Meeting Minutes 4 October 2000

Author:
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Scott Lett (Physiome Sciences Inc.)
Jian Li (Physiome Sciences Inc.)
Yi Ge (Physiome Sciences Inc.)
Melanie Nelson (Physiome Sciences Inc.)
Poul Nielsen (Bioengineering Institute, University of Auckland)
Penny Noble (Physiome Sciences Inc.)

1 Summary

This progress report sums up an Auckland teleconference on October 3, a subsequent meeting between Warren and Scott regarding low-level XML re-use, some MathML research Warren did and some XMI research Melanie did.

2 Auckland Teleconference

Melanie had gone over her earlier CellML metadata requirements work, and this had highlighted some issues which she wanted to raise with Poul and David at Auckland, who typically have strong views on pretty much everything. The issues document is recorded in Figure 1 for posterity. The Auckland teleconference was used both to discuss Melanie's metadata questions and to review the final versions of the meeting minutes from September 28, 1 October and 2 October.

The general consensus on the first metadata question was that metadata in general was strictly optional, with the exception of `name`, which is an identifier, and so not really metadata anyway.

Poul argued quite forcefully that the absence of metadata should not imply inheritance of metadata, with regard to the second question. This promotes re-use of model-parts, as we don't have to look for information anywhere other than in the model-part.

Finally, it was generally agreed that CellML must include a general facility for specifying reference information, but define some standard shortcut mechanism for making use of Medline UIs. This was a mildly disappointing result as Warren had been secretly hoping that Melanie would turn blue in the face.

With regard to discussion of the September 28 minutes, the idea of re-introducing a **`role`** attribute to variable declarations in order to differentiate state variables from other variables was unanimously thrown out, as unnecessary duplication of information.

Poul and David were fairly happy with the CellML extension mechanism (based on XML namespaces) and the script definitions described in the September 28 and October 2 minutes respectively. They re-iterated however their displeasure with the low-level XML re-use scheme, and in particular, with the idea of using it for class definition. Again, they talked about using a more high-level scheme, and again they kept weaseling out of doing any work themselves — luckily, Melanie was kind enough to volunteer to investigate this further. More discussion in Section 4.

3 Low-level XML Re-use

Poul and David had been pretty harsh on the ol' low-level XML Re-use scheme since Warren had proposed that it was the ideal candidate for the implementation of class definition (as described in the September 28 meeting minutes). Poul had gone so far as to question its very need for existence. Interestingly, so did Scott,

- **1. Should we allow all metadata to be optional, or make some required?**

  I think we are mostly going to want metadata to be optional, but here are some things that are on the metadata list for models and components that we may want to require:

  - **1.1 Name**

    Warren argues that this is not metadata, and so we can safely make it required. I don't care if its data, metadata, or Bob's uncle, I think every model and every component should have a name.

  - **1.2 ModelBuilder**

    One possibility: we could require this info for models, and assume the value is the same for components unless they specifically say otherwise. However, I do not have extremely strong feelings about requiring this info. I would say that we should make it at least strongly recommended, though.

  - **1.3 Mathematical problem type**

    The software folks tell me that they really want to have this info when we start having models that use math other than ODEs, and that in fact it might be impossible to have a solver that can always figure out what it should do without this information. I'll try to get more info from Scott before tonight's phone call. I think that this could be another strongly recommended piece of metadata, since the software could always just query the person running the model to tell it what sort of math is in there. However, we do need to make it possible for the software to have a sporting chance of using the info if it is provided– i.e., set up a system where the model builder can say what classification scheme he's using and provide the type info from that scheme. Scott has suggested using the GAMS classification of problem types: http://gams.nist.gov/

- **2. In general, I favor saying that a component is assumed to have the same metadata as the model that contains it unless the component specifically declares different metadata. Is this OK with everyone?**

- **3. Do we want to allow people to include reference info within the CellML file, or just to identify a reference (by a Medline UI or with some other UI), or allow them to do either?**

  I will argue until I am blue in the face (unlikely to be a pretty sight) that we should at least provide the option of just referring to a reference by a recognized UI, such as the Medline UI. It is a royal pain to have to type the reference info out, and it is a process that is prone to typos, which make it hard, if not impossible, for later users (particularly software) to figure out what reference the model builder really meant. However, I am not adverse to also providing a way for the user to include all of the reference details in the CellML document.

FIGURE 1: Melanie's metadata questions, written up for the 3 October Auckland teleconference.

who would probably have had to implement it at some point. He pointed out (yes, Warren had realised this) that, although it was useful as a cut and paste mechanism for hand-authoring XML files, it would be tricky for software to make use of when outputting, and even when the exact cutting and pasting was stored, any editing of the model would probably invalidate this.

The L2XR scheme's only saving grace was that it effectively provided the same functionality as a notebook — where the modeller can store "*diff*"s to a model, which might be a useful way of describing simulations. For example, given a model, we could use the L2XR scheme to describe concisely that model with a number of small changes, and associate this with the resulting dataset.

4 Class Definition

Melanie waded straight into the class definition thing by looking at the drowsiness-inducing XMI documentation. XMI is aimed at the serialization of UML data models, and is way too complex for our purposes — I wouldn't wish the coding of an XMI interpreter on anyone. So this was thrown out.

It appears that the best solution would be to define some vocabulary specifically aimed at defining component classes, with the single concept of **extension**. The extension mechanism would only allow the addition of information to a class definition, and not allow re-definition of or modifications to inherited information. Modification of information could be made however in class instances, and so something resembling the low-level XML re-use scheme would be necessary here.

A really dodgy quick example which kind-of demonstrates the functionality we'd like to see is given in Figure 2. The vocabulary probably could do with some improvement.

```
<class_definition name="my_really_cool_hh_gate">
  <extends_class_definition name_ref="regular_hh_gate" />

  <!-- add stuff here -->
  <variable name="new_variable" />
</class_definition>

<component name="fast_sodium_gate_1">
  <instance_of_class name_ref="my_really_cool_hh_gate">
    <insert_attribute xpath=".//variable[@name='new_variable]/@value">
      1.0
    </insert_attribute>
  </instance_of_class>

  <!-- add more instance-specific information here -->
</component>
```

FIGURE 2: A short example demonstrating the minimum set of component class definition functionality that we require for CellML.

Of course, if we really did want to milk the object-oriented angle, we would allow redefinition of information based on name. That is, if a subclass defines a variable with the same name as its base class, the latter definition overrides the former.

I think we might postpone further discussion on this issue until Poul Nielsen arrives at Physiome. Warren still advocates composition by cutting and pasting. Scott advocates composition by nesting (?? I think.)

Poul advocates anything that isn't composition by cutting and pasting. Melanie was left with no options to advocate.

5 Switches in MathML

The MathML specification is pretty hard to read and it is usually not clear how certain types are equations are supposed to be implemented. Warren spent some time on October 4 reviewing the best way to handle switches, a kind of behaviour fairly prevalent in electro-physiological models. A good example is the calculation of *alpha_h* (the alpha gating variable for the *h* gate in the sodium channel) from the Luo-Rudy I model. The switch is of the form:

```
             0      if V_m >= -40mV
alpha_h = {
             ...    if V_m < -40mV
```

The markup given in Figure 3 is based on the second example of Section 4.2.5 of the MathML 2.0 March 2000 working draft. Note that in addition to showing off the use of the **<condition>** element, I also added the **units** attribute from the CellML namespace to every bare number in the equations, potentially allowing processors to raise an error if there are inconsistencies, or convert the values appropriately. The **units** values are in plain text — some kind of mapping from text to unit markup would convert these to a machine-usable format.

6 Variables as Functions of Time

Many models incorporate hysteresis effects, where the model's state is dependent on a previous state. In fact all of a model's state variables (variables whose behaviour is described by differential equations) can be regarded as functions of time. Sometime soon we're going to have to make a big call about actually treating state variables as functions in the math — we'll leave that until Poul Nielsen arrives on the 10th.

Just for a laugh, Warren looked at some alternative methods for expressing the value of a variable at a specific point in time in MathML. A quick browse through the MathML documentation revealed two possible approaches to this, and they are shown in Figure 4.

7 Luo-Rudy II Hysteresis

The crazy Aucklanders were prepared to concede that, since the Luo-Rudy paper actually cites a particular iterative method for evaluating the calcium buffering in the SR, scripting was acceptable in CellML. However they were fairly adamant that its use should be minimized, and so Warren had a look into the specification of the calcium fluxes in the SR using MathML. (This is the nasty 2ms delay problem.) The result of his investigation is shown in Figure 5.

It would be reasonably tricky for software to work out what was really going on from the MathML in Figure 5, so perhaps this is a good place for the use of the **<semantics>** element, with a script in an **<annotation>**. Is it possible to write such a script without relying on implementation, ie. in a portable way?

```
<math
    xmlns="http://www.w3.org/1998/Math/MathML"
    xmlns:cellml="http://www.cellml.org/2000/CellML">
  <apply>
    <forall />
    <bvar><ci> V_m </ci></bvar>
    <condition>
      <apply><geq />
        <ci> V_m </ci>
        <cn cellml:units="millivolts"> -40 </cn>
      </apply>
    </condition>

    <apply><eq />
      <ci> alpha_h </ci>
      <cn cellml:units="dimensionless"> 0.0 </cn>
    </apply>
  </apply>

  <apply>
    <forall />
    <bvar><ci> V_m </ci></bvar>
    <condition>
      <apply><lt />
        <ci> V_m </ci>
        <cn cellml:units="millivolts"> -40 </cn>
      </apply>
    </condition>

    <apply><eq />
      <ci> alpha_h </ci>
      <apply><times />
        <cn cellml:units="dimensionless"> 0.135 </cn>
        <apply><exp />
          <apply><divide />
            <apply><minus />
              <cn cellml:units="millivolts"> -80.0 </cn>
              <ci> V_m </ci>
            </apply>
            <cn cellml:units="millivolts"> 6.8 </cn>
          </apply>
        </apply>
      </apply>
    </apply>
  </apply>
</math>
```

FIGURE 3: An implementation of the *alpha_h* switch equations in MathML.

```
<math
    xmlns="http://www.w3.org/1998/Math/MathML"
    xmlns:cellml="http://www.cellml.org/2000/CellML">

  <!--
    The first expression takes the limit of Ca as time -> t_o.
  -->
  <apply><limit />
    <bvar><ci> time </ci></bvar>
    <lowlimit><ci> t_o </ci></lowlimit>
    <ci> Ca </ci>
  </apply>

  <!--
    The second expression encodes "for all Ca, such that time = t_o". Note
    that if the <apply><forall /> were to be replaced with <set>, you'd have
    "the set of all Ca, such that time = t_o".
  -->
  <apply>
    <forall />
    <bvar><ci> Ca </ci></bvar>
    <condition>
      <apply><eq />
        <ci> time </ci>
        <ci> t_o </ci>
      </apply>
    </condition>
  </apply>
</math>
```

FIGURE 4: Two possible MathML encodings of the value of a variable (*Ca*) at a given time (`t_o`).

```
<math
    xmlns="http://www.w3.org/1998/Math/MathML"
    xmlns:cellml="http://www.cellml.org/2000/CellML">

  <!--
    First we define t_o. Note that "time" is the independent variable.
    The following chunk sets t_o equal to the set of all values of time,
    for which dV/dt is maximised.
  -->
  <apply><eq />
    <ci> t_o </ci>
    <set>
      <bvar><ci> time </ci></bvar>
      <condition>
        <apply><max />
          <bvar><ci> time </ci></bvar>
          <apply><diff />
            <ci> V </ci>
            <ci> time </ci>
          </apply>
        </apply>
      </condition>
    </set>
  </apply>

</math>
```

FIGURE 5: The calculation of `t_o`, the time corresponding to `V_dot_max`.