Meeting Minutes 26 July 2002
More 1.1 Import Problems / Solutions

Author:
Autumn A Cuellar (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Warren Hedley (Alliance for Cellular Signaling)
Catherine Lloyd (Bioengineering Institute, University of Auckland)
Melanie Nelson (CellML alumna)
Poul Nielsen (Bioengineering Institute, University of Auckland)
David Nickerson (Bioengineering Institute, University of Auckland)

1 Introduction

These minutes explore current problems with the import feature as discovered upon investigation of Catherine Lloyd's examples[1] implementing the rules of import currently set forth in the developer's CellML 1.1 Specification[2]. We also carry on discussions introduced in last week's meeting minutes[3].

2 Current Modelling Problems with 1.1

Catherine had a go at coding up some of the models she's coded in CellML 1.0 with the new import features of CellML 1.1. These would, of course, enable us to test our theories and maybe encounter problems we hadn't anticipated or hadn't worked out. I've only just had a chance to look To illustrate the first of these, we'll have a look at the Zeng *et al* re-use[4] of the Luo-Rudy II model. Figure 1 shows the three model treatment of just one of the components, the `"fast_sodium_current"`. The Luo-Rudy II model is a standard CellML 1.0 model; it can stand alone as a complete model. The `"fast_sodium_current"` component receives variables `"V"`, `"R"`, `"T"`, and `"F"` from the `"membrane"` component and variables `"Nao"` and `"Nai"` from the `"ionic_concentrations"` component. Using these passed in variables, the `"fast_sodium_current"` calculates `"E_Na"` and `"i_Na"`. The variable `"i_Na"` is then passed back to the `"membrane"` and `"ionic_concentrations"` components to re-calculate the membrane potential (`"V"`) and the sodium ionic concentrations (`"Nao"` and `"Nai"`).

The Zeng model reuses the `"fast_sodium_current"` component. Currently we simply import the calculated `"i_Na"` value through the `"import"` component to the Zeng `"membrane"` component. This `"i_Na"` value is used to calculate a new membrane potential. The problem is that the `"i_Na"` value obtained from the Luo-Rudy model is based on Luo-Rudy II membrane potentials and ionic concentrations which could be different from the Zeng membrane potential, especially if you run the model through several cycles of simulation. I think we either need a way of re-connecting the necessary component or explicitly stating how we expect the component to behave when we import a value to another component. For instance, does the entire Luo-Rudy II model run when you run the Zeng model?

To fix the above problem, we decided that a model should be split up into separate components with the foresight that another modeller may choose to use any one of the components in your model on its own. For example, the Luo-Rudy II model mentioned above would not be one whole model standing alone, it'd be a supermodel that imports a model that contains the `"fast_sodium_current"` component and its

[1]http://www.cellml.org/examples/examples/CellML_1.1/index.html
[2]http://www.cellml.org/private/unstable_cellml_spec/cellml_specifi cation.html
[3]http://www.cellml.org/private/progress_reports/20020719_meeting_minutes.html
[4]http://www.cellml.org/examples/examples/CellML_1.1/old_reuse_models_doc/zeng_model_1995_doc.html

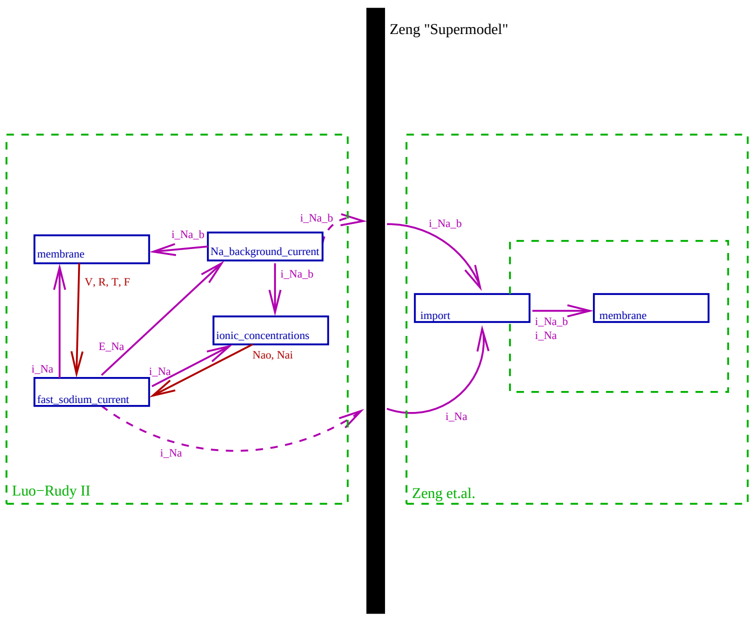

FIGURE 1: Component diagram of the Zeng and Luo-Rudy II models.

children components only and another model that contains the `"ionic_concentrations"` component only, etc. Then, since the LRII model would be made up of many separate sub-models, it'd be easy for a modeller coding up the Zeng model may pick and choose which component-models to re-use and code up new component-models that he/she needs.

At this point there are two clarifications we need to make:

- We'll stress again - when a model is imported, the entire model is imported, creating a new instance of the model, and will be run in its entirety by the application.
- Change to CellML notation (DTD) - a model may contain zero or more other **`<model>`** elements.

3 Import Element Name

We've again touched on the issue of the import element's name: **`<import_model>`** or just plain **`<import>`**? Do we expect a CellML model to be able to import anything besides another model? We've discussed this, and, no, we don't intend to allow a model to import anything besides another model. So whether we use **`<import_model>`** or **`<import>`** is simply a question of semantics. We do not want to imply that we might soon add an element **`<import_x>`** to CellML, the x a variable standing for who knows what (java_applet, XSL_script, etc, etc.), however, we do want to make it clear to new users or human readers that it is another CellML model being imported. I'm still a proponent of the **`<import_model>`** element.

4 Deep References

Team CellML has also decided that *deep references*, as we've come to call them, or references to imported models of imported models, will not be allowed. A model can only see the models it imports itself, I guess in much the same way that an encapsulating component cannot see the encapsulated components of its child. I'm worried that this decision will come back and bite us, but it seems easy enough to get around it- you can just import the model that your imported model also imports. Now that I think about it, we may need to put similar restrictions on importing models that we do on encapsulation so that we don't have cyclic imports (importing models that import the current model). I'll have to think about that for a minute.

5 `model` Attribute

As far as the whole `model` attribute is concerned, I'm still not happy with it. As Warren pointed out, "It is completely unintuitive what the semantics of the attribute are". It is not common in XML for an attribute to qualify another attribute rather than the element it's placed on. However, since we're not allowing deep references, I don't really have much of an excuse to not allow it. I don't think the use of the `model` attribute is any more XML-happy than the dot notation. But I cannot think of a third way to handle the references, so attribute it is, I guess.