BIOINFORMATICS APPLICATIONS NOTE

Vol. 00 no. 00 2011
Pages 1–3

JSBML: a flexible and entirely Java-based library for working with SBML

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

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

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Summary: The specifications of the Systems Biology Markup Language (SBML) define a standard for storing and exchanging biochemical models in XML-formatted text files. To perform higher-level operations on these models, e.g., numerical simulation or visual representation, an appropriate mapping to in-memory objects is required. To this end, the Application Programming Interface (API) library JSBML has been developed. In contrast to earlier approaches, JSBML has been especially designed for the Java™ programming language and can therefore be used on all platforms, for which a Java Runtime Environment is available. In this way, programs based on JSBML can, for instance, be released as Java webstart applications. JSBML's internal data structures have been completely developed from scratch based on the definitions in the SBML specifications but with respect to achieve the highest possible degree of compatibility to the existing library libSBML. JSBML supports all SBML levels and versions that are available today. In addition, JSBML provides modules that facilitate the development of CellDesigner plugins or ease the migration from a libSBML backend.

Availability: Source code, binaries, and documentation of JSBML can be downloaded under the terms of LGPL 2.1 at http://sbml.org/Software/JSBML.

Contact: jsbml-team@sbml.org

Supplementary information: Supplementary data are available at Bioinformatics online.

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 biochemical network models, and is supported by more than 200 software packages to date (October 2010). Much of this success is due to its clearly defined specifications and the availability of libSBML (Bornstein *et al.*, 2008), a portable, robust, and easy-to-use library.

LibSBML provides many methods for manipulating and validating SBML files through its Application Programming Interface (API). Originally written in C and C++, libSBML also provides automatically generated language bindings for Java™, MATLAB™, Perl, and many more. However, the platform independence of Java is compromised in libSBML due to the fact that the language binding is a wrapper around the C/C++ core and also due to libSBML's dependency on external XML parsers. Therefore, many software developers experience difficulties in the deployment of portable libSBML-based Java applications. The wrapped C/C++ code often hides internal states of the objects, which hampers the debugging process and sometimes it is not obvious which model elements point to each other. Further, the libSBML API and type hierarchy are not sufficiently intuitive from a Java programmer's perspective just because they were not designed directly for Java. For these reasons, several groups in the SBML community have mounted an open-source effort to develop a pure Java library for SBML. Here we present the JSBML project, whose products are freely available at the web site http://sbml.org/Software/JSBML.

The project's aim is to provide an SBML parser and programming library that maps all SBML elements to a flexible and extended type hierarchy. Where possible, JSBML strives to attain 100 % compatibility with libSBML's Java API, to ease a switch from one library to the other. Currently, JSBML supports all constructs for SBML up to the latest Level 3 Version 1 specification, including an API to add SBML extensions. At the moment, there are no plans to re-implement some of the more complex functions of libSBML, such as model consistency checking, SBML validation, and the conversion between different SBML Levels and Versions, since separate community efforts are expected to make them available to JSBML via web services.

2 A BRIEF OVERVIEW OF JSBML

The main achievement of the JSBML project is its extended type hierarchy that has been designed from scratch based on the SBML specifications only, but with respect to the naming conventions of methods and classes in libSBML. For each element that is defined in at least one of the SBML levels or versions, JSBML provides a corresponding class that reflects all of its properties. Classes for SBML elements that are not part of the

*to whom correspondence should be addressed

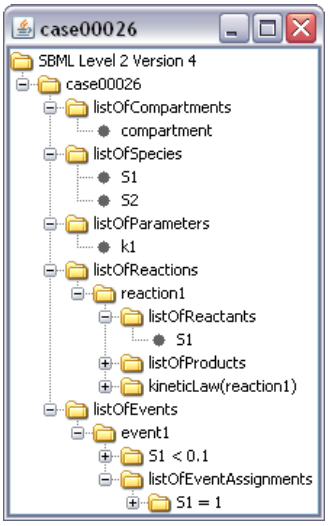
```java
import javax.swing.*;
import org.sbml.jsbml.*;

/** Displays the content of an SBML file in a {@link JTree} */
public class JSBMLvisualizer extends JFrame {

  /** @param document The sbml root node of an SBML file */
  public JSBMLvisualizer(SBMLDocument document) {
    super(document.getModel().getId());
    getContentPane().add(new JScrollPane(new JTree(document)));
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    pack();
    setVisible(true);
  }
  /** @param args Expects a valid path to an SBML file. */
  public static void main(String[] args) throws Exception {
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    new JSBMLvisualizer((new SBMLReader()).readSBML(args[0]));
  }

}
```

(a) Example source code using JSBML



(b) Example for SBML test case 26

Fig. 1: A minimal example using JSBML for reading and visualizing an SBML file. In JSBML, the abstract type `SBase` has become an interface for practical and technical reasons, which extends the Java interfaces `Serializable`, `Cloneable`, and `TreeNode`. The latter one allows users to perform any kind of recursive operations on any instance of `SBase` such as visualizing it in a `JTree`. The `Serializable` interface allows users to save all JSBLM objects in binary form or to send them via an internet connection without the need to explicitly write and parse SBML code again. With the help of the `Cloneable` interface, deep copies of all derived elements can be created.

most recent specification are marked as deprecated. The same holds for obsolete properties within a class. For elements sharing common properties, JSBML defines a superclass or an interface for them. For instance, the interface `NamedSBase` does not correspond to a data type in one of the SBML specifications directly, but serves as the superclass of all those instances of `SBase` that can be addressed by an identifier and a name. Similarly, all classes that may contain a mathematical expression implement the interface `MathContainer`, whereas system variables, i.e., `Compartment`, `Species`, `Parameter`, and `SpeciesReference`, are all derived from the type `Variable`. A full overview of this type hierarchy can be found in the supplementary file. JSBML also supports annotations, including MIRIRIAM (Le Novère *et al.*, 2005) and SBO (Le Novère *et al.*, 2006) with the help of the BioJava OBO parser (Holland *et al.*, 2008), and notes in XHTML form. Predefined units are automatically annotated with the correct Unit Ontology terms. The `Model` class provides several `find*` methods to query for SBML elements. Filters enable searching lists for elements with certain properties. All `ListOf*` elements implement Java's `List` interface making it possible to iterate over its elements and usage of generic types. Fig. 1 demonstrates how the hierachically structured content of an SBML file can be easily visualized in form of a tree. JSBML reads SBML code from various sources. Besides files and strings also URLs of SBML files can be accepted. Various constructors and create methods of all elements allow users to manipulate or build models in memory. Special parsers read mathematical expressions from C-language like formulas or MathML (since SBML Level 2) into abstract syntax trees, which are the only internal representation of formulas in JSBML. These trees can directly be written in MathML, formula strings, or LaTeX code and also be visualized in a `JTree` since they also implement `TreeNode`. This feature becomes particularly usefull in the debugging process. Although JSBML does not implement a fully-featured consistency check for SBML models, several exceptions help programmers to create invalid elements depending on Level/Version combination. An overdetermination check for the model has been implemented based on the algorithm of Hopcroft and Karp (1973), which also identifies the variables in algebraic rules. Furthermore, JSBML can automatically derive the unit of a mathematical expression. Whenever a property of some `SBase` is altered, an `SBaseChangeEvent` is fired that notifies dedicated listeners. In this way, graphical user interfaces can automatically react. JSBML supports logging, i.e., it is easily possible to create log files and to keep track of warnings, information or error messages. Furthermore, JSBML can be used as a communication layer between CellDesigner (Funahashi *et al.*, 2003) or libSBML and any other application. In this way, JSBML facilitates turning an existing application into a plug-in for CellDesigner, thereby translating between CellDesigner-specific annotations and SBO terms.

3 IMPLEMENTATION

JSBML is entirely written in the Java™ programming language version 1.5 and does not require the installation of any other software besides a Java Virtual Machine. It was successfully tested under MacOS X, Microsoft Windows XP, Vista, and Windows 7, and openSuSE 11.2 and Ubuntu Linux 10.04. JSBML is distributed in form of various JAR files (including or excluding required third-party libraries) and source code. In addition, a convenient ANT file with several options allows users to easily create customized JAR files. Being distributed under the terms of the Lesser GNU Public License (LGPL), the JSBML library can freely be used even in proprietary software.

4 CONCLUSION

JSBML is a young, ongoing software 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 plug-ins based on stand-alone programs with very little effort. One program, SBMLsqueezer (Dräger *et al.*, 2008), has already been re-implemented and released under version 1.3 using JSBML, a simulator, which is benchmarked on the SBML test suite will be available soon, and many other projects are planned.

ACKNOWLEDGEMENT

Funding: The development of JSBML is funded in part by a grant from the National Institute of General Medical Sciences (NIGMS, USA), funds from EMBL-EBI (Germany and UK), the Federal Ministry of Education and Research (BMBF, Germany) in the Virtual Liver Network (grant number 0315756) and the MedSys project Spher4Sys (grant number 0315384C), as well as the University of Tuebingen, Germany.

Conflict of Interest: none declared.

REFERENCES

Bornstein, B. J., Keating, S. M., Jouraku, A., and Hucka, M. (2008). LibSBML: an API Library for SBML. *Bioinformatics*, **24**(6), 880–881.

Dräger, A., Hassis, N., Supper, J., Schröder, A., and Zell, A. (2008). SBMLsqueezer: a CellDesigner plug-in to generate kinetic rate equations for biochemical networks. *BMC Systems Biology*, **2**(1), 39.

Funahashi, A., Tanimura, N., Morohashi, M., and Kitano, H. (2003). CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BioSilico*, **1**(5), 159–162.

Holland, R. C. G., Down, T., Pocock, M., Prlić, A., Huen, D., James, K., Foisy, S., Dräger, A., Yates, A., Heuer, M., and Schreiber, M. J. (2008). BioJava: an Open-Source Framework for Bioinformatics. *Bioinformatics*, **24**(18), 2096–2097.

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

Hucka, M., Finney, A., Sauro, H., and Bolouri, H. (2003). Systems Biology Markup Language (SBML) Level 1: Structures and Facilities for Basic Model Definitions. Technical Report 2, Systems Biology Workbench Development Group JST ERATO Kitano Symbiotic Systems Project Control and Dynamical Systems, MC 107-81, California Institute of Technology, Pasadena, CA, USA.

Le Novère, N., Finney, A., Hucka, M., Bhalla, U. S., Campagne, F., Collado-Vides, J., Crampin, E. J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B. E., Snoep, J. L., Spence, H. D., and Wanner, B. L. (2005). Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature Biotechnology*, **23**(12), 1509–1515.

Le Novère, N., Courtot, M., and Laibe, C. (2006). Adding semantics in kinetics models of biochemical pathways. In C. Kettner and M. G. Hicks, editors, *2nd International ESCEC Workshop on Experimental Standard Conditions on Enzyme Characterizations. Beilstein Institut, Rüdesheim, Germany*, pages 137–153, Rüdessheim/Rhein, Germany. ESEC.