The SBML Java™ library

Concept of JSBML

Compromise

- High compatibility to libSBML
- Java-like library

Main developers

Nicolas Rodriguez,

Alex Thomas,

Andreas Dräger

Mailing lists:

- private: jsbml-team@caltech.edu
- public: jsbml-development@googlegroups.com

How to get started?

Obtaining JSBML

- Every stable and experimental release is available for download at http://sourceforge.net/projects/jsbml/files/jsbml/
- Download the file `jsbml-X.Y-with-dependencies.jar`
- Once you have added it to the Java `CLASSPATH`, you can start working with JSBML.

How to compile JSBML-qual

Creating a JAR file with the latest code from the repository:

- Checkout the sources from sourceforge

```
svn co svn://svn.code.sf.net/p/jsbml/code/trunk JSBML
cd JSBML/core
ant jar
cd ../extensions/qual
ant jar
```

now, include the jar file from `core/build`, `core/lib`, `extension/qual/build`

Generating a big jar, including JSBML-qual:

```
cd ../..
cp -v extensions/qual/build/*.jar core/lib/
cd core
ant bigjar
```

now, you have a `jsbml-X.Y-with-dependencies.jar` that contains JSBML-qual as well

- Since JSBML 1.0β1 a precompiled version of JSBML-qual is already available for download at http://sourceforge.net/projects/jsbml/files/jsbml/

Visualizing the content of an SBML file


```java
import java.io.File;
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);
  }

  /**
   * Main routine. Note: this does not perform any error checking,
   * but should. It is an illustration only.
   * @param args Expects a valid path to an SBML file.
   */
  public static void main(String[] args) throws Exception {
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    new JSBMLvisualizer(SBMLReader.read(new File(args[0])));
  }
}
```

Example output when reading an SBML file with the code on the left from the SBML Test Suite with JSBML (on Mac OS X)

How does XML parsing work?

- Mapping between SBML elements and Java classes: `/jsbml-trunk/resources/org/sbml/jsbml/resources/cfg/SBMLCoreElements.xml`

- Then each `SBase` has a `readAttributes` and `writeAttributes` methods that take care of reading and writing the attributes of the element.

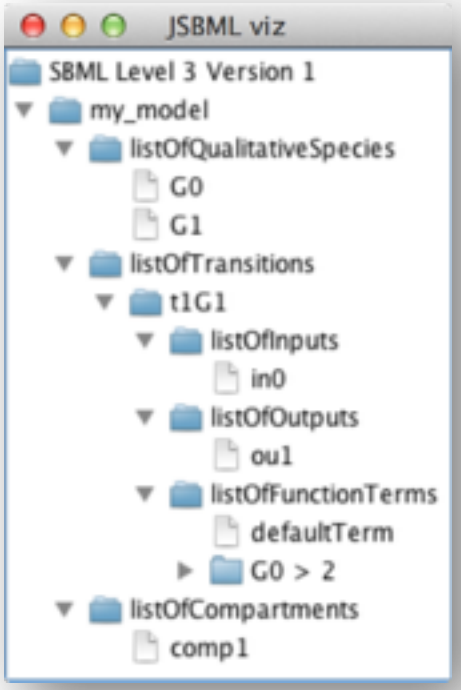
- The parsing is done in:
  - `org.sbml.jsbml.xml.stax`: main entry point of the parsing, using Stax.
  - `org.sbml.jsbml.xml.parsers`: parser independent of the underlying XML parsing library used.

- For extension packages a similar parsing/writing mechanism is used

Creating a new SBML model from scratch

```
public JSBMLexample() throws Exception {
  // Create a new SBMLDocument object, using SBML Level 3 Version 1.
  SBMLDocument doc = new SBMLDocument(3, 1);
  doc.addTreeNodeChangeListener(this);

  // Create a new SBML model, and add a compartment to it.
  Model model = doc.createModel("test_model");
  Compartment compartment = model.createCompartment("default");
  compartment.setSize(1d);

  // Create a model history object and add author information to it.
  History hist = model.getHistory(); // Will create the History, if it does not exist
  Creator creator = new Creator("Given Name", "Family Name", "Organization", "My@EMail.com");
  hist.addCreator(creator);

  // Create some sample content in the SBML model.
  Species specOne = model.createSpecies("test_spec1", compartment);
  Species specTwo = model.createSpecies("test_spec2", compartment);
  Reaction sbReaction = model.createReaction("reaction_id");

  // Add a substrate (SBO:0000015) and product (SBO:0000011) to the reaction.
  SpeciesReference subs = sbReaction.createReactant(specOne);
  subs.setSBOTerm(15);
  SpeciesReference prod = sbReaction.createProduct(specTwo);
  prod.setSBOTerm(11);

  // For brevity, WE DO NOT PERFORM ERROR CHECKING, but you should,
  // using the method doc.checkConsistency() and then checking the error log.

  // Write the SBML document to a file.
  SBMLWriter.write(doc, "test.xml", "JSBMLexample", "1.0");
}
```

How to create a simple qual model in JSBML?




```
int level = 3, version = 1;
SBMLDocument doc = new SBMLDocument(level, version);
Model model = doc.createModel("my_model");

// Creating the qualitative model extension and adding it to the document
QualModelPlugin qualPlugin = new QualModelPlugin(model);
model.addExtension(QualConstants.getNamespaceURI(level, version), qualPlugin);

// ListOfCompartments
Compartment comp1 = model.createCompartment("comp1");
comp1.setConstant(true);

// ListOfQualitativeSpecies
QualitativeSpecies g0 = qualPlugin.createQualitativeSpecies("G0", comp1, false);
QualitativeSpecies g1 = qualPlugin.createQualitativeSpecies("G1", comp1, false);

// ListOfTransitions
Transition t1G1 = qualPlugin.createTransition("t1G1");

// ListOfInputs
t1G1.createInput("in0", g0, InputTransitionEffect.consumption);
// ListOfOutputs
t1G1.createOutput("ou1", g1, OutputTransitionEffect.assignmentLevel);

// ListOfFunctionTerms
FunctionTerm defTerm = new FunctionTerm(level, version);
defTerm.setDefaultTerm(true);
defTerm.setResultLevel(0);

FunctionTerm ft1 = new FunctionTerm(level, version);
ft1.setResultLevel(1);

try {
  ft1.setMath(ASTNode.parseFormula("G0 > 2"));
} catch (ParseException exc) {
  exc.printStackTrace();
}

// G0 and G1
ASTNode andNode = new ASTNode(ASTNode.Type.LOGICAL_AND);
andNode.addChild(new ASTNode(g0.getId()));
andNode.addChild(new ASTNode(g1.getId()));

t1G1.addFunctionTerm(defTerm);
t1G1.addFunctionTerm(ft1);
```

The resulting Qual model

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" qual:required="true" level="3"
xmlns:qual="http://www.sbml.org/sbml/level3/version1/qual/version1" version="1">
  <model id="my_model">
    <qual:listOfQualitativeSpecies xmlns:qual="http://www.sbml.org/sbml/level3/version1/qual/version1">
      <qual:qualitativeSpecies qual:constant="false" qual:compartment="comp1" qual:id="G0"/>
      <qual:qualitativeSpecies qual:constant="false" qual:compartment="comp1" qual:id="G1"/>
    </qual:listOfQualitativeSpecies>
    <qual:listOfTransitions xmlns:qual="http://www.sbml.org/sbml/level3/version1/qual/version1">
      <qual:transition qual:id="t1G1">
        <qual:listOfInputs>
          <qual:input qual:transitionEffect="consumption" qual:qualitativeSpecies="G0" qual:id="in0"/>
        </qual:listOfInputs>
        <qual:listOfOutputs>
          <qual:output qual:transitionEffect="assignmentLevel" qual:qualitativeSpecies="G1" qual:id="ou1"/>
        </qual:listOfOutputs>
        <qual:listOfFunctionTerms>
          <qual:defaultTerm qual:resultLevel="0">
          </qual:defaultTerm>
          <qual:functionTerm qual:resultLevel="1">
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <apply>
                <gt/>
                <ci> G0 </ci>
                <cn type="integer"> 2 </cn>
              </apply>
            </math>
          </qual:functionTerm>
        </qual:listOfFunctionTerms>
      </qual:transition>
    </qual:listOfTransitions>
    <listOfCompartments>
      <compartment id="comp1" constant="true"/>
    </listOfCompartments>
  </model>
</sbml>
```

Recent Changes in JSBML with relevance for qual

- File `/core/resources/org/sbml/jsbml/resources/cfg/PackageParserNamespaces.xml` has been deleted
- `SBMLReader` and `SBMLWriter` now based on Java annotations → when using Eclipse and directly operating on the trunk, configuration needs to be updated as described here: https://code.google.com/p/spi/wiki/EclipseSettings
- libSBML now also supports qual and the SBML online validator validates qual models
- Naming conventions of qual objects: `QualitativeModel` is now deprecated (at the moment in the trunk, but at latest with the next release of 1.0 also in the stable version)
- Please use `QualModelPlugin` instead. Reason: Compatibility to libSBML's naming conventions
- Improved support for MAVEN: All pom.xml files have been updated

Modules

Download of modules

- LibSBML input/output:
  `svn co` svn://svn.code.sf.net/p/jsbml/code/trunk/modules/libSBMLio `libSBMLio`

- CellDesigner bridge:
  `svn co` svn://svn.code.sf.net/p/jsbml/code/trunk/modules/celldesigner `celldesigner`

- **Further modules:** Android, compare, libSBMLcompat

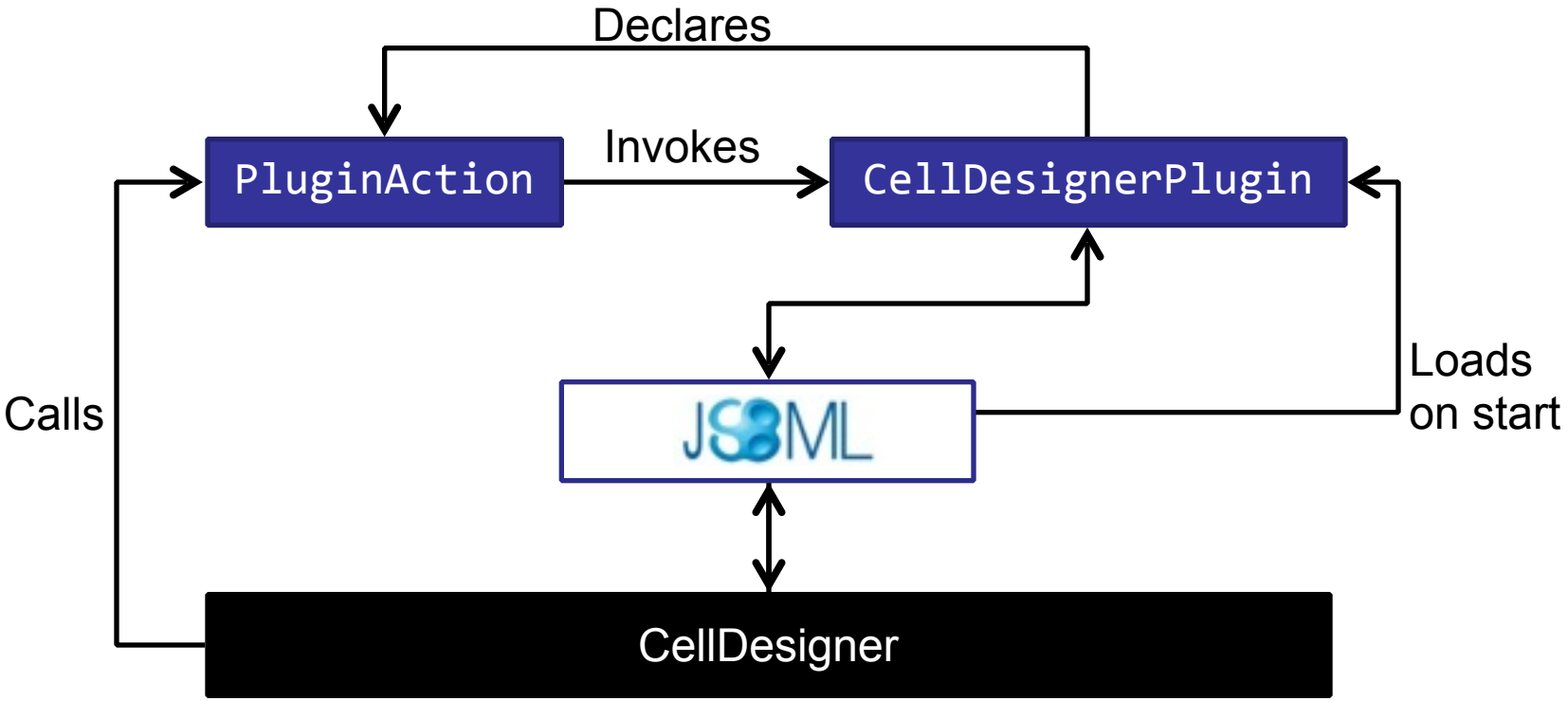
- LibSBML compatibility module for switching between libSBML and JSBML still under development

LibSBML module

```
public static void main(String[] args) {
  try {
    // Load LibSBML:
    System.loadLibrary("sbmlj");
    // Extra check to be sure we have access to libSBML:
    Class.forName("org.sbml.libsbml.libsbml");

    // Read SBML file using LibSBML and convert it to JSBML:
    SBMLInputConverter<org.sbml.libsbml.Model> reader = new LibSBMLReader();
    SBMLDocument doc = reader.convertSBMLDocument(args[0]);
    //SBMLDocument doc = new SBMLDocument(2,4);

    org.sbml.libsbml.SBMLDocument libDoc = reader.getOriginalModel().getSBMLDocument();
    //org.sbml.libsbml.SBMLDocument libDoc = new org.sbml.libsbml.SBMLDocument(2,4);
    doc.addTreeNodeChangeListener(new LibSBMLChangeListener(libDoc));

    // Run some application:
    new JSBMLVisualizer(doc);

  } catch (Throwable exc) {
    exc.printStackTrace();
  }
}
```

CellDesigner module

- Turning an existing application into a plugin for CellDesigner
- Only implementation of two abstract classes required



CellDesigner module: Example for a `PluginAction`

```
public class SimpleCellDesignerPluginAction extends PluginAction {
  /** The plugin that is triggered when this object receives appropriate actions. */
  private SimpleCellDesignerPlugin plugin;

  /** @param plugin */
  public SimpleCellDesignerPluginAction(SimpleCellDesignerPlugin plugin) {
    this.plugin = plugin;
  }

  @Override
  public void myActionPerformed(ActionEvent evt) {
    if (evt.getSource() instanceof JMenuItem) {
      JMenuItem item = (JMenuItem) evt.getSource();
      if (item.getText().equals(SimpleCellDesignerPlugin.ACTION)) {
        try {
          plugin.startPlugin();
        } catch (XMLStreamException exc) {
          JOptionPane.showMessageDialog(item, exc.getMessage(),
            exc.getClass().toString(), JOptionPane.ERROR_MESSAGE);
          exc.printStackTrace();
        }
      }
    } else {
      JOptionPane.showMessageDialog(null, "Unsupported source of action "
          + evt.getSource().getClass().getName(), "Invalid Action",
          JOptionPane.WARNING_MESSAGE);
    }
  }
}
```

CellDesigner module: Example for a `CellDesignerPlugin`

```
public class SimpleCellDesignerPlugin extends AbstractCellDesignerPlugin {

  public static final String ACTION = "Display full model tree";
  public static final String APPLICATION_NAME = "Simple Plugin";

  private PluginSBMLReader reader;

  /** Creates a new CellDesigner plugin with an entry in the menu bar. */
  public SimpleCellDesignerPlugin() {
   try {
      reader = new PluginSBMLReader(SBO.getPossibleEnzymes());
      addPluginMenu();
    } catch (Throwable exc) {
      exc.printStackTrace();
    }
  }

  @Override
  public void addPluginMenu() {
    PluginMenu menu = new PluginMenu(APPLICATION_NAME);
    PluginMenuItem menuItem = new PluginMenuItem(ACTION, new SimpleCellDesignerPluginAction(this));
    menu.add(menuItem);
  }

  /** Performs the action for which this plugin is designed.
   *  @throws XMLStreamException If the given SBML model contains errors.
   */
  public void startPlugin() throws XMLStreamException {
    Model model = reader.convertModel(getSelectedModel());
    model.getSBMLDocument().addTreeNodeChangeListener(new PluginChangeListener(this));
    new JSBMLvisualizer(model.getSBMLDocument());
  }
}
```

Some more details

Using annotation

```
Species species = model.createSpecies("species", comp1);
species.addCVTerm(new CVTerm(CVTerm.Qualifier.BQB_IS,
  "http://identifiers.org/go/GO:0006915",
  "http://identifiers.org/kegg.genes/hsa:321"));
species.addCVTerm(new CVTerm(CVTerm.Qualifier.BQB_IS_DESCRIBED_BY,
  "http://identifiers.org/pubmed/16333295"));
species.addCVTerm(new CVTerm(CVTerm.Qualifier.BQB_IS_ENCODED_BY,
  "http://identifiers.org/ensembl/ENSG00000085662"));
species.addCVTerm(new CVTerm(CVTerm.Qualifier.BQB_OCCURS_IN,
  "http://identifiers.org/kegg.reaction/R01787"));

/* This method call will return a List of Species that are annotated with the Qualifier
 * 'occursIn' and a resource attached to this qualifier that contains the String 'kegg'.
 */
model.getListOfSpecies().filter(new CVTermFilter(CVTerm.Qualifier.BQB_OCCURS_IN, ".*kegg.*"));
```

How to contribute

Creating a patch:

- Checkout the sources from sourceforge
  `svn co svn://svn.code.sf.net/p/jsbml/code/trunk JSBML`
- Do your modifications, then create a patch file:
  `svn diff > jsbml-patch.txt`
- Attach it to a tracker item or send it through the development list.

Bug tracker: http://sourceforge.net/p/jsbml/bugs/
Pivotal : https://www.pivotaltracker.com/projects/499447

Mailing lists:

- jsbml-development@caltech.edu: public list with discussion about the development of JSBML and support for users.
- jsbml-team@caltech.edu: private list for the JSBML team were anybody can send mails for support or bugs reports.

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

Thanks

http://sbml.org/Software/JSBML