CellML Specification

Draft — 2 March 2001

1 Introduction

1.1 Introduction to CellML

This document formally specifies CellML™, an XML-based language for describing and exchanging a wide range of mathematical models of cells and subcellular processes. CellML is being developed by scientists in the Bioengineering Research Group at the University of Auckland and at Physiome Sciences, Inc. The development of CellML is guided by an advisory board drawn from many different areas of biological modelling (see the project team[1] page on the CellML website for more information). CellML is being developed as an open standard, and all interested parties are encouraged to send feedback to info@cellml.org, or to the cellml-discussion[2] mailing list.

1.1.1 Purpose and scope of CellML

CellML is intended to support the definition of any type of model of a cell or subcellular process. Therefore, it uses a very general structure. CellML is also intended to facilitate the re-use of models and parts of models. It accomplishes this by using a component-based architecture. Models are split into logical subparts called components. These components are then connected together to form a model.

The scope of CellML is specifically limited to the definition of model structure. All other types of information that modellers need or want to include in a model document are incorporated using other languages. For instance, mathematics is included in CellML documents using MathML[3]. Metadata is included as RDF[4], using the Dublin Core's[5] schema wherever possible.

1.1.2 What is XML?

The CellML language is defined in terms of a meta-language called XML[6], which stands for e**X**tensible **M**arkup **L**anguage. XML is a standard published by the World Wide Web Consortium[7], the organisation responsible for defining many internet-related standards, most notably HTML. XML is essentially a means of adding structure to text documents, allowing machines to unambiguously associate text or binary data with a particular component in a document's data model.

XML is an appropriate medium for CellML because it is both human and machine readable. A model author can create a CellML document with a text editor or with any piece of CellML-compliant software. XML is a well-defined and widely used specification, and many free software utilities and libraries for the processing of XML already exist, simplifying the development of CellML software. XML has also been designed to be usable over the internet, making CellML suitable for the interchange of models between software and databases at different physical locations.

A quick introduction to XML[8] is available in the examples section of the CellML website.

[1] http://www.cellml.org/public/about/project_team.html
[2] http://www.cellml.org/public/mailing_lists/discussion.html
[3] http://www.w3.org/Math/
[4] http://www.w3.org/RDF
[5] http://purl.org/DC/
[6] http://www.w3.org/XML/
[7] http://www.w3.org/
[8] http://www.cellml.org/examples/introduction/xml_guide.html

1.1.3 Definition of "*model*"

A model is an idealized representation of the rules that govern the behaviour of a system. CellML supports both *quantitative* and *qualitative* models. Quantitative models represent these rules using mathematics. Qualitative models represent the relationships between objects in the system, without attempting to define mathematics to represent the behaviour of the objects.

The CellML specification covers three kinds of models: *complete*, *incomplete*, and *partial*. A *complete quantitative model* is one that can be simulated (i.e., the mathematical equations contained in the model can be solved). A *complete qualitative model* is one in which all objects of interest in a system are represented. An *incomplete model* is a work in progress. For instance, an incomplete quantitative model might not contain all of the equations necessary to simulate the behaviour of the system. A *partial model* is a description of one aspect of the system. Within that portion of the system, the description is complete. However, it still might not be possible to run a simulation of the model.

A *valid CellML document* may describe a complete, incomplete, or partial model. A *valid CellML model* must be complete. This specification does not attempt to limit the behaviour of processing software when confronted with invalid documents or models. It is recommended that software report errors to the modeller (at the very least).

1.2 Structure of the CellML Specification

1.2.1 Sections of the CellML specification

The CellML specification is divided into several sections, each of which discusses a particular aspect of CellML:

- **Fundamentals** — This section explains concepts used in all other sections of the specification, such as the definition of a valid CellML name and the use of XML namespaces in CellML.
- **Model Structure** — This section describes how models are organised in CellML. It includes an explanation of the use of a network of components to define a model and a discussion of variables in CellML.
- **Mathematics** — This section describes how to define mathematical equations and algorithms in CellML models.
- **Units** — This section explains the requirements for units in CellML and details how a modeller can define arbitrary sets of units.
- **Grouping** — This section explains how a model can be organised into logical encapsulation and geometric containment hierarchies by grouping components.
- **Reactions** — This section introduces the CellML syntax that makes it possible to define the chemical expressions that make up reaction/pathway models without resorting to MathML.
- **Metadata** — This section describes how to define metadata and associate it with models, model components, and other CellML elements.

A valid CellML model can be created using nothing beyond the material covered in the fundamentals, basic model structure, mathematics, and units sections of the specification. The concepts in the remaining sections of the specification allow modellers to build more meaningful models.

Each section of the specification is further divided into five subsections:

- **Introduction** — This subsection explains the purpose of the elements covered in the current section.
- **Basic Structure** — This subsection describes the new elements and attributes introduced in the current section of the specification and how they are combined.

- **Examples** — This subsection provides one or two basic examples of the correct use of the elements and attributes introduced in the current section of the specification. More extensive examples can be found in the examples section[9] of the CellML website.

- **Rules for CellML Documents** — This subsection provides formal rules for the use of the elements and attributes introduced in the current section to create valid CellML documents. These rules are specified as bulleted lists. Each rule may have an associated explanation, which appears directly after the rule, in square brackets ([]).

- **Rules for Processor Behaviour** — This subsection provides some rules for correct CellML processor behaviour with regards to the elements and attributes introduced in the current section.

1.2.2 Levels of CellML conformance

The rules in the CellML specification can be split into two groups: rules that define the syntax of a CellML document and rules that determine how software processing that document should behave. In the subsequent sections of the specification, the first set of rules are included in subsections titled Rules for CellML Documents, and the second set is in subsections titled Rules for Processor Behaviour.

The rules can also be split into two groups, each representing different *levels of conformance* to the specification. The majority of the rules in the CellML specification are part of the first level of conformance. Rules that are part of the second level of conformance are indicated as such by the inclusion of the phrase "*Level Two*" after the rule statement. The meaning of these conformance levels for documents and processing software is discussed below.

The levels of conformance to the CellML specification should not be confused with the features defined in different *versions* of the specification. As CellML is developed further, future versions of the CellML specification will add new elements to the language, which may add document and behaviour rules that affect both levels of conformance. Features that are expected to be added to CellML are documented in the Future Directions[10] part of the CellML website.

CellML conformance level one

The first level of CellML conformance is composed of the majority of rules in the CellML specification. Level one document rules generally specify how the different XML elements and attributes that make up the CellML vocabulary may be combined. A typical level one rule for a document is "Both the **<model>** and **<component>** elements can contain any number of **<units>** elements". Level one processor rules generally specify implied behaviour that is not immediately obvious from analysis of the XML. A typical level one rule for processor behaviour defines the scope of a units definition (see Section 5.5.1[11]).

A CellML Document is conformant to level one of the CellML specification if it complies with all level one rules for documents in the CellML specification.

A CellML processor is conformant to level one of the CellML specification if it can validate CellML documents against all level one rules for documents in the CellML specification and it follows all *appropriate* level one rules for processor behaviour in the CellML specification when interpreting CellML documents. The *appropriate* rules are those that relate to the intended use of the software (i.e., software that only renders the model need not address the scope of units definitions).

[9] http://www.cellml.org/examples/introduction/index.html
[10] http://www.cellml.org/public/documentation/future_directions.html
[11] http://www.cellml.org/public/specification/20010302/cellml_specification.html#sec_units_rules_scope_of_units

CellML conformance level two

The second level of CellML conformance rule is composed of all of the rules from level one plus additional rules that are marked as belonging to level two in the specification. Currently, there are no level two document rules. Level two processor rules generally specify complex interactions between objects defined in different parts of a CellML document. A typical example is the requirement that all mathematics within a model be self-consistent.

A CellML document is conformant to level two of the CellML specification if it complies with all level one and level two rules for documents in the CellML specification.

A CellML processor is conformant to level two of the CellML specification if it can validate CellML documents against all level one and level two rules for documents in the CellML specification and it follows all *appropriate* level one and level two rules for processor behaviour in the CellML specification when interpreting CellML documents. The *appropriate* rules are those that relate to the intended use of the software (i.e., software that only renders the model need not address the consistency of mathematics).