The Future of the CellML Specification

Status overview

- The latest stable specification is CellML 1.1. It was last changed in 2002, and marked as frozen in 2006.
- There has been discussion of many other possible features since then.
- The development of CellML 1.2 provides an opportunity for new features to be added.
- Community input on the specification will greatly aid this process.

Managing community input

- Initial messages about particular features get sent to cellml-discussion@cellml.org
- Discussion on specific features take place at: https://tracker.physiomeproject.org
- Decisions are made based on the consensus from discussions
- Unofficial drafts with the proposed changes are encouraged

Specification format

- The specification will be purely normative; examples and justification can go in a separate document.
- We are using DocBook (XML) to represent CellML 1.2.
- Mathematical equations are represented using MathML.
- DocBook gets converted into various formats as needed.

Sharing drafts

- We are using git, a distributed VCS to create unofficial drafts
- Anyone can easily create their own fork and make it world readable
- Changes can be merged between people using this workflow, keeping change history.
- No official central repository, but certain revisions are good bases for future work.

CellML design philosophy

- CellML 1.1 was inconsistent on some desgin aspects. Need a unifying philosophy for the core specification.
- Core CellML specifies only the underlying mathematics:
  - Declarative, not procedural
  - No biological or other domain specific information in the core
- Core CellML is general and not limited by what we anticipate can be computed.

Use of formal language

- CellML 1.2 drafts use well defined words in the style of an RFC specification.
- A number of ambiguities and contradictions from CellML 1.1 have been corrected in this process.
- Features like units conversions on connections, which must be implemented consistently for interoperability, are now mandatory

Secondary specifications

- CellML 1.0, 1.1, and drafts of 1.2 are too general for anyone to implement it entirely.
- Secondary specifications define a subset of CellML which software can implement entirely, allowing certainty in the model sharing process.
- Similar purpose, but more general, than the CellML 1.0/1.1 CellML Subset

Reactions

- Reaction elements do not fit with the underlying mathematics only philosophy of CellML
- Reactions should be in metadata, layered on top of the normal CellML mechanisms
- The reaction element is not in CellML 1.2 drafts.
- Sarala has worked out how to describe reactions in metadata as a best practice.

Containment

- CellML 1.1 provided a generalised grouping mechanism, and included definitions of encapsulation and containment.
- Encapsulation is for structuring the mathematics, containment describes the biology.
- Containment and user-defined groups don't belong in CellML core. Solution is to replace group with an encapsulation element.

Connection directionality

- In CellML 1.1, connections have directions
- This implies a procedure, and not a network of declarative mathematics
- Directionless connections would fit with the philosophy underlying CellML better.
- There is a draft implementing this.

Why structured types

- In CellML 1.1, everything is a real number (with units)
- CellML 1.1 models wanting to use matrices, vectors, sets, or λ-functions have to improvise
- This can make models inelegant, and it also makes model decomposition more complex

Types – built in types

- One proposal has been to create an in specification dictionary of datatypes, like vector_real.
- All real numbers in these types would have the same units.
- This system lacks generality; it would be necessary to wait for the next version of CellML to add new data types!

Types – new type element

- Another option for types is to define a series of new elements for deriving types (like setof, vectorof, and so on).
- This would look similar to the current unit system
- There would be a built in mechanism for real numbers with units
- Would add more complexity to CellML
- Re-usability would be limited without parameterised types

Types – typed λ calculus

- The third option is to make types first class mathematical objects (as in typed λ calculus) with associated variables
- Real number types include the units, so the units element won't be needed
- The relationships between types are specified with mathematical operators
- Connections, and functions which return types, allow for parameterised types.

Namespace policy

- CellML 1.1 changed the namespace on all elements, even those that didn't change
- From a compatibility standpoint, this is a bad thing
- A better approach is for namespaces to only change on elements that have changed their semantics
- This requires that CellML software check for unrecognised elements that look like they are from a future specification