



Model libraries with CellML 1.1





CellML

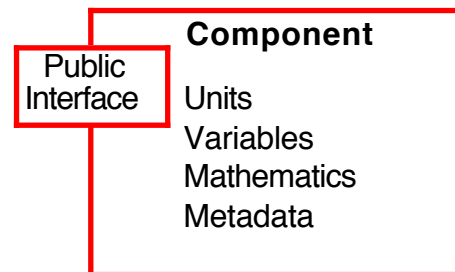
- CellML is designed to support the definition and sharing of models of biological processes.
- CellML includes information about:
 - Model structure (how the parts of a model are organizationally related to one another);
 - Mathematics (equations describing the underlying biological processes);
 - Metadata (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/>





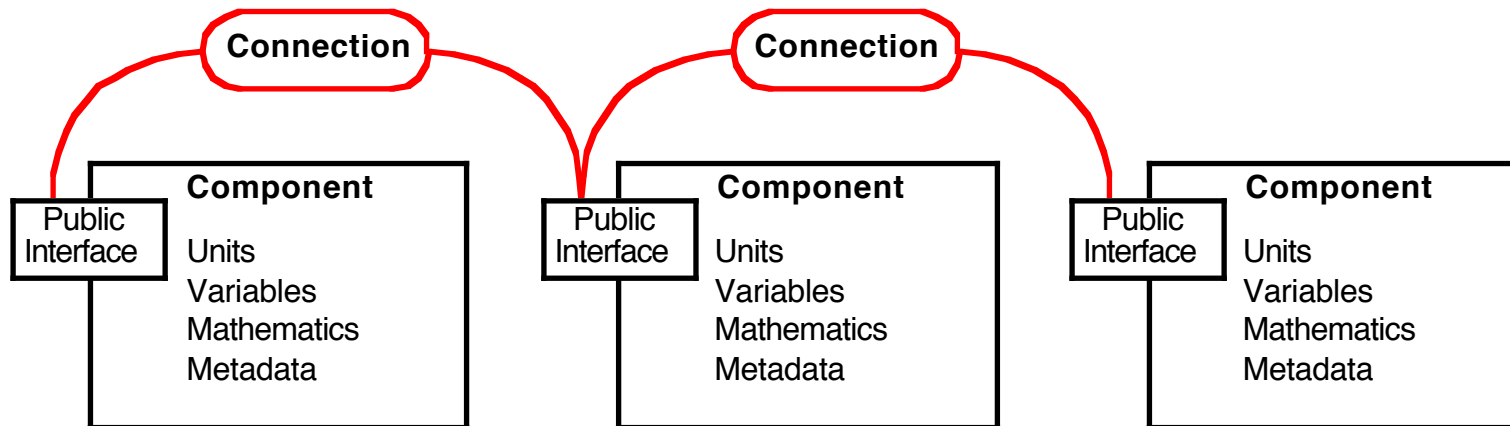
CellML components

- 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.



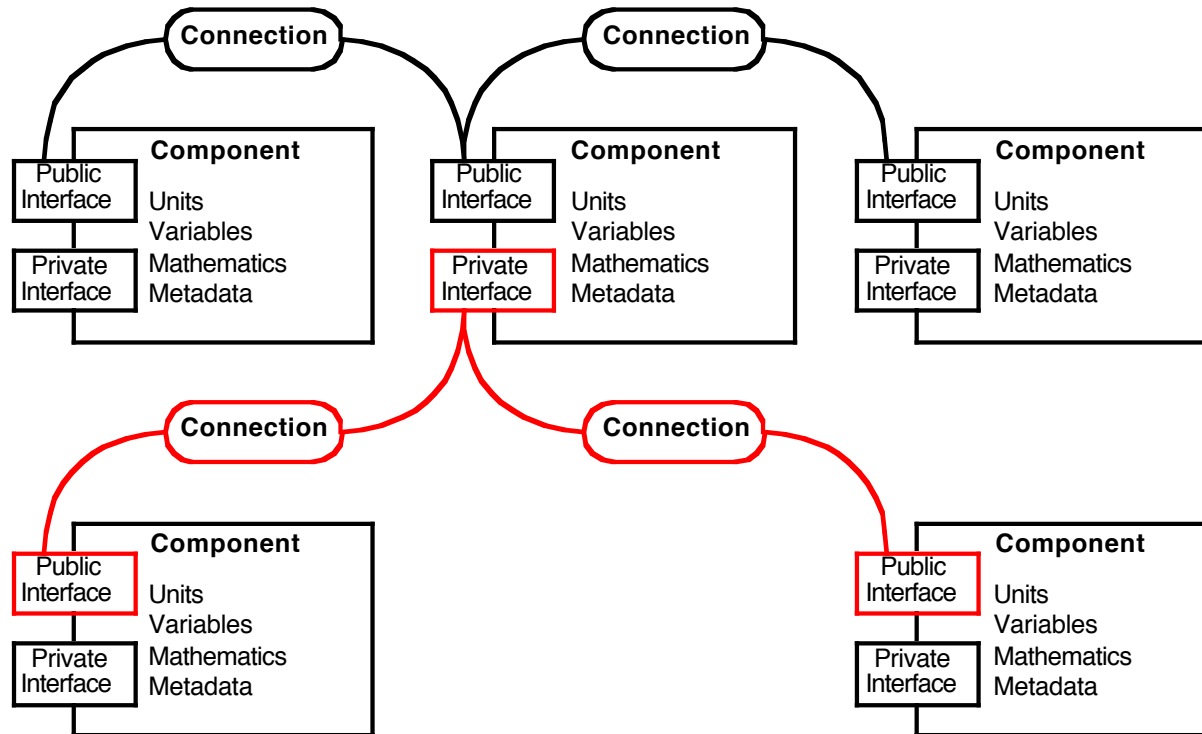
CellML connections

- *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*.



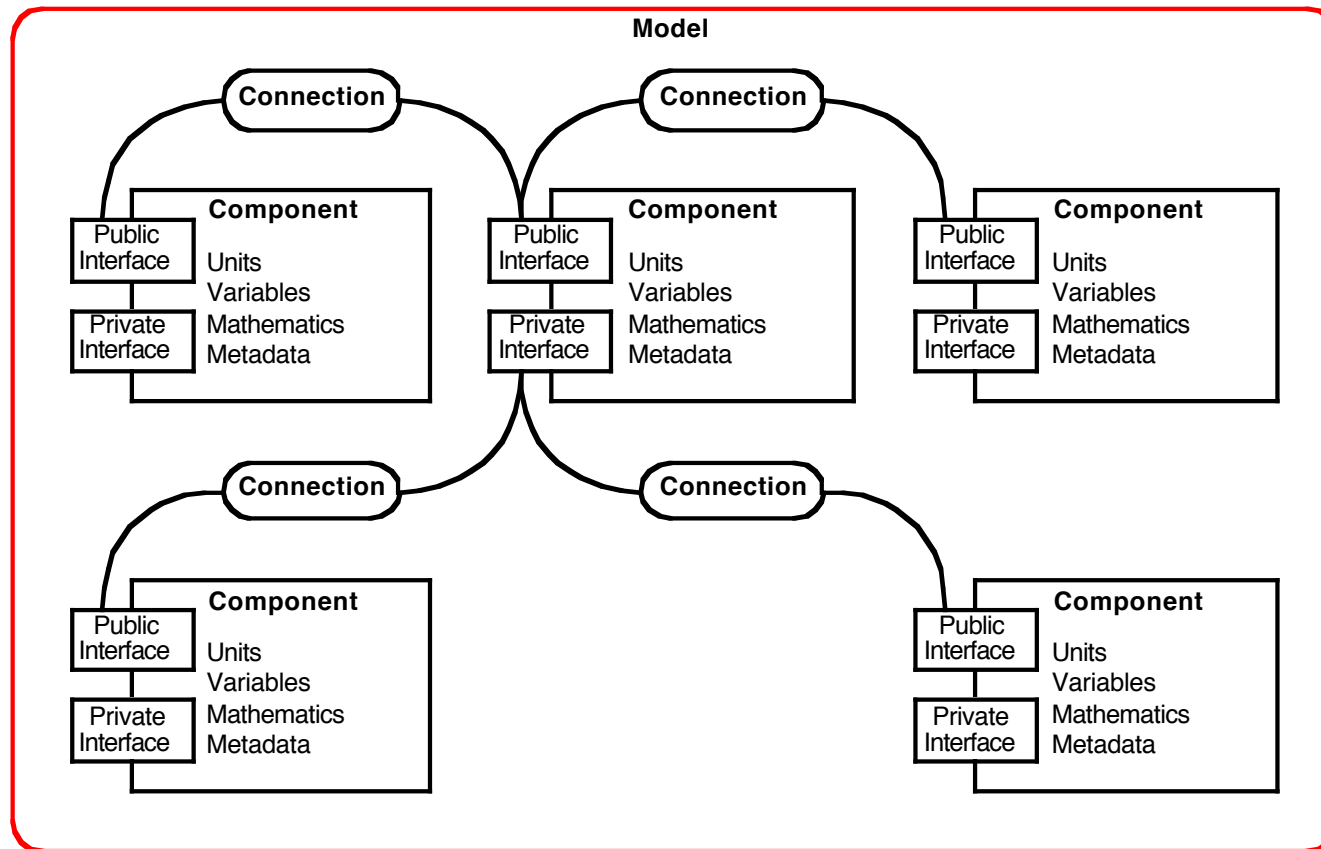
CellML encapsulation

- Encapsulation hierarchies are enabled using *private interfaces*.



CellML model

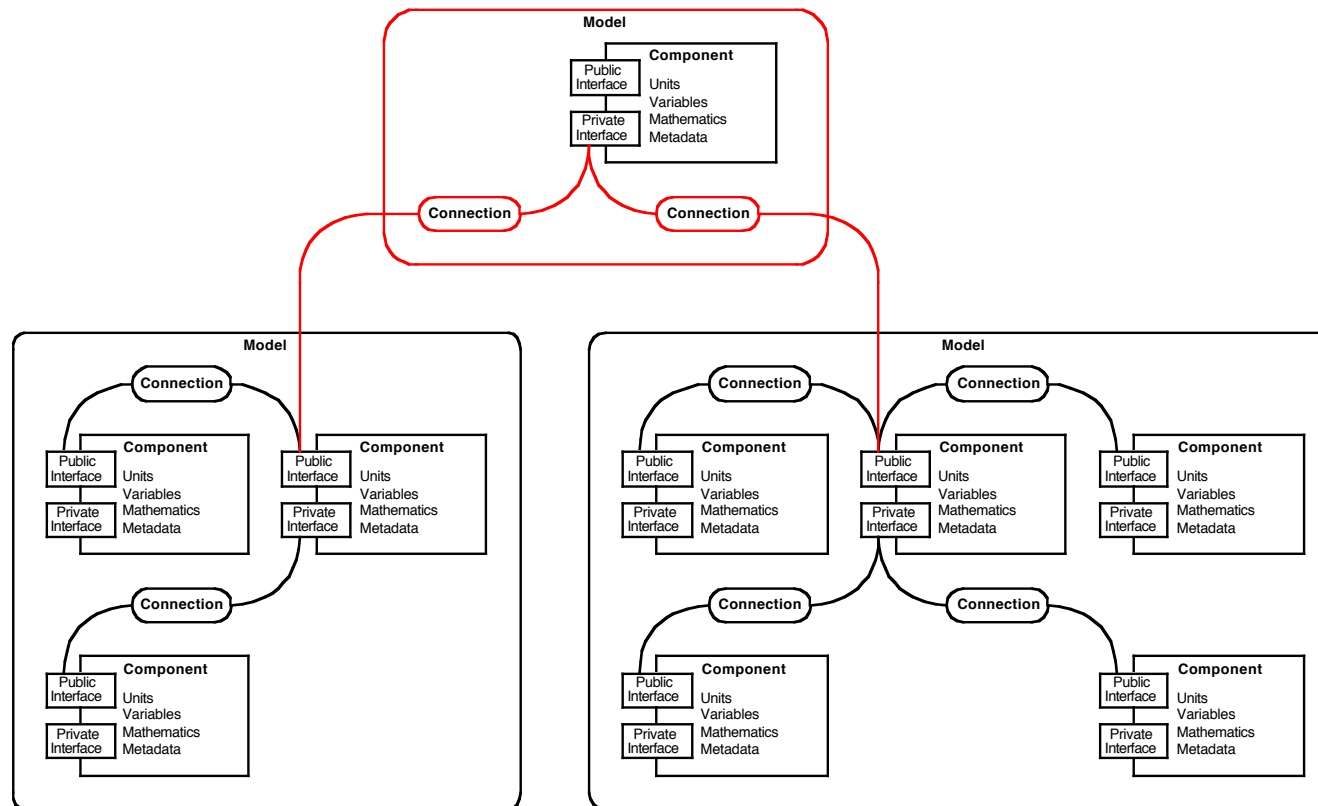
- A *model* is the root element for a CellML document. It is a container for components, connections, units, and metadata.





CellML import

- Model reuse is enabled by the *import* element.
- New models may thus be constructed by combining existing models into model hierarchies.



Model libraries

- Model reuse encourages the creation of model libraries.
- This is possible in CellML because there is no distinction between models as stand-alone entities and models as templates.
- Every import creates a new instance of the imported model in the importing model.
- The same model can be imported multiple times to create separate instances (with distinct identifiers) within the importing model.



Model libraries

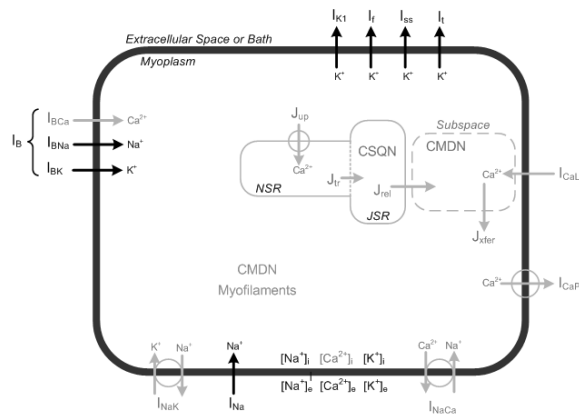
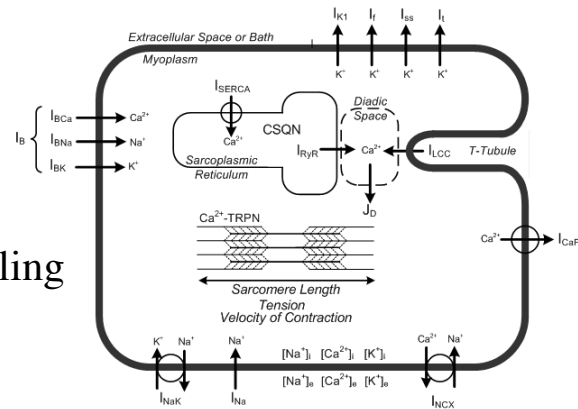
- Obvious candidates for reuse are existing CellML 1.0 models available in the model repository.
- Other candidates are the decomposition of existing models by identifying reusable generic (sub)models.
- These generic models are then formulated as new library models, making them available as basic building blocks for import into larger models.
- Useful generic models include collections of:
 - units (complicated combinations, non-SI definitions)
 - constants (codata fundamental physical constants)
 - processes (integrators, reactions, rate relations, ion channels, ...)
- Sometimes difficult to balance genericity versus conciseness.



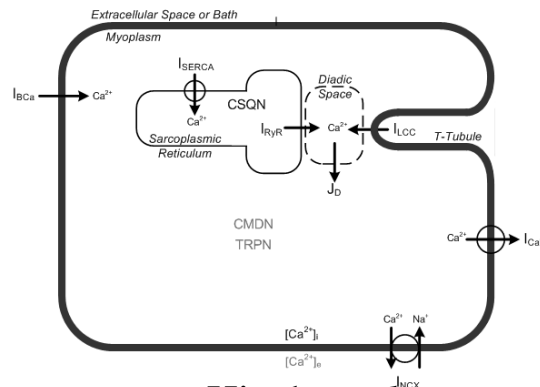


Combine models using CellML import

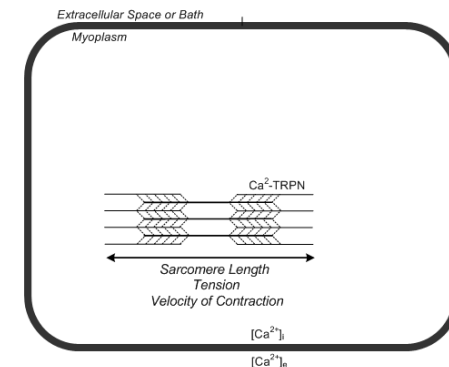
Terkildsen *et al.*
Integrated model of e-c coupling



Pandit *et al.*
cardiac action potential

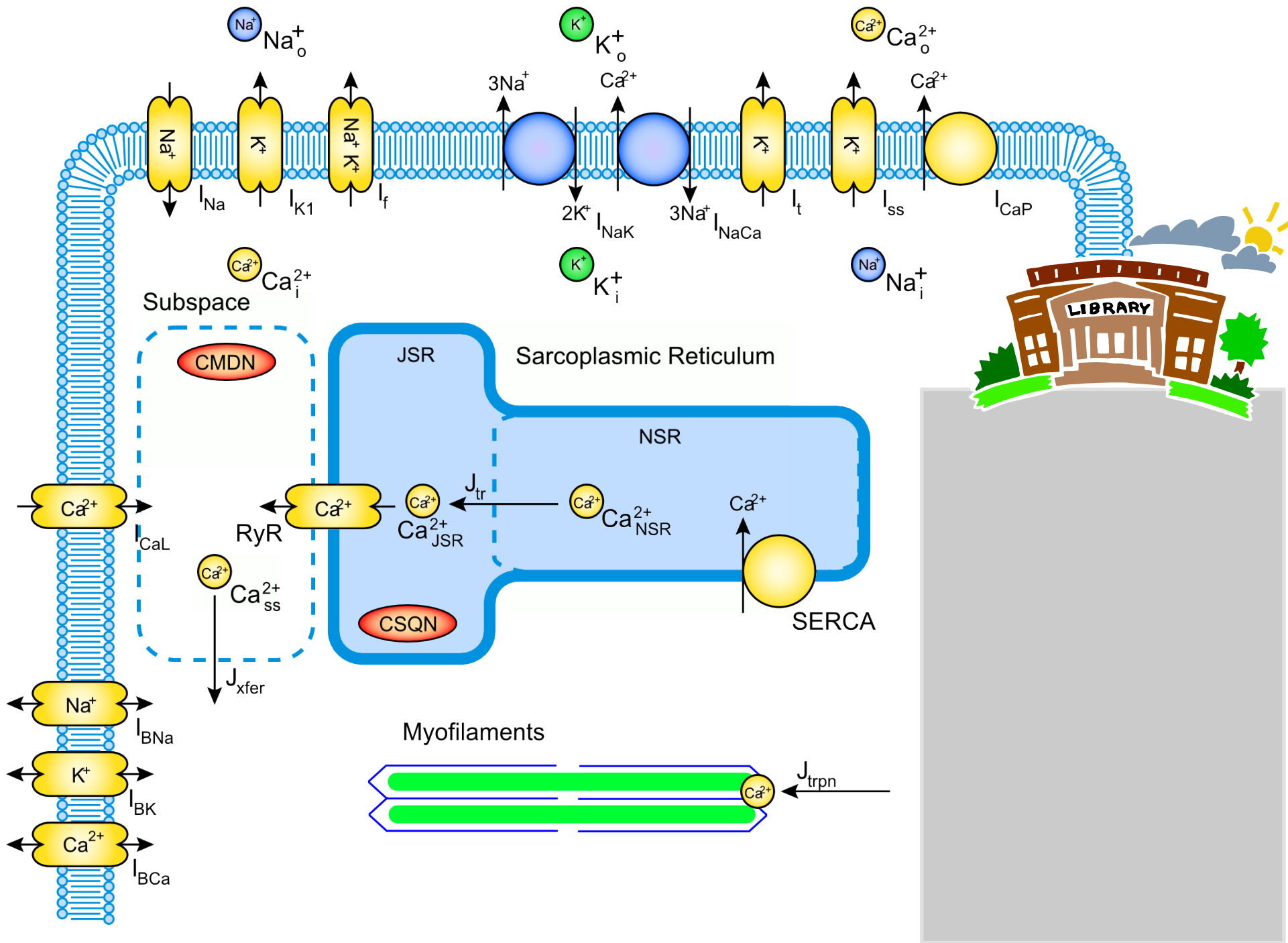


Hinch *et al.*
Ca-induced Ca release



Niederer *et al.*
myofilament mechanics





Best practice

- Most useful non-trivial library components describe clearly identifiable biophysical processes.
- Sarala Wimilaratne has given several examples of this approach in her PhD thesis on CellML model visualisation (Cooling 07 GCPR cycle, Hodgkin-Huxley 52, Nobel 62).
- We are compiling a list of best-practice examples based on the experience gained through the process of model decomposition.
- This work is still in its early stages – there is still much to be learned about which approaches offer the best long-term benefits. Mike Cooling has a poster in the ICSB conference.
- Others in this session will discuss the new tools that have been built to facilitate model reuse.



Best practice

- Put reusable mathematics in separate components, and use `<import>`s to instantiate these for use where appropriate.
- Use ‘*_delta*’ components to extensively connect multiple fluxes to species of interest.
- Use separate conversion components for connections where applicable.
- Build coarse-grained components from aggregations of finer-grained, biologically atomic components.
- Define `<units>` at the lowest level possible, `<import>`ing into higher level components as necessary.
- Separate out all parameter values into one or more non-mathematical CellML documents.
- Universal constants should be `<import>`ed from a non-mathematical CellML document (a standard based document on [UC] is recommended).
- If encapsulating, expose all potentially useful values using ‘`public_interface=“out”`’.

