# NeuroSpaces : Separating Modeling from Simulation

Hugo Cornelis,Erik De Schutter

February 8, 2002

**Abstract**

Modeling starts with sets of mathematical equations that have been put together by people who have a good intuition of the concepts behind the biological reality. A software system for neuronal modeling thus must manage sets of equations as well as the biological concepts.

A rigorous theoretical analysis on the software requirements for such a system has led us to a new fundamental data model. Based on this data model we have implemented an extensible software system that combines mathematical and neurobiological modeling and that still reflects intuitions like heterogenous components and hierarchical networks. When connected to a computation engine, the system turns into a simulator that is efficient in memory requirements, easy to use and allows complete inspection of the mathematical model.

## 1 Summary

During the last twenty years traditional monolithic software design has gradually been abandoned in favor of architectures with a small extensible core or kernel. The kernel manages a set of modules and components that engage in shared activities via semantically defined interfaces. Such systems are called collaborative systems. The current neuronal simulation packages – developed during the last decade – don't share this philosophy. It makes them hard to extend and difficult to use for large and complex neuronal models.

Recent developments try to overcome this situation in two ways : first there have been attempts to integrate the existing packages into a collaborative system. Although these attempts could hold a promise, they have been unsuccessful so far because the simulation package itself still suffers from its monolithic design and is difficult to interface with other software components. Second, attempts have been made to make calculation libraries accessible as object-oriented class hierarchies. These attempts have no direct relation with neuronal modeling with the consequence that few people in the neuroscience community are aware of the project. The classes are sometimes ill defined or do not address a number of important issues related to neuronal modeling.

Our approach has been different. After analyzing the structural software requirements for a neural simulation package we discovered that there is currently no data model available that covers all the modeler's needs. An example is the need to compose hierarchical networks. In Genesis such a network is composed with scripts that construct all the populations and then add links between individual cells. After the setup phase of the model, however, the topology of the network as well as the projections between the individual populations are lost i.e. the model has become a flat network instead of being a hierarchical network – the modeler's concept of the model. In structural terms it means that it must be possible to have projections at the different levels of the network hierarchy. Yet the requirement to inspect all the connections on one cell must not be violated, even if the cell is involved in many projections. Currently no single data model is able to fulfill both of these two requirements. Other examples of problematic needs are scaling of conductances and compound models of heterogenous intracellular mechanisms. Reducing these problematic needs we ended with a set of ultimate requirements.

Building further on known theoretical data structures that have been studied in informatics, we designed a new data model that is able to address all these requirements. The theoretical basis of this new data model is given by an ordered tree with relative paths to tree nodes encoded as integers. By assigning mathematical or biological semantics to the nodes of the tree, it becomes a natural representation of a model. The integer encoding scheme allows grouping of components at all biological levels and memory efficiency via prototyping of repetitive sub-models. Trivial manipulations of the integers associated with such a relative path allows a flat representation of components of the same type such that e.g. projections in a hierarchical network can be inspected via a single connection matrix and connections converging on the same synapse can be queried.

The implementation of this data model for neuronal modeling is called NeuroSpaces. It consists of a number of independent software components that cover different needs of extensibility. An example is the loadable module manager that supports the implementation of extensions covering specialized modeling semantics, such as setting up three-dimensional projection schemes or randomizing parameters. Another part of the system is a file parser that gives explicit support for model reuse and grouping into libraries.

Except for computer science related research NeuroSpaces also includes a number of benefits for neuronal modelers such as memory efficiency and complete model validation. A major design principle has been the separation of the specification of the mathematical model and the way of simulating this model as done by computation engines. Computation engines can now be implemented in the most efficient way while their complexity is hidden from the user by NeuroSpaces.

NeuroSpaces has been linked with Genesis and its highly optimized numerical computation engine (hsolve) and can be used to build models and simulate their behavior. Using subcellular components like dendritic segments and synapses as building blocks, we constructed libraries of detailed models of the different cell types found in the cerebellar cortex. These libraries have in turn been used to build networks with the same topology as the ones currently used to investigate the function of the cerebellar cortex. Since these libraries cover the complete spectrum from the subcellular up to the network level, the memory requirements of complex models and the setup time of computation engines can easily be evaluated. The structure of the network libraries can be used as a basis to build other network types.

After having tested this software system further our future plans are twofold : first we will be working on database integration. The libraries we are currently using are self-contained and declarative. This makes interaction possible with XML, the general agreed upon database exchange format. We are planning to implement additional software to setup simulations directly from databases.

Second we are investigating how to distribute models on parallel machines. Since the mathematical model is completely available for inspection, it can be used to calculate a workload for transparent parallelization of large simulations.