# Meeting Minutes 21 March 2001
# Connections and Grouping

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

## 1 Introduction

These meeting minutes document the final decision on the data model and syntax for connections and grouping in CellML. The decision was based on Melanie's meeting minutes of March 13 and 14 which contained some proposed schemes with analysis and discussion thereof. Subsequent meetings in Auckland and an urgent deadline forced the final decision to be made.

## 2 Connections In CellML

The syntax which will appear in the CellML 1.0 specification for a connection is shown in Figure 1.

```
<connection>
  <map_components component_1="membrane" component_2="na_channel" />

  <!-- Variables may have the same name in each component ... -->
  <map_variables variable_1="voltage" variable_2="voltage" />

  <!-- ... or have different names. -->
  <map_variables variable_1="na_current" variable_2="current" />
</connection>
```

FIGURE 1: The syntax for defi ning a connection between two components in CellML 1.0.

The decision to go with this syntax was based on the following sequence of decisions (more or less).

1. In the syntax in the 2001-03-02 draft of the CellML specification, the mapping of variables was dependent on the ordering of variable references in two locations, a system that was potentially error-prone for hand-coders of CellML documents. This led to the introduction of a **<map_variable>** element. The **<map_variable>** element was accepted because it (a) made it easy to see which variables were being mapped together, and (b) was concise. (Longer alternatives were considered and rejected.)
2. The syntax still had the **<component_ref>** elements, which now were inconsistent with the **<map_variab** elements, exhibiting the same ordering problem (although not in a confusing way, because there is always only two of them.) So this was changed to a **<between_components>** element with **component_1** and **component_2** attributes.

3. PFN and DPB had problems with the naming of the **<between_components>** element (which they thought was not informative enough) and suggested that the connection consist of a **<component_map>** element and multiple **<variable_map>** elements. WJH suggested that these "noun-based" element names suggested that the contents of element would be a complete map, which was not true. The "verb-based" element names that we ended up with do not have this association, and thus represent more accurately what is actually happening.

The solution is attractive because:

- It is pretty obvious what's going on.
- The element names are meaningful and accurate.
- The scheme is concise.
- The referencing scheme is internally consistent.
- It removes data model clashes with the **<variable_ref>** element that appears within **<reaction>** elements, and the **<component_ref>** element that appears in **<group>** elements, which were making it tricky to write a schema.

One issue that had concerned us during the syntax re-arrangement was whether a **<connection>** element must contain a **<map_variables>** element or not. The final decision was that it did, because it was unclear what the behaviour associated with an empty connection should be. We do not want applications to use empty connections to imply some sort of relationship, essentially compromising the portability of those documents.

### 2.1 Melanie's Objection

Just for the record, Melanie is not a big fan of the final syntax. She would prefer the **<map_components>** element to have been called **<between_components>** as was originally suggested. Unfortunately, Melanie was unavailable when the final decision was made, so tough cookies!

### 2.2 Poul's Objection

Also just for the record, Poul still thinks the **<map_components>** and **<map_variables>** elements should be renamed **<component_map>** and **<variable_map>** respectively. However, since he's not in New Zealand at the moment, tough cookies!

## 3 Grouping

It was generally agreed that the grouping syntax proposed in the 14 March meeting minutes was agreeable to all. The example of the syntax given in that document is reproduced in Figure 2.

The key features/rules of this syntax are:

- **<component_ref>** elements can be nested, allowing the specification of an entire hierarchical structure within a single **<group>** element.
- A **<group>** element may contain multiple **<relationship_ref>** elements. This implies that all specified relationships hold between all of the components referenced within the group.
- A **<group>** element may contain multiple **<component_ref>** elements at the top level. This implies that the referenced components are all at the top of hierarchical structures, or have equivalent status in the event that they do not contain child **<component_ref>** elements.

```
<group>
  <relationship_ref relationship="containment" />
  <component_ref component="cell">
    <component_ref component="cell_membrane">
      <component_ref component="sodium_current" />
      <component_ref component="calcium_current" />
    </component_ref>
  </component_ref>
</group>

<group>
  <relationship_ref relationship="encapsulation" />
  <component_ref component="cell_membrane">
    <component_ref component="sodium_current">
      <component_ref component="sodium_channel_1" />
      <component_ref component="sodium_channel_2" />
    </component_ref>
    <component_ref component="calcium_current">
      <component_ref component="L_type_calcium_channel" />
      <component_ref component="T_type_calcium_channel" />
    </component_ref>
  </component_ref>
</group>
```

FIGURE 2: The new CellML grouping syntax. The geometric containment group and the logical encapsulation group must be defined separately, since they only partially overlap.

- The different parent-child groupings that make up a hierarchy sharing the same relationship may be split up over multiple **<group>** elements. This allows submodels that include some hierarchical structure information to be pulled from a database and combined "as is" to form a larger model.
- The two relationships that CellML defines are now called **encapsulation** and **containment**.

This syntax rocks because:

- It's pretty obvious what's going on.
- An entire hierarchy can be stored in one place, making CellML documents much more human-readable.
- It removes the pesky major/minor component problem.
- It is only really a minor syntactical change from the current system.

The downside is:

- Pretty much the entire grouping section of the draft specification has to be rewritten.