David Nickerson
Auckland Bioengineering Institute
University of Auckland
New Zealand

CellML Editors

Poul Nielsen
2011-2014
NZ

Mike Cooling
2013-2015
NZ

David Nickerson
2011-2014
NZ

Alan Garny
2011-2013
France

Jonathan Cooper
2011-2013
UK

First a few use-cases

- Large scale tissue and organ models

  - Tissue level (bidomain equations):

$$\begin{aligned} \chi\left(C_m\frac{\partial V}{\partial t}+I_{\text{ion}}(\mathbf{u},V)\right)-\nabla\cdot\left(\sigma_i\nabla\left(V+\phi_e\right)\right) &= -I_i^{(\text{vol})}, \\ \nabla\cdot\left(\sigma_i\nabla V+\left(\sigma_i+\sigma_e\right)\nabla\phi_e\right) &= I_{\text{total}}^{(\text{vol})}, \\ \frac{\partial\mathbf{u}}{\partial t} &= \mathbf{f}\left(\mathbf{u},V\right), \end{aligned}$$

  - Cellular models encoded in CellML:

$$\frac{\mathrm{d}V}{\mathrm{d}t}=-\frac{I_{\text{ion}}(\boldsymbol{u},V)+I_{\text{stim}}}{C_m}$$

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}=\boldsymbol{f}(\boldsymbol{u},V)$$

  with many slight variations…

courtesy Jonathan Cooper

First a few use-cases

- Large scale tissue and organ models

  - Tissue level (bidomain equations):

$$\chi\left(C\frac{\partial V}{\partial t}+I_{\text{ion}}(\mathbf{u},V)\right)-\nabla\cdot\left(\sigma_i\nabla\left(V+\phi_e\right)\right) = -I_i^{(\text{vol})},$$

$$\nabla\cdot\left(\sigma_i\nabla V+\left(\sigma_i+\sigma_e\right)\nabla\phi_e\right) = I_{\text{total}}^{(\text{vol})},$$

$$\frac{\partial\mathbf{u}}{\partial t} = \mathbf{f}\left(\mathbf{u},V\right),$$

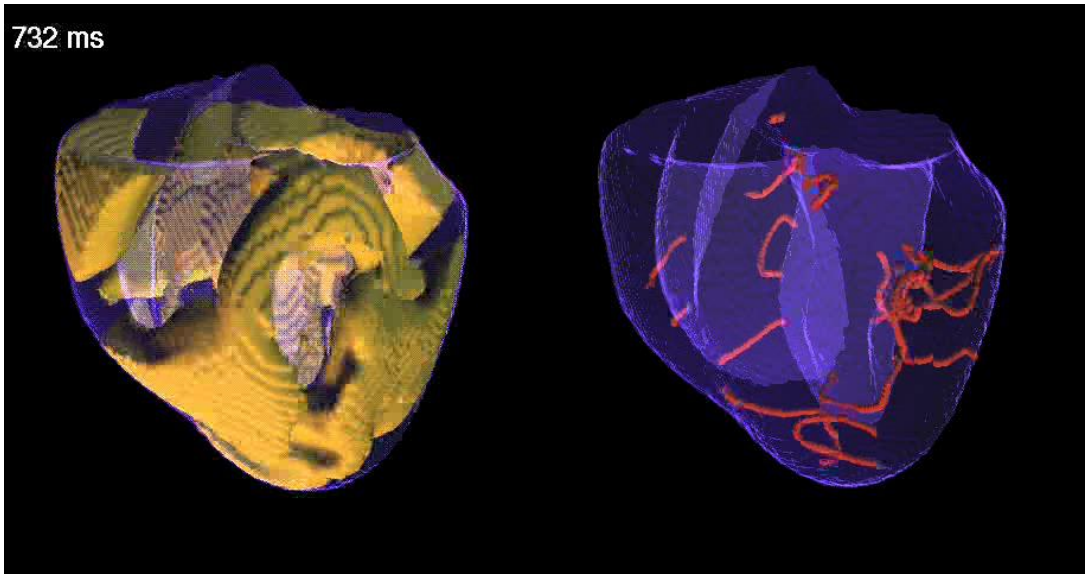
  - Cellular models encoded in CellML:

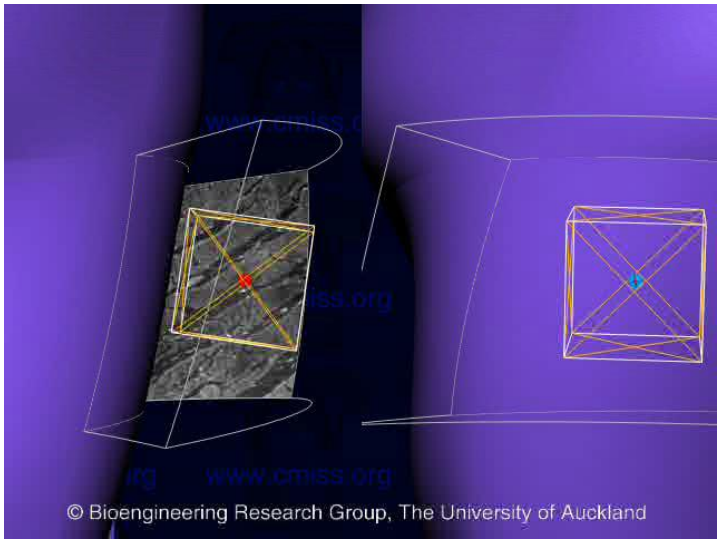
$$\frac{\mathrm{d}V}{\mathrm{d}t} = -\frac{I_{\text{ion}}(\boldsymbol{u},V)+I_{\text{stim}}}{C_m}$$

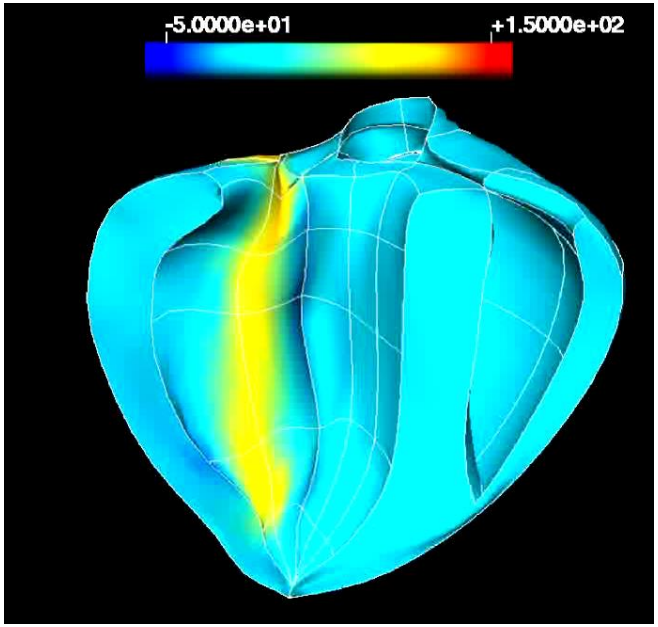
$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{u},V)$$

  with many slight variations…

courtesy Jonathan Cooper


732 ms


© Bioengineering Research Group, The University of Auckland


-5.0000e+01
+1.5000e+02

Accelerating model evaluation

- In software:
  - Automatic transformation of CellML descriptions into forms allowing more efficient simulation (lookup tables, partial evaluation, analytic Jacobians, solver-specific tweaks) – Jonathan Cooper/Chaste (Oxford)
- In hardware:
  - Making use of GPUs
  - Developing an automatic solution to accelerate CellML models with reconfigurable hardware e.g. FPGAs that to be used by OpenCMISS – Ting Yu (ABI)

Rheological models for 3D fluid mechanics simulations of blood flow in arteries – David Ladd (ABI)

Carreau Model

Simple single equation model, relatively numerically stable with bounds μ_∞ and μ_0.

$$\mu_{eff}(\dot{\gamma}) = \mu_\infty + (\mu_0 - \mu_\infty)\left(1 + (\lambda\dot{\gamma})^2\right)^{(n-1)/2}$$

Quemada Model

Semi-phenomenological: based on blood constituent material parameters and and concentrations.

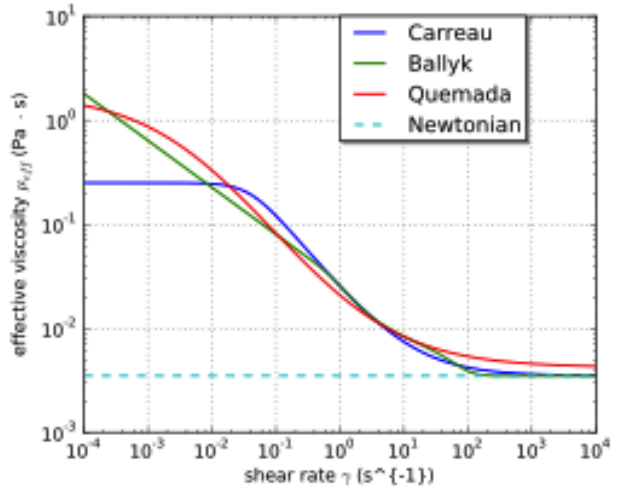
$$\mu_{eff}(\dot{\gamma}) = \mu_F\left(1 - \frac{\phi}{2}\frac{k_0 + k_\infty\sqrt{\dot{\gamma}_c/\dot{\gamma}}}{1+\sqrt{\dot{\gamma}_c/\dot{\gamma}}}\right)^{-2}$$

Ballyk Model

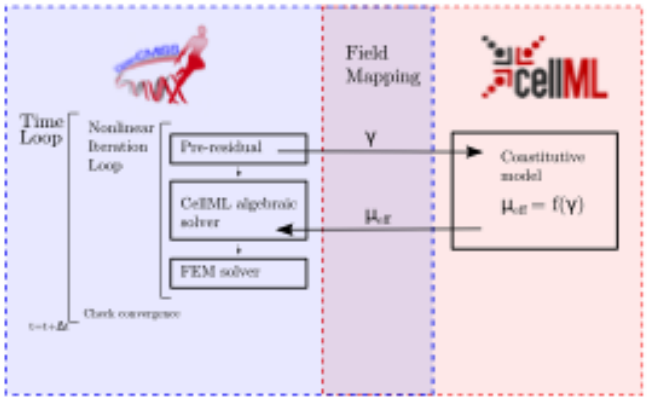
Exhibits higher viscosity at very low shear rates, better approximating near-wall stresses.

$$\mu_{eff}(\dot{\gamma}) = \lambda(\dot{\gamma})\,\dot{\gamma}^{n(\dot{\gamma})-1}$$
$$\lambda(\dot{\gamma}) = \mu_\infty + \Delta\mu\exp\left[-\left(1+\frac{\dot{\gamma}}{a}\right)\exp\frac{-b}{\dot{\gamma}}\right]$$
$$n(\dot{\gamma}) = n_\infty + \Delta n e^{\left[-\left(1+\frac{\dot{\gamma}}{c}\right)e^{\frac{-d}{\dot{\gamma}}}\right]}$$

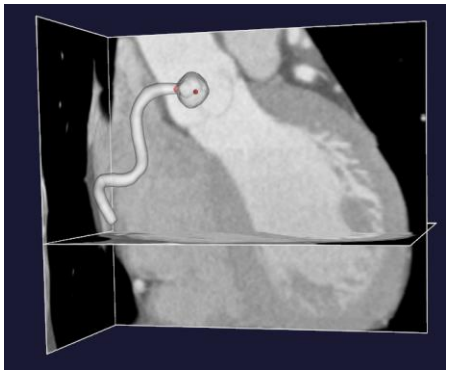
Viscosity Shear Response

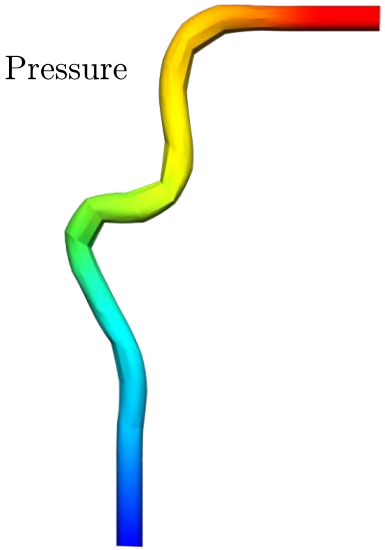


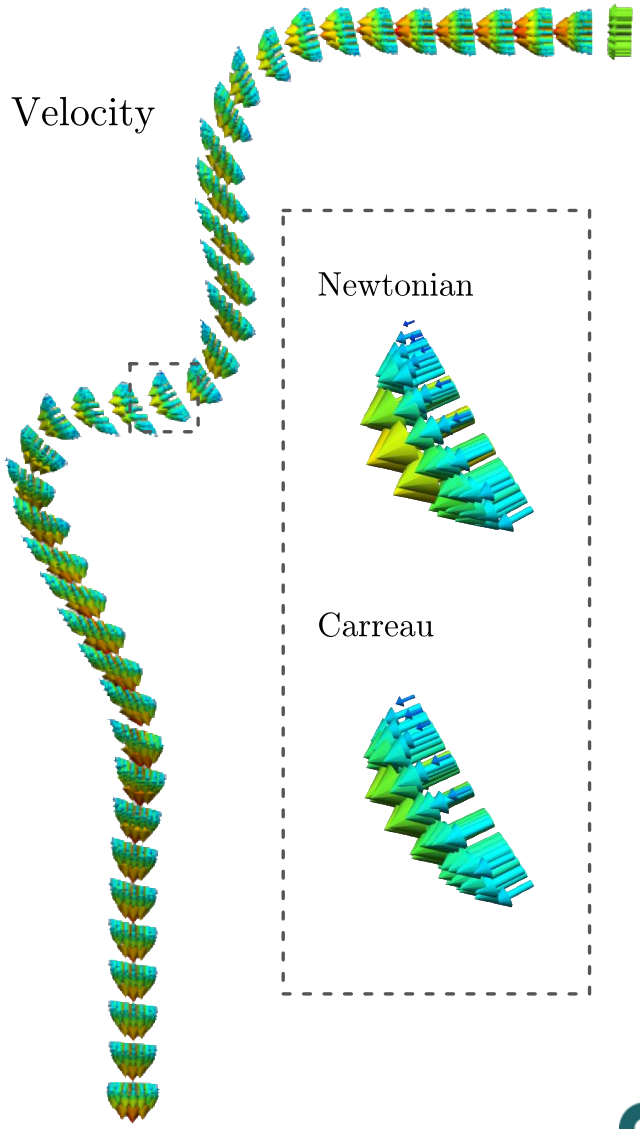
OpenCMISS/CellML Coupling



Rheological models for 3D fluid mechanics simulations of blood flow in arteries – David Ladd (ABI)

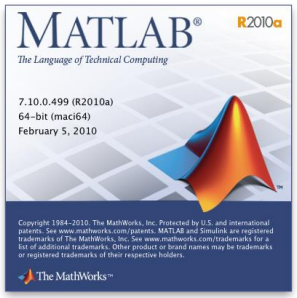






Model development workflows

Cell model development



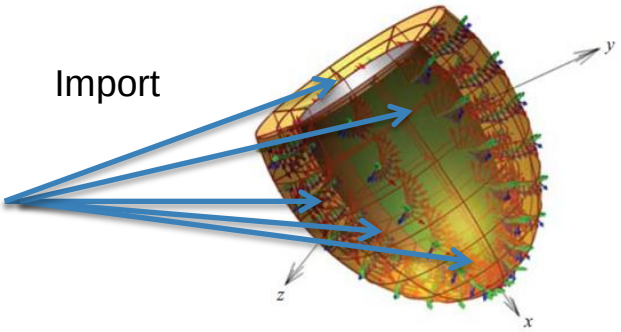
Translate

cellML

www.cellml.org

CMISS
www.cmiss.org

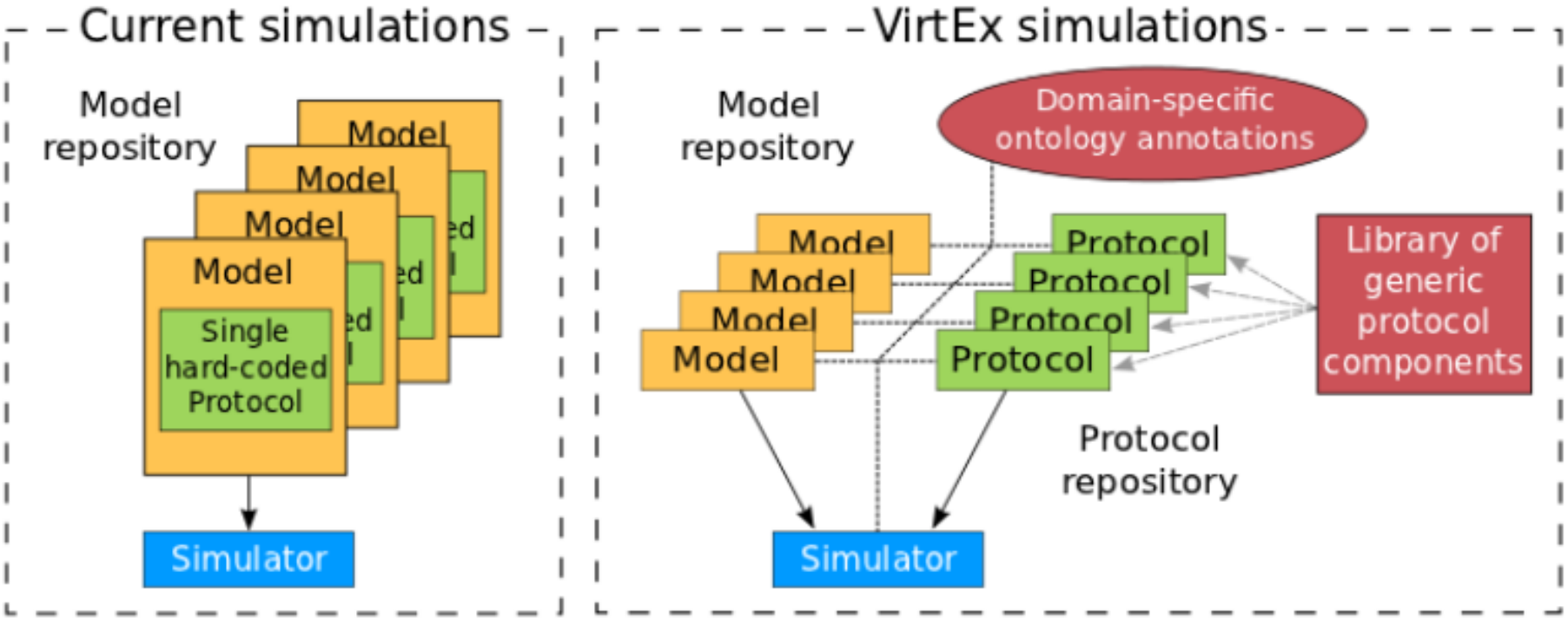
Import



- Standard finite elements & finite elasticity
- Cell model calculates active contraction for strains computed in tissue model

Functional Curation – Jonathan Cooper *et al* (Oxford)

- Separate model structure and experimental scenario

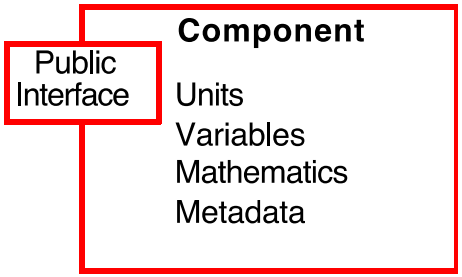


A brief overview of CellML as it is

- CellML is designed to support the definition and sharing of (lumped parameter) models of biological processes.
- CellML includes information about:
  - Model structure (how the parts of a model are organizationally related to one another);
  - Mathematics (equations describing the underlying biological processes);
  - Metadata (additional information about the model that allows scientists to search for specific models or model components in a database or other repository).
- A public repository of over 500 published signal transduction, electrophysiological, mechanical, and metabolic pathway processes is available at *http://models.cellml.org/*.

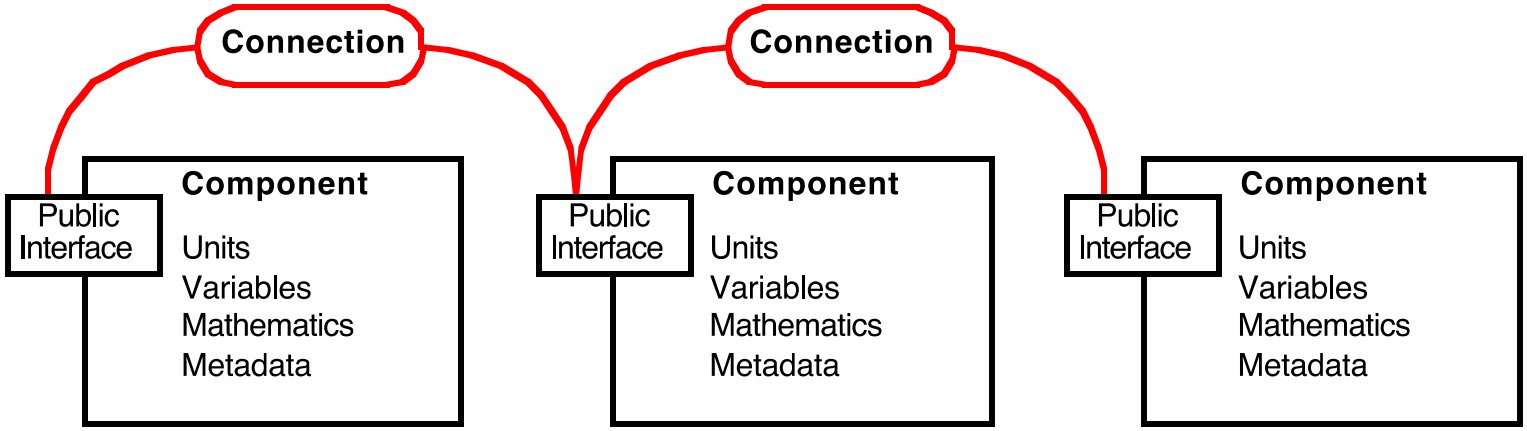
CellML components

- CellML has a simple structure based upon connected *components*.
- Components abstract concepts by providing well-defined interfaces to other components.
- Components encapsulate concepts by hiding details from other components.



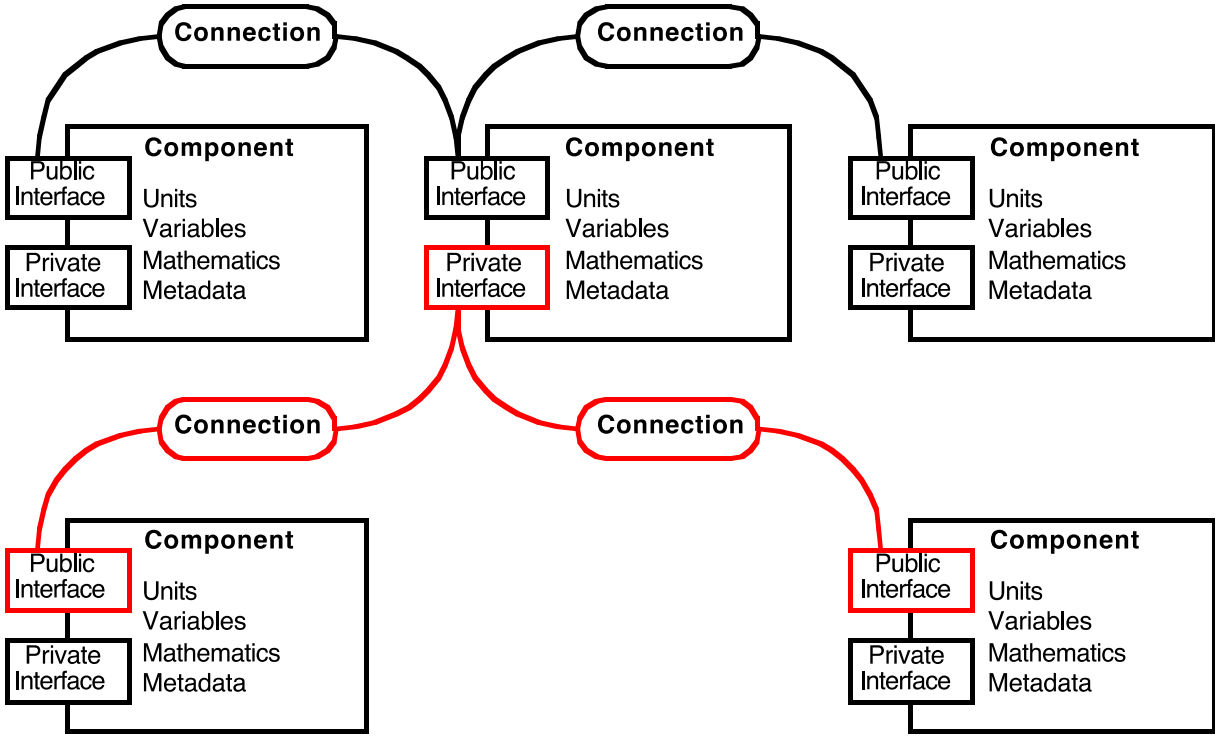
CellML connections

- *Connections* provide the means for sharing information by associating variables visible in the interface of one component with those in the interface of another component.
- Consistency is enforced by requiring that all variables be assigned appropriate physical *units*.



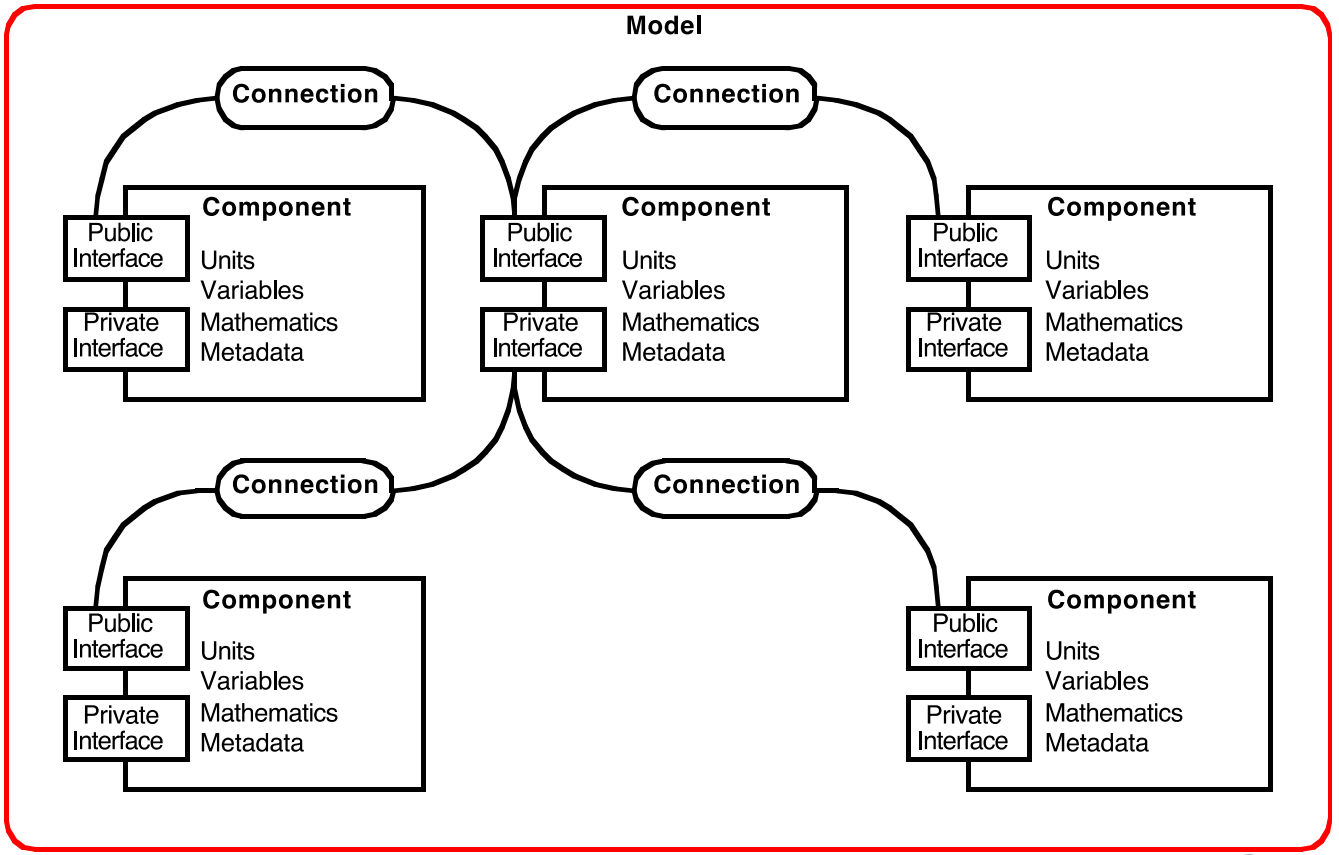
CellML encapsulation

- Encapsulation hierarchies are enabled using *private interfaces*.



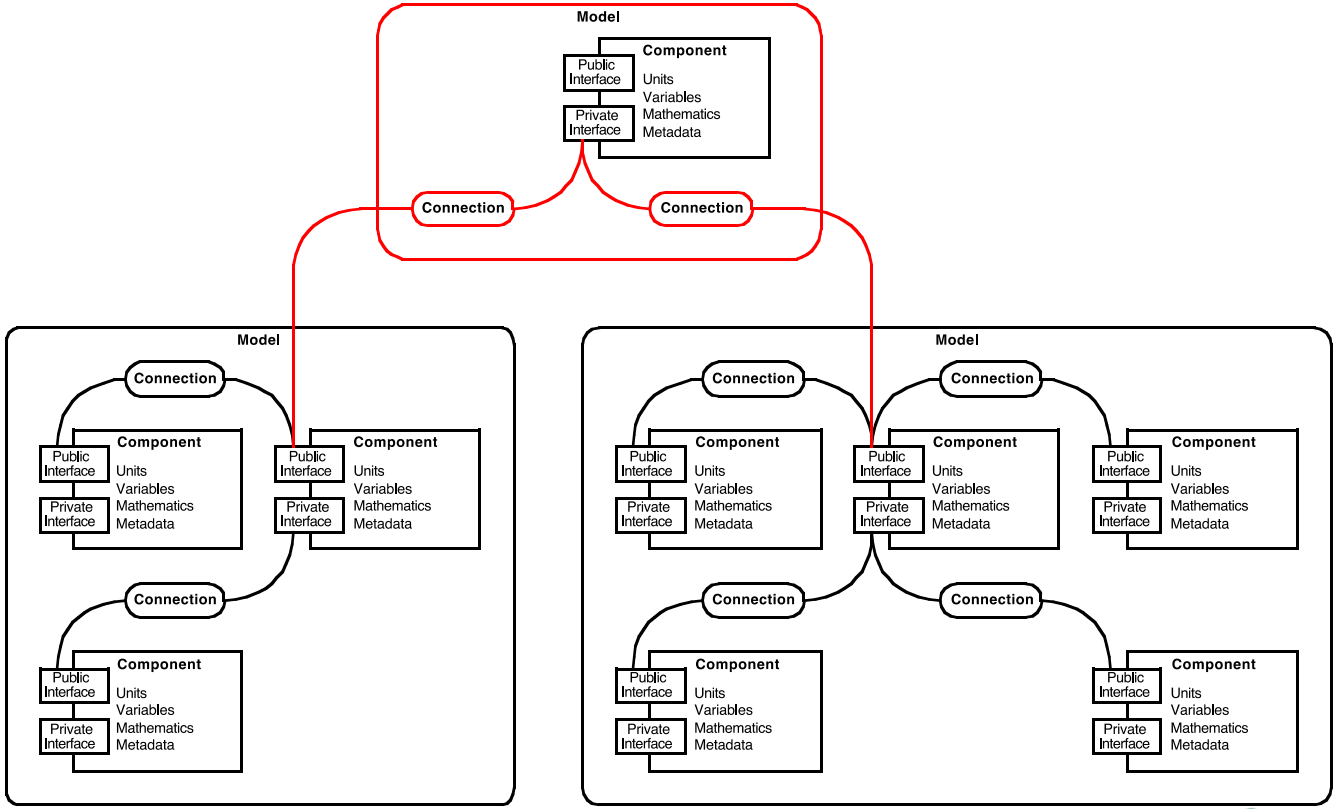
CellML model

- A *model* is the root element for a CellML document. It is a container for components, connections, units, and metadata.



CellML import

- Model reuse is enabled by the *import* element.



CellML 1.2 – the next version

- Tidy up the specification (normative + informative).
- Establish the process for future development.
- Incremental change from CellML 1.1
- Timely software support:
  - Need for API support;
  - Ease the burden on tool developers to support CellML 1.2.
- Grounded in concrete, real world, use-cases.

'Minor' changes

- Remove the directionality of connections.
- Simplify the encapsulation mechanism.
- Remove the reaction construct – the final bit of biology hanging around in the language!

- Various miscellaneous clarifications in the language of the specification, including:
  - Produce a 'normative' version with concise technical details; and
  - An 'informative' version with more explanation and demonstrations/ examples.

Core + secondary specifications

- CellML 1.2 Core specification:
  - Basic concepts;
  - As permissive as possible;
  - Foundation for future versions of the specification.
- "CellML 1.2" includes a collection of secondary specifications which place restrictions on what is valid to express in CellML 1.2 Core.
- Software support, for example:
  - Editing and visualisation tools likely to support the core specification;
  - Simulation tools only able to support some secondary specification(s).

Example: Mathematics

- Core specification: any MathML can be expressed.
- Secondary specification: only index-1 DAE systems and algebraic expressions consisting of this set of MathML operators are valid.
- "Fixes" current ambiguity regarding the CellML subset of MathML.
- Provides a restricted set of mathematical expressions which simulation tools can reasonably be expected to support.

Example: '`evaluatedAt`' operator

- A new operator (`csymbol`) introduced in CellML 1.2 Core to evaluate a variable at a specific "time".
- Allows the concept of delayed differential equations, infinitesimal delays, etc. to be expressed.
- CellML 1.2 Secondary restricts the usage of this operator to:
  - Infinitesimal delays (reset rules, like SBML events);
  - Setting initial values (i.e. all non-infinitesimal occurrences of `evaluatedAt` operator refer to the same value of "time").

Future development

- Draft guidelines for feature requests and proposals: http://www.cellml.org/specifications/development
- Normative specification provides explicit points of reference for future proposals.
- Secondary specifications can be developed for any purpose:
  - Tools can be unambiguous about their support for "CellML";
  - Popular and well-supported secondary specifications likely candidates for adoption into official CellML specifications.
- Timely method for easing restrictions placed by previous specifications in a compatible manner:
  - DDEs will not be allowed in CellML 1.2, but may be allowed with no change required to the Core specification.

Specification development

- Using reStructuredText with Sphinx.
- Source available on GitHub: http://github.com/cellml.
- Latest draft rendered via Read the Docs: http://cellml-specification.readthedocs.org/.

Software support – http://cellml.org/tools

- Large scale, high performance, multi-scale, multi-physics
  - CMISS – http://cmiss.org (CellML 1.0)
  - Chaste – http://www.cs.ox.ac.uk/chaste/ (CellML 1.0)
  - OpenCMISS – http://opencmiss.org
- Simulators
  - CSim – http://code.google.com/p/cellml-simulator
  - JSim - http://www.physiome.org/jsim (units validation)
- CellML environments
  - COR – http://cor.physiol.ox.ac.uk (CellML 1.0, Windows only)
  - OpenCell – http://opencell.org
  - OpenCOR – http://opencor.ws
- Antimony – http://antimony.sourceforge.net/

OpenCOR – http://opencor.ws

- Cross-platform modeling environment to organize, edit, simulate and analyze models
- Written in Qt/C++
- Plugin-based application, e.g.,
  - CellMLModelRepository
  - CellMLAnnotationView
  - CVODESolver, ForwardEulerSolver, …
  - SingleCellView
- "*Easy*" to extend and customize the application to meet your requirements.
- Integrated HTML help system to create interactive tutorials

Useful links

- http://www.cellml.org/
- http://www.cellml.org/workshop
- https://github.com/cellml
- CellML mailing lists: http://lists.cellml.org/

Acknowledgements

- The CellML community for all their input.
- The CellML team at the Auckland Bioengineering Institute.
- Funding from: Maurice Wilkins Centre, VPH-NoE, VPH-SHARE, 2020 Science, VPR.