Linking computational neuroscience simulation tools - a pragmatic approach to component-based development [1]

F. Howell (fwh@anc.ed.ac.uk) [a]
R. Cannon (cannon@anc.ed.ac.uk) [a,b]
N. Goddard (Nigel.Goddard@ed.ac.uk) [a]
H. Bringmann (hbringma@uos.de) [a]
P. Rogister (paulro@anc.ed.ac.uk) [a]
H. Cornelis (hugo@bbf.uia.ac.be) [b]

[a] *Institute for Adaptive and Neural Computation, Division of Informatics, University of Edinburgh, 5 Forrest Hill, Edinburgh EH1 2QL, Scotland*

[b] *Theoretical Neurobiology, Born-Bunge Foundation, University of Antwerp, Universteitsplein 1, 2610 Wilrijk, Belgium*

Abstract

Many problems in computational neuroscience require sophisticated software systems that are beyond the development scope of a single individual or research group. Realizing such systems with minimal redundant effort requires cooperation among software develpers and the adoption of design strategies and technology from the software industry.

We are working on neuroscience-specific frameworks for modeling tools aimed specifically at maximising the benefits from of investment in software development by encouraging the reuse of software components and at facilitating model development by establishing shared formats for model description.

The techniques employed include component technology for coupling parts of applications, XML file standards based on NeuroML for model and data exchange between applications and databases, and peer-peer web based indexing of models and modules.

This paper describe progress to date in the modularization of simulation and analysis functions from NEURON, Catacomb and NEOSIM.

Key words: Neural simulation tools, modelling, simulation, software

[1] Supported by: NIH (MH-57358), FWO Belgium

1 Introduction

Besides its scientific goals, computational neuroscience also presents significant software engineering challenges. Modeling biological systems at the range of levels now demanded will eventually require tools ranking amongst the most advanced software systems in use.

The development of such systems is clearly beyond the resources of individual research groups, and therefore requires technology which allows each group to make maximum use of development work elsewhere. Likewise, model development would benefit substantially from the sharing and reuse of models between research groups.

The software industry is facing closely related problems in the development of software for commercial and industrial applications, and is generating a continual stream of new technology to help tackle them. We are harnessing this work in the neuroscience context to allow software developers and modelers to carry on working in their own area of expertise, but to ensure that the results of this work are more widely accessible.

2 Current state of multi-user neuroscience software

The choices facing a researcher interested in modelling at some level of computational neuroscience are to use a simulation system such as NEURON [5] or GENESIS [1] aimed at a particular type of modelling, or write their own software from scratch, frequently implemented in C, C++, Java or Matlab.

A critical issue is what researchers do when a software tool *nearly* does what is required. For example, if one would like to run a network model in NEURON but to use a feature not yet available, such as the display of a clickable connectivity graph showing when particular links are active, then the choices at present are either to ask the author to implement it or to write one's own simulation code from scratch which includes the particular display feature.

It would be more efficient if the researcher were able to write a module for the new feature in a simulator-independent way, couple it to an existing simulation environment, and make the module available for others. At present there are numerous practical barriers to this kind of component based development which must be addressed before modular software development and reuse becomes the norm in neuroscience.

3 Working modules

There are already a number of areas where self-contained modules have been shown to work; NEURON `.mod` files for mechanisms and templates for defining reusable cells; GENESIS packages and messages, which allow a way to append user code, and the NEOSIM event interface which allows linking modules via a restricted send/handle spike interface.

Our aim is to extend the range of situations in which module-based technology can be used, to include interfaces for modules to perform:

- Synaptic dynamics
- Ion channels
- Artificial / ball and stick / reconstructed compartmental neurons
- Projection patterns
- Structured populations

The systems biology community is faced by similar modeling problems, and has come up with the Systems Biology Workbench [12](SBW) as a partial solution. They have a rather different starting point, with several highly developed modeling systems in widespread use and covering broadly the same problem domain. Their solution has been to implement a loose coupling between these applications based on message passing. This works well from the point of view of the user, who can run the same model under different systems, but does not reduce the software engineering requirements because each system still must be independently maintained.

4 Module technologies

Modularisation and reusable software components are widely used to develop business software, with a number of technologies available, e.g. CORBA [7], RMI [8], JavaBeans [9], .NET [13]. In the scientific community, as well as the SBW mentioned above, there is a recent effort at LLNL [6] focussed on components for parallel computations.

The approach we have adopted is based on Java archive files (.jar) to package modules into self contained units, JNI for C++/Java coupling, and the NeuroML dialect of XML for model / parameter specification.

The precise choice of component technology is less significant than making the step from developing monolithic applications to developing components. Once modules are available in some format, software bridges can be built to

link the different component technologies.

5 Example interfaces

We have developed a number of interfaces for specific types of module, including `RunnableCell`, `PreSynapse`, `PostSynapse`, `Projection`. For example, the RunnableCell interface includes a few methods which a module which can receive spikes can implement:-

```
package neuroml.sim.run;
interface RunnableCell {
  void handleSpike(double t, int inputConnection);
  void advance(double t);
  void reset();
}
```

Modules which implement such an interface can be developed by different people, and loaded on the fly by a module aware simulation package. For more documentation see the websites `www.compneuro.org/javadoc` and `www.neuroml.org`. Figure 1 illustrates a number of modules linked using the Catacomb workbench.

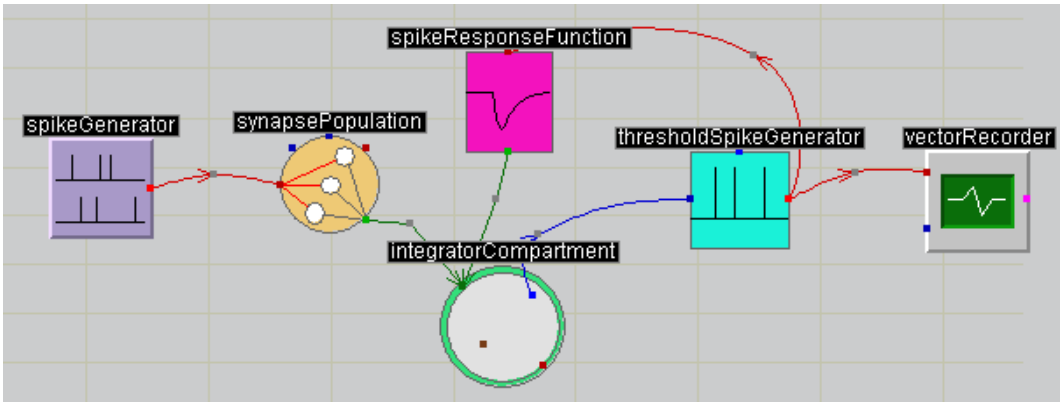

Fig. 1. A Catacomb cell model connecting components for spike generation, integration and plotting. Moving to distributed development means that modellers can develop new component types and link them to the system.

6 Model descriptions

For simulation model descriptions, we use NeuroML [4], an XML declarative language. This has the advantages of being extensible, easy for software and people to read, and it leverages the industry investment into XML technology.

Describing models in XML rather than a programming language also separates out the key parts of the model from the details of its implementation.

The key technology which is part of the NeuroML development kit is *data binding*, to allow data in an object tree (within a simulator) to be transparently mapped to/from a NeuroML file without having to implement a parser. The schema for the NeuroML sublanguage is defined using a set of class definitions, or in XML. We have also used the NeuroML development kit within a viewer for the CoCoMac connectivity database [10].

7 Peer-peer exchange

One consequence of moving from a few monolithic simulation packages to a scenario of a large number of component developers is that it becomes essential to provide tools that allow people to search for and publish modules. To address this issue we have developed peer-peer web based exchange and indexing tools (`www.neuroml.org/napster`) based on the Napster model of distributed content (people keep modules/models on their own website which they can customise and control), and centralised indexes (so it becomes simple to perform a global search for modules or models, either from a website or an application).

We specifically avoided a centralised database because of the perceived transfer of ownership and control which takes place when items are submitted to such databases. Despite efforts to the contrary, some degree of control inevitably shifts away from the model or module creator towards the database host, a situation which is very unappealing for those investing in software development.

8 Conclusions

We have begun the process of moving from monolithic software packages to collections of modules. The parts of the puzzle we have implemented include a stable model description language (NeuroML), working Java and C++ based mechanisms for module interaction (.jar files, interfaces, JNI), and a peer-peer system for publishing and finding out about modules and models.

The process has to be evolutionary rather than revolutionary because of the large amount of time and effort invested in existing general simulation packages and the models which depend upon them. We are currently working on incrementally adding interfaces to more simulation packages, with the ultimate

aim of making it easier to develop, test, distribute and explore sophisticated models of circuits and cells.

References

[1] J.M. Bower, D. Beeman (eds), The Book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System, 2nd ed. (Springer-Verlag, New York, 1998).

[2] R. Cannon, Catacomb, `www.compneuro.org/catacomb`.

[3] N. Goddard, G. Hood, F. Howell, M. Hines, E. De Schutter, NEOSIM: Portable large-scale plug and play modelling, Neurocomputing 38 (2001), 1657-1661.

[4] N. Goddard et al, Towards NeuroML: Model description methods for collaborative modeling in neuroscience", Phil. Trans. Roy. Soc, series B, (Aug 2001), vol 356, issue 1412.

[5] M.L. Hines, N.T. Carnevale, The NEURON simulation environment, Neural Comput. 9 (1997), 1179-1209.

[6] S. Kohn, Scientific component technology at LLNL, `www.llnl.gov/CASC/components`

[7] OMG, CORBA, `www.corba.org`

[8] Sun Microsystems, RMI, `java.sun.com/products/jdk/rmi`

[9] Sun Microsystems, JavaBeans, `java.sun.com/products/javabeans`

[10] R. Kotter et al, CoCoMac, `www.cocomac.org`

[11] R.C. Cannon, F.W. Howell, N.E. Goddard, E. de Schutter, Non-Curated distributed databases for expwerimnetal data and models in neuroscience, (*submitted to Network: Comput. Neural Syst. 2002*)

[12] M. Hucka et al, SBML/SBW, `www.cds.caltech.edu/erato`

[13] Microsoft, .NET, `www.microsoft.com/net`