Meeting Minutes 24 May 2002

Component Re-use Continued

Author:
Autumn A Cuellar (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Catherine Lloyd (Bioengineering Institute, University of Auckland)
David Nickerson (Bioengineering Institute, University of Auckland)
Poul Nielsen (Bioengineering Institute, University of Auckland)

1 Introduction

These minutes continue the discussions started on May 8[1] concerning component re-use.

2 Importing Units from a Different File

To facilitate component re-use, we need a general way to refer to another model and its sub-elements. Unfortunately, the method shown in Figure 4 of the May 8 Meeting Minutes[2] just isn't going to cut it because you will not always need to map two existing **<units>** or **<component>**s. For instance, if an ontology of units is defined, it would be silly to redefine the units in your model and map them to the ontology's units as in Figure 1.

```
<units name="millimole" />

<model_import name="units_ontology" uri="http://www.example.com/units.xml" />

<connection>
  <map_models model_1="units_ontology" model_2="." />
  <map_units units_1="millimole" units_2="millimole" />
</connection>
```

FIGURE 1: Mapping two units definitions from different models together. You can see how nonsensical this method is.

The scheme shown in Figure 2 has been proposed to deal with this issue. The **<model_ref>** of the May 8 Meeting Minutes has been changed to **<model_import>**, but the **name** and **uri** attributes have been kept. I will further discuss this element in Section 3. The units defined for the variables "Cm" and "I_st" refer to units defined in the imported model by first identifying the model by its given name followed by a dot and the units definition contained in the model referenced. Using this method, we would not need a **<map_models>** element as described in the May 8 Meeting Minutes.

We recognize the need to set limitations on the use of this import method. This method could only be used to identify *top-level* elements in an imported model. In this document, top-level elements are the

[1] http://www.cellml.org/private/progress_reports/20020508_meeting_minutes.html

[2] http://www.cellml.org/private/progress_reports/20020508_meeting_minutes.html

```
<model_import name="units_ontology" uri="http://www.example.com/units.xml" />

<component name="membrane">
  <variable name="Cm" units="units_ontology.microF" />
  <variable name="I_st" units="units_ontology.microA_per_microF" />

  ...

</component>
```

FIGURE 2: The `<model_import>` identifies the URI of the file being imported and gives the model a name by which it will be referred in the current model. The variable definitions can then refer to `<units>` defined in the imported model by using the imported model's name and the `<units>`'s name separated by a dot.

immediate child elements of the CellML `<model>` element: the `<units>`, `<component>`, `<group>`, and `<connection>` elements. Also, we haven't really discussed recursive importing of models, i.e. a model imported into an imported model. We're saving conversations about how to deal with recursive importation for another day.

3 The `<model_import>` Element

Since we've come to the conclusion that a `<model>` element is simply a container element, a "CellML model" does not have any singular requirements (besides the rules outlined in the CellML Specification[3]). In other words, no CellML elements are *necessary* before a model becomes a "CellML model", and a "CellML model" is *sufficient* even if it only contains one child CellML element so long as it is still valid CellML. By this reasoning, we feel that we will have to make a restrictive change to the next version of the CellML specification because Section 3.2.2[4] of the CellML 1.0 Specification states that a `<component>` element may be used as the root element of a CellML document. In the future, a `<model>` element must be the root element of any CellML document.

Two other minor clarifications: first, a model can only import another valid CellML model. Software, of course, would be expected to check for errors in the imported model just as it would for the current model. Secondly, the `name` attribute on the element does not have to be the value of the name attribute of the `<model>` root element.

[3] http://www.cellml.org/public/specification/cellml_specification.html

[4] http://www.cellml.org/public/specification/cellml_specification.html#sec_structure_component_element