BIOINFORMATICS APPLICATIONS NOTE Vol. 27 no. 15 2011, pages 2167–2168
doi:10.1093/bioinformatics/btr361

Systems biology

Advance Access publication June 22, 2011

JSBML: a flexible Java library for working with SBML

Andreas Dräger[1,*,†], Nicolas Rodriguez[2,†], Marine Dumousseau[2], Alexander Dörr[1], Clemens Wrzodek[1], Nicolas Le Novère[2], Andreas Zell[1] and Michael Hucka[3,*]

[1]Center for Bioinformatics Tuebingen (ZBIT), University of Tuebingen, Tübingen, Germany, [2]European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, UK and [3]Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA, USA

Associate Editor: Jonathan Wren

ABSTRACT

Summary: The specifications of the Systems Biology Markup Language (SBML) define standards for storing and exchanging computer models of biological processes in text files. In order to perform model simulations, graphical visualizations and other software manipulations, an in-memory representation of SBML is required. We developed JSBML for this purpose. In contrast to prior implementations of SBML APIs, JSBML has been designed from the ground up for the Java™ programming language, and can therefore be used on all platforms supported by a Java Runtime Environment. This offers important benefits for Java users, including the ability to distribute software as Java Web Start applications. JSBML supports all SBML Levels and Versions through Level 3 Version 1, and we have strived to maintain the highest possible degree of compatibility with the popular library libSBML. JSBML also supports modules that can facilitate the development of plugins for end user applications, as well as ease migration from a libSBML-based backend.

Availability: Source code, binaries and documentation for JSBML can be freely obtained under the terms of the LGPL 2.1 from the website http://sbml.org/Software/JSBML.

Contact: jsbml-team@sbml.org

Supplementary information: Supplementary data are available at Bioinformatics online.

Received on March 29, 2011; revised on May 23, 2011; accepted on June 12, 2011

1 INTRODUCTION

The XML-based Systems Biology Markup Language (SBML, Hucka *et al.* 2003) is the *de facto* standard file format for the storage and exchange of quantitative computational models in systems biology, supported by more than 210 software packages to date (March 2011). Much of this success is due to its clearly defined specifications and the availability of libSBML (Bornstein *et al.*, 2008), a portable, robust, full-featured and easy-to-use library.

LibSBML provides many methods for the manipulation and validation of SBML files through its Application Programming Interface (API). Primarily written in C and C++, libSBML also provides automatically generated language bindings for Java™, among other programming languages. However, the full platform independence brought by the use of Java is limited in libSBML because the binding is only a wrapper around the C/C++ core, implemented using the Java Native Interface (JNI). As a consequence, some software developers experience difficulties deploying portable libSBML-based Java applications. Several groups in the SBML community thus began to develop their own Java libraries for SBML. To avoid needless duplication, some of these groups recently pooled their efforts and created JSBML, an open-source project to develop a pure Java library for SBML.

The primary aim of the project is to provide an API that maps all SBML elements to a flexible and extended Java type hierarchy. Where possible, JSBML strives for 100% compatibility with libSBML's Java API, to ease the transition from one library to the other. There are currently no plans to re-implement the more complex functionalities of libSBML, such as model consistency checking, SBML validation and conversion between different SBML Levels and Versions; separate community efforts are under way to provide such libSBML facilities via web services.

The software produced by the project is freely available from http://sbml.org/Software/JSBML.

2 IMPLEMENTATION

A key achievement of the JSBML project is the development of an extended type hierarchy, designed from scratch based on the SBML specifications, but still following the naming conventions of methods and classes in libSBML. For each element defined in at least one SBML Level/Version combination, JSBML provides a corresponding class reflecting all its properties. SBML elements or attributes not part of higher SBML Levels (removed or made obsolete) are marked as deprecated. JSBML defines superclasses or interfaces for elements that share common properties. For instance, the interface `NamedSBase` does not directly correspond to a data type in one of the SBML specifications, but serves as the superclass of all `SBase`-derived classes that can be addressed by an identifier and a name. Similarly, all classes that may contain a mathematical expression implement the interface `MathContainer`. A full overview of this type hierarchy can be found in the Supplementary Material associated with this article. JSBML also supports SBML *notes* in XHTML format, as well as SBML *annotations*, including MIRIAM identifiers (Le Novère *et al.*, 2005) and SBO terms (Le Novère *et al.*, 2006). When building JSBML, the latest SBO OBO export can directly be downloaded and parsed (Holland *et al.*, 2008). The `Model` class provides several methods, all beginning with the name `find*`, for querying SBML elements. Filters enable users to search lists for elements that possess

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

(a)

```
1  import javax.swing.*;
2  import org.sbml.jsbml.*;
3
4  /** Displays the content of an SBML file in a {@link JTree} */
5  public class JSBMLvisualizer extends JFrame {
6
7    /** @param document The sbml root node of an SBML file */
8    public JSBMLvisualizer(SBMLDocument document) {
9      super(document.getModel().getId());
10     getContentPane().add(new JScrollPane(new JTree(document)));
11     pack();
12     setVisible(true);
13   }
14   /** @param args Expects a valid path to an SBML file. */
15   public static void main(String[] args) throws Exception {
16     new JSBMLvisualizer((new SBMLReader()).readSBML(args[0]));
17   }
18 }
```

(b)

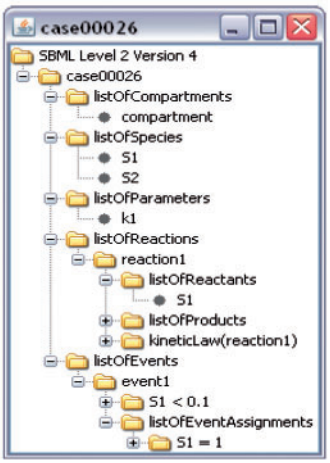

Fig. 1. (**a**) The SBML parser in JSBML understands the hierarchical data structure of SBML; (**b**) Example for SBML test case 26. Using JSBML for reading and visualizing an SBML file. The type `SBase` extends the Java interfaces `Serializable` for saving JSBML objects in binary form or sending them over a network connection, `Cloneable` for creating deep object copies and `TreeNode`. The last interface allows callers to apply any recursive operation, such as using `JTree` for display [see (b) for an example].

specific properties. All `ListOf*` elements in JSBML implement Java's `List` interface, making iteration and the use of generic Java types possible. Figure 1 demonstrates how the hierarchically structured content of an SBML file can be easily visualized in the form of a tree.

JSBML includes parsers that read mathematical formulas in both MathML format and an infix formula syntax. Internally, it converts formulas into an abstract syntax tree representation; it can write out the trees in MathML, infix and LaTeX formula notations. In addition, although JSBML does not implement full-featured consistency checking of SBML models, it does throw Java exceptions in some situations to prevent users from creating invalid content. It implements a check for overdetermined models using the algorithm of Hopcroft and Karp (1973); this is also used to identify variables in algebraic rules. Further, JSBML can automatically derive the units of a mathematical expression. Whenever a property of some `SBase` is altered, an `SBaseChangeEvent` is fired that notifies dedicated listeners. As one possible application, graphical user interfaces could automatically react when the model is changed. Using modules, JSBML capabilities can be further extended; it can therefore be used as a communication layer between an application and libSBML or CellDesigner (Funahashi *et al.*, 2003)—this also facilitates turning an existing application into a plugin for CellDesigner.

3 CONCLUSION

JSBML is an ongoing project that provides comprehensive and entirely Java-based data structures to read, write and manipulate SBML files. Its layered architecture allows for the creation of Java Web Start applications and CellDesigner plugins based on stand-alone programs with very little effort. New versions of SBMLsqueezer (Dräger *et al.*, 2008) and Biomodels Database (Li *et al.*, 2010) have already been released using JSBML. Many other projects are planned.

Funding: National Institute of General Medical Sciences (NIGMS, USA); funds from EMBL-EBI (Germany, UK); Federal Ministry of Education and Research (BMBF, Germany) in the projects Virtual Liver and Spher4Sys (grant numbers 0315756 and 0315384C). The grant number for the NIH grant that was, among others, used for the JSBML article reads 2R01GM070923.

Conflict of Interest: none declared.

REFERENCES

Bornstein,B.J. *et al.* (2008) LibSBML: an API Library for SBML. *Bioinformatics*, **24**, 880–881.

Dräger,A. *et al.* (2008) SBMLsqueezer: a CellDesigner plug-in to generate kinetic rate equations for biochemical networks. *BMC Syst. Biol.*, **2**, 39.

Funahashi,A. *et al.* (2003) CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BioSilico*, **1**, 159–162.

Holland,R.C.G. *et al.* (2008) BioJava: an open-source framework for bioinformatics. *Bioinformatics*, **24**, 2096–2097.

Hopcroft,J.E. and Karp,R.M. (1973) An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, **2**, 225.

Hucka,M. *et al.* (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**, 524–531.

Le Novère,N. *et al.* (2005) Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat. Biotechnol.*, **23**, 1509–1515.

Le Novère,N. *et al.* (2006) Adding semantics in kinetics models of biochemical pathways. In Kettner and Hicks (eds), *2nd International ESCEC Workshop. Beilstein Institut, Rüdesheim, Germany*, ESEC, Rüdessheim/Rhein, Germany, pp. 137–153.

Li,C. *et al.* (2010) BioModels database: an enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Syst. Biol.*, **4**, 92.