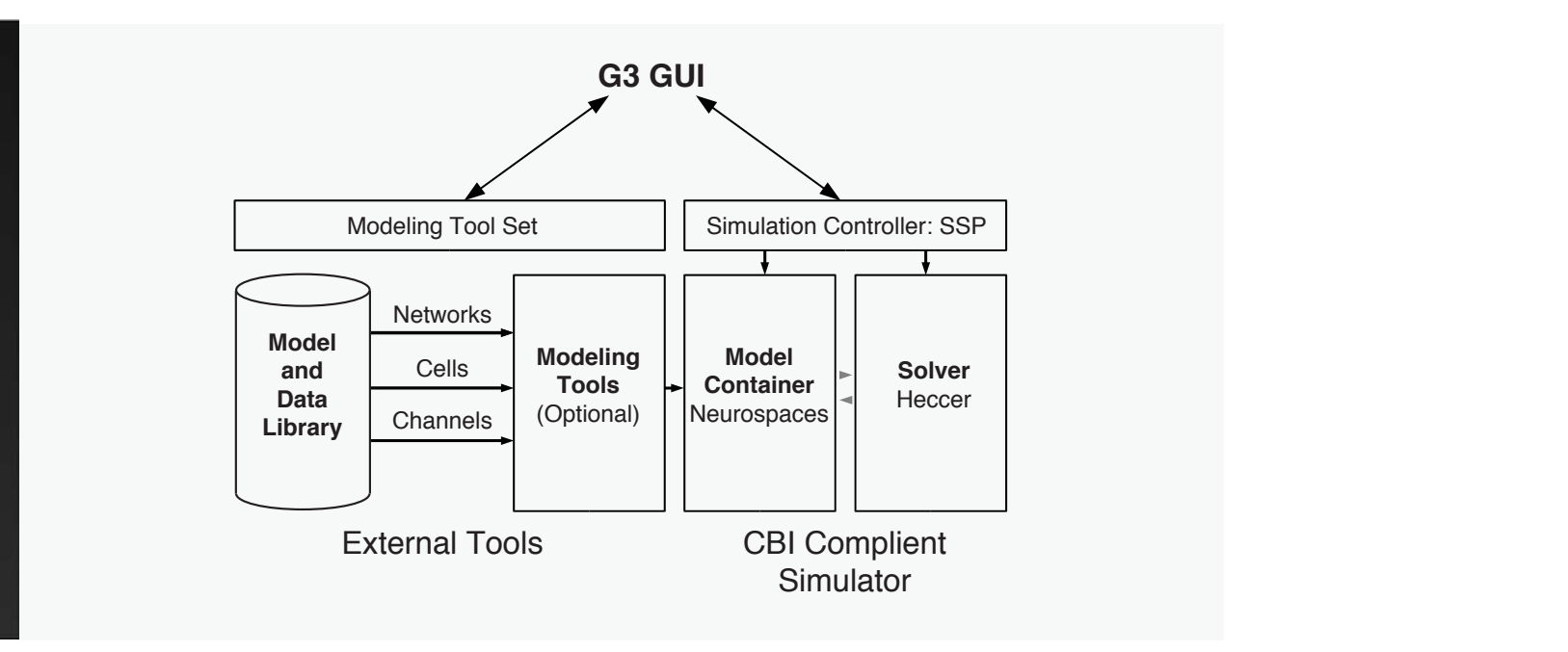


Figure 6: Neurospaces



C. Neurospaces Project

Compliant with the CBI architecture, the Neurospaces project embodies several completely modular software components implemented in a state-space theoretical framework (Figure 6). It supports a global namespace that is separated from solver instances, thereby separating and allowing for optimization of the numerical core independently of the modeling package. Neurospaces imports GENESIS .p files, NeuroML, and .swc cell morphology files, and exports NeuroML files. Currently, the following G3 compatible modules have been implemented:

1) *Heccer*: A fast compartmental solver based on the GENESIS *hsolve* that can be instantiated from C, Perl, or other scripting languages.
2) *Neurospaces:* Middle-ware dealing with biological entities and end-user concepts instead of mathematical equations.
3) *Simple Scheduler in Perl:* SSP can be constructed to bind Neurospaces and Heccer and activate them correctly, such that they work together on a single simulation.

Other modules in development include:

1) *Neurospaces Studio:* Tools for graphical browsing and command line usage.
2) *Geometry Library:* A general purpose library containing essential geometrical operators not commonly found in other geometrical libraries.
3) *Reconstruct Interface:* Supports conversion of contours exported by the Reconstruct software to the Neurospaces declarative NDF format.
4) *Project Browser:* Inspect projects and simulation results.

REFERENCES and URLS

Beeman D. (2005) GENESIS Modeling Tutorial. Brains, Minds, and Media. 1: bmm220 (urn:nbn:de:0009-3 2206). (http://www.brains-minds-media.org).
Bower JM & Beeman D, (1998) The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System, Third (Internet) Ed., www.genesis-sim.org/GENESIS/bog/bog.html.
Nenadic Z, Ghosh, BK & Ulinski P. (2003) Propagating waves in visual cortex: A large scale model of turtle visual cortex, J Computational Neuroscience 14:161-184.

The GENESIS home page: http://genesis-sim.org/GENESIS
The MOOSE project home page: http://sourceforge.net/projects/moose/
The Neurospaces project home page: http://neurospaces.org