# Meeting Minutes 22 September 2000

Author:
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Melanie Nelson (Physiome Sciences Inc.)
Poul Nielsen (Bioengineering Institute, University of Auckland)

## 1 Summary

Yet again, this document is not just limited in scope to meeting minutes, but also reports on progress on behind-the-scenes CellML development, which generally equates to E-mail conversations between Warren and Melanie. Topics of discussion included possible XML implementations of qualitative pathway model information, and scripting support.

## 2 Qualitative Pathway Models

To recap: Auckland had originally declared that the concept of a model without math was ridiculous, and that qualitative model information was really metadata or documentation. Melanie and Prasad at Physiome had gone to considerable trouble convincing the Auckland team that something like `A + B -> C` could in fact be a complete model, and that they needed to be able to record the roles of the different variables in a reaction. Melanie and Prasad pulled this off so well that the Auckland team had devoted some time to thinking about the best way to do this at subsequent meetings. Auckland was overjoyed to hear later (they thought) that in fact they only needed to be able to associate a group of variables with a reaction. With this in mind they reached the grouping solution proposed in the September 19 minutes.

During subsequent discussions with Physiome, we realised that they do in fact hope to also be able to differentiate between the inputs and outputs in a reaction. It was Melanie who first proposed a good-looking solution, as shown in Figure 1.

```
<group>
  <relationship class="is-a-participant-in" />
  <major_component_ref name_ref="CaM_activate" />
  <minor_component_ref name_ref="Ca" side="1" />
  <minor_component_ref name_ref="CaM" side="1" />
  <minor_component_ref name_ref="CaCaM" side="2" />
</group>
```

FIGURE 1: Melanie's proposed solution (actually the best of two) for adding variable role information to the grouping mechanism previously developed for associating variables with a reaction.

As the Auckland team had already been convinced that we'd need to be able to differentiate "`activates`" from "`inhibits`" anyway, Warren proposed that we may as well add that back to the requirements. Melanie then proposed the XML solution shown in Figure 2.

This looked initially like a simple extension of the grouping mechanism used for encapsulation and geometry information, but Warren wasn't happy with it. Basically, this grouping was referencing components rather than variables, and thus relied on the assumption that there is only one variable per component

```
<group>
  <relationship class="is-a-participant-in" />
  <major_component_ref name_ref="mels_pathway" />
  <minor_component_ref name_ref="a" participant="reactant" />
  <minor_component_ref name_ref="b" participant="catalyzes" />
  <minor_component_ref name_ref="c" participant="inhibits" />
  <minor_component_ref name_ref="d" participant="activates" />
  <minor_component_ref name_ref="e" participant="product" />
</group>
```

FIGURE 2: Melanie's second stage solution for adding variable role information to the grouping mechanism previously developed for associating variables with a reaction.

and only one variable is received by the reaction component from each participant. In addition to this, it would force the modeller to specify pathway models with one component per participant, which might be unnecessary in many cases, or downright impossible for combined electrophysiological-pathway models.

So Warren proposed the solution shown in Figure 3. In this solution, the variables taking part in a reaction are referenced directly, and the grouping is placed inside the component where the reaction takes place (this would allow multiple reactions to be specified inside one component). He also voiced the opinion that some kind of generalised variable grouping mechanism (previously unconsidered) could be used for associating properties like units with sets of variables.

```
<component name="warrens_reaction">
  <variable name="A" public_interface="in" />
  <variable name="B" public_interface="in" />
  <variable name="C" public_interface="in" />

  <group>
    <relationship class="is-a-participant-in" />
    <variable_ref name_ref="A" participation="reactant" />
    <variable_ref name_ref="B" participation="catalyst" />
    <variable_ref name_ref="C" participation="product" />
  </group>
</component>
```

FIGURE 3: Warren's initial solution for specifying qualitative information about reactions, which takes the form of variable grouping *inside* of a component.

This solution was the recipient of some pretty harsh abuse during the Auckland team's meeting of September 22. Poul and David were fairly resistant to the idea of adding variable grouping to CellML at this late stage. They argued that it added complexity and wasn't a requirement, so why have it? This was a pretty concise and effective argument, and Warren was persuaded that the general grouping of variables wasn't needed. However, he stubbornly refused to admit that the core information set marked up in his solution wasn't the best one, something which Poul and David were forced to accept. So we looked at introducing a new object to the data model called `reaction` which fits in amongst the mathematical information associated with a component. A more complete serialization of a component is given in Figure 4.

```
<component name="warrens_reaction">
  <variable name="A" public_interface="in" />
  <variable name="B" public_interface="in" />
  <variable name="C" public_interface="in" />

  <mathml:math xmlns:mathml="http://www.w3.org/1998/Math/MathML">
    ...
  </mathml:math>
  <script type="text/ecmascript"> ... </script>
  <reaction>
    <variable_ref name_ref="A" participation="reactant" />
    <variable_ref name_ref="B" participation="catalyst" />
    <variable_ref name_ref="C" participation="product" />
  </reaction>
</component>
```

FIGURE 4: A more acceptable version of Warren's initial solution for specifying qualitative pathway model information which doesn't imply any kind of variable grouping.

Poul and David were reasonably happy with this, but also suggested some further changes. It was pointed out that the `<script>` element (that comes from the document publishing side of XML) should probably be replaced with the MathML equivalent (a more model/math-oriented way of looking at things.) And if the `<script>` element was to become MathML, their reasoning went, that would leave the `<reaction>` element looking like the odd one out, so why not look at ways of expressing qualitative pathway information in MathML? Of course, no-one had any particularly good ideas about the best way to do this. Of note were the two methods shown in Figure 5 (given in code/math form so it's obvious what's going on). It was also suggested that CML (Chemical Markup Language) might have tags to handle this kind of information.

```
true = reactant(A);
true = reactant(B) | catalyst(B);
true = product(C);

{ A , B , C } -> { T , F };
```

FIGURE 5: Some of the ideas proposed for the specification of pathway model information in MathML.

## 3 Math and Scripting Support

One of the other topics of discussion during the September 22 meeting was the specification and use of math and scripting information in CellML. There had never been any plan to throw out the MathML specification of ODEs proposed in CellML '99. The Auckland team was, and still is, very opposed to using script for the specification of model formulation, which they believe can always be specified using MathML. However since the Luo-Rudy II model appears to recommend using an iterative method to evaluate the value of some state variable, it was accepted that at least some support for specifying algorithms should be incorporated into MathML (as long as its use was discouraged).

Ecmascript (formerly Javascript) seems a good bet, as it has a fairly concise specification, is intuitive to use, and a number of free interpreters already exist. Javascript is used primarily for adding functionality to hypertext documents for publishing on the web, and is generally encapsulated by a **`<script>`** tag, which may enclose a script or refer to a script in an external document using a **`src`** attribute. A **`type`** is required to specify the language of the script, which should be in the form of a content-type (eg. “text/ecmascript”).

One of the major headaches with incorporating scripting support into CellML is the specification of an appropriate set of rules for the ways in which ODEs specified in MathML should interact with functions or objects defined in scripting language. Some of the issues are (in no particular order):

1. What does a function call look like? How do we pass arguments and receive return values into ODEs? The Auckland team would prefer to use the appropriate mechanisms specified in MathML for this, but as yet nobody has had time to properly research this area. A quick scan showed that much of the functionality in this area in MathML 1.0 (April 1998) has been deprecated and replaced in the most recent MathML 2.0 working draft (March 2000).
2. Are there MathML tags for actually defining script or wrapping script definitions? This would be cool!
3. How should the ODE integrator deal with scripts? Must we place the restriction that all scripts be executed *instantly* in model-time? Is it possible to have a script that describes a process occurring over several steps in the main integration? How would we indicate that? I think the Auckland team would ban this kind of use of scripts, limiting their use to evaluation of variables at a single point in time.

## 4 Component Reuse and Class Definition

With about three minutes left to discuss things (a rude interruption by Andrew Pullan this time) the Auckland team moved on to component re-use and class definition. Warren’s original ontology ideas (see the meeting minutes from August 25) were again brought up. Basically a class definition or template can use exactly the same data model as the type of object that the author wishes to create a template for, where some associated object flags the processing application that this is an *abstract* (in an object-oriented sense) object, and need not be instantiated in the network. Additionally some sort of flag is needed on data object instances to associate the instance with the class definition.

To the annoyance of Poul and David, both these requirements are satisfied by the CRIM scheme developed by Warren in April 2000 (and appearing soon on the CellML preview website). A class definition could be implemented using the **`<define>`** element, perhaps with a modifying attribute that differentiates it from a low level XML re-use definition. An instance of the class would be implemented using the appropriate **`<??_copy>`** element, with whatever modifications and additions to the class definition are necessary to achieve the desired functionality.

Even Warren would admit however that this means of class definition would be fairly un-intuitive and unnecessarily complex. Some new scheme may be needed.