Multi-level Neural Modelling: An Application of Object-oriented Software Engineering

[*]F. A. Henskens, [!]P. J. Johnston, [*]W. McGowan

[*]Department of Computer Science & Software Engineering
[!]Centre for Mental Health Studies
University of Newcastle
N.S.W. 2308
email: henskens@cs.newcastle.edu.au, pjohnsto@mail.newcastle.edu.au, bill_mcgowan@impulseairlines.com.au

KEYWORDS:

Object-oriented, modelling, neuroscience, applications

ABSTRACT

Neu-Model, an ongoing project aimed at developing a neural simulation environment that is extremely computationally powerful and flexible, is described. It is shown that the use of good Software Engineering techniques in Neu-Model's design and implementation is resulting in a high performance system that is powerful and flexible enough to allow rigorous exploration of brain function at a variety of conceptual levels.

1. INTRODUCTION

A recent development in the fields of Cognitive Science and Neuroscience has been the use of computational models as a means of understanding mental processes and brain dynamics [1; 2; 3]. The use of such models defines the sub-discipline of Computational Neuroscience, which aims to elucidate the mechanisms of neural information processing and population dynamics, through a methodology of incorporating biological data into complex mathematical models. These models are to a greater or lesser degree inspired or constrained by the functional architecture of the brain, and the cells that comprise it. For instance, sets of linked partial differential equations may be used to simulate the time and voltage dependent behaviours that are displayed in cellular membrane conductances, and which have been observed and described in the cellular electrophysiology literature. Such models are used to simulate experimental results, generate novel hypotheses and provide insights into how the brain performs computations. This type of methodology differs fundamentally from classical modelling approaches in computer science, specifically in the field of Artificial Intelligence [4], where the physical structure of the brain is ignored in efforts to duplicate its function.

The field of Computational Neuroscience has concerned itself with modelling the brain at specific, but different, levels of abstraction, ranging from extremely biologically detailed models of single neurons (nerve cells) [5], to highly abstracted models focussing upon the computational properties of specific configurations of cellular circuitry [6]. Multi-level modelling [7] is an emerging approach which aims to bind different levels of abstraction, allowing a coherent and integrated investigation of brain function.

A number of simulation environments currently exist [1, 3, 8, 9], however, these model at a particular level of biological realism, and none of them allow a multi-level approach. Additionally, most were researcher-need specific and not engineered to produce compute-efficient solutions, an important issue because sufficient processing power is a major impediment in the field.

From the Software Engineering perspective, a large and complex task of this kind provides an ideal opportunity to prove and extend the latest methodologies. This paper describes Neu-Model, an ongoing project aimed at developing a neural simulation environment that is extremely computationally powerful, and flexible enough to allow rigorous exploration of a variety of conceptual levels.

2. COMPUTATIONAL NEUROSCIENCE AND LEVELS OF MODELLING

Neural models are comprised of an array of simple processing elements ("neurons"), which are interconnected (ie. by "synapses"), and which represent dynamic parallel processing systems. The field encompassing the use of such models has variously been labelled connectionism, artificial neural network modelling, realistic brain modelling and computational neuroscience - the choice of terminology most often been influenced by the degree of biological realism with which the neural models are constrained. The choice of the two extremes of highly abstract models versus highly realistic models also tends to reflect the types of question being addressed. Generally speaking this dichotomy can be considered as reflecting either:

- top-down approaches to understanding the kinds of computations that might underlie specific mental processes, (ie. involved for instance in the storage and retrieval of memories, or the transformation of an array of light on the retina into a meaningful 3 dimensional scene) (ie. [10; 11; 12]), or
- bottom up approaches to understanding the types of specific cellular and sub-cellular processes that might effect population behaviours (ie. [13, 14]).

The former type of modelling strategy usually involves an *a priori* computational theory of how a specific task might be performed, and then attempts to show how a simple neural

model might perform that computation. This type of model's properties are typically predicated more upon the specific pattern of connectivity or circuitry than on the range of behaviour displayed by the individual processing elements, which is usually modelled as a simple input-output function [15]. Abstracted representations of neural activity may be modelled by simple linear or non-linear functions, which describe either activity/non-activity at any given moment, or a time-averaged gross activation level.

At the more biologically constrained end of the spectrum (the latter strategy), models tend to have more complex processing elements and statistically constrained connectivity patterns that rely upon the existence of large numbers of elements to achieve their effects. These models employ a methodology known as compartmental modelling, which is based upon sets of linked partial differential equations modelling the electrotonic properties of the cellular membrane [16], and the active time and voltage dependant behaviours of various ionic conductances across the cellular membrane [17]. These properties of cellular behaviour can be described using cable equations [16] which represent the passive electrotonic properties of the cell, in conjunction with linked non-linear generalised partial differential equations (PDEs) which describe the time and voltage dependent behaviours of active cellular mechanisms. These PDEs take parameters describing known physiological measures and are "fitted" to observed physiological functions. Values for parameters and fitting curves describing dynamic functions are taken from the cellular physiology literature. This modelling strategy allows an arbitrary level of complexity in terms of the explicit modelling of realistic dendritic tree structures and the inclusion of numerous parameters modelling the effects on the intracellular and extracellular conductances operating on varying temporal scales.

The relative merits of these two approaches are clear. Abstract models are easy to understand and relatively computationally inexpensive but often it is hard to see what they might have to do with actual brain structure and function (apart from wishful thinking on the part of the modeller). Realistic models are much less transparent, there is a greater number of variables and therefore a much larger behavioural state space, and they are much more computationally expensive than abstract models. However they yield data that can be more directly compared with measures of real brain activity (such as EEG and single unit recordings). Moreover, there is a much stricter isomorphism between the thing been modelled and the model itself.

A methodological approach is emerging, not that takes a middle path between these extremes, but rather that takes a number of middle paths. This approach is called multi-level modelling. The aim is to use a combination of top down and bottom up modelling strategies that attempt to show isomorphisms between models at different levels of description, and that converge upon a coherent solution through the iterative application and relaxation of constraints (ie. [18]).

3. EXISTING NEURAL MODELLING ENVIRONMENTS

Whilst a number of neural modelling software packages and class libraries exist, as a rule they are aimed at modelling a specific level of abstraction/realism. For example GENESIS [3], CONICAL [8] and NEURON [9] allow the modelling of single neurons and small networks with a high degree of biological detail. PDP++ [1] allows the modelling of larger neural networks, but places considerable constraints upon the level of detail at which individual elements can be modelled. Specific-level modelling is problematic, as it does not include support for the systematic exploration of the effects of tightening or loosening the realistic constraints, or for thoroughly exploring the relative contributions of circuitry and cellular complexity in emergent population behaviours.

It may be suggested that multi-level modelling could be achieved by integrating the (discrete level) results obtained from existing packages, so that, in effect, a number of packages would cooperate to produce experimental results. Such an approach would at best be extremely difficult, involving significant investigator or programmer effort in modifying the outputs of each source modelling environment to render them suitable for the target environment. This effort would be essentially on a 'one-off' basis, and need to be repeated for each individual experiment. It was deemed that the overheads of such an approach were unacceptable, and no further effort was expended on it.

Whilst each existing environment has contributed to the advancement of neural modelling, their construction reflects their medical-researcher and temporal origins, and does not exemplify modern software engineering practice, leading to problems with classic issues such as adaptability, extensibility and portability. Incorporating new components or new functionality into these packages, for example, involves programming, a skill not necessarily associated with medical researchers.

As the complexity of models increases due to, for instance, large numbers of interacting components, performance impacts on the researcher's ability to experiment. This may be attributed in part to the fact that existing modelling environments (except CONICAL) are interpreted rather than compiled, and in part to the actual construction of the software itself. Lack of software performance increases the exposure of experiments to system failure, and impacts negatively on other users of the computing equipment. In the event of failure, and in the absence of support for persistence of results exhibited by the current environments, it becomes necessary to start from the beginning if an experiment if it is unexpectedly interrupted. Finally, with the exception of PGENESIS, a parallel version of GENESIS which requires a supercomputer to provide any real benefit, the existing modelling packages have not been constructed to provide support for performance enhancement through the use of arrays of parallel processors.

4. NEU-MODEL ARCHITECTURE

The use of *objects* in modelling [19] has become increasingly popular in the past decade, supported by the definition of languages such as C++ [20] and Java [21]. The use of objects in system construction allows designers to view components (of systems being modelled) in terms of function, abstracting over implementation details. This approach has the dual benefits of reducing unnecessary complexity and facilitating software re-use. Specification of such models has been largely standardised by the use of Unified Modelling Language (UML) [22]. Formalism within UML is defined in [23]. Implementation of object-oriented models (programming) is fundamentally different from the previously widely accepted procedural approach [24] with emphasis on information hiding, with objects being seen purely in terms of their procedural interface.

Object-oriented software systems typically comprise arbitrarily complex networks of objects, and construction of objects themselves out of more fundamental objects. Thus a system can be seen as comprising *components* that may themselves be constructed from more fundamental components. In this way the use of object-orientation for software systems parallels the construction of structures found in nature. The networks themselves may be different at different abstraction levels, but their utility may be enhanced, particularly from the aspect of component replacement, by viewing them in terms of *positions* and the objects that occupy those positions [25] as further developed in, for example, [26]. Modelling according to this paradigm allows the required flexibility in terms of levels of abstraction, with the additional benefit of complexity reduction when complex components are replaced by (more simply implemented but) functionally equivalent state machines (whose behaviour mimics that of the replaced component).

Neu-Model has been constructed using modern object-oriented design principles. In accordance with the latest practice, the user view (with which the user interacts, typically a graphical user interface or GUI) is separated from the model implementation. The view and model entities may thus be implemented using the respectively optimal programming languages. For example the interpreted language Java provides excellent portability, and is ideal for implementation of the view, which requires a high degree of interaction with the host operating system (OS), while the fully compiled C++, for which portability is most a problem from the point of view OS interaction, is ideal for the model to provide performance. Communication between the view and the model is achieved using message passing, provided by the OS if both view and model execute on the same computer, or using a transport mechanism such as CORBA [27] to exploit powerful computer servers in a distributed environment.

The model itself is further decomposed into as a single controller component and multiple, extensible (in number), substitutable and adaptable neural components (some CONICAL compartmental components have been used, others are new with Neu-Model). Neural components may be organised into arbitrarily complex networks, with the relative orientation of each component defined using the Position abstraction [25, 26]. This provides an architecture in which components are substitutable in isolation from changes to the network structure. Run-time binding allows researchers to redefine or substitute components without the need for a full system rebuild between experiments.

5. EVALUATION

The standard GENESIS single-level tutorials were performed using Neu-Model, and the results matched empirical data as well as GENESIS, confirming Neu-Model's compatibility with the research-community accepted existing system. For example Figure 1 shows the modelling of two single compartment neurons (created as a pair of multi-compartmental neurons with gated channels using the Hudgkin-Huxley parameters connected with a delay link) connected along a pathway with a 15msec attenuation delay when at 60 msec into the simulation a current from a probe is injected into neuron1 (this is equivalent to demonstration 5 of the GENESIS simulation library).

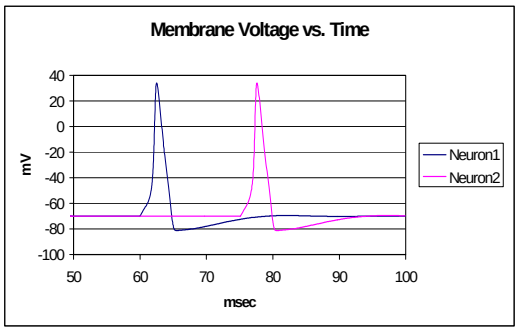

Figure 1. Two Neurons at the Same Abstraction Level.

Support for multi-level modelling was evaluated using a number of combinations of elements at different abstraction levels. For example Figure 2 shows the modelling of a pair of neurons, one modelled using a compartmental model (using Hudgkin-Huxley parameters as in Figure 1) and connected along a pathway with a 15msec attenuation delay to a connectionist style sigmoid neuron, when at 60 msec into the simulation a series of pulses from a probe is injected into the compartmental neuron. Notice that the behaviour of the sigmoid neuron differs from that of the HH neuron, with differently shaped spikes and more rapid return to rest state. Such a combination of abstractions produced modelled results that had not previously been achieveable.

Performance comparisons show that Neu-Model compares well with the other popular modelling environments. Figure 3 shows that only CONICAL outperforms Neu-Model for a simple two-node network. This is expected because, Neu-Model must instantiate significantly more objects than CONICAL to achieve the infrastructure for any simulation.

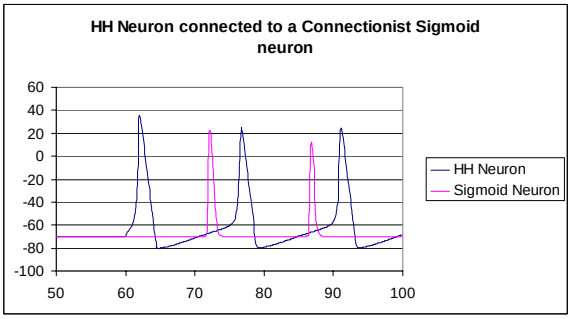

Figure 2. Two Neurons at Different Abstraction Levels.

As shown by Figure 4, the infrastructure-object instantiation overhead decreases with neural-network size, so that for larger networks Neu-Model performance matches and eventually outperforms that of all other systems (including CONICAL). It should be noted that this level of performance and scaleability is achieved while providing unmatched versatility.

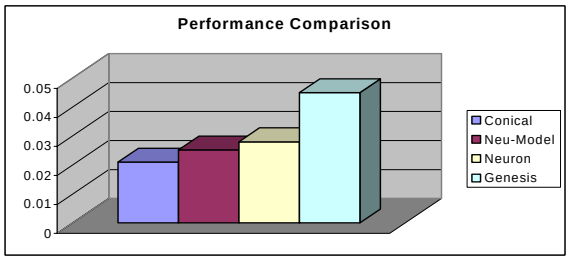

Figure 3. Performance Comparison for Two-Neuron Network (ms of processing time per cycle)

As further described in Section 6, Neu-Model is still in development, and the GUI is as yet incomplete. This necessitates the use of scripting language for model definition (use of scripting is also required by the existing modelling systems). Complexity of use of Neu-Model was measured in terms of lines of script required to achieve equivalent models. As shown in Figure 5, Neu-Model is significantly easier to use than its competitors. It is expected that this advantage will be extended when the GUI is complete.

6. FUTURE WORK

Given the fact that nervous system components 'learn' (their behaviour is modified by previous events) it is possible that technology such as that used to implement *software agents* [28] or *mobile agents* [29] may be applicable. Software agents are capable of learning and developing while their related structures, the mobile agent, are independent entities sent out on the Internet to research and return data and information. If agents were applicable to this problem, an added benefit of their use would be their ability to exploit computers providing arrays of processing elements [30].

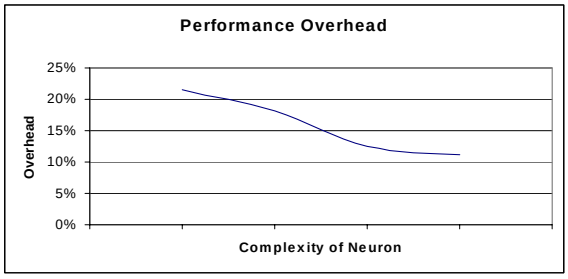

Figure 4. Comparison of Overhead and Model Complexity.

At present Neu-Model provides a textual user interface, with users defining neural components using scripts, and output provided in text files. The compute engine, however, has been constructed to support message-based input, and construction of a GUI interface to provide such input and visually display output is in progress. Output, of course, may if required be directed to some other module, for instance the output of one simulation may be directed to form (part of) the input to another.

Other areas of future work include interfacing to message transport mechanisms to support networks of simulator instances, the introduction of arbitrarily-positioned probes to inject and collect signals at various physical positions throught the network of neural components, and importantly support for persistence of results, allowing both restart after unexpected system failure and electronic comparison of the results of different abstraction levels.

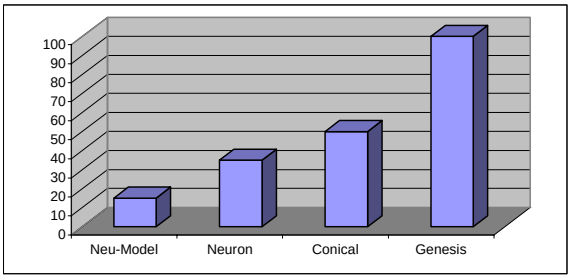

Figure 5. Comparative Modelling Complexity.

7. CONCLUSION

This paper presents a new modelling environment designed to assist neuroscientists in their quest for better knowledge of the brain and its function. The environment, named Neu-Model, provides explicit support for multi-level modelling, and is the first environment to do so.

Neu-Model's.architecture incorporates the most recent advances in object-oriented software engineering. It thus provides a great deal of flexibility for the medical researcher, and facilitates future modification end extension of its functionality.

As demonstrated, Neu-Model is already able to duplicate and extend on the functionality and performance of existing

modelling environments. Development is continuing, and will be reported in the future literature.

REFERENCES

[1] Rumelhart, D.E., & McClelland, J. (1986) *"Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol1"* Cambridge Mass. MIT Press.

[2] Churchland, P. S. & Sejnowski, T. L. (1992) *"The Computational Brain"*, Cambridge Mass. MIT Press.

[3] Bower, J.M., Beeman, D., (1995) *"The book of Genesis"*, Telos, Springer-Verlag.

[4] McCarthy, J. and Hayes, P. J. (1969) *"Some Philosophical Problems from the Standpoint of Artificial Intelligence"*, Machine Intelligence, 4: pp463-502.

[5] DeSchutter, E., and Bower, J.M. (1994a) *"An Active Membrane Model of the Cerebellar Purkinje Cell: I. Simulation of current clamps in slice"*, J. Neurophysiol. 71, 375-400.

[6] Rolls, E. (1996) *"A theory of hippocampal function in memory"* Hippocampus 6:601-620.

[7] Cervantes F., Weitzenfeld A (1996) *" Multi-Level Simulation Methodology: A Computational and Experimental Approach to Neural Systems"* http://pitts.rhon.itam.mx/English/Project/Multilev.htm

[8] Strout, J.J. (1996) *"CONICAL: The Computational Neuroscience Class Library"* http://cajalnt.ucsd.edu/jstrout/conical/

[9] Hines, M., (1994) *"NEURON"*, in Neural Network Simulation Environments, Editor J. Skrzypek, Kluwer Academic Publishers.

[10] Rolls, E. (1996) *"A theory of hippocampal function in memory"* Hippocampus 6:601-620.

[11] McClelland, J. & Goddard, N. (1996) *"Considerations arising from a complementary learning systems perspective on hippocampus and neocortex*", Hippocampus 6:654-665.

[12] Marr D, and Poggio, T. (1976) "*Cooperative computation of stereo disparity."* Science 194:283-287.

[13] Hasselmo, M. Wyble P., & Wallenstein (1996) *"Encoding and retrieval of episodic memories: role of cholinergic and gabaergic modulation in the* hppocampus", Hippocampus 6:693-708.

[14] Klopp, J., Johnston, P., Halgren, E., Nenov, V. (1999) *"Wide-band spectral power fluctuations characterize the response of simulated cortical networks to increasing stimulus intensity"*.

[15] McCulloch, W.S. & Pitt, W.H. (1943) *"A logical calculus of ideas immanent in nervous activity"*. Bull. Math. Biophysiscs 5:115-133.

[16] Rall, W. (1964*) "Theoretical significance of dendritic trees for neuronal input-output relations"*. In: Neural Theory and Modeling. Reiss, R. F. ed. Stanford University Press, pp. 73-97.

[17] Hodgkin, A. L. and Huxley, A.F. (1952d) *"A quantitive description of membrane current and its application to conduction and excitation in the nerve."* J. of Physiol. 117:500-44.

[18] Cervantes F., Weitzenfeld A (1996) *" Multi-Level Simulation Methodology: A Computational and Experimental Approach to Neural Systems"* http://pitts.rhon.itam.mx/English/Project/Multilev.htm

[19] Rumbaugh, J., Blaha, M., Premerlani, W. & Eddy, F. (1991) *"Object-Oriented Modelling and Design"* Prentice-Hall.

[20] Stroustrup, B. (1991) " *The C++ Programming Language, Second Edition*", Addison-Wesley.

[21] Arnold, K. and Gosling, J. (1996) *"The Java Programming Language"* The Java Series, Reading, Mass.: Addison-Wesley.

[22] Booch, G., Rumbaugh, J., Jacobson, I. & Rumbaugh, J. (1999) *"The Unified Modelling Language User Guide"* Addison-Wesley.

[23] Warmer, J. & Kleppe, A. (1999) *"The Object Constraint Language: Precise Modelling with UML"* Addison-Wesley.

[24] Meyer, B. (1997) *"Object-Oriented Software Construction"* Prentice-Hall.

[25] Aho, A., Hopcroft, J. & Ullman, J. (1983) *"Data Structures and Algorithms"* Addison-Wesley.

[26] Goldberg, A. & Robson, D. (1989) *"Smalltalk-80: The Language"* Addison-Wesley.

[27] *"CORBA 2.3.1/IIOP Specification"*, available electronically from http://www.omg.org/library/c2indx.html

[28] Sloman, A. *"What sort of architecture is required for a human-like agent?"* invited talk at Cognitive Modelling Workshop, AAAI96, Portland Oregon.

[29] Vitek, J. & Tschudin, C. (1997) *"Lecture Notes in Computer Science 1222"* Springer-Verlag.

[30] Fitzpatrick, S., Harmer, T., Stewart, A., Clint, M. & Boyle, J. (1997) *"The automated transformation of abstract specifications of numerical algorithms into efficient array processor implementations"* Science of Computer Programming 28(1): 1-41.