Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Neurocomputing 58–60 (2004) 1123–1128
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Neural systems integration

Michael Arnold[a,*], Terrence Sejnowski[a], Dan Hammerstrom[b], Marwan Jabri[b]

[a] *Computational Neurobiology Laboratory, The Salk Institute for Biological Studies, 10010 North Torrey Pines Road, La Jolla, CA 92037, USA*
[b] *OGI School of Science and Engineering, 20000 NW Walker Road, Beaverton, OR 97006, USA*

Abstract

A need is identified to build models of the central nervous system that are semi-complete, applied within multiple contexts to multiple tasks, using methodologies that span multiple levels of abstraction. The issues and constraints in building such models are discussed with respect to completeness, validation, cost, scalability and robustness. An approach currently being explored is described that is suited to the creation of large heterogenous models by small independently collaborating research groups. It is based on a network model interface, a software wrapper that abstracts the interaction between a generic component and a generic framework.

Keywords: Systems integration; Biological modeling; Software engineering

1. Introduction

A sparsity of constraints is a frequent problem when modeling the complex systems found in the brain. For any given level of abstraction, models containing more than a handful of neurons can rarely be well constrained, particularly if not at the motor or sensory periphery. At any point in time, the boundary conditions of the system being modeled are usually only partially known. Solutions include looking at the consistency of the model within a wider context, looking at the utility of the model and combining constraints from different levels of abstraction.

These approaches argue for the ability to model systems which are semi-complete and applied to multiple tasks or in multiple contexts, using methodologies that span

* Corresponding author. Tel.: +1-858-4534100; fax: +1-858-5870417.
E-mail addresses: mikea@salk.edu (M. Arnold), terry@salk.edu (T. Sejnowski), strom@ece.ogi.edu (D. Hammerstrom), marwan@ece.ogi.edu (M. Jabri).

0925-2312/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.neucom.2004.01.176

multiple levels of abstraction. Such modeling exercises can be characterized as large, heterogenous and collaborative. Issues arise from the increased size and complexity, the heterogeneity and the increased requirement for collaboration. The following discussion is given within the context of neural systems, but may apply equally well to other domains such as cell signaling.

Two themes can be identified. The first seeks to define methodologies that can deal with the problem of sparse constraints. The second deals with the practical issues arising in their application. The creation of large heterogenous models applied within multiple contexts is beyond the scope of a small research group, yet there is the need to do so outside of the context of large, well-funded, highly structured and heavily managed research initiatives. This paper focuses not on the first theme, what is a good methodology, but on the second, the practical issues arising from their application within the fragmented and less well-funded context of many small, independent, collaborating research teams. An approach is described that is currently being tested in practice.

2. Constraints

The essence of any approach is to integrate the work from many researchers. The optimal path depends on the resources available and the operating environment. Highly structured, managed and funded initiatives can afford to impose strong constraints on individuals, in part because they can also impose comprehensive training. By contrast, a solution geared to an academic research environment must avoid imposing harsh constraints on the individual researcher, in particular the imposition of specific modeling environments or sets of tools. Such impositions often meet with resistance, due to a perceived reduction in immediate effectiveness, lack of suitability, interference with creativity and requirement for extensive training.

Two paths are available for the integration of a model as a component in to a larger system. The rebuilding approach is for the integrators to re-build published work within some target modeling environment. The integration approach is to integrate an existing model as a software component, stitching the components together within a framework. Both approaches have their strengths and weaknesses regarding completeness, validation, sufficient understanding, cost, scalability, robustness and performance. A component described as a piece of software is a complete description, yet it may be difficult both to gain sufficient understanding of, and to validate, its function. A component described by a publication is rarely a complete description, yet reproducing the model in the target modeling environment both validates and ensures sufficient understanding of its function. The relative cost and performance are harder to gauge. Rebuilding may have a higher initial cost, but depending on the quality of the software components, may have a lower overall cost than the integration approach. Performance and robustness may be difficult to maintain for the integration approach, limiting the scalability it offers.

What is optimal is further constrained by the target implementation, however a system of components, as well as individual components and subsets of components, may need to be investigated within multiple contexts and tasks. Different contexts place

different constraints on the target implementation, with requirements varying from interacting with a simulated world in virtual time, to a real world in real time, to specific robotic or prosthetic platforms and the use of specialized hardware. A further variability will be introduced by the different questions being asked within a broad group of collaborating researchers. The constraints on the target implementations are therefore largely unknown.

3. The network model interface

An approach is described that favors integration over rebuilding, that allows the integration of existing software, and that is invariant to the target implementation. The emphasis is on flexibility, resutability and low barriers to use, requiring a low level of training and presenting low risk. It is targeted towards collaborations between many, small, independent research groups. It does not directly address the problems of validation and sufficient understanding.

The approach is based on the concept of a network model interface (NMI) [1]. The NMI defines the interaction between a generic model component and a generic simulation framework. It is applied as a software wrapper around a component. Frameworks can support the wrapper without knowledge of a component's internals. The NMI does not define a framework nor does it address framework issues, other than what is implicit in the definition of the interface. The motivation for distinguishing the NMI from the framework is to separate the implementation of a component from its deployment, as the target implementation cannot generally be well-defined. The goal is to allow the implementation of reusable components that are framework-independent. The choice of a framework may depend on many issues that the designer of a component need not or cannot be aware of. This is in contrast to approaches focused on the design of a framework or simulation environment to match a given set of constraints.

A component interacts with a framework in three areas: the passing of data, control and configuration. A master–slave relationship is enforced with the framework calling the component. The scope of a component is strictly limited to itself. It has no knowledge of the other components within the model and is directly dependent on the framework for all information external to itself. It is the responsibility of the framework to deal with synchronization and timing issues through the regulation of how and when control is passed to a component.

The NMI abstracts the flow of data between components as a network of nodes and connections. This network connectivity explicitly defines the scope for all possible interactions between components, which is limited to the communication of data, and which is managed by the framework. Control can only be passed to a component from the framework, in the form of a timing token that allows the component to move forward to a specified point in time. Control could be implicitly passed between components as data, but this would be a matter for the components generating and interpreting the data, and would be transparent to the framework.

The interaction between a framework and a component can be abstracted into a number of steps (Fig. 1). First, passing configuration information to a component.

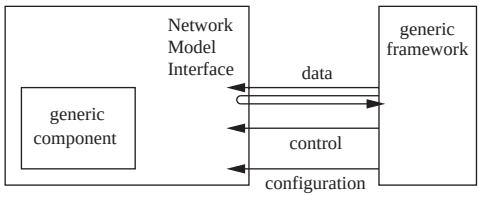

Fig. 1. The network model interface is a software wrapper that abstracts the interaction between a generic component and a generic framework, as the passing of control, data and configuration.

Second, requesting the component to create any internal structures. Third, requesting from the component information about the data communication channels that must be set up on its behalf. Fourth, requesting the component to reset itself in preparation for execution. Fifth, running the component by repeatedly (a) passing input packets, (b) passing control so that the component can move forward in time, and (c) requesting output packets.

The NMI may be used to encapsulate both software components and specialised hardware. Components have been developed by a number of groups [2] that encapsulate a range of functionality, such as the numerical integration of arbitary networks of conductance-based neurons using field programmable gate arrays, motor and sensory systems such as mechanical arms and cameras, simulated tasks such as table tennis, and models of brain areas such as the cerebellum, parts of the ventral pathway, pre-frontal cortex and basal ganglia (Fig. 2). Frameworks exist that support the automatic integration of any NMI component. Interoperability between frameworks is addressed by components that encapsulate simulation environments such as Matlab, allowing arbitary run-time definable segments of an environment's native code to be executed as a component. Details of modeling projects based on the NMI are given in [2].

4. Discussion

A component approach to building large neural systems where participants share work as software or hardware components could be broken in to four stages. First would be a low cost and low committment, knowledge sharing primary stage with low barriers to participation. An infra-structure could facilitate and disseminate the identification of hardware and software components that participants could potentially make available. This would be followed by an identifying secondary stage, with low to intermediate cost and committment. In this stage the infrastructure facilitates the identification of opportunities for the sharing of components between participants. Third would be a facilitating secondary stage, with intermediate cost and committment. Here, the infrastructure facilitates the physical sharing of components by providing support and documentation for the process of encapsulating work as shareable components, using a mechanism such as the NMI. The final stage would be a high cost and committment tertiary stage, where components are brought together to build large neural systems.

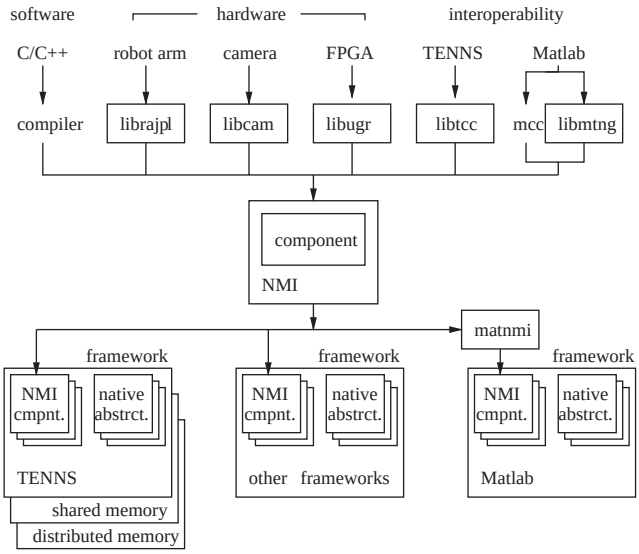

Fig. 2. The network model interface allows components providing a range of functionality to be automatically integrated into frameworks that support the interface. This introduces a flexibility for the end user, and an invariance to the target implementation for the component designer. For details of available NMI resources refer to [2].

What is left largely unaddressed by component approaches such as that based on the NMI, are some important systems integration and engineering issues. This is not a reference to the software engineering issues, but to those relating to the integration of existing components versus their re-implementation. In practice, the function of a component will never be transparent. This makes it difficult to both validate the component and to gain a sufficient understanding of its function to be able to apply it within a larger system. Managing this transparency of function is the major issue in the integration of neural systems. This is reflected in those approaches based around the use of markup languages to structure the knowledge domain for biological models [3].

The described approach, as illustrated using the network model interface, differs from others in that it focuses on how large neural systems can be implemented in the short term and with limited resources. This acknowledges the unknown nature and complexity of the integration problem, and the need to identify the issues from direct experience. The value of this approach is that it represents a shortest path to allow the problems to be studied at first hand.

Acknowledgements

This work was supported by NASA grant number NCC 2-1253 and the SpikeFORCE IST-2001-35271 grant from the European Community.

References

[1] M.P. Arnold, The network model interface, Technical Report, The Salk Institute for Biological Studies, 2002. Available at www.cnl.salk.edu/~mikea/doc/libnmi/libnmi.html.

[2] M.P. Arnold, NMI simulation resources, Technical Report, The Salk Institute for Biological Studies, 2002. Available at www.cnl.salk.edu/~mikea/doc/nmisim/nmisim.html.

[3] N.H. Goddard, M. Hucka, F. Howell, H. Cornelis, K. Shankar, D. Beeman, Towards neuroml: model description methods for collaborative modelling in neuroscience, Philos. Trans. R. Soc. Lond. B. Biol. Sci. 356 (1412) (2001) 1209–1228.