Meeting Minutes 19 July 2002

CellML 1.1 Namespaces & Identifier Issues

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

1 Introduction

Still ironing out problems with CellML 1.1 Specification. Warren had a look at what I'd written up for the new Importing Models[1] section and had some constructive criticism to offer. In the following sections we discuss the two main issues concerning the 1.1 Specification that need to be addressed: version information in namespaces and how to identify imported models.

2 CellML 1.1 Namespace Issue

Last year Warren consulted the XML community (see the xml-dev discussion group archives[2]) asking for suggestions of how to contain version information in an XML document. This is a much debated topic, as seen by the number of methods the XML community has come up with to deal with this information. As Warren points out, the Dublin Core uses different namespaces to represent version changes, the W3C's XSLT language uses a **`version`** attribute on the root element, and the MathML language uses neither of the above. Practically every method has its advantages and its drawbacks, but since there is no industry standard of how to encapsulate changes to a language, each group is left on its own to solve this messy little problem.

Warren chose to use the Dublin Core solution: each new version of CellML should belong to its own CellML namespace. As far as I can tell, his motivation was the vision that CellML libraries could be composed of incomplete XML documents, documents that did not have root elements, generally components of oft used biological compartments, species, etc, that could be re-used and incorporated into full models. For incomplete XML documents, the standard document type declaration identifying the document to validate against (generally a decent tip to version information on its own) would be inapplicable. Furthermore, since he anticipated the use of CellML components not contained in a root **`<model>`** element, he wanted to relay the version information without having to tag a **`version`** attribute to every element.

So how does Warren's decision of last year affect us this year? Well, it leaves us with the main drawback that people complain about when mentioning the namespaces-as-versioning technique: XPath information gets messed up (yep, that's the technical explanation). The problem is that namespaces provide a unique name for your elements and attributes so that an application built to understand the CellML 1.0 language not only doesn't understand the newly introduced elements, but it doesn't understand any of the elements because **`<http://www.cellml.org/cellml/1.1#connection>`** is not the same element as **`<http://www.cellml.org/cellml/1.0#connection>`**.

The whole point of these ramblings is that we realized we needed to 1) decide if the changes that we had made in the CellML Specification were enough to warrant a namespace update (since we could have made

[1] http://www.cellml.org/private/unstable_cellml_spec/import_model.html
[2] http://lists.xml.org/archives/xml-dev/200105/msg00037.html

the changes under the same namespace as the MathML folks did) and 2) clarify treatment of new elements under same or other namespace in Specification depending on what we decided. After some discussion, our conclusions were that the entire CellML language would be redefined under the new 1.1 namespace, but applications that understand CellML 1.1 should also be written to understand 1.0.

3 CellML 1.1 Identifier Dot Notation

Poul has been expressing concern over our proposed use of the dot notation to identify models from which we are importing parts. He and I recently attended a workshop held by the SBML people, and at this two day meeting, they spent a good deal of time debating on the limitations of the identifiers, which, like ours, consist of only alphanumeric characters and the underscore (_). I won't go into the details of the debate, but in the end they decided to use two identifiers, one using the current notation to be used by the computer, the other expanded to include any Unicode character to be used by the user.

Following these long discussions, Poul became more convinced that if we want to keep our identifier names in line with the SBML group's, the dot notation that we had originally quite liked might cause us difficulty in the future. His suggestion is to tag a **`model`** attribute to any of the elements that may use an imported feature (or in the case of the connection elements have a **`model_1`** and **`model_2`** attribute) as shown in Figure 1.

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

<component name="membrane">
  <variable name="Cm" units="microF" model="units_vocabulary" />
  <variable name="I_st" units="microA_per_microF" model="units_vocabulary" />

  ...

</component>
```

FIGURE 1: The use of a **`model`** attribute instead of the formerly proposed "dot notation". Would obviously have to clarify the meaning of the **`model`** attribute in spec, i.e. the units are from the model `"units_vocabulary"`, not the variable.

I for one am not entirely convinced this is the way to go for two reasons:

- Poul's argument that we need to remain somewhat consistent with the SBML group has been for the most part rendered invalid because they have chosen to keep an internal identifier that is the same as what we are currently using. Thus, the *dot notation* does not find implementation problems in SBML.
- Use of a **`model`** attribute makes it difficult to refer to what I'll call *inherited models*, models imported by imported models.

David B. and I had a brief conversation about how to get around this, and he suggested that there should be some logical way to accomplish what the dot notation did without using the dot. In other words, instead of `a.b.c` you have (a, (b, c)) where the brackets indicate placement of XML start and end tags. See Figure 2 for an example of how to encode this notation. If you can think of a better way, please let us know.

Tune in to the next riveting installment of CellML Meeting Minutes to see which solution we accept.

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

<component name="membrane">
  <variable name="Cm">
    <model_ref ref="complex_units_vocabulary">
      <model_ref ref="simple_units_vocabulary" units="microF" />
    </model_ref>
  </variable>
  <variable name="I_st">
    <model_ref ref="complex_units_vocabulary" units="microA_per_microF" />
  </variable>

  ...

</component>
```

FIGURE 2: In this example, the model imported referred to as `"complex_units_vocabulary"` itself imports another model referred to by the name `"simple_units_vocabulary"`.