Published in final edited form as:
Bioinformatics. 2008 March 15; 24(6): 880–881. doi:10.1093/bioinformatics/btn051.

LibSBML: An API Library for SBML

Benjamin J. Bornstein[*,2], **Sarah M. Keating**[*,1], **Akiya Jouraku**[3], and **Michael Hucka**[1]

[1]Biological Network Modeling Center, California Institute of Technology, Pasadena, CA, USA.

[2]NASA Jet Propulsion Laboratory, Pasadena, CA, USA.

[3]School of Fundamental Science and Technology, Keio University, Yokohama, Japan.

Abstract

Summary—LibSBML is an application programming interface library for reading, writing, manipulating and validating content expressed in the Systems Biology Markup Language (SBML) format. It is written in ISO C and C++, provides language bindings for Common Lisp, Java, Python, Perl, MATLAB and Octave, and includes many features that facilitate adoption and use of both SBML and the library. Developers can embed libSBML in their applications, saving themselves the work of implementing their own SBML parsing, manipulation, and validation software.

Availability—LibSBML 3 was released in August 2007. Source code, binaries and documentation are freely available under LGPL open-source terms from http://sbml.org/software/libsbml.

Contact—sbml-team@caltech.edu

1 INTRODUCTION

The Systems Biology Markup Language (SBML) is an XML-based format for encoding computational models of biochemical reaction networks (Finney et al., 2006; Hucka et al., 2003). It is today the de facto standard for exchanging and storing biological/biochemical models, allowing for easier systematic reuse of such models in systems biology research.

Since SBML is based upon XML, software developers can potentially support SBML using off-the-shelf XML parser libraries. Nevertheless, it is more convenient and efficient for developers to start with a higher-level API tailored specifically to SBML and its distinctive features. We developed libSBML to address this need.

The availability of an application programming interface (API) library such as libSBML presents an opportunity to go beyond simply providing support for reading, writing and manipulating SBML objects: many additional capabilities can be provided, such as checking the consistency and validity of models. Building these features directly into a popular and free API library helps uniformly improve the quality of models being produced by the majority of SBML-using software.

2 LIBSBML FEATURES

LibSBML 3 is a mature software library providing a wealth of features and capabilities. We summarize some of them in the following paragraphs.

*The first two authors should both be regarded as First Author.

1. Seamless support for different SBML Levels and Versions

With the continued evolution of SBML has come multiple SBML Levels and Versions within the Levels (Finney et al., 2006), with models “in the wild” existing in all variants. LibSBML provides a uniform API that seamlessly covers all SBML Levels and Versions, making it significantly easier for software developers to support the different definitions in their applications.

2. Support for multiple ways of handling SBML

LibSBML can read and write an SBML model to/from a file or string; create a model de novo; manipulate an existing model and (where possible) convert models between Levels and Versions of SBML.

3. Object-oriented model structure

SBML model components are represented as an object hierarchy in libSBML mirroring SBML's structure. Each object class defines the member variables and methods applicable to that kind of SBML model component. For example, the Compartment object has member variables for the size, number of spatial dimensions, and other attributes of an SBML compartment, and getter and setter methods provide the means of accessing, setting and unsetting the attributes' values. An SBML model in libSBML consists of an overall SBMLDocument object instance containing instances of ListOfComparments, ListOfSpecies, etc., each of which in turn contains lists of instances of the relevant objects.

4. Support for all SBML constructs, including notes and annotations

SBML specifies that notes (for humans) and annotations (for software) can be attached to individual SBML components within a model. LibSBML provides two interchangeable approaches to handling the contents of notes and annotations: as text strings, and as XML objects. This provides flexibility for calling applications and transparently addresses differences in the format of notes and annotations that arose over the course of SBML's evolution.

5. Support for MathML and text-string mathematical formulas

Mathematical expressions are encoded using simple infix text strings in SBML Level 1 and using a subset of MathML (Ausbrooks et al., 2001) in SBML Level 2. LibSBML provides a uniform interface to both forms. It stores formulas internally using Abstract Syntax Trees (AST) and provides methods for reading and writing either text-string or MathML forms. LibSBML can also translate expressions between these two forms—a feature that application software developers find convenient.

6. Validation of SBML

LibSBML uses the Visitor Pattern (Gamma et al., 1995) to provide a means of walking down the hierarchical object structure and applying validation and constraint tests to model components. This is used to validate an SBML model according to the Level and Version of SBML declared by the model. Validation checks performed include identifier consistency (e.g., making sure that all parameters are actually defined in the model), MathML consistency, unit consistency (see below), and many others. LibSBML implements all the validation rules defined by the SBML Level 2 Version 3 specification (Hucka et al., 2007). Failures are reported using libSBML's uniform error reporting facility, which logs errors and warnings in the top-level SBMLDocument object. Finally, libSBML provides the scaffolding permitting user applications to define their own additional validators using the same framework employed internally by libSBML.

7. Verification of unit consistency

LibSBML implements rules for performing unit and dimensional analysis to help ensure that a model's various components and mathematical formulas are self-consistent. The rules use the underlying validation and error reporting framework described above. Applications can opt to disable these checks if they prefer.

8. Support for MIRIAM annotations

The Minimum Information Requested in the Annotation of biochemical Models (MIRIAM; Le Novère et al., 2005) defines guidelines for encoding and annotating quantitative models. LibSBML provides API methods for creating and manipulating MIRIAM compliant annotations. Relevant information can be defined using the methods and libSBML will generate the appropriate MIRIAM-compliant RDF annotations.

9. Support for SBO

The *Systems Biology Ontology* (SBO; Le Novère et al., 2007) is a machine-readable ontology tailored specifically for computational modeling in systems biology. Annotating a model with SBO terms adds semantic information that can permit better software interpretation of the model's mathematical structure and (potentially) the ability to translate a model between different mathematical frameworks. LibSBML provides support for adding and working with SBO terms in SBML models.

In addition to these features, LibSBML provides many other capabilites. Interested readers are invited to visit the libSBML website (http://sbml.org/software/libsbml) to learn more.

3 IMPLEMENTATION AND DISTRIBUTION

LibSBML is written in ISO C and C++ and currently provides language bindings for C, C++, Common Lisp, Java, Python, Perl, MATLAB and Octave. It requires installing an underlying XML parser library and can be used with either libXML (http://xmlsoft.org/), Expat (http://expat.sourceforge.net/) or Xerces (http://xerces.apache.org/).

LibSBML is supported on Linux, Windows and MacOS X, and is distributed in both source-code form and as precompiled dynamic libraries for Windows. Extensive user documentation is provided; it is generated with Doxygen (van Heesch, 2007) and available both online and for downloading.

LibSBML is distributed under the Lesser GNU Public License (LGPL), which among other benefits, allows the software to be used in proprietary application software.

ACKNOWLEDGEMENTS

We warmly thank the following individuals for code contributions and bug fixes: Frank Bergmann, Bill Denney (Octave language bindings), Andrew Finney, Christoph Flamm (Perl language bindings), Ralph Gauges (SBML Layout extension support), Martin Ginkel (Lisp bindings), Damon Hachmeister, Johan Hattne, Stefan Hoops (first interface to the Expat XML parser), Ben Kovitz, Konstantin Kozlov, Michael Lawrence, Rainer Machné, and Jacek Puchalka. We also thank all other people who used, tested, reported, and otherwise contributed to the development of libSBML over the years.

FUNDING

The development of libSBML has been supported by the following organizations: the National Institutes of Health (USA) under grants R01 GM070923 and R01 GM077671; the International Joint Research Program of NEDO (Japan); the JST ERATO-SORST Program (Japan); the Japanese Ministry of Agriculture; the Japanese Ministry of Education, Culture, Sports, Science and Technology; the BBSRC e-Science Initiative (UK); the DARPA IPTO Bio-Computation Program (USA); the Army Research Office's Institute for Collaborative Biotechnologies (USA); the Air Force Office

of Scientific Research (USA); the California Institute of Technology (USA); the University of Hertfordshire (UK); the Molecular Sciences Institute (USA); the Systems Biology Institute (Japan); and Keio University (Japan).

REFERENCES

Ausbrooks, R.; Buswell, S.; Dalmas, S.; Devitt, S.; Diaz, A.; Hunter, R.; Smith, B.; Soiffer, N.; Sutor, R.; Watt, S. 2001. Mathematical Markup Language (MathML) Version 2.0 (Second Edition) W3C Recommendation 21 October 2003

Finney, A.; Hucka, M.; Bornstein, BJ.; Keating, SM.; Shapiro, BE.; Matthews, J.; Kovitz, BL.; Schilstra, MJ.; Funahashi, A.; Doyle, JC.; Kitano, H. Software Infrastructure for Effective Communication and Reuse of Computational Models. In: Szallasi, Z.; Stelling, J.; Periwal, V., editors. System Modeling in Cell Biology: From Concepts to Nuts and Bolts. MIT Press; Cambridge, Mass: 2006.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley; Boston: 1995.

Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin II, Hedley WJ, Hodgman TC, Hofmeyr J-H, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le Novère N, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J. The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models. Bioinformatics 2003;19:524–531. [PubMed: 12611808]

Hucka, M.; Finney, AM.; Hoops, S.; Keating, SM.; Le Novère, N. Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions. 2007. Available from Nature Precedings, http://hdl.nature.com/10101/npre.2007.58.2

Le Novère N, Finney A, Hucka M, Bhalla US, Campagne F, Collado-Vides J, Crampin EJ, Halstead M, Klipp E, Mendes P, Nielsen P, Sauro H, Shapiro B, Snoep JL, Spence HD, Wanner BL. Minimum Information Requested in the Annotation of biochemical Models (MIRIAM). Nature Biotechnology 2005;23:1509–1515.

Le Novère, N.; Courtot, M.; Laibe, C. Adding Semantics in Kinetics Models of Biochemical Pathways. In: Kettner, C.; Hicks, MG., editors. 2nd International ESCEC Workshop on Experimental Standard Conditions on Enzyme Characterizations. Beilstein Institut; Rüdesheim, Germany: 2006.

van Heesch, D. Doxygen. 2007. Available via the World Wide Web at http://www.stack.nl/~dimitri/doxygen/index.html