Meeting Minutes 31 August 2000

Author:
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
Poul Nielsen (Bioengineering Institute, University of Auckland)
David Bullivant (Bioengineering Institute, University of Auckland)

1 Summary

Warren started the August 31 meeting off by raising an important issue that had come up as a result of the encapsulation in the pathway example from the August 24 minutes. There was more discussion about the limits of the ontology definition, and how far it should be extended into defining component libraries. We also talked about specifying the set of legal and illegal relationships in the ontology. (Reminder: I meant to bring up defining variable classes in the ontology but didn't.)

2 More On Encapsulation

As the reader will undoubtedly recall from earlier meeting minutes, the idea behind adding some kind of "*is-encapsulated-by*" flag to a connection is to hide the encapsulated component(s) from the network in which the encapsulator (is that a word?) resides. Connections could not exist between components on different levels of complexity within a model. However we will undoubtedly need to get variables from components adjacent (i.e., connected) to the encapsulator down to the encapsulated components. These variables would obviously have role *in* in the encapsulator, but in order to get them down a level, we would have had to re-map them to variables with role *out*. How would you indicate that these variables could only be passed down the encapsulation connections? It's not even worth thinking about!

Why not instead have implicit inheritance of variables along connections with an *is-encapsulated-by* flag? All variables in the encapsulator are automatically mapped to variables of the same name with role *in* in the encapsulated components, unless they are explicitly connected to another variable from another component on the same complexity level. It is unlikely that naming problems would occur, because an encapsulated network would never really be inserted under some arbitrary node in the graph, but only developed in conjunction with its encapsulator, or developed as a refinement in function for an existing component in a network. If name mapping is a problem, it may be appropriate to have a role *inherited*, and specify the name and location of the target variable inside the variable declaration. (Come to think of it, maybe this would be a way to have a general inheritance scheme.)

3 More On Ontology

In earlier meeting minutes I proposed that class-definitions have entirely the same data model as actual instances of components and connections in the network. I didn't mean to suggest that the ontology was the preferred place to define a component library for a software package, although this is definitely an option. A good reason not to do this is that it is impossible to create a template for an entire mini-network in the ontology definition.

This is best demonstrated in terms of an example. It might be desirable to have a component library that contained an object consisting of a membrane with a calcium and a sodium channel, something which might best be expressed as a network with some required inputs to each of the components. The best approach to the creation of a component library where some objects are entire networks is to declare some base classes such as membrane and channel in the ontology along with preferred connections between the two (i.e., channels can be inside and/or encapsulated by membranes), and then create self-contained but unfinished networks in terms of instances of these classes.

The thing that we haven't really covered in previous discussions of ontology is the specification of relationships between components in a network. In the typical definition of an ontology for electro-physiological models the model author might define classes compartment, membrane and channel, and want to specify that a channel can be placed inside of a membrane but not inside of a compartment. To do this the ontology must contain a connection definition, where the component references that would typically appear in the `between` section (see data model from the August 25 meeting minutes) are class definition references. The relationship information that then appears in the connection indicates relationship information that may exist on connections between instances of the specified classes. For example, if the model author defines a connection between classes channel and membrane, they might add a geometric relationship of type *is-in* and the *is-encapsulated-by* encapsulation relationship flag. These relationships might then be present in a model where instances of classes channel and membrane are connected.

The most elegant approach to indicating that a relationship may not occur between two components is to simply not allow relationships between components that aren't defined in the ontology. For example, to prevent the model user from placing a channel inside a compartment, the model author does not put any geometric relationship information on a connection between the two class definitions in the ontology. This would not prevent the user from mapping variables in an instance of class channel to variables in an instance of class compartment, but would prevent them adding any geometric information to that connection.

It would still be possible to define mappings in a connection class definition that could be "inherited" by all instances of that connection class. For instance, a class definition for a membrane might include the definition of a voltage variable (V_m) and the class definition for a compartment might include a definition of the same variable, where this variable has a role value of `input`. It might make sense to declare that in all connections between instances of class compartment and instances of class membrane contain a mapping between the voltage variable definition in the membrane and the input in the compartment. This removes some unnecessary specification from the actual model definition.

Note that in order to allow geometric relationships between components of the same class, the ontology may also contain connection definitions that are between two components of the same class. This might be useful, for instance, in the definition of an ontology for some simple electro-physiological models, where the extra-cellular subspace is connected directly to the T-Tubule subspace. In cases like this, we need to be able to allow two kinds of geometric relationship on the same connection, both *is-in*, and *is-next-to*.

In the data model presented in the meeting minutes of August 25, it was indicated that a connection might have a `class` property in the same manner as a component. It turns out that, with the scheme described above and the requirement that there can only be one connection between any two components in a model, it is no longer strictly necessary to associate an identifier with a connection definition.