The CellML API: An update

Andrew Miller – ak.miller@auckland.ac.nz

AUCKLAND BIOENGINEERING INSTITUTE
THE UNIVERSITY OF AUCKLAND NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Introduction

The CellML API is a reusable contract between an application and an implementation of the CellML API.

In Auckland, we produce both the API and a complete implementation of the API.

This talk will cover:

1. Briefly: An overview of the API.
2. What has changed in the last year.

Structure of the API

Key parts of the API

1. Interfaces and Object Model
2. DOM (optional for implementations)
3. MathML DOM
4. RDF API
5. API Core
6. Older Extension Modules (Integration, Export Language Definition, Code Generation, Validation, Units Simplification / Expansion, MathML-to-Language services, Context)

IDL and Language Support

Interfaces written in IDL.

IDL processed to produce bindings (e.g. C++ headers) and bridges between bindings.

API written in C++ against one particular binding.

Can currently call the API from C++, Java, Python. Previously: Javascript (XPCOM) and over CORBA.

Generics & Reflection Service

AUCKLAND BIOENGINEERING INSTITUTE
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

The CellML Generics & Reflection Service (CGRS) is a new service that allows applications to:

1. Retrieve information about the types, operations, and attributes in the API programmatically (reflection)
2. Write generic code that can call API operations and set and retrieve attribute data in response to user requests, without needing to hard code the structure of the API into the application.

This is useful for:

1. Making better language bindings for dynamic languages.
2. Creating generic user interfaces that allow access to the full API.

Better Python Bindings

AUCKLAND BIOENGINEERING INSTITUTE
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

The CellML API now has significantly better Python bindings.

Built on the Generics & Reflection Service.

Dynamically object-orientated, not interface orientated – can call any operation / access any attribute supported by an object, without worrying about what interface it is on.

Supports Python's iterator protocol – easier & more idiomatic iteration through models.

TeLICeM Service

AUCKLAND BIOENGINEERING INSTITUTE
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

We created a new format called TeLICeM (“text language for the input of CellML models”) as a common text-based input format for CellML models (including their content MathML).

```
def model SimpleDAENonLinear as
  def comp main as
    var a : dimensionless { init: 1 };
    var d : dimensionless;
    var c : dimensionless;
    var t : dimensionless;
    def math as
      d(a)/d(unknown-element) = (4{unit: ""} / 7{unit: ""}) * (2{unit: ""} * power(d, 3{unit: ""}) + c);
    enddef;
    def math as
      2{type: "integer",unit: ""} * (-1{type: "integer",unit: ""} * a + power(d, 3{unit: ""}) + c) = 0{type: "integer",unit: ""};
    enddef;
    def math as
      3{type: "integer",unit: ""} * (power(d, 3{unit: ""}) + -1{type: "integer",unit: ""} * (3{type: "integer",unit: ""} * c)) = 0{type:
"integer",unit: ""};
    enddef;
  enddef;
enddef;
```

TeLICeM Service parses and serialises TeLICeM format.

SED-ML Support

AUCKLAND BIOENGINEERING INSTITUTE
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

The CellML API now includes code that lets you work with SED-ML (simulation experiment description markup language).

It could be separated out in future – SED-ML isn't CellML specific. It's convenient to include it in the CellML API for now.

There are two parts:

SED-ML Processing Service – traverse through a SED-ML description – one interface per SED-ML element.

SED-ML Running Service – run a simulation on a model and get results (currently on CellML models supported, but designed to be extensible).

You can also use the running service to just apply SED-ML changes to models if you want to do your own simulations.

Documentation

AUCKLAND BIOENGINEERING INSTITUTE
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

One major item of feedback we had about the API in our survey was that we needed to improve our documentation.

We have now consolidated all our documentation in one place, at http://cellml-api.sf.net/

It is now based entirely on the Doxygen included within the source tree.

We have also added more tutorials to help you get started with using the API.

There is still lots of room for improvement on documentation – let us know if there is something specific you think is poorly documented so we can improve it.

Building & Binaries

Another theme in the feedback in our survey on the API was that the API was hard to build, and that it would be nice if there were binaries available.

We have streamlined our build system using CMake – so it is easier to build on most platforms.

The libxml2 dependency made building harder on Windows, because it forced people to find compatible libxml2 binaries or build libxml2 themselves. We fixed that by including a version of libxml2 with the CellML API that gets built by CMake.

Our build system now automatically produces and uploads CellML API binaries on Linux (x86 / x86_64), Windows (x86) and Mac (PPC, x86, x86_64 universal binaries) – binaries are packed as an 'SDK', which most users can use instead of building from source.

Questions

AUCKLAND BIOENGINEERING INSTITUTE
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Questions?

Comments?

Suggestions?

Criticism?