The Computational Biology Initiative

The CBI Architecture for Computational Simulation of Realistic Neurons and Circuits in the GENESIS 3 Software Federation.

Hugo Cornelis[1], Michael Edwards[1], Allan D. Coop[2], James M. Bower[1]. The Computational Biology Initiative.

[1]Research Imaging Center and [2]Dept. Epidemiology and Biostatistics, UT Health Science Center, San Antonio, Texas 78229, USA.

Biology

Model Data

Model Tools

- Model version history manager: Track and record history of model development.
- Data base connector. Acquire model from data base such as neuromorph.org, convert it to a format that can be understood by the model container.
- Model and result analyzer. Inspect a model stored in the model container and extract, for example, morphological or connectivity characteristics for quantification of model components,
- Model and result annotator. Automatic or manual text attachment to whole or part of model
- Automatic model constructor: For example, parameter searching, development of dendritic morphology, software components that embed algorithms for automatic model construction.

Biology Model Container

- Allows a user to define a model in terms of biological properties such as spine, morphology, circuits and their connectivity.
- Stores Biological Model: Model is available for inspection by other software modules.
- Translates hierarchical biological model to an expanded numerical model for solvers.
- Translates model connectivity to connectivity between solvers.

Experiment Model Container

- Allows definition of the experiment in terms of actions taken on the biological model.
- Stores model of stimulus paradigm and output definitions.
- Defines and stores a hierarchical sequence of actions (e.g. start and stop times of current injection) and their dependencies.

North-bound interfaces

Graphic User Interface

model & results

- Connects to other modeling components to provide a GUI for the data base connectors and the biological and experimental model containers.
- Provides different views of the model and the possible work flows between them.
- Color-code morphology and network behavior according to user specified activity.
- Plot temporal evolution of solved variables.

simulation control

- Buttons for interaction with the controller, e.g. to start and stop a simulation.
- Buttons and dialog boxes to specify experimental model inputs and output, e.g. stimulate model at a particular location.
- Browser provides access to sets of related simulations and results.

Simulation Control

Scripts & Applications

- Applications and tutorial functions written in high-level scripting languages.
- Connect model and stimulus using a procedural paradigm.
- These applications have a focus on specific research questions or educational tutorials.
- They define user work flows to run simulations and to prepare results for analysis.

Implementation Libraries

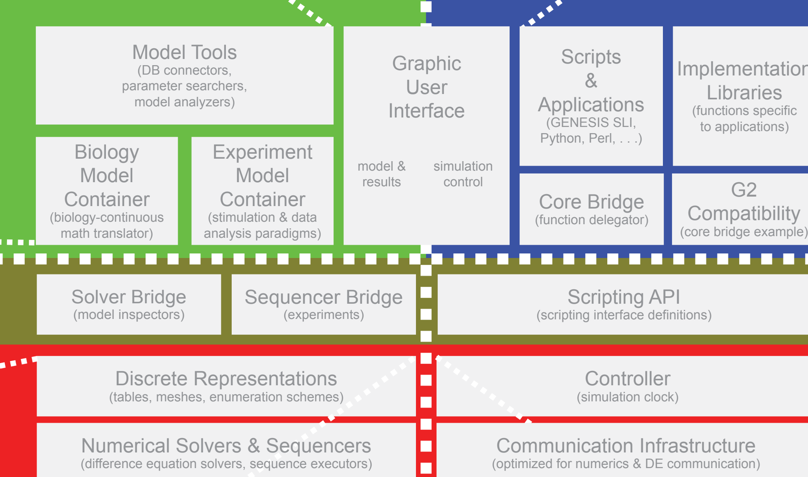
- Contain glue functions to/and external libraries for a variety of purposes, for example for result analysis.
- Contains neuroscience specific libraries that may be implemented directly.
- Examples include, the GNU Scientific Library (GSL), the Perl Data Language (PDL), and the Scientific Library for Python (SciPy).

Core Bridge

- Translates procedural descriptions of a simulation (including legacy) by decomposition into modeling vs. implementation control.
- The core bridge delegates procedural scripting statements to the appropriate software components in the simulator.

G2 Compatibility

- A specific implementation of the core bridge provides backward compatibility with the GENESIS 2 software platform.
- Currently this implementation of a core bridge is complete for simple single neuron simulations and near complete for the Purkinje cell model.



Solver Bridge

- Data Binding Framework: Gives a simple 1-to-1 mapping/translation of a numerically expanded mathematical representation of a model into data structures that match a specific solver implementation.
- Decouples the numerical backend from model containers: Allows for separate optimization of model containers and solvers.

Sequencer Bridge

- Data binding translates the sequence of stimulus actions and output definitions to data structures specific to a sequencer implementation.
- Decouples sequencer backend from the experiment model containers to allow independent optimization of both.
- Gives simple 1-to-1 mapping/translation of model data with sequencer.

Scripting APIs

- A series of specifications that glue the functionality of other software components to existing scripting languages. The API may (preferentially) be automatically generated via SWIG.
- Additional code may be required to translate low level APIs to high level APIs typical of scripting languages.

Discrete Representations

- Internal software publication (invisible to users) of the numerical representations of a model, e.g. discretized tables of channel gate kinetics, dendritic morphology meshes, Hines enumeration of compartmentalized morphology.
- Annotation of internal publications for identification purposes and intelligent reuse.

Controller

- Activates the solver and the communication infrastructure as required.
- Contains the global simulation time clock.
- Contains the core functions such as those that start and stop a simulation.

Numerical Solvers & Sequencers

- Low-level backends at the software and/or hardware levels that solve numerical equations using specific algorithms.
- Discretization and tabulation of fixed mathematical functions according to a user setable accuracy specific to the solvers.
- Sequence executors provide a physical implementation of an experimental protocol *in computo*.

Communication Infrastructure

- Establishes run-time communication between different solvers and output elements working on the same model.
- Optimized for communication of array based (numerical) data.
- Differential implementation for serial as opposed to parallel hardware.

Numerics

Simulation Data

South-bound interfaces

Implementation Control

The Problem

The repetitive incorporation of foreign source code into an application ultimately makes the source code structure so complicated that the core software becomes difficult, if not impossible, to extend.

The resulting stand-alone applications become monolithic and their life cycles are moved from extension to maintenance.

A Solution

We propose that in computational neuroscience two fundamental axes can be employed to partition the functionality of a neural simulator:

1. In the computational realm there is a distinction between data and control.
2. In the simulation realm there is a distinction between the biological model and its numerical implementation.

	Data	Control
High-level	Biology	User Workflows
Low-level	Numerics	Scheduling

Relationship between the four functional modules of a neural simulator.

Horizontal and vertical interactions maintain software modularity, whereas, diagonal interactions lead to monolithic software architectures.

The CBI Architecture

- The CBI simulator software architecture places stand-alone software components into the logically layered diagram illustrated above.
- These layers map naturally to the occurence of high-level data (e.g. biological concepts) versus low-level data (e.g. numerical values).
- Independent of the technology used, the diagram can be employed, by the user and developer communities to communicate about the global concepts and functions present in the software.

The Advantages

- Single component complexity is reduced compared to that of the total software system.
- Components can be documented and tested as isolated entities in terms of inputs and outputs. Facilitates communication among developers.
- Unnecessary or obsolete components are easily replaced, or removed from the architecture.
- A component can easily be tested stand-alone.
- Importantly, the CBI architecture clearly delineates the scope of new development.

Conclusions

- Each box illustrated above represents a class of stand-alone software components. Gluing together the appropriate subsets of these components results in different functional applications.
- The Neurospaces Project follows this approach to implement the core of a new GENESIS simulator.
- The federated CBI architecture we propose has important advantages for both the software developer and user communities.

References & Web Links

Eckerson WW (1995) Three tier client/server architecture: Achieving scalability, performance, and efficiency in client server applications. *Open Inf. Syst.* **10**: 3-20.

Galis, A (2000) Multi-Domain Communication Managment. CRC Press: Boca Raton, FL.

GENESIS 2: http://www.genesis-sim.org/
GENESIS 3: http://genesis-sim.org/
Neurospaces: http://www.neurospaces.org/