CellML Requirements

Authors:
Melanie Nelson (Physiome Sciences Inc.)
Warren Hedley (Bioengineering Institute, University of Auckland)
Contributors:
David Bullivant (Bioengineering Institute, University of Auckland)
Mark Grehlinger (Physiome Sciences Inc.)
Kam-Chuen Jim (Physiome Sciences Inc.)
Melanie Nelson (Physiome Sciences Inc.)
Dave Nickerson (Bioengineering Institute, University of Auckland)
Poul Nielsen (Bioengineering Institute, University of Auckland)
Prasad Ramakrishna (Physiome Sciences Inc.)
Jeremy Rice (Physiome Sciences Inc.)

This document details the user requirements for cell model documents that conform to v1.0 of the CellML standard. The main user group is biological modelers who will be creating cell models in CellML. However, the needs of biologists who will be running and viewing models created in CellML and programmers who will be writing programs that use CellML are also considered.

The requirements expressed in this document may be implemented using one of two primary methods: (1) by direct incorporation into the CellML data model (which will be serialized to CellML elements and attributes); (2) by including a separate language that handles the information required (as we have done with MathML). This document does not detail which method will be used, since this is not of interest to the biological modelers who are the main user group.

1 It must be possible to unambiguously specify in the CellML document all information needed to run or represent the model.

Not all models will contain math. These models are qualitative models constructed solely to help the user organize his information about a biological system. The CellML document must be able to contain all of the information needed by processing software to accurately represent the model. Note that this does not mean that the CellML document must contain exact instructions about how every piece of software must render the model. It means only that the CellML document must contain enough information to allow the software to apply its model rendering rules and create an accurate representation of the model.

When a model does contain math, the CellML document describing the model must contain all of the information needed by the software to run the model.

2 It must be possible to indicate the biological rules and definitions to which the model described in the CellML document adheres.

Users have requested the ability to define generic types of components and limit how components of a given type may be used in a model. This will speed up the model building process and facilitate the reuse of model components. However, it would be very difficult to define one universal set of rules and definitions that could be incorporated directly into the CellML data model, and even more difficult to get universal acceptance of these rules and definitions. Instead, we will make it possible to associate a set of rules and definitions with a model. These rules and definitions may be used by processing software to limit the use of certain model components and to support “biologically smart” behavior of the software during the model building process. For instance, Physiome Sciences might define a set of rules and definitions for cardiac modeling in which two types of components, channels and membranes, are defined. Physiome’s rules may specify that a component of type “channel” can only be used in conjunction with another component of type

"membrane". A software package could use these rules to require that each channel created in a new model be associated with a membrane. Other companies or labs might also define a set of rules and definitions for cardiac modeling, and there would be no requirement that these two sets of rules agree or even be compatible. It would be up to the user to choose a set of rules for his or her models to follow.

Note that these rules are only important during the process of building a model. Once the model is built, it must be possible to run the model without accessing the rules (to allow interoperability between software applications). How these rules and definitions are used is left entirely to the discretion of the processing software. Some applications may choose to ignore the rules, others may choose to strictly enforce them.

3 It must be possible to represent qualitative as well as quantitative information.

As stated in Section 1, some models will not actually contain math that would allow them to be solved. In these sorts of models, it must be possible to represent qualitative information such as "A activates B through some unknown pathway".

When biologists are first elucidating a signal transduction pathway, they draw diagrams indicating their understanding of which proteins are involved in the pathway, and how they interact. At this point, they probably won't have enough information to put in equations describing the mechanisms of interaction, but they will be able to draw a diagram that would show the known events in the pathway, such as the model shown in Figure 1 for the initial steps in vision (more information about this signal transduction pathway can be found in most biochemistry texts).

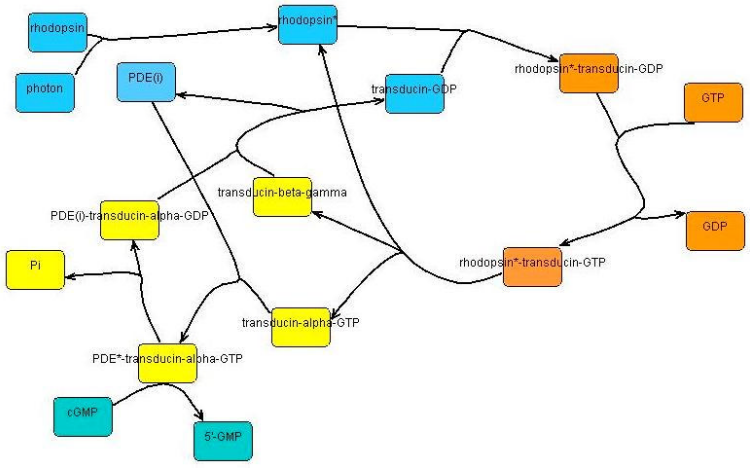

FIGURE 1: An example of a pathway model where the reaction pathways may contain no mathematical relationships, but contain only qualitative information.

The biologist will probably also be able to fill in some metadata about some of the elements in the pathway. We have a requirement to be able to store the information that would allow the exchange of these pathways. Note that we must be store enough information to allow a processing program to reproduce a correct representation of the pathway, but we do not want to store information about specific rendering details, such as whether the line connecting two elements is solid or dashed.

4 It must be possible to group portions of a model.

The modeler must be able to group a portion of the model. The organization of these groups must be at the discretion of the modeler, unless he is working under a set of rules that limits the allowable model organization (see Section 2). These groups will be called model components, or simply components, throughout the rest of this document.

4.1 It must be possible to create a logical hierarchy of components.

The modeler must be able to define a hierarchical organization of components. This allows the modeler to group together a set of related components and treat them as a single component. However, the detailed information about the individual components in the group must not be lost: the modeler must be able to "drill down" from the top level to the lower levels. There must not be a limit to the number of levels in the hierarchy.

Note that these hierarchies need not have any spatial connotation.

This requirement is particularly important for reaction pathways. For instance, consider the set of reactions:

```
A + B -> AB
AB + C -> ABC
ABC -> AB +D
```

It should be possible for the modeler to represent these as:

```
A + B + C -> AB + D
```

But the detailed breakdown of reactions should be preserved "underneath" this representation.

Suppose the modeler wishes to group the reactions shown in orange in Figure 1 into a single node in a new model (shown in Figure 2). This has the effect of hiding the details about how activated rhodopsin (rhodopsin*) catalyzes the decomposition of transducin into its subunits (transducin-α and transducin-$\beta\gamma$). Note that the details shown in Figure 1 are not removed, they are only hidden. The details must be preserved as "subreactions" of the reactions shown here.

4.2 It must be possible to encapsulate model components.

The modeler must be able to define the necessary inputs required to use a component, and define the output that will be produced by the component. Other modelers must be able to use this component without worrying about anything more than the required inputs and provided outputs if they choose. Note that it will also be possible for a modeler to take another modeller's encapsulated component and break the encapsulation. The CellML document must make it possible for the original modeler to create the encapsulation. The behavior of a particular processing application with respect to allowing later users to break the encapsulation is not part of the CellML specification, i.e., it will be left to the discretion of the application.

The pathways shown in Figure 1 and Figure 2 are only part of the signal transduction system that regulates vision. Given such a pathway with the necessary equations included, it must be possible for the modeler to encapsulate this pathway and allow another modeler to use it essentially as a "black box", and add it to a larger pathway. In the example shown, the inputs might be: the concentration of rhodopsin, transducin-GDP, GTP, PDE(i), and cGMP, as well as the wavelength and intensity of light. The outputs might be: the concentration of 5'-GMP, GTP, GDP, and Pi.

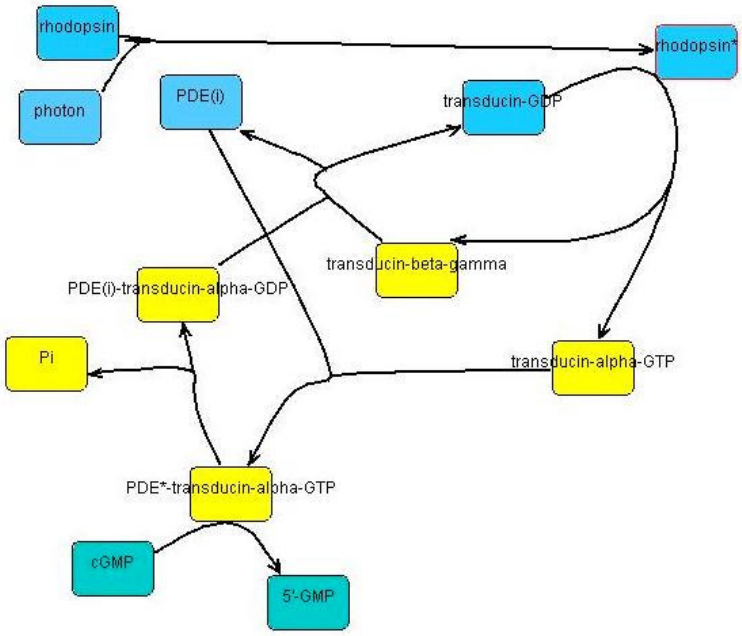

FIGURE 2: The pathway model from Figure 1, where the orange portion of the network has been encapsulated by a single node in the new model.

5 It must be possible to connect model components together. The ways in which model components may be connected must be left to the discretion of the modeler.

The modeler must be able to define connections between his model components in whatever way he deems appropriate. However, he may choose to build his model according to a previously defined set of rules and definitions (see Section 2) that would limit how model components may be connected. Note that it will be possible to pass variables between components via these connections.

6 It must be possible to define a component once and reuse it in multiple locations within the same model.

The modelers want to be able to reuse a component without resorting to physically copying that component. It must be possible to modify the values input into the component when it is reused.

7 It must be possible to define variable scope and access privileges.

For modeling convenience and model conciseness, modelers need to be able to define the scope and access privileges for their variables. Variable scope determines where the variable can be referenced, and the lifetime of the variable. Possible scope values include: local, global, function, file, and class. Variable access privileges determine which objects can access the variable. Possible values include: private, protected, friend, and public. Note that these values for scope and access privileges are included only as examples.

8 It must be possible to define a group of properties and to assign these properties to model components.

The modeler needs to be able to define a set of properties once and then assign these properties to as many components as he wants. Properties are things such as state variables, processes such as diffusion, and structural constraints.

9 It must be possible to define a hierarchical organization for access to properties.

The modeler needs to be able to specify that by default one component inherits a set of properties from another component.

For instance, the modeler might define two components: one that represents the cytosol, and another that represents the nucleus. The modeler may wish to represent the fact that some small molecules diffuse freely across the nuclear membrane by having the nucleus component inherit the concentrations of these molecules from the cytosol component.

10 It must be possible to represent certain types of biological information.

This information can be split into specific information needed to handle known models, and general information, which often supports storage of some portion of the specific information.

Specific Information

1. Electrogenic and non-electrogenic transport processes [Examples: Hodgkin-Huxley, Luo-Rudy, etc.]
2. Reaction pathways [Examples: Michaelis-Menten kinetics; Bhalla and Iyengar, Science 283, pp 381-387 (1999)]
3. Signalling pathways [Example: Bhalla and Iyengar, Science 283, pp 381-387 (1999)]
4. Cell/compartment structure [Example: Virtual Cell; Schaff et al, Biophys J. 73, pp 1135-1146 (1997)]

General Information

1. Spatial variation of state
2. Spatially-varying processes
3. Geometric relationships between components (such as component A is inside component B) [Examples for items 1-3: Jafri and Keizer, Biophys J. 69, pp 2139-2153 (1995); Jafri, J. Theor Biol 172, pp 209-216 (1995)]
4. Non-geometric relationships between components (such as component A modifies component B) [Example: Bhalla and Iyengar, Science 283, pp 381-387 (1999)]
5. Inhomogeneous/stochastic distribution of molecular species (when the number of molecules is too small to be handled with a continuous variable for concentration) [Example: Morton-Firth et al, J Mol Bio 286 pp 1059-1074 (1999); see also McAdams and Arkin, Trends in Genetics 15 pp 65-69 (1999)]

11 It must be possible to represent certain types of mathematical information.

Most Important

1. Ordinary differential equations (stochastic and deterministic) [Examples: deterministic = DiFrancesco-Noble, etc; stochastic = Predicting temporal fluctuations in an intracellular signaling pathway. Morton-Firth and Bray, J Theoret Biol (1998)]
2. Discrete and logical algorithms (includes Boolean) [Example of need for algorithms: Luo-Rudy (Ca2+-dynamics)]
3. Differential-algebraic expressions [Example: Modelling Myocardial Ischaemia and Reperfusion, Ch'en, Vaughan-Jones, Clarke, Noble, Proc. Bioph. (year?); Section 5.3 of Murray, "Mathematical Biology", 2 Ed., Springer (1993)]
4. Linear algebra/matrix math [Example: The Escherichia coli MG1655 in silico metabolic genotype: its definition, characteristics, and capabilities. Edwards JS, Palsson BO Proc Natl Acad Sci U S A 2000 May 9 97:10 5528-33]
5. Partial differential equations (stochastic and deterministic) [Example: Chapter 9 of Murray, "Mathematical Biology", 2 Ed., Springer (1993)]

Can Be Postponed

1. Petri Nets [Example: Hybrid Petri Net Representation of Gene Regulatory Network, Matsuno, Doi, Nagasaki, Miyano, Pacific Symposium on Biocomputing 5:338-349 (2000)]
2. Integral-differential equations [Example: Integro-differential equations and the stability of neural networks with dendritic structure. Bressloff PC Biol Cybern 1995 Aug 73:3 281-90]

12 The modeler must be able to define the units to be used in a model or model component.

A modeler must be able to choose his or her own units.

13 It must be possible to store the data produced by running a model and the configuration of the model and simulation conditions that produced this data.

This information is not part of the model itself, but the modelers want to be able to connect a data set to the model and initial conditions from which it was produced. The modelers need to be able to exchange this information with other users in conjunction with the model.

14 It must be possible to associate formatted text and graphical documentation with the model and its components.

This refers to unstructured documentation, as opposed to more structured metadata (which is dealt with in Section 16). The modelers want to be able to add formatting to their text documentation (such as bold or italicized text, headings, lists, etc.).

15 There must be a mechanism by which extensions can be safely added to CellML.

Groups developing software that uses CellML must be able to add extensions to the core language in such a way that other groups' software will not break when it reads a CellML file that uses the first group's extensions.

16 Metadata Requirements.

16.1 Models and Model Components.

This list indicates only the information identified by users as required. All references to "model" can also refer to a model component.

1. Name = primary name of the model
2. Alias = alternative name for the model (can have more than one)
3. Model builder = person responsible for coding the model in CellML (can have more than one)
4. Primary reference = report in which the model was originally described
5. Additional reference = reference that provides further information relevant to the model (can have more than one)
6. Species = the species for which the model is relevant
7. Sex = whether the model was defined for male, female, or either sex
8. Classification = classifying information about the model (Note: this will likely be linked in with the solution to the requirement for the ability to define a set of rules/definitions to which the model adheres)
9. Description = a short, unformatted text description of the model
10. CellML coding date = date on which the model was coded in CellML
11. Model-builder comments = comments of the person who coded the model into CellML
12. Limitation = brief text describing the limitations/scope of the model (can have more than one; each limitation needs to be associated with the author (who is not necessarily the model builder) and the date on which the limitation was noted)
13. Biological entity = the entity that the model represents (can be more than one)
14. Copyright = the copyright protecting the model
15. Publisher = the entity responsible for making the CellML model public
16. Modification = text describing a change made to a model (can have more than one; each modification needs to be associated with the model builder making the modification and the date on which the modification was made)
17. Derivation = the derivation of the model from another model or model component (can have more than one; each derivation must be associated with information about whether or not derivations were made to the parent model/component. Each derivation can also be associated with an optional comment about the derivation.)

18. Mathematical problem type = the type of mathematical problem encoded in the math associated with the model. The problem type could be identified using a controlled vocabulary or some other existing classification scheme.

Additional metadata may be accepted. For instance, if we choose to use the Dublin Core schema for some of our metadata, it also includes:

1. Subject (similar to the keywords list idea from early CellML documents)
2. Contributor (entity that contributes to the resource)
3. Type ("the nature or the genre of the content of the resource"-this would always be "CellML model")
4. Format ("physical or digital manifestation of the resource")
5. Identifier (A unique identifier for the resource. This may be a URL or it may be some other identifier)
6. Language (The "language of the intellectual content of the resource". We should consider adding this to the recommended metadata if we allow models to have descriptions, etc. in non-English languages.
7. Relation ("A reference to a related resource")
8. Coverage (the scope of the resource: this could be made roughly equivalent to our "species" metadata, but I do not think we should force the match.)

16.2 Variables.

It should be possible to associate the following metadata with variables:

1. Variable Status: Modelers may wish to indicate to the user the status of a variable within their model (or partial model). For instance, some variables may be important inputs, which a user might want to play with when running simulations, while other variables are important outputs, which the user might want to keep track of during a simulation.

16.3 Supporting Information.

1. CellML must be able to store the following information about a person:
   (a) Name
   (b) Contact information
      i. Mailing address
      ii. Phone number(s)
      iii. Fax number
      iv. E-mail address
   (c) Affiliation (institution/corporation for which the person works)
2. CellML must be able to store identifying information about references. It must be able to store this either as a complete reference (title, author, journal, etc.) or as a unique identifier from a public database such as Medline.

17 Test Cases.

1. Luo-Rudy
2. Jafri-Rice-Winslow
3. A pathway model with no associated math
4. A pathway model with associated math
5. An example model from Adam Arkin
6. An example model from Dennis Bray
7. HMT
