brains, minds media
ISSN: 1861-1680

XML for Model Specification in Neuroscience

An Introduction and Workshop Summary

Sharon Crook[a]*, David Beeman[b], Padraig Gleeson[c] and Fred Howell[d]

[a] Department of Mathematics and Statistics and School of Life Sciences, Arizona State University, Tempe, Arizona, USA
[b] Department of Electrical and Computer Engineering, University of Colorado, Boulder, Colorado, USA
[c] Department of Physiology, University College, London, UK
[d] Institute for Adaptive and Neural Computation, University of Edinburgh, Edinburgh, UK

*Corresponding Author: Tel: + (480)965-0403, Fax: + (480)965-0461, email: sharon.crook@asu.edu

urn:nbn:de:0009-3-2282

Abstract. One of the main roles of the Neural Open Markup Language, NeuroML, is to facilitate cooperation in building, simulating, testing and publishing models of channels, neurons and networks of neurons. MorphML, which was developed as a common format for exchange of neural morphology data, is distributed as part of NeuroML but can be used as a stand-alone application. In this collection of tutorials and workshop summary, we provide an overview of these XML schemas and provide examples of their use in down-stream applications. We also summarize plans for the further development of XML specifications for modeling channels, channel distributions, and network connectivity.

Keywords: XML, NeuroML, ChannelDB, GENESIS, MorphML, neuroConstruct

Citation: Crook S, Beeman D, Gleeson P, Howell F (2005). XML for Model Specification in Neuroscience - an Introduction and Workshop Summary. Brains, Minds and Media, Vol. 1, bmm228 (urn:nbn:de:0009-3-2282).

An Introduction to XML in Neuroscience - *Sharon Crook*

What is the eXtensible Markup Language *(XML)*[1]*?* XML is a portable format for computer documents where data are surrounded by text descriptions called tags. Tags are ordinary text that is meant to provide a clear, concise and understandable context for data. Due to this self-describing representation, programs can parse documents easily. The tags define data objects called language elements, and the hierarchy of relationships among the language elements can be used to create an XML schema.

[1] XML, http://www.w3.org/XML

What are some of the potential benefits of XML? Because each language element in an XML schema can be equivalent to an object class in a programming language such as Java or C++, XML facilitates the generation of code for reading and writing data documents. The use of XML also makes it easy to validate documents and create database tables for data element storage and access. All of these aspects of XML lead to its greatest benefit—its use facilitates communication and collaboration.

Are there XML applications for neuroscience data or modeling? The following list provides examples of applications under development that can be used for representing neuroscience data and models. The first two are described in more detail in subsequent sections.

NeuroML[2] supports the use of declarative model specifications for neuroscience modeling efforts at different scales, from intracellular mechanisms to networks of reconstructed neurons.

MorphML[3] provides a common format for exchange of neuronal morphology data. It can also be used to specify cell structure for modeling efforts as part of NeuroML.

BrainML[4] is an application for representing time series data, spike trains, experimental protocols, and other data relevant to neurophysiology experiments.

SBML[5] (Systems Biology Markup Language) is an application for specifying models of biochemical reaction networks such as metabolic networks, cell-signaling pathways and gene regulatory networks.

CellML[6] is designed for the specification of biological models of cellular and sub-cellular processes such as calcium dynamics, metabolic pathways, signal transduction, and electrophysiology.

MathML[7] provides the means for describing the structure and content of mathematical notation in order to serve, receive, and process mathematics on the web. Other XML applications often use MathML language elements for representing mathematical equations.

Many of these XML applications have associated tools for data analysis, data visualization, simulation, or data storage. Some also have tools for further development of the XML application itself. For example, the NeuroML Development Kit[8], described in more detail in the next section of this article, can be used to parse and generate NeuroML to and from a Java object tree.

An additional benefit of using XML is the availability of commercial and free development software. For example, there are many commercial products available for schema development, validation, and documentation such as Altova's XMLSpy[9]. Software such as Sun's Java Architecture for XML Binding (JAXB)[10] can be used to bind an XML schema to schema-derived classes and will create an application program interface for reading and writing XML documents.

[2] NeuroML, http://www.neuro.org/
[3] MorphML, http://www.morphml.org/
[4] BrainML, http://www.brainml.org/
[5] SBML, http://www.sbml.org/
[6] CellML, http://www.cellml.org/
[7] MathML, http://www.w3.org/Math
[8] NeuroML Develpoment Kit, http://www.neuroml.org/projects/ndk.html
[9] Altova's XMLSpy, http://www.altova.com/
[10] Sun's Java Architecture for XML Binding (JAXB), http://java.sun.com/xml/jaxb

What are the potential liabilities of XML? It should be noted that most of the advantages of XML can be accomplished with good discourse, good design and good documentation. The major liability of XML is that files are extremely verbose compared to non-XML formats, so performance may suffer. In addition, the same features that make XML valuable for communication and collaboration can make it more difficult to protect private data.

An Introduction to NeuroML - *Fred Howell*

NeuroML is an ongoing effort to improve uptake of XML for software relating to neuroscience modeling. It is not a single standard XML language - rather a collection of related XML projects for modeling different aspects and levels of neural systems, from intracellular mechanisms and ion channels to networks of reconstructed neurons. This tutorial introduces the aims, techniques and future plans of the NeuroML project.

Aim

The aim of the NeuroML project is to move model specifications from programs to a declarative XML format.

The challenges

Building working models of the circuits of neural tissue is one of the best ways of furthering our understanding of how these complicated biological machines work. But models are often themselves complex programs, understandable only to the researcher who developed them. In order to give these models a longer shelf life and to make them more transparent, it is useful to express such models in a declarative fashion. XML is an ideal representation for the complex structure of models, as it is an open file format and is capable of representing arbitrarily complex structures. However, devising a standard language for models in neuroscience is challenging because of the variety of levels of scale and the various amounts of detail in models.

The schematic of the interactions in a synapse (Collins et al 2005) shown in Figure 1 reveals some of the complex 3D interactions among intracellular pathways, receptors and the synaptic machinery. SBML provides a method for describing the basics of intracellular pathway modeling but provides no formalism for the spatial aspects of 3D interactions. It also does not include support for modeling the electrical properties of receptors, a critical aspect of most neural models.

Models of single neurons often include detailed *morphology* data from reconstructed cells. Detailed models of dendrites also require data from electron microscopy (EM) level reconstructions. (Figure 2 below shows an EM slice through rat hippocampus.) The MorphML XML standard was designed as an exchange format for morphology data, to complement the existing proprietary formats of reconstruction systems such as Neurolucida. In addition to morphology data, building a working model of a neuron requires access to physiology data including information about the types and distributions of ion channels throughout the cell membrane. It would help the modeling task substantially if more experimental data were available in open file formats. XML based formats for experimental metadata, as well as a cultural shift towards data publication will help this process.

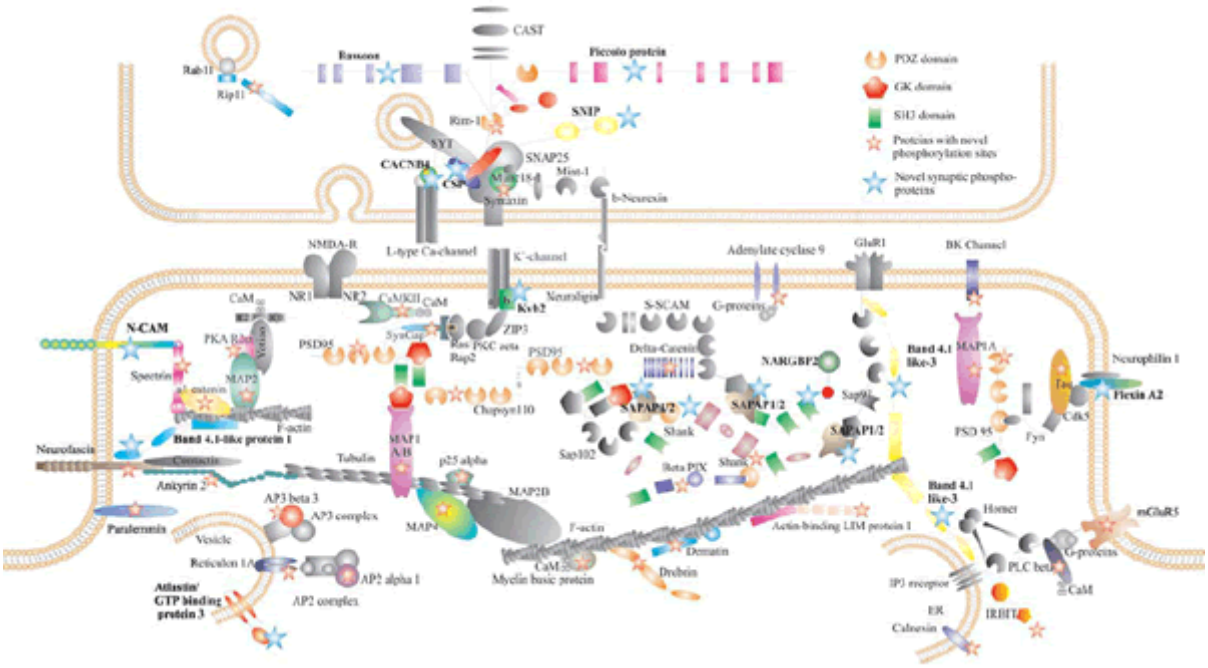


Figure 1 - Interactions in a synapse (from Collins et al 2005). Figure reprint kindly permitted by the *American Society for Biochemistry and Molecular Biology*.

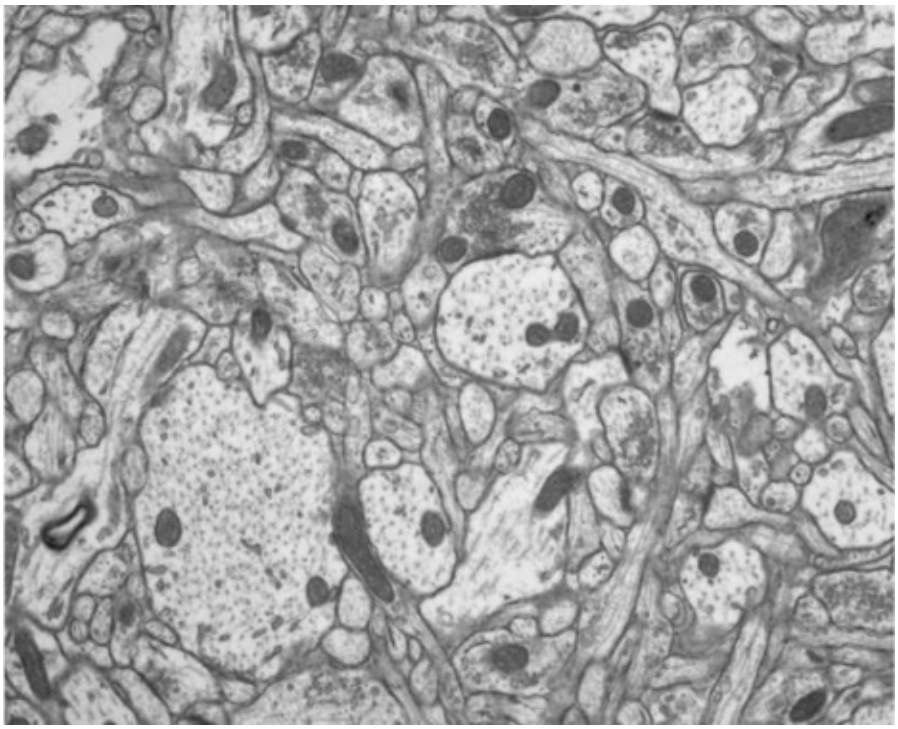


Figure 2 - EM of slice from rat hippocampus (data from Synapse Web[11], Medical College of Georgia)

[11] Synapse Web, http://synapses.mcg.edu/

Connectivity data is especially hard to obtain, but essential to building network models. Figure 3 shows an example model of part of the cerebellar cortex circuitry (Howell et al. 2000). This model was built using a script program for the GENESIS[12] simulator; the aim of NeuroML is to allow networks such as this to be specified declaratively in XML.

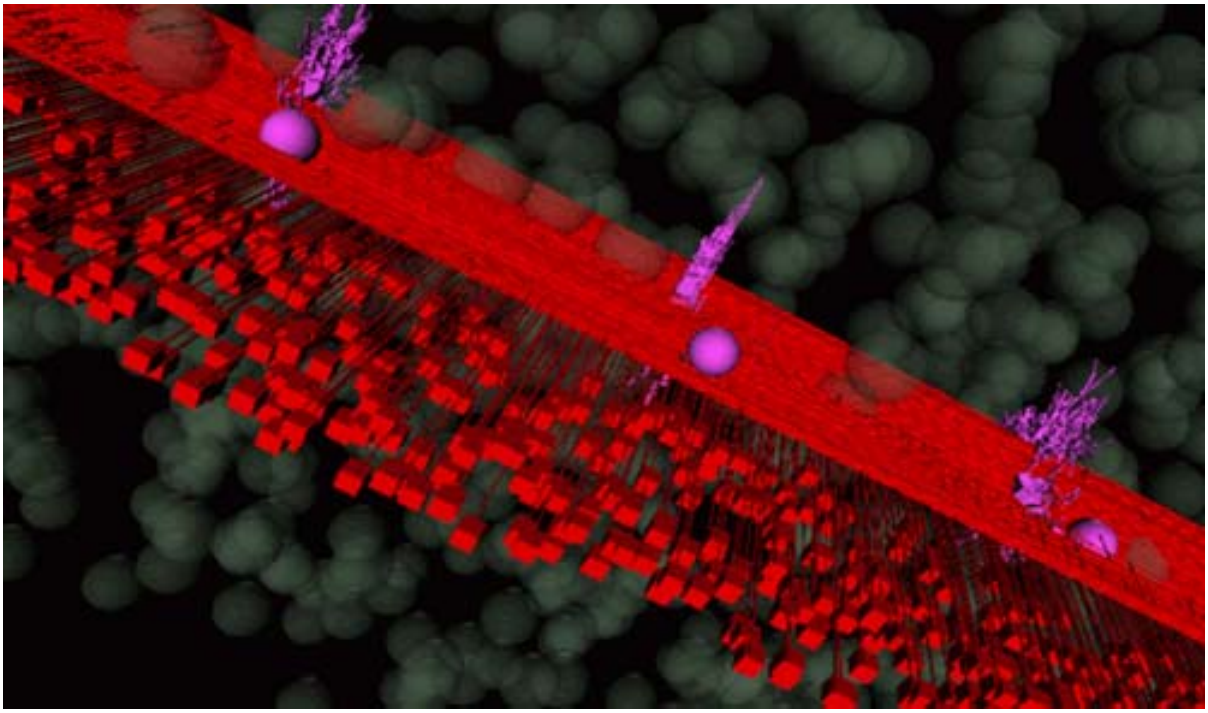


Figure 3 – Example from cerebellar cortex circuitry based on model in [2]

Why XML?

One important aspect of XML is that it provides a language-independent way to store structured information. It has achieved huge industry momentum because of its simplicity and flexibility as well as its relation to the HTML standard for web pages. It is used for representing data rather than as a programming language; because of this, using XML for model specification encourages declarative specifications rather than code. This facilitates automated transformation of model specifications to multiple other formats; programs have to be recoded by hand.

Why not XML?

There are a number of disadvantages to using XML as a representation.

- It is more cumbersome to edit by hand than programming language sources because of its redundancy. Many prefer to use it as an output file format for software tools.
- The format is verbose, and file sizes can be significantly larger than binary formats. Because of this, many tools compress the XML source.

[12] GENESIS, http://www.genesis-sim.org/

- It is slower to parse than ad hoc text formats. However, XML parsing libraries are available for every major programming language.
- It is not suitable for binary data. Although one can embed binary data in an XML file using the BASE64 coding, this defeats the original objective of creating readable files.

For the purposes of designing a common exchange format, however, the advantages of clarity and self-description outweigh the disadvantages.

Components of a simulation model

A typical simulation model in neuroscience is composed of scripts for setting up the model, parameter search routines, as well as custom code as shown in Figure 4. Often the simulation parameters are set within the code, and other researchers must read the code to find the details of the model. In order for other researchers to simulate the model, they must install the same version of the simulator, and compile any additional extensions.

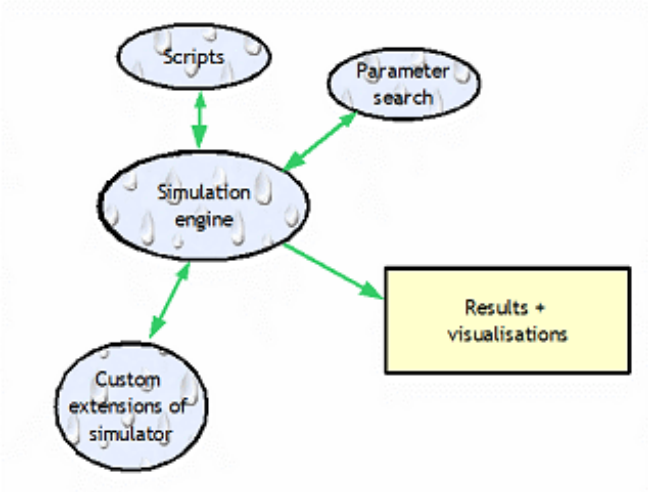


Figure 4 – Schematic of typical model

Aiming towards declarative model specifications

Ideally, the model specification and parameters would be separated from the simulation code and expressed in a declarative form rather than as a program as shown in Figure 5. This would make models easier to understand and also remove the dependence on a particular version of the simulator. It is possible for another simulator to run a declarative model, but a model expressed as a program has to be rewritten to run on another simulator.

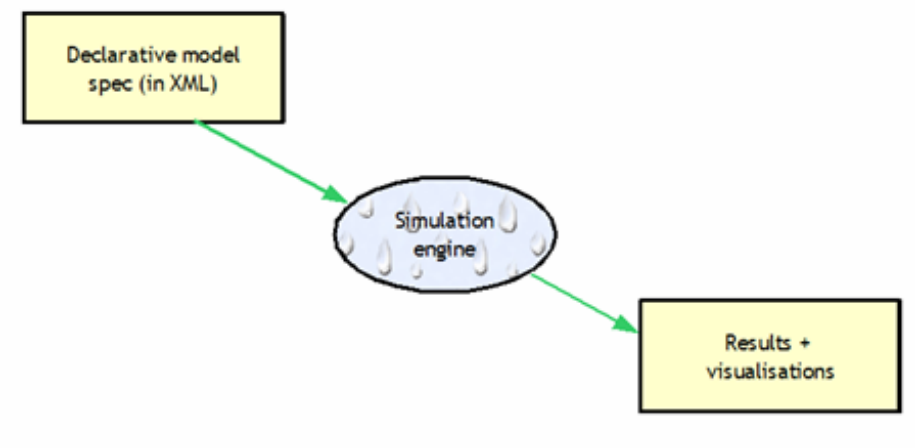


Figure 5 – Schematic of ideal model

Publishing models

Models are currently more like *programs* than *publications*; however, models expressed declaratively in XML are more likely to outlast the simulator they were written for. Robert Gentleman from Harvard has launched a *campaign for reproducible research* noting that many papers based on complex computer models do not include enough information to allow other researchers to reproduce the results; the *real* information about the model is embedded in the model code which is not often published. He notes that if published results are not easily reproducible by other researchers, the paper becomes more like an *advertisement* for the research than being the research itself. Moving model specification to XML makes the publication process simpler, but ideally the model specification should also include links to any experimental data used to derive parameters or validate results. The ModelDB[13] database is a valuable repository of published models.

Why is standardization for neural models hard?

The problem of standardizing model descriptions for neuroscience models is difficult for several reasons. As demonstrated above, neural modeling is done at a variety of levels of scale, from protein interactions to large-scale networks of neurons. In addition, different simulators have different and changing capabilities that create a moving target for attempts to create any standards.

Union or intersection?

A central design question for XML formats for neural models is whether we should attempt to address the intersection of capabilities of a number of simulators and standardize one aspect of modeling; or alternatively, if we should address the *union* of capabilities and represent all aspects of all possible models.

[13] ModelDB, http://senselab.neuron.yale.edu/

The advantage of the *intersection* approach is that it is manageable; by restricting the scope to one modeling method, it becomes possible to converge on an agreed notation for one class of models. For example, the SBML standard provides a common notation for differential equation models of intracellular pathways, and the MorphML standard provides a specification for dendritic and axonal structures in neurons. The disadvantage of the intersection approach is that it only provides for a small part of the model with the rest of the model specification coded in a simulator specific way, perhaps using script programs.

The essence of the *union* approach is for the XML model specification to be a complete description of the model - it can be used as the file format for a simulator. The advantage of this is that no additional files are needed to run the model; the XML is a complete description. The disadvantage is that attempting to define this "standard" would be a Sysiphan task. The only way it would work would be for each simulator developer to "just use XML" and to code representations in simulator-specific XML.

The *intersection* approach is most attractive for making the resulting model descriptions useable by a number of tools with overlapping functions; however, it will only encompass a subset of model descriptions which will often require supplemental code or extensions to the XML standard to actually run. Simulator developers are likely to migrate towards their own XML formats, making the union of XML formats inevitable. Thus the questions for future XML model descriptions in neuroscience are:

- Which types of models are simple enough and stable enough for a fixed XML standard to be appropriate?
- For all the other parts of a model for which no standard exists, should they be expressed in arbitrary XML or should they at least use a common method for serializing object models?

What is "NeuroML"?

The NeuroML project was started to address the issues of model specification using XML. The website includes documentation for a standard way to map object models to XML, with a sample implementation provided in a Java development kit. The emphasis is on making it easy to define any object model and to serialize it in XML. Thus NeuroML addresses the *union* problem of expressing any level of model, with the standard upgraded from "just use XML" to "use XML which represents a systematic coding of an object tree". The NeuroML Development Kit performs this systematic coding automatically so that only a single line of code is required to read or write an arbitrarily complex object tree.

Within NeuroML, the focus on defining *object models* first, rather than on defining the particular sequence of XML elements and attributes (as expressed by an XML Schema), is important. This approach is similar to the approach taken by the SBML, MAGE-ML[14], and Open Microscopy[15] teams. The problem with starting by defining XML tags first is that the resulting format requires custom code to parse. It is harder to write code that copes with arbitrary XML than it is to write generic code that deals with arbitrary object models. Important applications of such generic code are user interfaces that can cope with arbitrary object models (e.g. Catacomb by R. Cannon[16]) and parsers.

[14] MAGE-ML, http://www.mged.org/Workgroups/MAGE/mage.html
[15] Open Microscopy, http://www.openmicroscopy.org/
[16] Catacomb, http://www.enorg.org/

The problem of defining standards for particular types of models remains. The NeuroML Development Kit includes sample object models for *channel*, *cell* and *network* levels; it is likely that the most appealing subjects for standardization are the channel and single compartmental neuron specifications since the techniques used by many simulators to model at these levels are agreed and stable. However, different simulators implement network model descriptions differently. One of the problems with network descriptions is that they often depend on the detail at lower levels. For example, a simulator-independent method for specifying a connectivity rule based on receptor densities at the dendrite would require standards for all levels from networks down to channels.

Other XML languages

As mentioned in the introduction, other XML standards relevant to model specification in neuroscience include SBML for intracellular pathway models, and MathML for equations. BrainML is a candidate schema for representing experimental data. In addition, the Axiope[17] project has developed an object model editor for experimental metadata.

Practicalities

Question: I'm creating a simulator and I'd like to use NeuroML- what do I do?

- Separate out the declarative aspects of the model specification.
- Serialize the model into XML, using the NeuroML development kit (in Java) or your own code.
- If any other developers are creating similar models, see if you can agree on a common set of classes to describe the models.

The NeuroML Development Kit

The NeuroML home page includes the Java-based development kit for defining object models and for generating and reading XML. It uses a technique called data binding for automating the process of generating and parsing XML from objects in memory. This reduces the programmer burden of reading arbitrarily complex XML model descriptions to using a single line of code and makes defining object models as simple as defining classes. The resultant XML uses class and field names, but contains no Java-specifics (unlike the basic serialization routines provided with Java and other programming languages such as Python).

- Your program can read in an XML document using:

```
Object o = XMLIn("file.xml");
```

- And write one using:

```
Object o = new MyComplexStructure();
XMLOut(o,"file.xml");
```

[17] Axiope, http://www.axiope.com/

Example network definition

This example taken from the NeuroML Development Kit defines a "Network" as having a set of elements (which can be neurons or other networks), and a set of projections.

```
package neuroml.model.network;
import neuroml.core.*;
public class Network extends Element {
    /** A network has a set of elements - can be populations or
        individual cells*/
    public Set elements = new Set("ElementRef");
    /** A network also defines a set of projections between
        elements   */
    public Set projections = new Set("Projection");
}
```

The example below defines a simple 3D grid for the structure of a population of cells - just x, y and z sizes.

```
public class Grid3DStructure extends PopulationStructure {
    public int xsize=1;
    public int ysize=1;
    public int zsize=1;
}
```

Schema format

The development kit uses a restricted subset of Java as a schema definition language. This includes:

- Simple types: int, double, string
- Collections, references
- Classes and inheritance

The created schema is provided in alternative formats such as XML-Schema and UML diagrams using reflection. The choice of schema format is less important than the choice of starting with an object model and generating the XML in a systematic fashion. The NeuroML website includes details on the XML coding chosen by the development kit.

The place of embedded scripts in model descriptions

Scripts are extremely convenient for coding loops for running simulations and for setting up ad hoc connectivity patterns, but including code in a model description makes that description less portable. It is likely that cutting edge models will always include an element of scripting. Thus the aim should be to express as much as possible of a model description declaratively, but accept that some models will always be programs.

Future plans

Currently, many simulators are adopting their own XML formats for serializing model descriptions. As outlined above, common standards are working well where the domain is stable and agreed, but questions remain concerning how much standardization is useful for other domains. It is likely that the next modeling domain where standards will be addressed will be in the development of ChannelML for specification of ion channel models and NeuronML for single cell modeling. More experience with formalisms for building network models will be needed before a NetworkML specification will be feasible.

NeuroML for Model Specification in ChannelDB and GENESIS - *David Beeman*

Introduction

ChannelDB[18] (Beeman an Bower 2004) is an implementation of a database of ionic conductance models (channels) to be used in biologically realistic neuronal simulations. When choosing a representation for such models, a problem arises: **the procedure for constructing a single model can be described in many different ways.**

Although a model may be unambiguously specified by a set of equations and parameter values, it is the function of a simulation script to tell the simulator how to implement the model. Because of differences in simulator design, simulation scripts for GENESIS (Bower and Beeman 1998) and NEURON[19] (Hines and Carnevale 1997) will look quite different from each other. Consequently, it is very difficult to convert a simulation written for one simulator to one for another.

The solution to this problem will be obvious to the participants of this workshop: **use XML to establish a standard format for a declarative representation, NOT a simulator-dependent procedural representation.**

An example

As an example, consider the set of equations used to describe the Hodgkin-Huxley model of the potassium conductance found in the squid giant axon (Hodgkin and Huxley 1957).

$$G_k = g_K n^4$$

$$\frac{dn}{dt} = \alpha_n(V)(1-n) - \beta_n(V)\,n$$

[18] channelDB, http://www.modelersworkspace.org/channeldb/ChannelDB.html
[19] NEURON, http://www.neuron.yale.edu/

$$\alpha_n(V) = \frac{0.01(10-V)}{exp\left(\frac{10-V}{10}\right)-1}$$

$$\beta_n(V) = 0.125\ exp(-V/80)$$

Here, the conductance G_K of this channel is given in terms of a gating variable n that satisfies a first-order differential equation involving two voltage-dependent rate variables. Three possible ways to specify the rate variables for this model would be to

- Represent the equations in a form that can be parsed into statements that can be evaluated by a simulator or a by computer language such as Java or C++.
- Store tabulated values of the rate variables.
- Use a parameterized form (A + BV) / (C + D exp((E + V)/F)) and store only the six parameter values.

The ChannelDB solution

The ChannelDB implementation allows all three of the representations above, and is based on the XML format used in the NeuroML model description language (Goddard et al. 2001). The main features are:

- XML representation of a Java Hodgkin-Huxley object with attributes for maximal conductance, and a set of gates and their exponents.
- Gate objects have an attribute indicating dependence on voltage or an ion concentration, and objects for the forward and backward rate variables.
- The NeuroML development parser converts between XML channel representations and Java objects.
- Simple Java string manipulation commands are used to produce a simulation script from information in the fields of the DBChannel object.
- The prototype database and interface creates commented GENESIS scripts from stored XML channel descriptions.

NeuroML representation of the K channel

This procedure results in the following XML representation for the Hodgkin-Huxley model potassium conductance:

```
<neuroml class="DBChannel" description="Hodgkin-Huxley squid K
 channel"
 author="Dave Beeman"
 keywords="Hodgkin-Huxley potassium squid delayed rectifier"
 uniqueID="10262778758662F22@dogstar.colorado.edu"
 notes="An implemention of the GENESIS K_squid_hh channel"
```

```
 Erest="-0.07V">
   <channels>
      <channel name="K_squid_hh" class="HHChannel"
       permeantSpecie="K" Erev="0.09V" Gmax="360.0S/m^2"
       ivlaw="ohmic">
         <gates>
            <gate name="X" class="HHVGate" timeUnit="sec"
            voltageUnit="V"
            vmin="-0.1" vmax="0.05"
            instantCalculation="false" useState="false" power="4">
              <forwardRate class="ParameterizedHHRate" A="-600.0"
               B="-10000.0" C="-1.0" D="1.0" E="0.060" F="-0.01"/>
              <backwardRate class="ParameterizedHHRate" A="125.0"
               B="0.0 C="0.0" D="1.0" E="0.07" F="-0.08"/>
           </gate>
         </gates>
         <log author="Dave Beeman" date="Jul 9, 2002 11:11:15 PM"
               literatureReference="A.L. Hodgkin and A.F. Huxley,
                J. Physiol. (Lond) 117, pp 500-544 (1952)">
         </log>
      </channel>
   </channels>
</neuroml>
```

Some classes defined for ChannelDB

NeuroML provides a number of templates, or classes, that may be used for this description. In the example above, the K_squid_hh channel is derived from the HHChannel class, which has certain properties such as a reversal potential Erev, and a conductance density Gmax. It also possesses a voltage activated Hodgkin-Huxley gate, derived from the HHVGate class. The gate contains objects representing the forward and backward rate variables. In this case, the equations for the rate variables are represented with a parameterized form, derived from the ParameterizedHHRate class. Other objects and attributes specify descriptive information about the model that would be useful in a database search.

Some of the NeuroML classes used with ChannelDB are:

DBChannel: Wrapper class that is used to contain any channel model that is stored in ChannelDB, along with some descriptive information.

HHChannel: Class used for all the Hodgkin-Huxley type channels in the database.

HHVGate: Used as a member of the gates set of a HHChannel. It contains forward and backward rate objects that depend on voltage, as well as some additional fields to describe the gate.

HHCGate: An ion concentration-dependent gate, analogous to the voltage-dependent HHVGate. It provides an additional field for a reference to the object that provides the source of the ion concentration.

HHRate: A superclass for the specialized forms for the rate variables.

ParameterizedHHRate: A subclass of HHRate that expresses rate variables in a parameterized form typical of many Hodgkin-Huxley type rate equations where rate equation is (A + BV) / (C + D exp((E + V)/F)).

EquationHHRate: A subclass of HHRate that expresses the rate variables as equations.

TabulatedHHRate: A subclass of HHRate that allows a gate's forwardRate or backwardRate to be specified by a table at equally spaced voltage (or ion concentration) points.

ConcenPool: Describes a single shell model for a ion concentration pool with a buildup of concentration proportional to an incoming current and a time constant for decay. The object providing the source of concentration to a HHCGate is typically formed from this class. The source of currents is provided by a set of objects of class CurrentSource.

CurrentSource: Used by ionic concentration pools to provide information about the object that provides an ionic current.

Unfinished business and open questions

There is more yet to be done to establish a complete simulator-independent XML representation for ion channel models. It is hoped that this workshop will provide a start for a cooperative effort to address some of the following issues:

- Extend NeuroML to provide representations for more detailed multi-shell models of calcium diffusion.
- Implement a more sophisticated representation of literature references than the simple string that is currently used in the NeuroML software. (We have proposed a schema for the Modeler's Workspace based on BibTeX.)
- Create software to convert ChannelDB descriptions to NEURON and other simulator formats.
- Implement the HHCVGate, a two-dimensional gate depending on both voltage and concentration. (Note that the Traub Ca-dependent K channel model uses a form that can be expressed as a product of a HHVGate and a HHCGate.)
- Implement Borg-Graham or Lytton-Sejnowski temperature-dependent channel models with the NeuroML ThermodynamicHHVGate.
- Is there a better way for a concentration-dependent channel model to reference the models that provide the source of ionic currents and concentrations?
- NeuroML allows any string to be used for the equation used in the EquationHHRate. How much standardization should there be for the equation format and for the names of the independent variables and parameters?

NeuroML and GENESIS 3 development plans

The GENESIS simulator is undergoing a major redevelopment effort, and will use an XML representation for model specifications. Standards that will emerge from this workshop will play an important role in GENESIS version 3.

Figure 6 shows the components of GENESIS 2 as seen by the GENESIS user. Although GENESIS and its scripting language produce modular object-oriented simulations that are easy to modify and extend, this is not the case with the source code for GENESIS 2. The core modules of GENESIS were written in the late 1980's in C, and the modules are much more tightly coupled than the diagram would suggest.

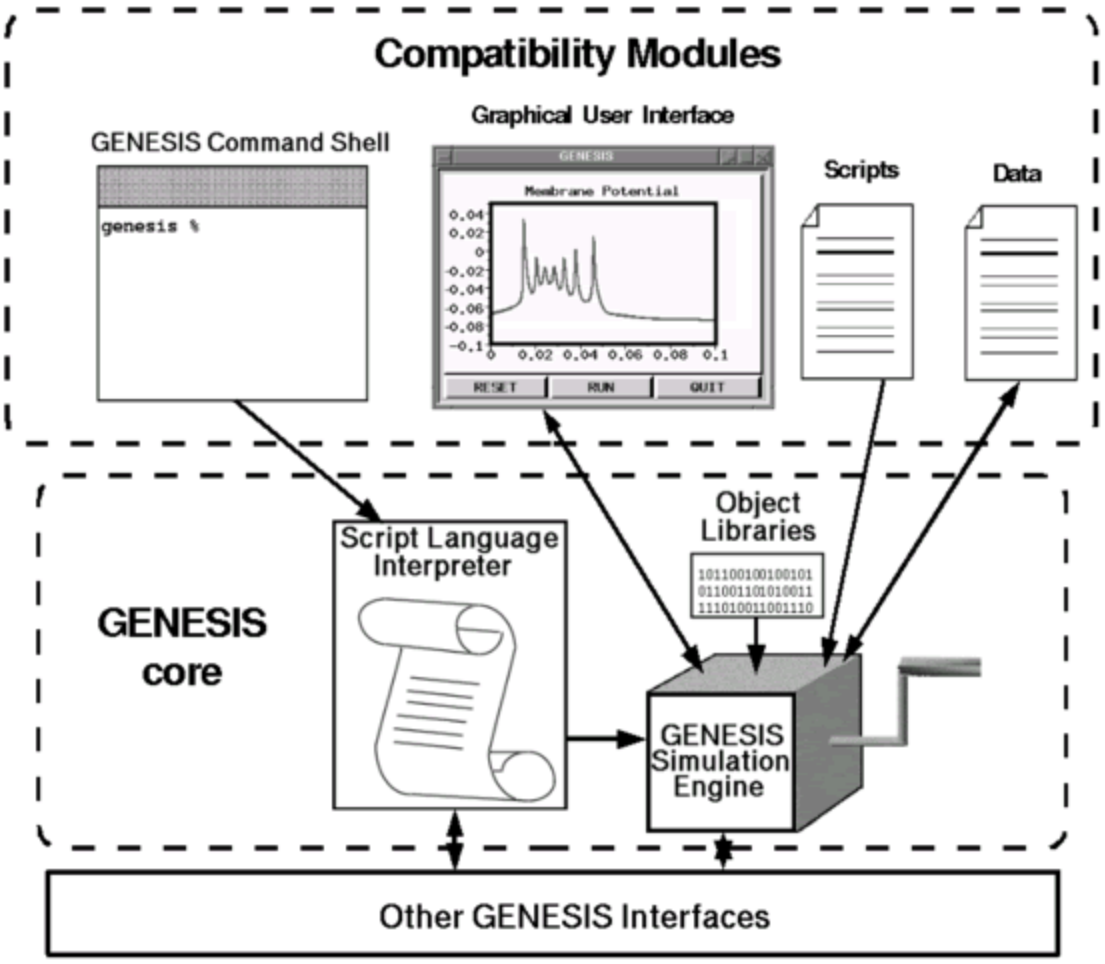


Figure 6 – Components of GENESIS 2

Upinder Bhalla of the National Centre for Biological Sciences in Bangalore, India wrote the X Windows XODUS GUI interface for GENESIS and did much of the initial GENESIS development. He has been rewriting the GENESIS base code in C++ as the Messaging Object Oriented Simulation Environment

(MOOSE)[20]. This reimplementation in an object-oriented language makes it possible to "unbundle" the components of GENESIS as shown by the dashed lines, and to create other interfaces to the GENESIS core.

MOOSE provides:

- Improved Messaging between GENESIS objects
- Faster, smaller, cleaner implementation
- Portability to MS Windows and non-UNIX platforms
- Improved equation solvers
- Ability to use multiple script parsers and user interfaces

GENESIS 3 will add:

- A modern graphical user interface
- XML representation of models
- Backwards compatibility with GENESIS 2
- Tutorials and educational applications

Figure 7 shows how using the MOOSE core for GENESIS 3 will allow multiple external interfaces to GENESIS 3.

There are two main priorities for GENESIS 3 development. The first is to re-implement the X Windows graphical user interface of GENESIS in Java. This will allow the use of more modern user-friendly interfaces and will allow GENESIS to be used on non-UNIX platform such as Windows. A Java GUI for GENESIS will also allow tutorial simulations to be run over the Internet from a GENESIS server. The second is to develop XML-based representations of channel, cell, and network models. Although the present GENESIS cell parameter file format will continue to be supported for backwards compatibility, we plan to use a standard simulator-independent XML model description to import models into GENESIS. This will make it possible for modelers to more easily exchange simulations and simulation components, regardless of the simulator used.

[20] MOOSE – Messaging Object Oriented Simulation Environment, http://www.genesis-sim.org/GENESIS/newgrant/MOOSE-plans.html

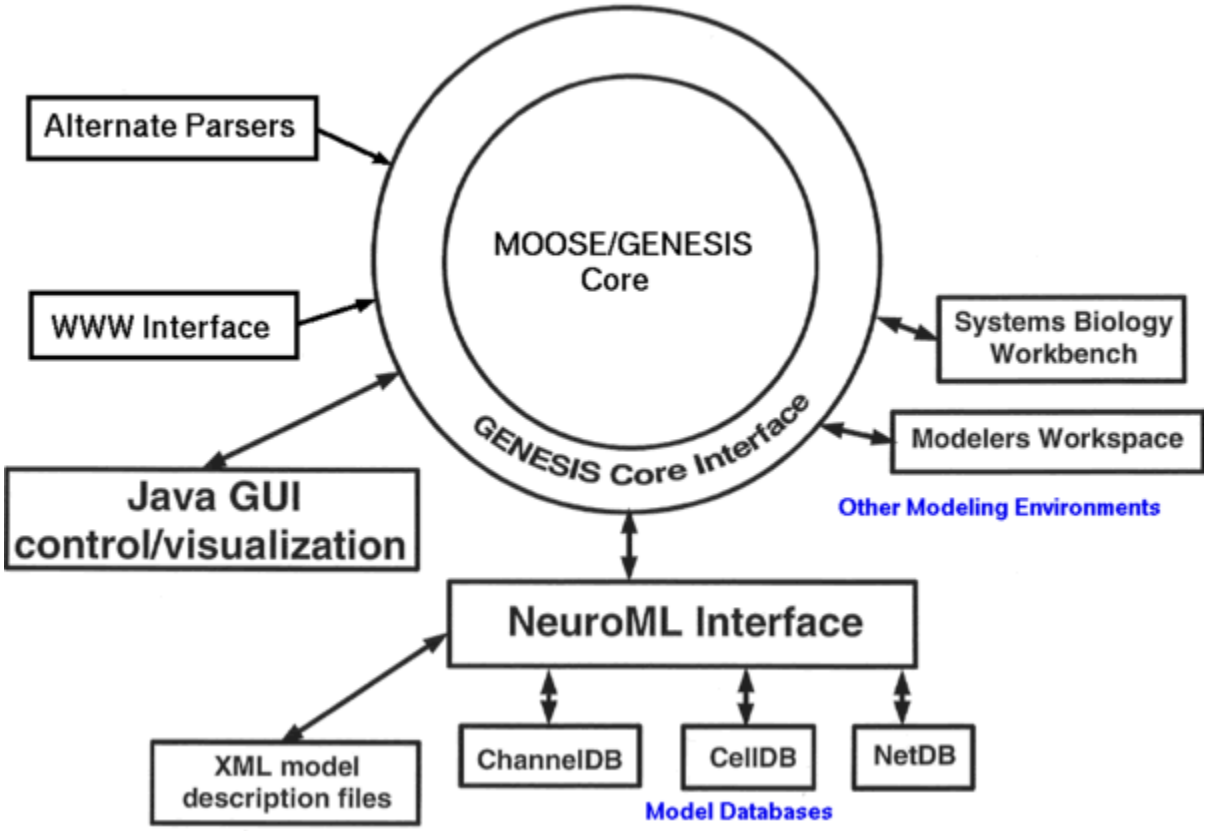


Figure 7 - External interfaces to GENESIS 3

MorphML: An XML Application for Neuronal Morphology Data - *Sharon Crook*

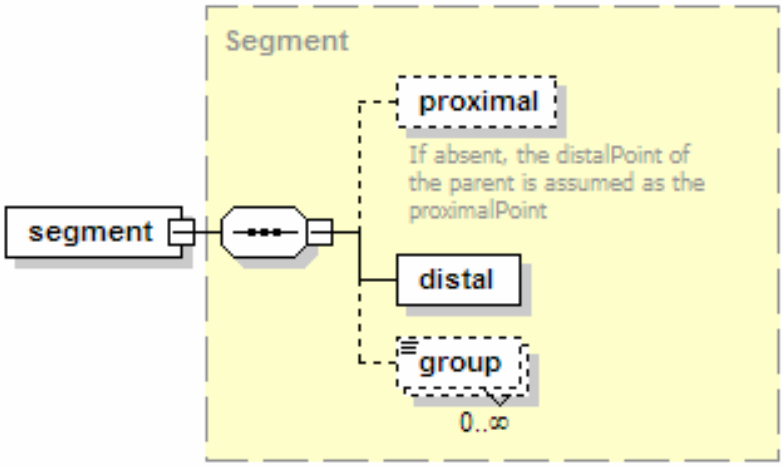
Introduction

Neuronal morphology data are important for the study of many areas of neuroscience including development, aging, and pathology and are crucial for the development of structurally accurate compartmental models (Cannon et al. 2002). The complexity of problems in neuroscience often requires that research across disciplines be combined. However, existing neuroanatomical data have been obtained using many different neuron tracing systems. Generally, these systems represent neuron arborizations using a collection of points, diameters and connections in three dimensions.

MorphML was developed as a common data format for neuronal morphology data. It can be used separately from NeuroML as a stand-alone XML application, but it can also be used as a representation for neuron morphology for compartmental models described using NeuroML. In

particular, MorphML is intended to facilitate data archiving, data and model exchange, database creation, and data and model publication for morphology data (Qi and Crook 2004).

element **Cell/segments/segment**



namespace	http://morphml.org/morphml/schema/1.0.0
type	**Segment**
children	**proximal distal group**

attributes	Name	Type	Use
	id	xsd:nonNegativeInteger	required
	name	xsd:string	optional
	parent	xsd:nonNegativeInteger	optional
	cable	xsd:nonNegativeInteger	optional

source: `<xsd:element name="segment" type="Segment" maxOccurs="unbounded"/>`

Figure 8 – Example of a MorphML element

An example

The MorphML language elements describe objects used to represent neuron cell bodies, branching dendrites and axons, spines and varicosities, fiducials, etc. An example of XMLSpy generated documentation for the language element for a *segment* is shown in Figure 8. Segments are connected to represent branching structures, where each *segment* is a truncated cone or frustum. The geometry of a segment is determined by the cross-sections at the ends of the segment. Each cross-section is defined by the 3D location (x, y and z) of the center point as well as an associated *diameter*. Note that some applications require that the diameters at the opposite ends of a *segment* have the same value so that the segments are all cylinders.

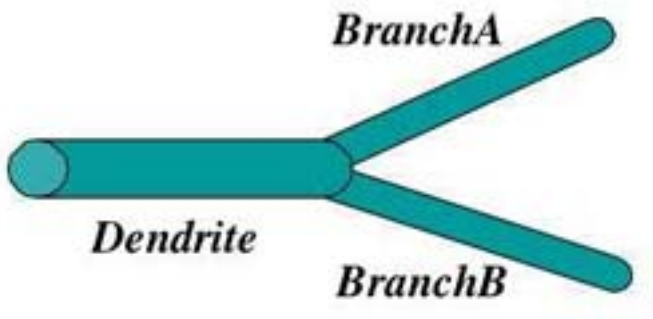


Figure 9 – Schematic of branching structure

An example of a branching structure is shown in Figure 9. The XML fragment below provides a representation for this branching structure in MorphML.

```
<segments>
 <!--   Segment: Dendrite, ID: 1-->
 <segment id="1" name="Dendrite">
       <proximal x="0" y="0" z="0" diameter="4"/>
       <distal x="16" y="0" z="0" diameter="4"/>
 </segment>
 <!--   Segment: BranchA, ID: 2-->
 <segment id="2" name="BranchA" parent="1">
       <proximal x="16" y="1" z="0" diameter="2"/>
       <distal x="32" y="9" z="0" d="2"/>
 </segment>
 <!--   Segment: BranchB, ID: 3-->
 <segment id="3" name="BranchB" parent="1">
       <proximal x="16 y="-1" z="0" diameter="2"/>
       <distal x="32" y="-9" z="0" diameter="2"/>
 </segment>
</segments>
```

Applications of MorphML

Currently, MorphML is being used as the format for data and model specification in a number of software applications. One example is the Virtual Ratbrain Project[21] which includes a database for peer reviewed 3D cellular anatomical data of the rat brain as well as visualization and analysis tools. In this project, data is stored in MorphML format, and one of the available tools is a MorphML Viewer. NeuroConstruct[22], which is introduced below, is another application using MorphML. In addition, both the NEURON simulation environment and GENESIS are including the MorphML format in future developments.

[21] Virtual Ratbrain Project, http://www.ratbrain.org/
[22] NeuroConstruct, http://www.neuroconstruct.org/

The practical use of XML specifications- *Padraig Gleeson*

Introduction

XML has emerged over the past few years as the de-facto standard for information exchange on the Internet. Data are formatted so that a target application knows exactly where each piece of information can be found. In the field of neuroscience too, this way of exchanging information can be useful, as the number of published models illustrating different aspects of the functioning of the nervous system grows. Many papers contain detailed information on cell morphologies, connectivity, the mechanics of voltage-gated ion channels, and the distribution of channels on neuronal membranes. Currently this information can be published in tabular form in the paper itself, or in accompanying files in one of a number of different simulator formats. Model data from these papers, which might be reusable for other models, can be extracted from the files only if the syntax of the simulator scripting language is known and the scripts are well written.

The NeuroML, MorphML, and ChannelML initiatives seek to codify this knowledge to make it easier to publish data in a form that will be readily usable by the neuroscience community in general. We describe here how a computational neuroscience application might use models published using these specifications.

Linking NeuroML, MorphML, and ChannelML specifications

Neuroscientists are interested in relating experimental findings at the cellular and sub-cellular levels to the overall behavior of neurons. Models that are developed using experimental anatomy and electrophysiology data are often published, and ideally this information should be easily accessible for a wide range of scientists to analyze and reuse for their own areas of study. Any application for the general neuroscience community should present the concepts dealing with cell morphology and cell processes in a form that is familiar for neuroscientists and should leave the computational details in the background as much as possible. An intuitive user interface for creating the cell models would also increase the appeal of such an application.

Cell morphology

Many computational neuroscience models containing detailed morphological reconstructions of cells like that shown in Figure 10 have been published in recent years. See for example (De Schutter and Bower 1994) (Poirazi et al. 2003). The cell morphology data are usually provided in the format associated with the simulation environment, and such biophysical parameters as channel densities are usually embedded in the file that specifies the morphology. This simulator-dependent format makes it quite difficult to incorporate a cell model developed in NEURON, for example, with a cell that has been developed for the GENESIS environment. Separation of the morphology from the channel distribution would allow for greater flexibility for different investigators to study cell function under different electrical conditions, for example with different channel densities or additional membrane conductances.

Figure 10 – Purkinje cell (from De Schutter and Bower 1994)

Cell processes

Electrophysiological processes such as voltage-gated ion channels and synaptic mechanisms on the membranes of neurons are responsible for cell firing behavior. These processes can be represented by systems of differential equations that are solved by the various simulation platforms. The implementation of these simulations can be quite complex, and the scripts are difficult to read for anyone not familiar with the syntax of the simulator scripting language. However, most neuroscientists are familiar with the main elements of the cell process models such as the Hodgkin-Huxley formalism for the differential equations, the maximum conductance for a channel, etc.

Figure 11 demonstrates how the XML representation for the important aspects of a model might be handled. Templates are created for important cellular processes common to many computational neuroscience models such as double or single exponential synaptic mechanisms or Hodgkin-Huxley type channel mechanisms. The essential parameters that take the model from a theoretical framework to an instance describing a specific experimentally investigated process are required to complete the template. Documentation should make the required fields underlying the model formalism unambiguous. Different sets of parameters lead to differing instances of similar cellular processes. The XML file containing the data, formatted according to the template, can then be published, and it is instantly accessible to any application familiar with the file format. A mapping of the model to a number of simulation environments can be associated with each model. This mapping should be well tested and should represent an implementation of the model in line with the best practices of the simulation environment.

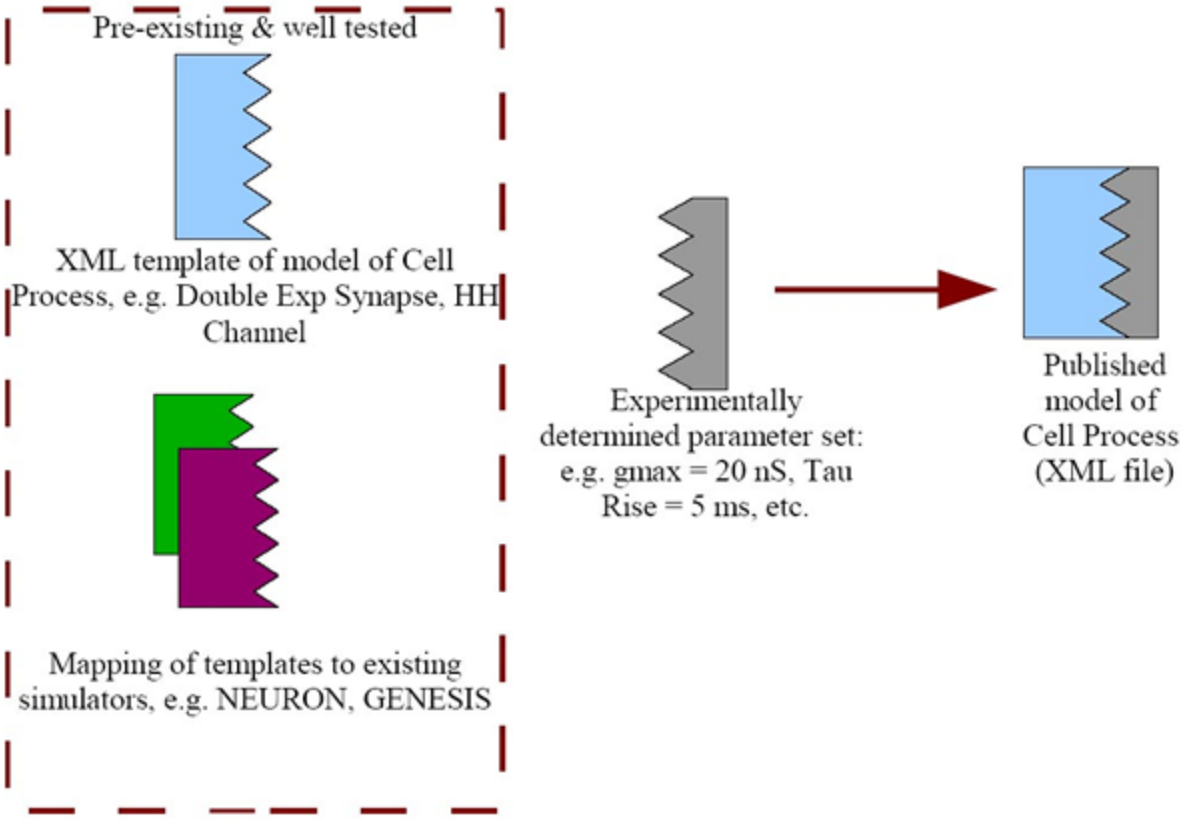


Figure 11 – Schematic of XML model representation

Investigators need an application for gathering these published models together, linking cellular processes with cell morphology, mapping the cell models into one or more of the native simulation scripting languages, and running the simulations.

ISSN: 1861-1680

neuroConstruct: An application to facilitate handling of XML specifications

The Silver Laboratory in the Department of Physiology at University College London is developing an application, neuroConstruct, that will facilitate the linking of the XML model templates as described above. neuroConstruct features a 3D interface for visualizing cell morphologies and imports morphology data in various existing formats, including MorphML. Channel mechanisms can be placed on cells to form biologically realistic neuronal models. Networks of these cell prototypes can then be constructed, and the final model can be mapped to various simulation formats such as GENESIS or NEURON for execution. Development of neuroConstruct is taking place in parallel with the development of XML standards and provides an example of how real applications can make use of the emerging standards.

Current Issues and Future Development: A Workshop Summary - *Sharon Crook*

After much discussion, one of the items of consensus was that it is worthwhile for the neural modeling community to create a common data format under the umbrella of NeuroML for many of the most widely used model formalisms. Because researchers focus on anatomical, physiological, or modeling characteristics at different levels of scale (e.g. sub-cellular, cellular, circuit and system levels), the easiest way to develop NeuroML will be to create several domain-specific description languages such as MorphML and ChannelML and then link them through NeuroML. This modular approach will also help potential users understand the underlying schema and will make it easier to take advantage of existing XML applications. For example, BrainML has elements for describing experimental electrophysiological data that could be extended to incorporate descriptions for model-generated data. Other existing XML applications might be used for describing models of sub-cellular biochemical reactions or other cellular processes (SBML or CellML). NeuroML will define logical connections among these independent domains as needed.

In order to follow up on this idea, working groups were created for further development of NeuroML core technology and for further development of specifications for cell morphology data, for models of ion channels, for channel distributions, and for cell connectivity. These working groups will function through e-mail lists administered through a SourceForge development website for NeuroML[23]. This website will also facilitate version control, development of documentation, and code distribution.

[23] SourceForge development website for NeuroML, http://sourceforge.net/projects/neuroml

ISSN: 1861-1680

References

Beeman D., and Bower J. M. (2004) Simulator-independent representation of ionic conductance models with ChannelDB, Neurocomputing 58-60: 1085-1090.

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

Cannon, RC, FW Howell, NH Goddard, E De Shutter (2002) Non-curated distributed databases for experimental data and models in neuroscience. Network: Computation in Neural Systems 13, 415-428.

Collins M.O., Yu L., Coba M.P., Husi H., Campuzano I., Blackstock W.P., Choudhary J.S., and Grant S.G. (2005). Proteomic analysis of in Vivo phosphorylated synaptic proteins. J Biol Chem 280:7.

Goddard N., Hucka M., Howell F., Cornelis H., Shankar K., and Beeman D. (2001) Towards NeuroML: Model Description Methods for Collaborative Modelling in Neuroscience, Philosophical Transactions of the Royal Society B. 356: 1209-1228.

Hines M. and Carnevale N. T. (1997). The NEURON Simulation Environment. Neural Computation 9: 1179-1209.

Hodgkin A. and Huxley A. (1952) A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. (London). 117: 500--544.

Howell F., Dyhrfjeld-Johnsen J., Maex R., De Schutter E., and Goddard N. (2000) A large scale model of the cerebellar cortex using PGENESIS. Neurocomputing 32:1041-1046.

Poirazi P, Brannon T, Mel BW. (2003) Pyramidal neuron as two-layer neural network. Neuron 37: 989-99.

Qi, W and SM Crook (2004) Tools for neuroinformatic data exchange: An XML application for neuronal morphology data. Neurocomputing 58-60C, 1091-1095.

De Schutter E. and Bower J.M. (1994) An active membrane model of the cerebellar Purkinje cell. I. Simulation of current clamps in slice. Journal of Neurophysiology 71: 375-400.