CellML Specification
Draft — 2 March 2001

3 Model Structure

3.1 Introduction

Any model can be described as a network of connections between self-contained components. A component is a functional unit that corresponds to a physical compartment, event, or species or that is just a convenient modelling abstraction. Components contain variables and mathematical relationships that manipulate those variables. Connections contain mappings between the variables of connected components.

3.2 Basic Structure

3.2.1 Definition of a model

A CellML model may be a complete, functional model; an incomplete model; or a partial model (as defined in Section 1.1.3[1]). A model is declared with the **`<model>`** element. This is the usual root element for a CellML document. The **`<model>`** element for the simple electro-physiology example[2] from the examples section of the CellML website is shown in Figure 4.

```
<model
    name="simple_electro_physiological_model"
    xmlns="http://www.cellml.org/2001/03/cellml"
    xmlns:cellml="http://www.cellml.org/2001/03/cellml">

  ...

</model>
```

FIGURE 4: The root element of an XML document containing a CellML model description is the **`<model>`** element shown above.

The **`<model>`** element has a **`name`** attribute that allows this model to be unambiguously referenced by other models. For instance, this would be necessary if the model were to be combined with other models or partial models to create a larger model. The namespace declarations on the **`<model>`** element shown in Figure 4 are discussed in Section 2.2.3[3].

A **`<model>`** element may contain any number of the elements in the following list in any order. The recommended practice is for elements placed within the **`<model>`** element to appear in the order given in the following list. This allows people to quickly find certain kinds of information within a CellML document.

- **`<units>`** — A modeller can declare a set of units to use in the model, as described in Section 5[4].

[1]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_intro_definition_of_a_model
[2]http://www.cellml.org/examples/examples/electrophysiological_models/basic_ep_models/basic_ep_model_doc.html
[3]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_fundamentals_extending_cellml
[4]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_units

- **`<component>`** — Components are the smallest functional units in a model. Each component contains variables that represent the key properties of the component and mathematics that describe the behaviour of the portion of the system represented by that component.
- **`<group>`** — Groups allow the modeller to define logical and physical relationships between components. Groups are defined using the **`<group>`** element, as discussed in Section 6[5].
- **`<connection>`** — Connections are used to connect components to each other and to map variables in one component to variables in another. Connections are defined using the **`<connection>`** element, as discussed in Section 3.2.4.

The **`<model>`** element (and indeed any of the elements in a CellML document) may define metadata to provide context for that object. This metadata might include documentation, citations from literature, or a modification history for the current CellML object. Adding metadata to a CellML document is discussed in detail in Section 8[6].

3.2.2 Definition of components

Constructing a model from multiple components encourages the re-use of components. For instance, an electro-physiological model of a cell might be organised into components that represent various ion channels. All of the mathematics that describe the behaviour of the L-type calcium channel would be defined in a single component representing this particular ion channel. If a modeller wished to re-use the portion of the model representing the L-type calcium channel in another model, he or she would only need to copy this component.

A **`<component>`** element is used to declare a CellML component. It may only be used inside a **`<model>`** element or as the root element of a CellML document. A **`<component>`** element that is the root of a CellML document does not define a complete model. It would probably be part of a library of standard components that could be imported and used in models. Eventually, CellML will include a mechanism that simplifies such re-use of components. At the present time, the component would need to be physically copied into a model document to be used in that model.

A CellML **`<model>`** may contain any number of **`<component>`** elements. Each **`<component>`** must have a **`name`** attribute, the value of which is a unique identifier for the component within the current model. The value of the **`name`** attribute is used to reference the component in other parts of the model, such as in connections and groups.

A **`<component>`** may contain any of the elements in the following list in any order. Again, recommended practice is for elements placed within the **`<component>`** element to appear in the order given in the following list.

- **`<units>`** — A modeller can declare a set of units to use within the component, as described in Section 5[7].
- **`<variable>`** — A component may contain any number of **`<variable>`** elements, which define variables that may be mathematically related in the equation blocks contained in the component. Variables are discussed in Section 3.2.3.
- **`<reaction>`** — A component may contain **`<reaction>`** elements, which are used to provide chemical and biochemical context for the equations describing a reaction. It is recommended that only one **`<reaction>`** element appear in any **`<component>`** element. The definition of reaction information is described in Section 7[8].

[5]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_grouping
[6]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_metadata
[7]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_units
[8]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_reactions

- **`<math>`** — A component may contain a set of mathematical relationships between the variables declared in this component. These equations are marked up using MathML, as discussed in Section 4[9].

A **`<component>`** element is also a sensible place to define metadata, using the syntax presented in Section 8[10].

The definitions of two **`<component>`** elements are included in the example described in Section 3.3.

3.2.3 Definition of variables

Models are usually developed to simulate the behaviour of a number of variables that have physiological significance. Each variable in the model belongs to a single component, which may contain equations or scripts that modify the value of that variable. The value of a variable may be passed through connections into other components. The variable must also be declared in these components, which can then use the value of the variable in their own equations and scripts but may not modify it.

The **`<variable>`** element is used to declare a CellML variable. It can only be used inside a **`<component>`** element. Variables must define a **`name`** attribute, the value of which must be unique across all variables in the current component. The name of a variable is used when referencing variables inside connections (see Section 3.2.4) and reactions (see Section 7[11]). All variables must also define a **`units`** attribute. The value of this attribute must correspond to one of the keywords in the CellML units dictionary or the value of the **`name`** attribute of a **`units`** element defined within the current component or model, as described in Section 5[12].

A **`<variable>`** element may also have the following attributes:

- **`initial_value`** — This attribute provides a convenient means for specifying the initial or default value of a scalar variable in a simulation with time as the independent variable. The variable's value may be reset or modified in equations or scripts in the current component. The initial values of variables need not be set in the model definition at all; they could instead be set in a configuration file loaded separately by the model processor.
- **`public_interface`** — This attribute specifies the interface exposed to components in the parent and sibling sets (see below). The public interface may have a value `"in"`, `"out"`, or `"none"`. The absence of a **`public_interface`** attribute implies a value of `"none"`.
- **`private_interface`** — This attribute specifies the interface exposed to components in the encapsulated set (see below). The private interface may have a value `"in"`, `"out"`, or `"none"`. The absence of a **`private_interface`** attribute implies a value of `"none"`.

When a variable is declared with either a **`public_interface`** or **`private_interface`** attribute value of `"in"`, then the value of that variable can be imported from another component. Otherwise, a variable's value must be set and modified in the current component. The variable is then said to *belong to* or be *owned by* the current component.

Whether or not a component may obtain the value of a variable in another component depends on the **`public_interface`** and **`private_interface`** attributes on the variable declaration, and the place of the two components in the encapsulation hierarchy. Encapsulation allows the modeller to hide a complex network of components from the rest of the model and provide a single component as an interface to the hidden network. Encapsulation effectively divides the network into layers, where connections between the layers may only be made through the interface components.

[9]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_mathematics
[10]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_metadata
[11]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_reactions
[12]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_units

The components to which any given component may connect can be divided into three distinct classes. The set of all components encapsulated by the current component is referred to as the *encapsulated set*. If the current component is encapsulated, then the encapsulating component is referred to as the *parent*, and the set of all other components encapsulated by the same parent is referred to as the *sibling set*. If the current component is not encapsulated, then it has no parent and the sibling set consists of all other components in the model that are not encapsulated. The encapsulation hierarchy and its effects on variable mapping are described in Section 6[13].

Eventually, it will be possible to specify the temporal and/or spatial variation of a variable's value using FieldML[14]. The capability to include FieldML is still under development. At the present time, all variables must have singular values.

3.2.4 Definition of connections

Connections provide the mechanism for mapping variables declared within one component to variables in another component, allowing information to be exchanged between the various components in the network. There will be many such mappings present in a network. The mapping of variables involves the transfer of a variable's value from one component to another. This transfer may involve a conversion to account for the units in which each component expects the variable's value to be defined. (More information on units conversion can be found in Section 5[15].)

The complete set of variable mappings between any two components constitutes a connection. Only one connection may be created between any two components in a model. Each connection contains a pair of component references, indicating the two components involved in the connection. The two component references each contain an ordered list of variable references. Each variable contained in the first ordered list is mapped to the corresponding variable contained in the second ordered list. Mapping depends only on the order of the **`<variable_ref>`** elements in the two **`<component_ref>`** elements. It is not necessary for the variables that are to be mapped to each other to have the same name. However, the interface attributes of each pair of variables must be compatible — an `"out"` variable in one component's interface must map to an `"in"` variable in the other component's interface. The direction of each mapping is determined by the value of the **`public_interface`** attributes on the two variables: the value is always passed from the variable with an interface value of `"out"` to the variable with an interface value of `"in"`. The value of a variable declared with an interface value of `"out"` may be passed out to any number of variables in other components declared with interface values of `"in"`. The component to which a variable belongs is found by following the variable back from `"in"` to `"out"` interfaces defined by the model's connections.

The **`<connection>`** element is used to declare a CellML connection. It can only appear inside a **`<model>`** element.

Two **`<component_ref>`** elements are used to reference the components involved in the connection. Each **`<component_ref>`** element must define a **`component`** attribute, the value of which is the name of the component being referenced. Currently, this component must be defined within the current **`<model>`** element. It is anticipated that it will eventually be possible to reference components from other models, allowing models to be connected into larger models.

The **`<variable_ref>`** element is used to reference the variables being mapped between the two components in the connection. Each **`<component_ref>`** element may contain multiple **`<variable_ref>`** elements, but the number of **`<variable_ref>`** elements in each of the two **`<component_ref>`** elements within a **`<connection>`** must be equal. Each **`<variable_ref>`** element must define a **`variable`** attribute, the value of which is equal to the **`name`** attribute value of a variable declared in the component referenced by the parent **`<component_ref>`** element.

[13] http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_grouping
[14] http://www.physiome.org.nz/sites/physiome/fieldml/pages/index.html
[15] http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_units

The CellML example discussed in Section 3.3 demonstrates the definition of a **`<connection>`** element.

3.3 Examples

Figure 5 contains a portion of the CellML encoding of the Hodgkin-Huxley squid axon model published in 1952. The excerpt contains the definitions of the components corresponding to the membrane and the sodium channel, and the connection between the two components. Most of the complexity from the full model definition has been left out for conciseness and clarity. This example is only used to demonstrate the standard use of the **`<component>`**, **`<variable>`**, and **`<connection>`** elements.

The **`membrane`** component declares four variables, which are divided into four types. The first variable is called **`V`**, and it represents the membrane voltage in the model. It has a **`public_interface`** attribute value of `"out"`, which indicates that the variable "*belongs*" to this component and that its value may be obtained by other components in the model via connections. It references a units definition by the name of **`millivolt`** (this definition is not included here), and is given an initial value of -75.0 millivolts.

The second and third variables are **`time`** and **`I_Na`** (sodium current). They are both declared with a **`public_interface`** attribute value of `"in"`, which indicates that their value is obtained from another component via a connection. Finally, a variable **`C`** (capacitance) is declared. This **`<variable>`** element defines neither a **`public_interface`** or a **`private_interface`** attribute. Both of these attributes therefore assume the default value of `"none"`, which means that the variable belongs to the current component and is not visible to other components in the model.

After the variable declarations, a **`<math>`** element in the MathML namespace is used to define an equation relating **`V`** to the other variables. Only the values of the variables belonging to a component may be mathematically modified in that component. The equation that appears in the full model is too lengthy to include here. Instead, a nonsense equation is included to demonstrate the use of mathematics in components. This equations is:

$$^{d\mathtt{V}}/_{d\mathtt{time}} = {}^{\mathtt{i\_Na}}/_{\mathtt{C}}$$

The **`sodium_channel`** component declares three variables, all of which represent quantities that were also declared in the membrane component. The **`I_Na`** variable declared in this component has a **`public_interface`** attribute value of `"out"`, indicating that the sodium current belongs to this component. The value of the sodium current is calculated in this component, although the actual math has been omitted.

Finally, a **`<connection>`** element references the **`membrane`** and **`sodium_channel`** components (using **`<component_ref>`** elements) and maps the **`V`** and **`I_Na`** variables in each component together. Variables are mapped according to the order of **`<variable_ref>`** elements within each **`<component_ref>`** element.

3.4 Rules for CellML Documents

The following are the rules for using the **`<model>`**, **`<component>`**, **`<variable>`**, **`<connection>`**, and **`<component_ref>`** elements, and **`<variable_ref>`** elements within **`<component_ref>`** elements.

3.4.1 The `<model>` element

1. **Allowed use of the `<model>` element**

   - A **`<model>`** element must contain only the following elements, which may appear in any order:

```
<model
    name="hodgkin_huxley_model_excerpt"
    xmlns="http://www.cellml.org/2001/03/cellml">

  <component name="membrane">
    <!-- the following variable is used in other components -->
    <variable
        name="V" initial_value="-75.0"
        public_interface="out" units="millivolt" />
    <!-- the following variables are imported from other components -->
    <variable name="time" public_interface="in" units="millisecond" />
    <variable name="i_Na" public_interface="in" units="microA_per_cm2" />
    <!-- the following variable is only used internally -->
    <variable name="C" initial_value="1.0" units="microF_per_cm2" />
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply><eq />
        <apply><diff />
          <bvar><ci> time </ci></bvar>
          <ci> V </ci>
        </apply>
        <apply><divide />
          <ci> i_Na </ci>
          <ci> C </ci>
        </apply>
      </apply>
    </math>
  </component>

  <component name="sodium_channel">
    <variable name="i_Na" public_interface="out" units="microA_per_cm2" />
    <variable name="time" public_interface="in" units="millisecond" />
    <variable name="V" public_interface="in" units="millivolt" />
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply cellml:name="i_Na_calculation"><eq />
        <ci> i_Na </ci>
        ...  <!-- a function of V & time -->
      </apply>
    </math>
  </component>

  <connection>
    <component_ref component="membrane">
      <variable_ref variable="V" />
      <variable_ref variable="i_Na" />
    </component_ref>
    <component_ref component="sodium_channel">
      <variable_ref variable="V" />
      <variable_ref variable="i_Na" />
    </component_ref>
  </connection>

</model>
```

FIGURE 5: A small portion of the CellML definition of the Hodgkin-Huxley squid axon model from 1952. This excerpt contains the definition of the components corresponding to the membrane and the sodium channel, and the connection between them. Much detail has been omitted, but this example clearly demonstrates the relationship between the `<component>`, `<variable>` and `<connection>` elements. See the text for more information.

  - `<units>`, `<component>`, `<connection>`, and `<group>` elements in the CellML namespace,
  - metadata framework elements, as described in Section 8[16].

  [Recommended practice is to define the CellML namespace child elements in a `<model>` element in the order stated above.]

- Each `<model>` element must define a `name` attribute.

2. **Allowed values of the `name` attribute**

- The value of the `name` attribute must be a valid CellML identifier as discussed in Section 2.2.1[17].

3.4.2 The `<component>` element

1. **Allowed use of the `<component>` element**

- A `<component>` element must contain only the following elements, which may appear in any order:
  - `<units>` and `<variable>` elements in the CellML namespace,
  - `<math>` elements in the MathML namespace,
  - metadata framework elements, as described in Section 8[18].

  [Recommended practice is to define the CellML and MathML namespace child elements of a `<component>` element in the order stated above. Note that it a `<component>` element must not appear inside another `<component>` element. Such nesting could be intended to indicate a logical encapsulation relationship, a geometric containment relationship, or some other relationship between the two components. There is no reason to assume that the nesting hierarchy produced for one type of relationship would be consistent with the hierarchy produced for other types of relationship. Therefore, CellML defines these relationships using the `<group>` element, rather than nesting of `<component>` elements.]

- Each `<component>` element must define a `name` attribute.

2. **Allowed values of the `name` attribute**

- The value of the `name` attribute must be a valid CellML identifier as discussed in Section 2.2.1[19].
- The value of the `name` attribute must be unique across all `<component>` elements contained in the same `<model>` element.

3.4.3 The `<variable>` element

1. **Allowed use of the `<variable>` element**

- A `<variable>` element must contain only the following elements, which may appear in any order:
  - metadata framework elements, as described in Section 8[20].

[16]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_metadata
[17]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_fundamentals_identifiers
[18]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_metadata
[19]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_fundamentals_identifiers
[20]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_metadata

- Each `<variable>` element must define a `name` attribute and a `units` attribute. It may also define `public_interface`, `private_interface`, and `initial_value` attributes.

2. **Allowed values of the `name` attribute**

   - The value of the `name` attribute must be a valid CellML identifier as discussed in Section 2.2.1[21].
   - The value of the `name` attribute of a `<variable>` element must be unique across all `<variable>` elements contained in the same `<component>` element.

     [Two variables in the same component may not have the same name. However, two variables in different components can have the same name, and a variable can have the same name as its parent component.]

3. **Allowed values of the `units` attribute**

   - The value of the `units` attribute must either be one of the keywords defined in the standard dictionary or the value of the `name` attribute on a `<units>` element defined in the current component or model.

     [The dictionary and the units element are described in Section 5[22].]

4. **Allowed values of the `public_interface` attribute**

   - If present, the value of the `public_interface` attribute must be `"in"`, `"out"`, or `"none"`.
   - If not present, its value defaults to `"none"`.

5. **Allowed values of the `private_interface` attribute**

   - If present, the value of the `private_interface` attribute must be `"in"`, `"out"`, or `"none"`.
   - If not present, its value defaults to `"none"`.

6. **Proper use of the `public_interface` and `private_interface` attributes**

   - A `<variable>` element must not define both `public_interface` and `private_interface` attributes with values equal to `"in"`.

     [A variable's value may only be obtained via one mapping.]

7. **Allowed values of the `initial_value` attribute**

   - If present, the value of the `initial_value` attribute must be a real number.
   - The absence of an `initial_value` attribute implies nothing.

     [The absence of this attribute would usually mean either that the variable does not need an initial value or that this value will be supplied in a parameter file or by the user at the time simulations using the model are run.]

8. **Proper use of the `initial_value` attribute**

   - An `initial_value` attribute must not be defined on a `<variable>` element with a `public_inter` or `private_interface` attribute with a value of `"in"`.

     [These variables receive their value from variables belonging to another component.]

[21]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_fundamentals_identifiers
[22]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_units

3.4.4 The `<connection>` element

1. **Allowed use of the `<connection>` element**

   - A `<connection>` element must contain only the following elements, which may appear in any order:
     - `<component_ref>` elements in the CellML namespace,
     - metadata framework elements, as described in Section 8[23].
   - Each connection element must contain exactly two `<component_ref>` elements.

3.4.5 The `<component_ref>` element in a `<connection>` element

1. **Allowed use of the `<component_ref>` element in a `<connection>` element**

   - A `<component_ref>` element in a `<connection>` element must contain only the following elements, which may appear in any order:
     - `<variable_ref>` elements in the CellML namespace,
     - metadata framework elements, as described in Section 8[24].
   - Each `<component_ref>` element in a `<connection>` element must contain at least one `<variable_ref>` element.
   - The two `<component_ref>` elements in a `<connection>` element must contain an equal number of `<variable_ref>` elements.
   - Each `<component_ref>` element must define a `component` attribute.

2. **Allowed values of the `component` attribute**

   - The value of the `component` attribute must equal the value of the `name` attribute of a `<component>` element contained within the current `<model>` element.
   - Every `<connection>` element in a model must contain a unique pair of `component` attributes on the child `<component_ref>` elements.

     [There can only be one connection between any two components in a network. This prevents setting up inconsistent, circular, or duplicate variable mappings between any two components in the network. However, it does not prevent a model author from creating inconsistent mathematical relationships between the variables.]

3.4.6 The `<variable_ref>` element in a `<component_ref>` element

1. **Allowed use of the `<variable_ref>` element in a `<component_ref>` element**

   - A `<variable_ref>` in a `<component_ref>` element must contain only the following elements, which may appear in any order:
     - metadata framework elements, as described in Section 8[25].
   - Each `<variable_ref>` element must define a `variable` attribute.

2. **Allowed values of the `variable` attribute**

[23]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_metadata
[24]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_metadata
[25]http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_metadata

- The value of the `variable` attribute on a `<variable_ref>` element in a `<component_ref>` element must equal the value of the `name` attribute of a `<variable>` element contained in the `<component>` element referenced by the parent `<component_ref>` element.

3. **Proper use of the `<variable_ref>` element to map variables to each other**

   [The rules for mapping a variable to other variables depend on the encapsulation hierarchy of the component that owns the variable. This hierarchy divides the rest of the components in the model into *parent*, *sibling*, and *encapsulated* sets, as described in Section 3.2.3. The `public_interface` attribute defines the availability of a variable to the parent component and components in the sibling set. The `private_interface` attribute defines the availability of a variable to components in the encapsulated set.]

   - Variables with a `public_interface` or `private_interface` attribute value of `"in"` must be mapped to variables with a `public_interface` or `private_interface` attribute value of `"out"`.
   - A variable with either a `private_interface` or `public_interface` attribute value of `"in"` may be mapped to no more than one other variable in the model.

     [Note that a similar restriction does not apply to variables with interface values of `"out"`. An output variable can be mapped to multiple input variables in various components in the current model. It is up to the modeller to ensure that these mappings are consistent.]
   - A variable with a `public_interface` attribute value of `"in"` may be mapped to a single variable owned by a component in the sibling set, provided the target variable has a `public_interfac` attribute value of `"out"`, or to a single variable owned by the parent component, provided the target variable has a `private_interface` attribute value of `"out"`.
   - A variable with a `public_interface` attribute value of `"out"` may be mapped to variables owned by components in the sibling set, provided the target variables have `public_interface` attribute values of `"in"`. It may also be mapped to variables owned by the parent component, provided the target variables have `private_interface` attribute values of `"in"`.
   - A variable with a `private_interface` attribute value of `"in"` may be mapped to a single variable owned by a component in the encapsulated set, provided the target variable has a `public_interface` attribute value of `"out"`.
   - A variable with a `private_interface` attribute value of `"out"` may be mapped to variables owned by components in the encapsulated set, provided the target variables have `public_interface` attribute values of `"in"`.

3.5 Rules for Processor Behaviour

3.5.1 Mapping of variables

- The lists of `<variable_ref>` elements are ordered. The first `<variable_ref>` element in the first `<component_ref>` element maps to the first `<variable_ref>` element in the second `<component_ref>` element, and so on. Variable mappings specifically do *not* depend on variable names.