BIOINFORMATICS APPLICATIONS NOTE

Vol. 00 no. 00 2011
Pages 1–2

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], 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
[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 standards for storing and exchanging models in XML-formatted text files. To perform model simulations, graphical visualizations, and other software manipulations, it is necessary to map the SBML representation to an in-memory form. We developed JSBML for this purpose. In contrast to earlier approaches, 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 applications such as CellDesigner, 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.

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 (Mar. 2010). 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 of Java is compromised in libSBML because the Java binding is essentially only a wrapper around the C/C++ core, implemented using the Java Native Interface. As a conseqeuence, many software developers experience difficulties deploying portable libSBML-based Java applications. For this and other reasons, several groups in the SBML community began to develop their own Java libraries for SBML. To avoid needless duplication of effort, some of these groups later 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 library 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 reimplement the more complex functionality of libSBML, such as model consistency checking, SBML validation, and conversion between different SBML Levels and Versions; separate community efforts are underway to provide such libSBML facilities via web services.

We describe JSBML in the rest of this article. The fruits of the project are freely available from http://sbml.org/Software/JSBML.

2 A BRIEF OVERVIEW OF JSBML

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 of 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 data associated with this article. JSBML also supports SBML *notes* in XHTML format, as well as SBML *annotations*, including MIRIAM (Le Novère *et al.*, 2005) and SBO (Le Novère *et al.*, 2006). When building JSBML, the latest SBO file 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 specific properties. All `ListOf*` elements in JSBML implement Java's `List` interface, making

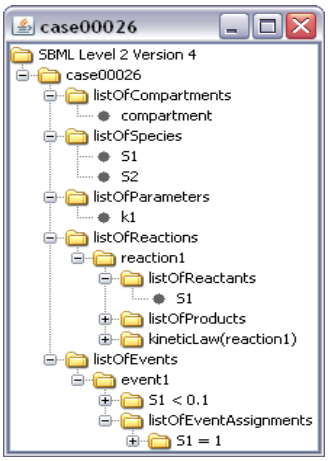
*to whom correspondence should be addressed

```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)));
    pack();
    setVisible(true);
  }
  /** @param args Expects a valid path to an SBML file. */
  public static void main(String[] args) throws Exception {
    new JSBMLvisualizer((new SBMLReader()).readSBML(args[0]));
  }
}
```

(a) The SBML parser in JSBML understands the hierarchical data structure of SBML.



(b) Example for SBML test case 26.

Fig. 1: 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 1b for an example).

iteration and the use of generic Java types possible. Fig. 1 demonstrates how the hierachically structured content of an SBML file can be easily visualized in 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. In this way, graphical user interfaces can automatically react when the model is changed. Using modules, JSBML capabilites can be further extended; this allows JSBML to 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 IMPLEMENTATION

JSBML is written entirely in Java version 1.5 and does not require additional non-Java software. It is distributed in source-code form as well as precompiled JAR files, with different JAR distributions available depending on whether required third-party libraries are included. We also provide a convenient build file offering several options for users to easily create customized JAR files. Since it is distributed under the terms of the Lesser GNU Public License (LGPL), JSBML can freely be used even in proprietary software.

4 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 some programs have already been released using JSBML (Dräger *et al.*, 2008, 2009). A simulator, benchmarked on the SBML Test Suite, will be available soon, and many other projects are planned.

ACKNOWLEDGEMENT

Authors' contribution: NR and AD contributed equally to this work.

Funding: The development of JSBML is funded by a grant from the National Institute of General Medical Sciences (NIGMS, USA), funds from EMBL-EBI (Germany, UK), and the Federal Ministry of Education and Research (BMBF, Germany) in the projects Virtual Liver and Spher4Sys (grant numbers 0315756 and 0315384C).

Conflict of Interest: none declared.

REFERENCES

Bornstein *et al.* (2008). LibSBML: an API Library for SBML. *Bioinformatics*, **24**(6), 880–881.

Dräger *et al.* (2008). SBMLsqueezer: a CellDesigner plug-in to generate kinetic rate equations for biochemical networks. *BMC Syst. Biol.*, **2**(1), 39.

Dräger *et al.* (2009). SBML2LaTeX: Conversion of SBML files into human-readable reports. *Bioinformatics*, **25**(11), 1455–1456.

Funahashi *et al.* (2003). CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *BioSilico*, **1**(5), 159–162.

Holland *et al.* (2008). BioJava: an Open-Source Framework for Bioinformatics. *Bioinformatics*, **24**(18), 2096–2097.

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

Hucka *et al.* (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**(4), 524–531.

Le Novère *et al.* (2005). Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat. Biotechnol.*, **23**(12), 1509–1515.

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