Meeting Minutes 2 March 2004

Autumn A Cuellar

- **Robert Muetzelfeldt's e-mail exchange with Poul**. Robert Muetzelfeldt is a coordinator for the Simile [http://simulistics.com/] project. Following is a copy of the e-mail exchange.

  Poul asked us to think about how well the CellML concepts could map to the Simile package.

  > Dear Robert
  >
  > On 2004 Mar 02, at 23:30, Robert Muetzelfeldt wrote:
  >
  > \> Dear Poul,
  > \>
  > \> Thanks for getting back to me so quickly.
  > \>
  > \>\> I am sorry that you had no luck mailing to info@cellml.org. We will
  > \>\> look into this.
  > \>
  > \> Thanks.
  > \>
  > \>\>\> All this does mean that our submodels may be part of the mathematical
  > \>\>\> specification of the model - unlike, as I read, in CellML.
  > \>\>
  > \>\> You will find that in CellML 1.1 model hierarchies can be defined using
  > \>\> the import mechanism.
  > \>
  > \>
  > \>\>\> 1. What is the CellML take on handling disaggregation (e.g.
  > \>\>\> populations of
  > \>\>\> interacting cells)?
  > \>\>
  > \>\> CellML 1.1 can already model populations of interacting components. The
  > \>\> way to achieve this is to create separate models of each species and
  > \>\> import those models, multiple times, into a super- or enclosing-model.
  > \>\> Each import will effectively create a new instance of the imported
  > \>\> model. The multiple instances can then be linked using separate
  > \>\> connections to create the desired model of interacting components.
  > \>
  > \> It seems to me, from what I read in the CellML 1.1 spec and your wording here,
  > \> that CellML's mechanisms for handling multiple instances of something requires
  > \> a separate reference to each instance (i.e. a separate import element). Is that
  > \> right?
  >
  > Yes.
  >
  > \> In that case, it is a long way short of being able to model, as one can in
  > \> Simile, a collection of (say) 100,000 cells simply by modelling one of them and
  > \> specifying that there are 100,000 of them - each behaving according to the
  > \> same rules, but each able to have its own values for parameters, derived
  > \> quantities, and state variables.   Moreover, in CellML (as I understand it), would
  > \> not interactions between the imported instances have to be made explicitly for
  > \> each interaction?   In Simile, the association submodel concept (similar to UML

> association) enables interactions to be specified once. Thus, specifying a
> spatial grid (of potentially very large size) with diffusion between neighbouring
> cells involves a very simple Simile model, with a multiple-instance 'cell'
> submodel and a 'is_a_neighbour' association submodel.   Additionally, in
> Simile the multiple-instance submodel can be defined as a population
> submodel, in which case it would include model elements specifying how to
> dynamically create and destroy instances: again, a concept that I didn't think
> CellML could represent.

You are correct in your interpretation of CellML 1.1's lack of special facilities to handle large-scale collections of similar components. This is an issue that we spent some time on during the design of the language. In the end we decided (for now) to keep the language as simple as possible. One of the consequences of this is that every component is an instantiation. The import facility allows a limited form of templating, but is inadequate for very large scale models. We felt that, for the time being, this would still enable CellML to act as a specification for a wide class of models, but would rely on simulation applications to deal with the issues of multiple instantiation and associated connections.

It should be noted that MathML currently has the ability to declare structured objects, such as arrays, matrices, lists, and sets, which could enable large scale models to be described. However, we don't encourage the use of such facilities because it compromises the reusability of the resulting model and, in our opinion, is a poorer (although more compact and arguably computationally efficient) representation of the underlying processes.

>
> Now, some of this is trivial to represent: for example, it's easy enough to include
> an additional attribute of 'number_of_instances' for a CellML component.
> Specifying associations, with information flow between instances, requires
> somewhat more thought.   And of course developing a simulator that knows
> what to do with such specs is a significant undertaking. It's because I thought
> (and still think, from what you've written) that CellML is not currently able to
> express such concepts that I thought you might be interested in such a
> language.

As I mentioned earlier, we are working on adding a full typing mechanism to CellML by coupling the language to ontology descriptions, such as OWL. This move will enable much more general mechanisms for handling the large-scale problems that you mention, without having to introduce specific new keywords into CellML.

>
>>> Do you see potential for integration with Simile, where
>>> (say) a CellML single-cell model can be embedded in a Simile model?
>>
>> Yes, but such facilities already exist in CellML.
>
> If my above analysis is correct, the there is a big difference between CellML's
> approach and what Simile can offer. Enabling a CellML model to be a
> submodel in Simile, and exploiting Simile's mechanisms for handling multiplicity
> and interactions, could be a very efficient method for CellML to be used in
> highly-disaggregated models in the short term.   In the longer term, our
> particular approach to handling these concepts, characterised by quite a high
> level of abstraction, might be of some interest as CellML/FieldML/AnatML etc
> develop and merge.

Yes, we will look at your specifications.

>
>>> (I
>>> think there is a lot of potential in allowing each submodel to use its
>>> own
>>> diagrammatic - and possibly symbolic - language).
>>
>> I agree. We are currently exploring the integration of ontologies into
>> the language to enable the integration of formal specifications of such
>> domains of knowledge into CellML.
>
> I look forward to seeing how this develops.
>
>>> I note your plans to link
>>> up with AnatML and FieldML: do you envisage that these will give you
>>> sufficient expressiveness?
>>
>> AnatML will become CellML (or whatever name we give to the base
>> language) plus appropriate anatomical ontologies. FieldML is a separate
>> specification that enables the description of smooth spatially (and
>> temporally) varying quantities. At the moment such concepts are handled
>> within CellML as linked discrete components (I think that you do the
>> same in Simile). FieldML will give CellML much more flexibility by
>> enabling distributed parameter (as opposed to lumped parameter) models
>> to be described.
>
> You are right that Simile requires modellers to undertake discretisation (e.g.
> spatial) 'outside the model', so that yes, they do have to specify a smooth field in
> Simile using 'discrete components'.   The difference with CellML is (I think, as
> said above) that we can define the multiple-instance aspect very concisely (we
> specify the discretised component just once).   We have from time to time
> thought about a modelling front end that would enable users to specify
> continuous fields, and to have a tool that generated a spec for a Simile model
> with an appropriate discretisation. FieldML sounds like it could be ideal for this
> purpose.

Distributed parameter (finite element and boundary element) modelling is our
bread and butter. Nevertheless, the development of the FieldML specification is
an ongoing process.

>>>
>>> 2. Currently, Simile's data model is that each submodel maintains
>>> information on the components (incl. submodels) inside it - rather
>>> similar, I
>>> think, to SBML.   In moving towards XML, I have experimented with using
>>> the nesting of XML elements to manage this nesting implicitly.   This
>>> seems
>>> neat - and could open the door to model construction using Xinclude -
>>> but:
>>> a - it is a rather rigid way of capturing hierarchical information; and
>>> b - does not allow submodel-inclusion-by-reference.
>>> So at them moment I'm torn between going down that route - or switching
>>> across to the CellML approach, and the high degree of flexibility that
>>> brings
>>> in, including the ability to define one's one nesting relationship.
>>

>> We are very happy with the solution defined in CellML 1.1. It seems (to
>> us) a simple, natural, and flexible way of composing models from
>> submodels. In a sense it provides a very simple typing/instantiation
>> mechanism. We are endevouring to expand the power of this by adding a
>> much more general typing mechanism through the integration of
>> ontologies.
>
> I still like your approach - and I'm still pushing ahead with the having the XML
> tree reflect the model nesting hierarchy!   It's not restricting us just now; it
> makes browsing models (using IE's built-in XML viewer) really nice; and it's
> easy enough to transform into your more general notation should we so chose.
>
>> Please let me know if you have any comments on the above - I think that
>> we both have much to gain by continuing the dialogue...
>
> Great. I'd especially like to hear your reaction to my interpretation of your import
> mechanism versus Simile's multiplicity/association concepts.
>
> Best wishes,
> Robert

Best wishes
Poul

Poul Nielsen
Bioengineering Institute
The University of Auckland
New Zealand

- Matt's Update:

    - The second 4th year project offered is to do a **MathML API in Python**. Andre: Why? We already have a MathML API in C++. Matt: This API will not be using the DOM Lvl 2 interface. Shane: Will there be plans to converge these two APIs?

    - Still working on the CellML **Road Map** for tools.

    - **Anatomy ontology web interface** is almost finished. Just waiting on For Mike Lovell-Smith to clean up some of the relationships. It draws pictures of some of the various things referenced (bones, muscles). The next step is to get feedback.

- Poul's Update: Stephanie Jor has finished her summer work. She has been setting up compound structures (arrays, lists) and variables for numbers (integers, complex). Poul is looking to get Stephanie set up for part time work through the next year.

- Autumn, Check on why people can't send to `<info@cellml.org>` and circulate web documents for Team CellML to edit.