JSBML

The SBML Java™ library

Concept of JSBML

- Compromise:
  - High compatibility to libSBML
  - Java-like library

- Main developers
  - Nicolas Rodriguez and Andreas Dräger
  - Both available during the meeting any time to answer JSBML-related questions

JSBML

How to get started?

Obtaining JSBML

- `https://sourceforge.net/projects/jsbml/files/jsbml`

- Download the file `jsbml-X.Y-with-dependencies.jar`.

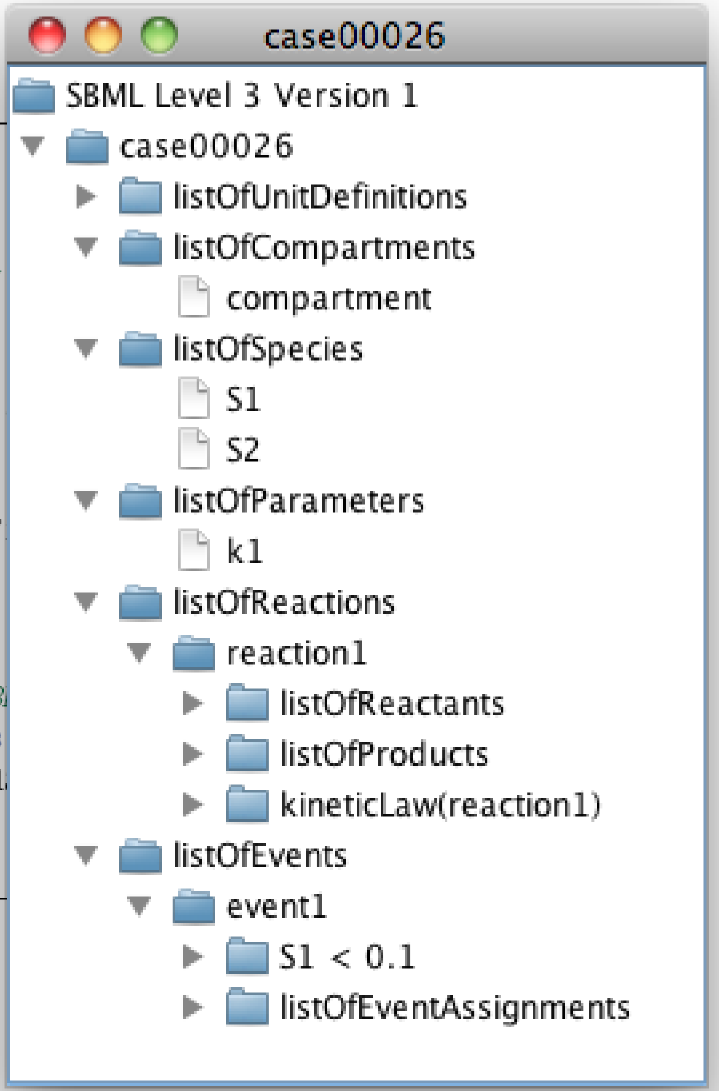
- Once you have added it to the Java `CLASSPATH`, you can start working with JSBML.

Visualizing the content of an SBML file

```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]));
  }
}
```

Visualizing the content of an SBML file

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

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

  /** @param document The sbml root node of an
  public JSBMLvisualizer(SBMLDocument document)
    super(document.getModel().getId());
    getContentPane().add(new JScrollPane(new JT
    pack();
    setVisible(true);
  }
  /** @param args Expects a valid path to an SB
  public static void main(String[] args) throws
    new JSBMLvisualizer((new SBMLReader()).read
  }
}
```



```
case00026
SBML Level 3 Version 1
▼ case00026
    ▶ listOfUnitDefinitions
    ▼ listOfCompartments
        compartment
    ▼ listOfSpecies
        S1
        S2
    ▼ listOfParameters
        k1
    ▼ listOfReactions
        ▼ reaction1
            ▶ listOfReactants
            ▶ listOfProducts
            ▶ kineticLaw(reaction1)
    ▼ listOfEvents
        ▼ event1
            ▶ S1 < 0.1
            ▶ listOfEventAssignments
```

Creating a new model

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

/** Creates an {@link SBMLDocument} and writes it's content to disk. **/
public class JSBMLexample implements SBaseChangedListener {
  public JSBMLexample() throws Exception  {

    // Create a new SBMLDocument, using SBML level 2 version 4.
    SBMLDocument doc = new SBMLDocument(2, 4);
    doc.addChangeListener(this);

    // Create a new SBML-Model and compartment in the document
    Model model = doc.createModel("test_model");
    model.setMetaId("meta_"+model.getId());
    Compartment compartment = model.createCompartment("default");
    compartment.setSize(1d);

    // Create model history
    History hist = new History();
    Creator creator = new Creator("Given␣Name", "Family␣Name",
      "My␣Organisation", "My@EMail.com");
    hist.addCreator(creator);
    model.setHistory(hist);

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

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

    // Write the SBML document to disk
    new SBMLWriter().write(doc, "test.sbml.xml", "ProgName", "Version");
  }
```

How to compile jsbml-qual

Creating a patch:

- Checkout the sources from sourceforge

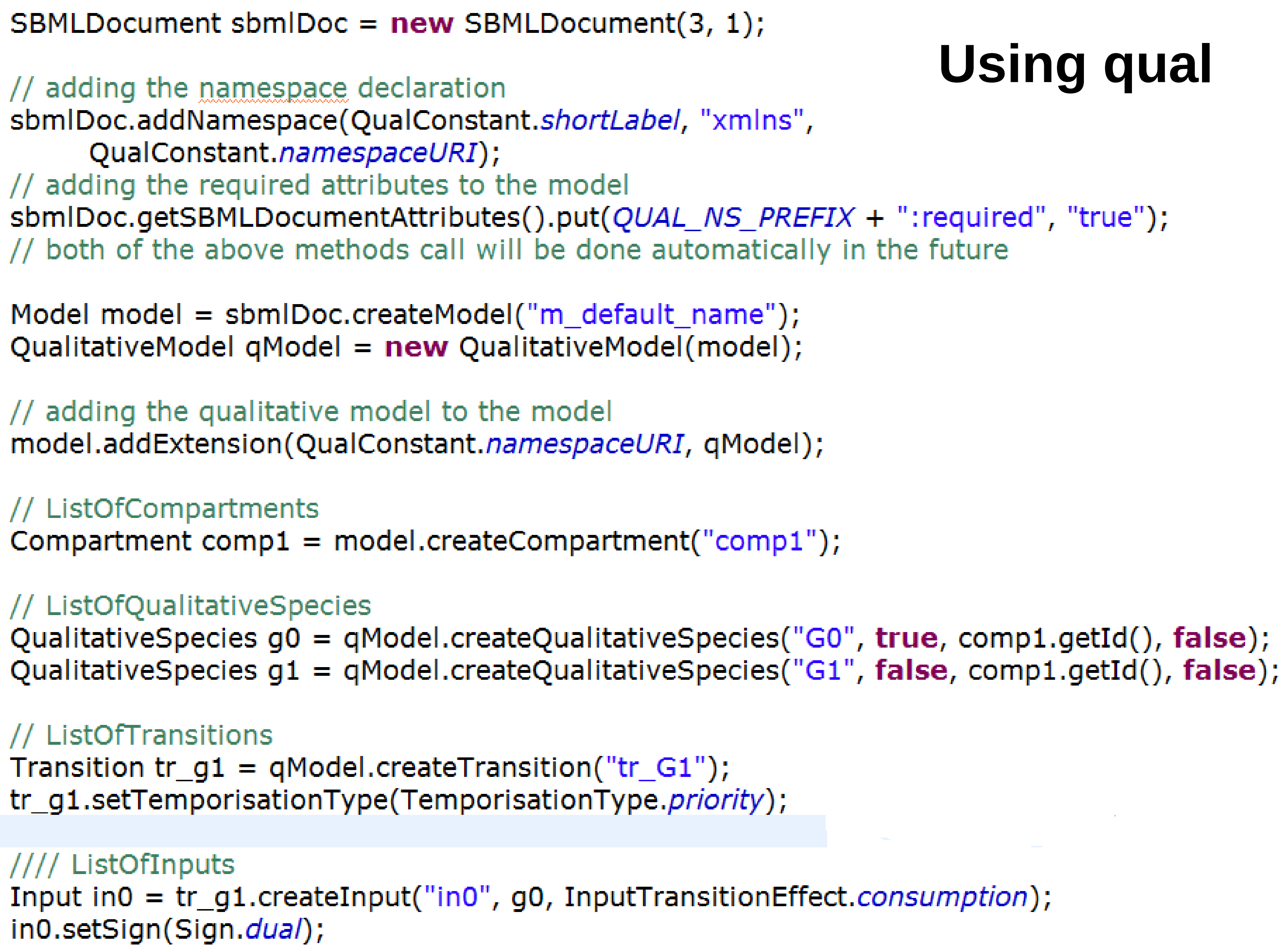
```
svn co https://jsbml.svn.sourceforge.net/svnroot/jsbml/trunk jsbml
cd jsbml/core
ant jar
cd ../extension/qual
ant jar
now, includes the jar file from core/build, core/lib, extension/qual/build
```

- Generating a big jar, including jsbml-qual:

```
cp extension/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
```

- `All of this will be automatised and we will provides the pre-compile jar files in the future`

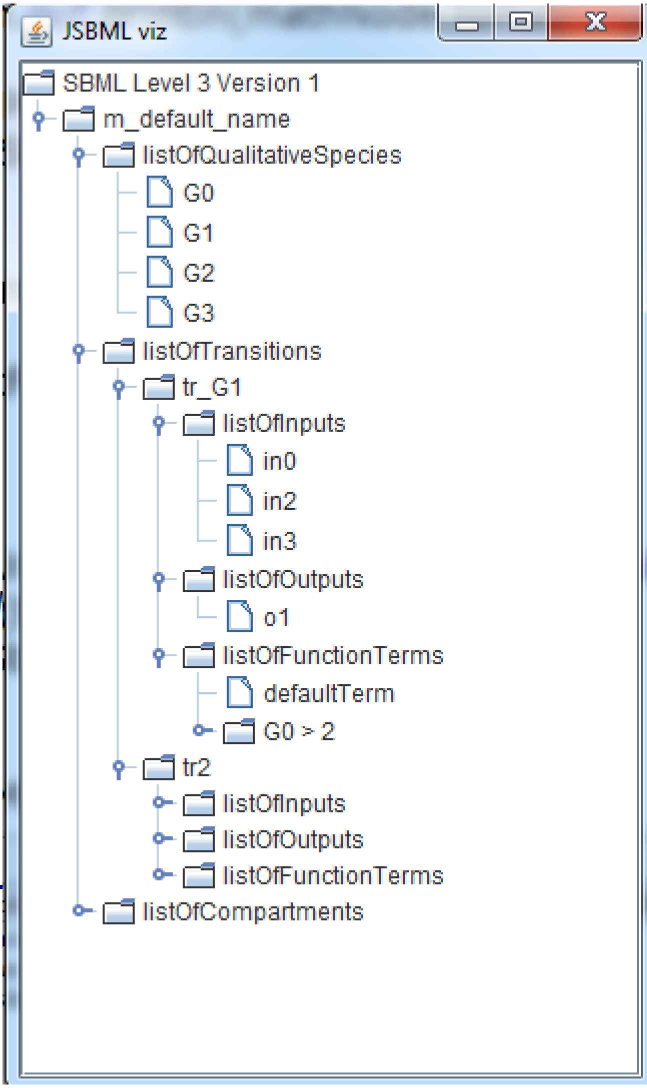
Using qual



```
SBMLDocument sbmlDoc = new SBMLDocument(3, 1);

// adding the namespace declaration
sbmlDoc.addNamespace(QualConstant.shortLabel, "xmlns",
        QualConstant.namespaceURI);
// adding the required attributes to the model
sbmlDoc.getSBMLDocumentAttributes().put(QUAL_NS_PREFIX + ":required", "true");
// both of the above methods call will be done automatically in the future

Model model = sbmlDoc.createModel("m_default_name");
QualitativeModel qModel = new QualitativeModel(model);

// adding the qualitative model to the model
model.addExtension(QualConstant.namespaceURI, qModel);

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

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

// ListOfTransitions
Transition tr_g1 = qModel.createTransition("tr_G1");
tr_g1.setTemporisationType(TemporisationType.priority);

//// ListOfInputs
Input in0 = tr_g1.createInput("in0", g0, InputTransitionEffect.consumption);
in0.setSign(Sign.dual);
```

```
//// ListOfOutputs
Output out1 = tr_g1.createOutput("o1", g1, OutputTransitionEffect.assignmentLevel);

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

FunctionTerm ft1 = new FunctionTerm();
ft1.setResultLevel(1);

ASTNode mathNode = null;
try {
    mathNode = ASTNode.parseFormula("G0 > 2");
    ft1.setMath(mathNode);
    ft1.setTemporisationMath(new TemporisationMath());
    ft1.getTemporisationMath().setMath(ASTNode.parseFormula("G0 == 1"));
} catch (ParseException e) {
    e.printStackTrace();
}

// G0 and G1
ASTNode andNode = new ASTNode(ASTNode.Type.LOGICAL_AND);
andNode.addChild(new ASTNode("G0"));
andNode.addChild(new ASTNode("G1"));

tr_g1.addFunctionTerm(defTerm);
tr_g1.addFunctionTerm(ft1);
```

```
//// ListOfOutputs
Output out1 = tr_g1.createOutput("o1", g1, OutputTransitionEffect.assignmentLevel);

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

FunctionTerm ft1 = new FunctionTerm();
ft1.setResultLevel(1);

ASTNode mathNode = null;
try {
    mathNode = ASTNode.parseFormula("G0 > 2");
    ft1.setMath(mathNode);
    ft1.setTemporisationMath(new TemporisationMath());
    ft1.getTemporisationMath().setMath(ASTNode.parseFormu
} catch (ParseException e) {
    e.printStackTrace();
}

// G0 and G1
ASTNode andNode = new ASTNode(ASTNode.Type.LOGICAL
andNode.addChild(new ASTNode("G0"));
andNode.addChild(new ASTNode("G1"));

tr_g1.addFunctionTerm(defTerm);
tr_g1.addFunctionTerm(ft1);
```

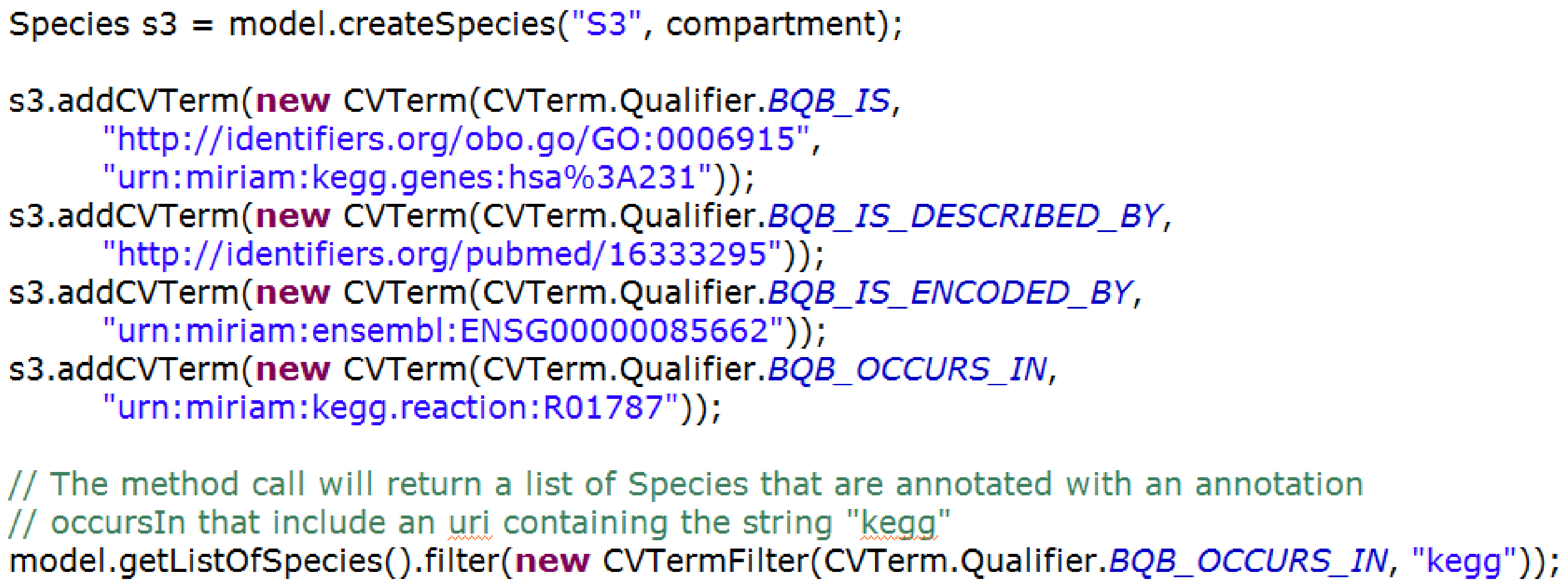


```
JSBML viz
SBML Level 3 Version 1
  m_default_name
    listOfQualitativeSpecies
      G0
      G1
      G2
      G3
    listOfTransitions
      tr_G1
        listOfInputs
          in0
          in2
          in3
        listOfOutputs
          o1
        listOfFunctionTerms
          defaultTerm
          G0 > 2
      tr2
        listOfInputs
        listOfOutputs
        listOfFunctionTerms
    listOfCompartments
```

JSBML

Some more details

Using annotation



```
Species s3 = model.createSpecies("S3", compartment);

s3.addCVTerm(new CVTerm(CVTerm.Qualifier.BQB_IS,
        "http://identifiers.org/obo.go/GO:0006915",
        "urn:miriam:kegg.genes:hsa%3A231"));
s3.addCVTerm(new CVTerm(CVTerm.Qualifier.BQB_IS_DESCRIBED_BY,
        "http://identifiers.org/pubmed/16333295"));
s3.addCVTerm(new CVTerm(CVTerm.Qualifier.BQB_IS_ENCODED_BY,
        "urn:miriam:ensembl:ENSG00000085662"));
s3.addCVTerm(new CVTerm(CVTerm.Qualifier.BQB_OCCURS_IN,
        "urn:miriam:kegg.reaction:R01787"));

// The method call will return a list of Species that are annotated with an annotation
// occursIn that include an uri containing 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 “https://jsbml.svn.sourceforge.net/svnroot/jsbml/trunk jsbml“ JSBML
```

- Do your modifications, then create a patch file:

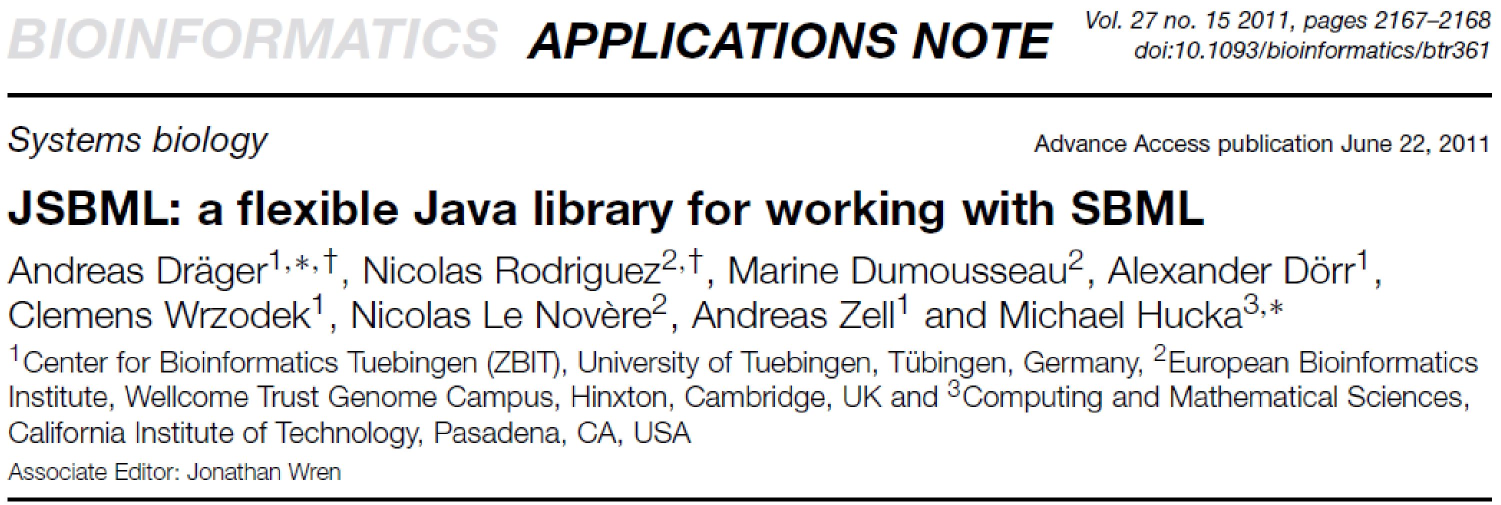
```
svn diff > jsbml-patch.txt
```

- Attach it to a tracker item or send it through the development list.

Bug tracker: http://sourceforge.net/tracker/?group_id=279608&atid=1186776
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

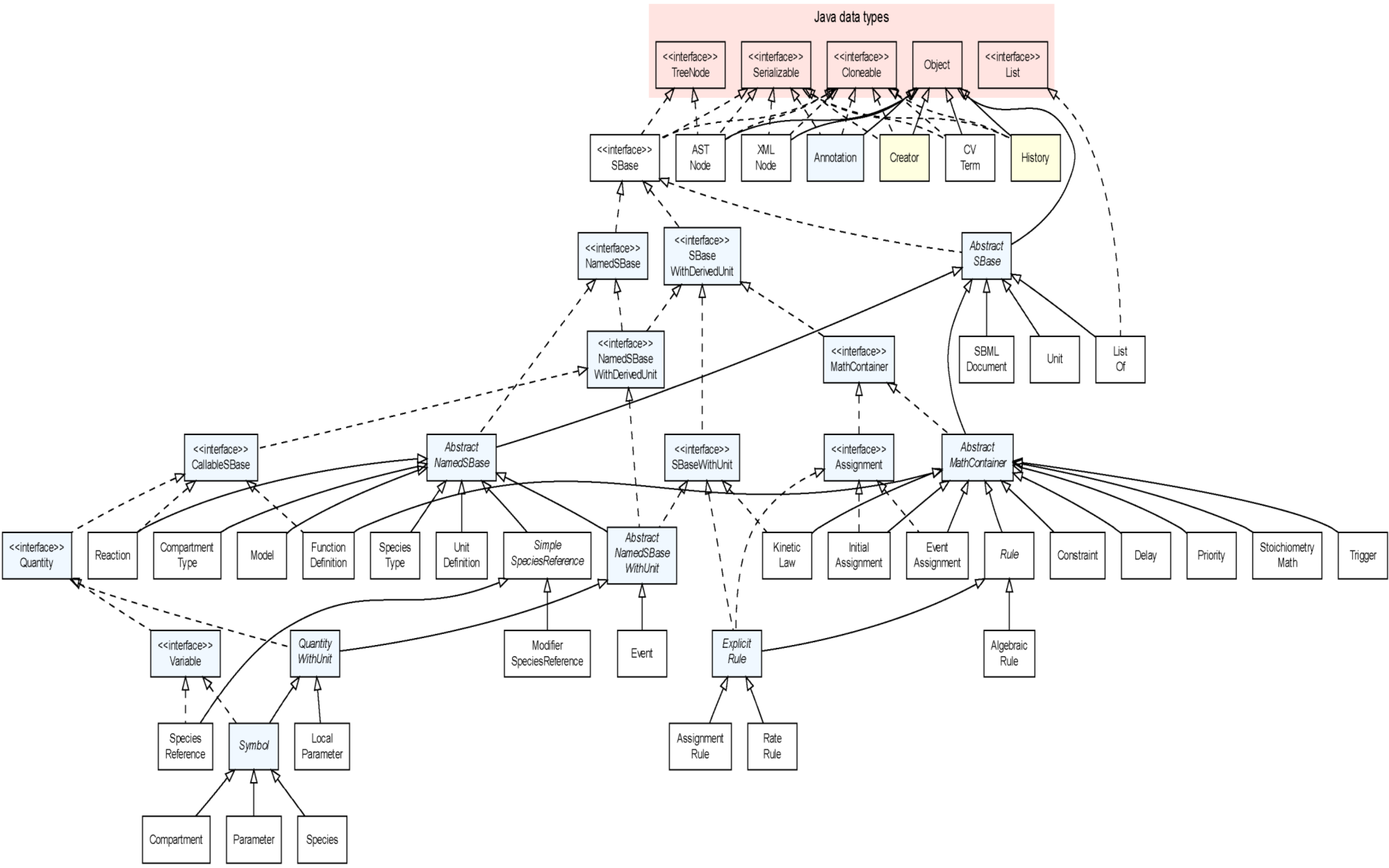
XML parsing ?

```
/jsbml-trunk/resources/org/sbml/jsbml/resources/cfg/PackageParserNamespaces.xml
/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.

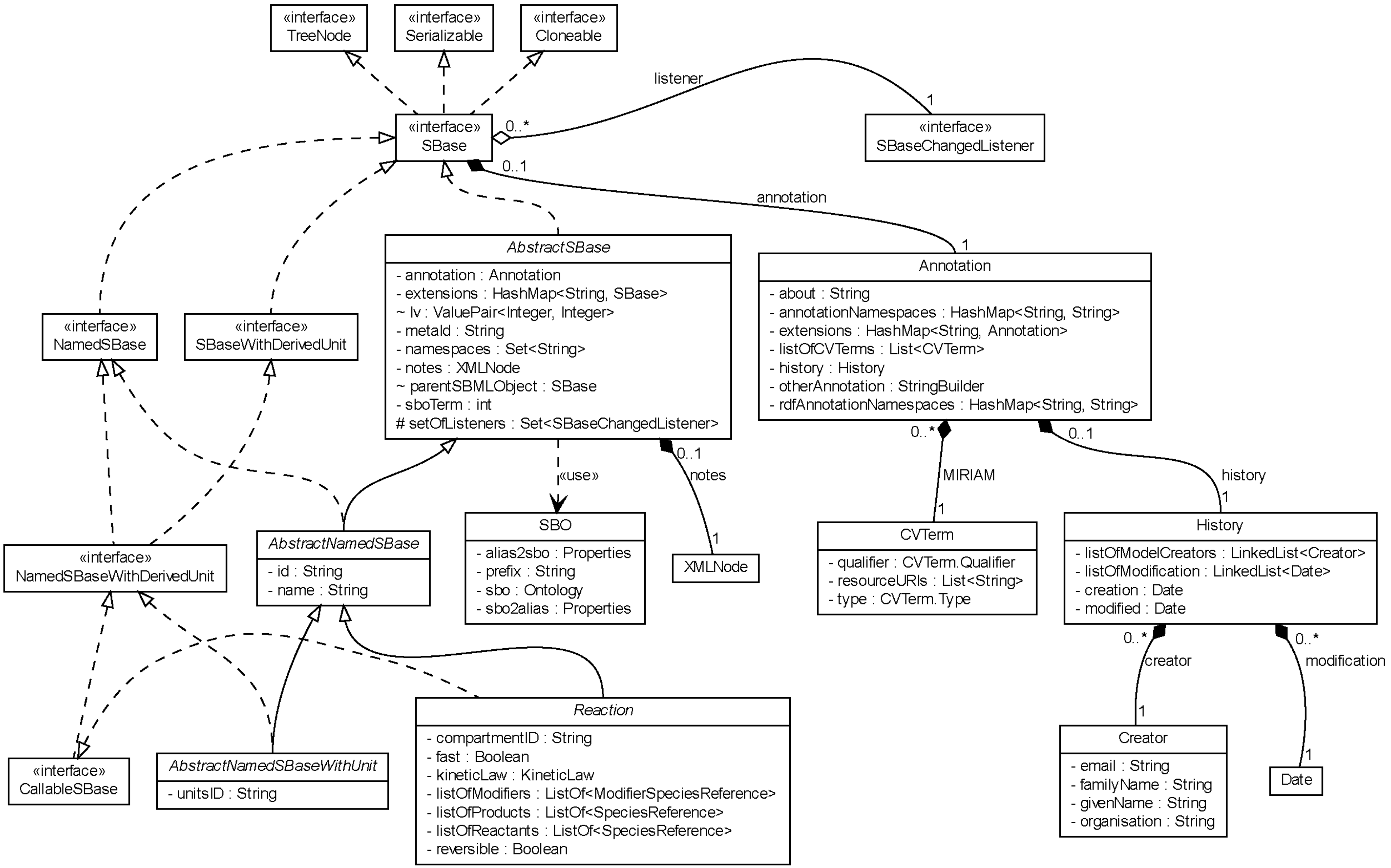
- The parsing is done in:
  - `org.sbml.jsbml.xml.stax`: main entry point of the parsing, using Stax.
  - `org.sbml.jsbml.xml.parsers`: parser independent of the underlying XML parsing library used.

Data types

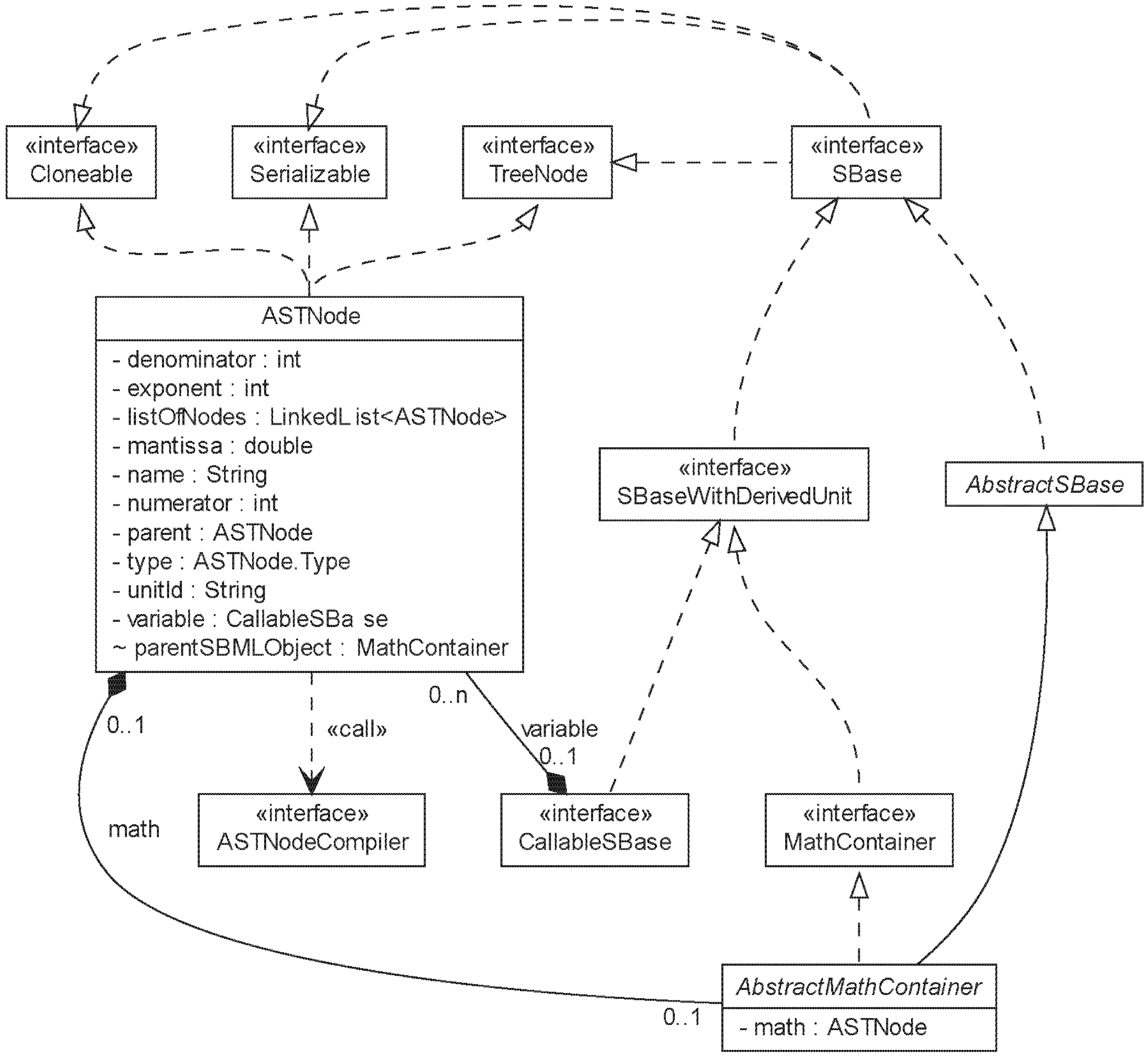


Java data types
<<interface>> TreeNode
<<interface>> Serializable
<<interface>> Cloneable
Object
<<interface>> List
<<interface>> SBase
AST Node
XML Node
Annotation
Creator
CV Term
History
<<interface>> NamedSBase
<<interface>> SBase WithDerivedUnit
Abstract SBase
<<interface>> NamedSBase WithDerivedUnit
<<interface>> MathContainer
SBML Document
Unit
List Of
<<interface>> CallableSBase
Abstract NamedSBase
<<interface>> SBaseWithUnit
<<interface>> Assignment
Abstract MathContainer
<<interface>> Quantity
Reaction
Compartment Type
Model
Function Definition
Species Type
Unit Definition
Simple SpeciesReference
Abstract NamedSBase WithUnit
Kinetic Law
Initial Assignment
Event Assignment
Rule
Constraint
Delay
Priority
Stoichiometry Math
Trigger
<<interface>> Variable
Quantity WithUnit
Modifier SpeciesReference
Event
Explicit Rule
Algebraic Rule
Species Reference
Symbol
Local Parameter
Assignment Rule
Rate Rule
Compartment
Parameter
Species

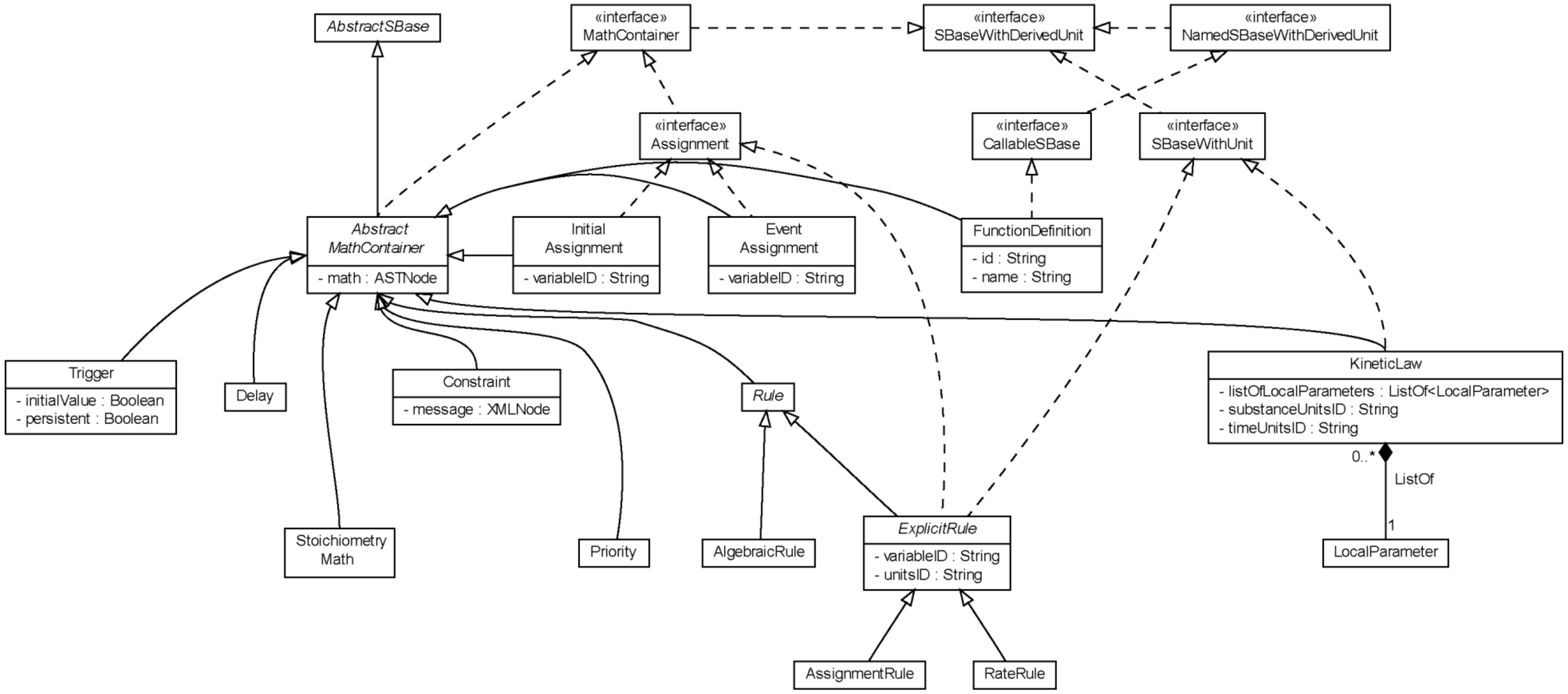
Closer look at the interface `SBase`


«interface» TreeNode
«interface» Serializable
«interface» Cloneable
«interface» SBase
listener
0..*
1
«interface» SBaseChangedListener
0..1
annotation
1
AbstractSBase
- annotation : Annotation
- extensions : HashMap<String, SBase>
~ lv : ValuePair<Integer, Integer>
- metaId : String
- namespaces : Set<String>
- notes : XMLNode
~ parentSBMLObject : SBase
- sboTerm : int
# setOfListeners : Set<SBaseChangedListener>
Annotation
- about : String
- annotationNamespaces : HashMap<String, String>
- extensions : HashMap<String, Annotation>
- listOfCVTerms : List<CVTerm>
- history : History
- otherAnnotation : StringBuilder
- rdfAnnotationNamespaces : HashMap<String, String>
«interface» NamedSBase
«interface» SBaseWithDerivedUnit
«use»
0..1
notes
1
0..*
MIRIAM
1
0..1
history
1
«interface» NamedSBaseWithDerivedUnit
AbstractNamedSBase
- id : String
- name : String
SBO
- alias2sbo : Properties
- prefix : String
- sbo : Ontology
- sbo2alias : Properties
XMLNode
CVTerm
- qualifier : CVTerm.Qualifier
- resourceURIs : List<String>
- type : CVTerm.Type
History
- listOfModelCreators : LinkedList<Creator>
- listOfModification : LinkedList<Date>
- creation : Date
- modified : Date
0..*
creator
0..*
modification
1
1
Reaction
- compartmentID : String
- fast : Boolean
- kineticLaw : KineticLaw
- listOfModifiers : ListOf<ModifierSpeciesReference>
- listOfProducts : ListOf<SpeciesReference>
- listOfReactants : ListOf<SpeciesReference>
- reversible : Boolean
«interface» CallableSBase
AbstractNamedSBaseWithUnit
- unitsID : String
Creator
- email : String
- familyName : String
- givenName : String
- organisation : String
Date

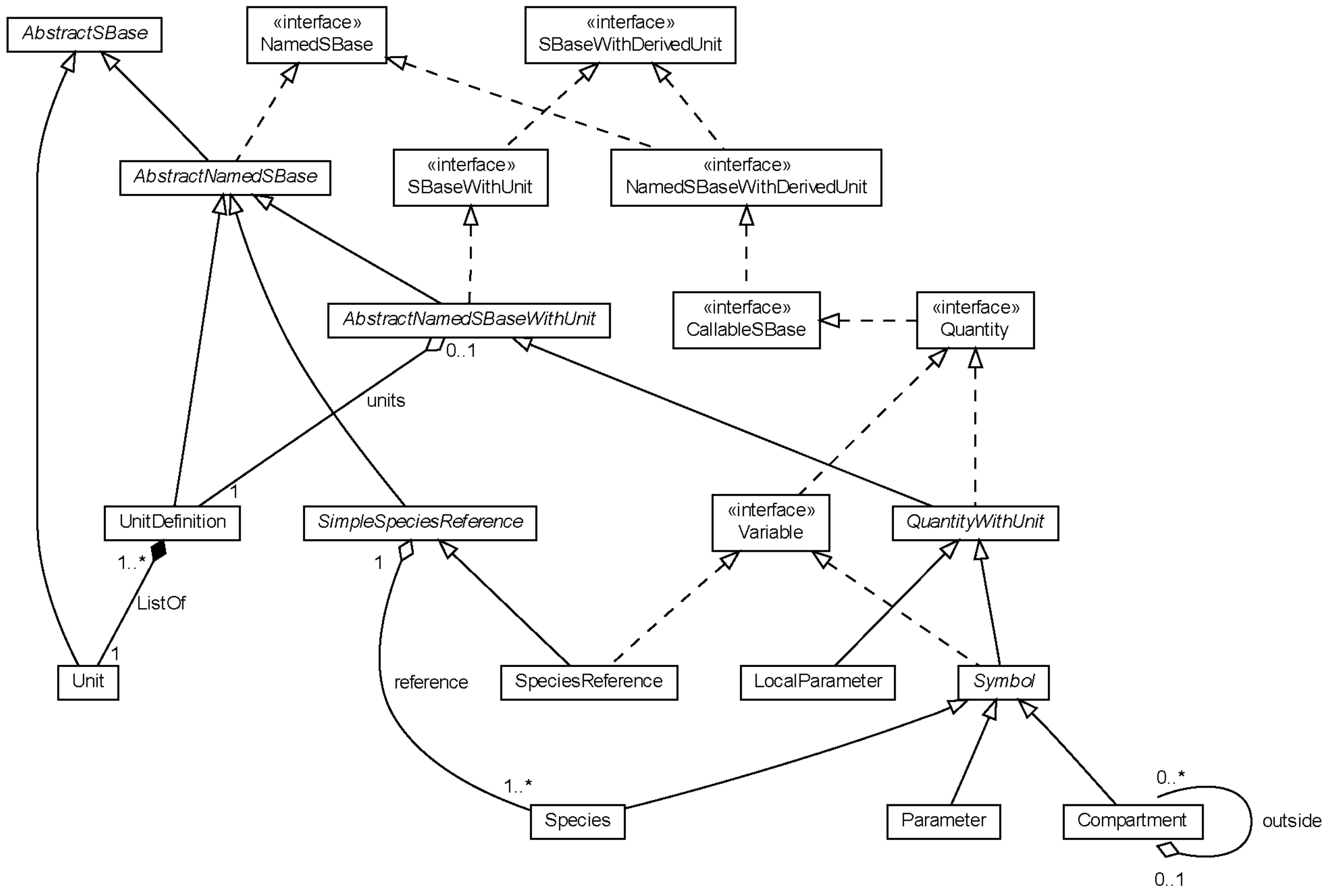
Representation of mathematical equations as ASTNode



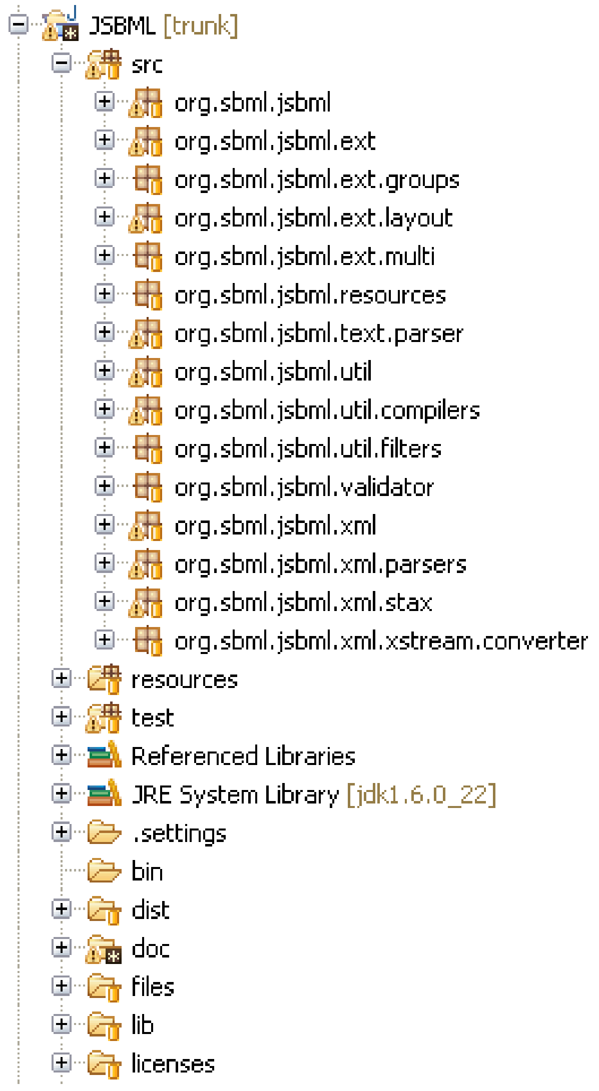
Closer look at the interface `MathContainer`



The relationship between instances of the interface `Variable`



Package structure



```
JSBML [trunk]
  src
    org.sbml.jsbml
    org.sbml.jsbml.ext
    org.sbml.jsbml.ext.groups
    org.sbml.jsbml.ext.layout
    org.sbml.jsbml.ext.multi
    org.sbml.jsbml.resources
    org.sbml.jsbml.text.parser
    org.sbml.jsbml.util
    org.sbml.jsbml.util.compilers
    org.sbml.jsbml.util.filters
    org.sbml.jsbml.validator
    org.sbml.jsbml.xml
    org.sbml.jsbml.xml.parsers
    org.sbml.jsbml.xml.stax
    org.sbml.jsbml.xml.xstream.converter
  resources
  test
  Referenced Libraries
  JRE System Library [jdk1.6.0_22]
  .settings
  bin
  dist
  doc
  files
  lib
  licenses
```

[illegible]

JSBML

Modules

Download of modules

- LibSBML input/output:

```
svn co "https://jsbml.svn.sourceforge.net/svnroot/jsbml/modules/libSBMLio/"
      libSBMLio
```

- CellDesigner bridge:

```
svn co "https://jsbml.svn.sourceforge.net/svnroot/jsbml/modules/cellDesigner"
    cellDesigner
```

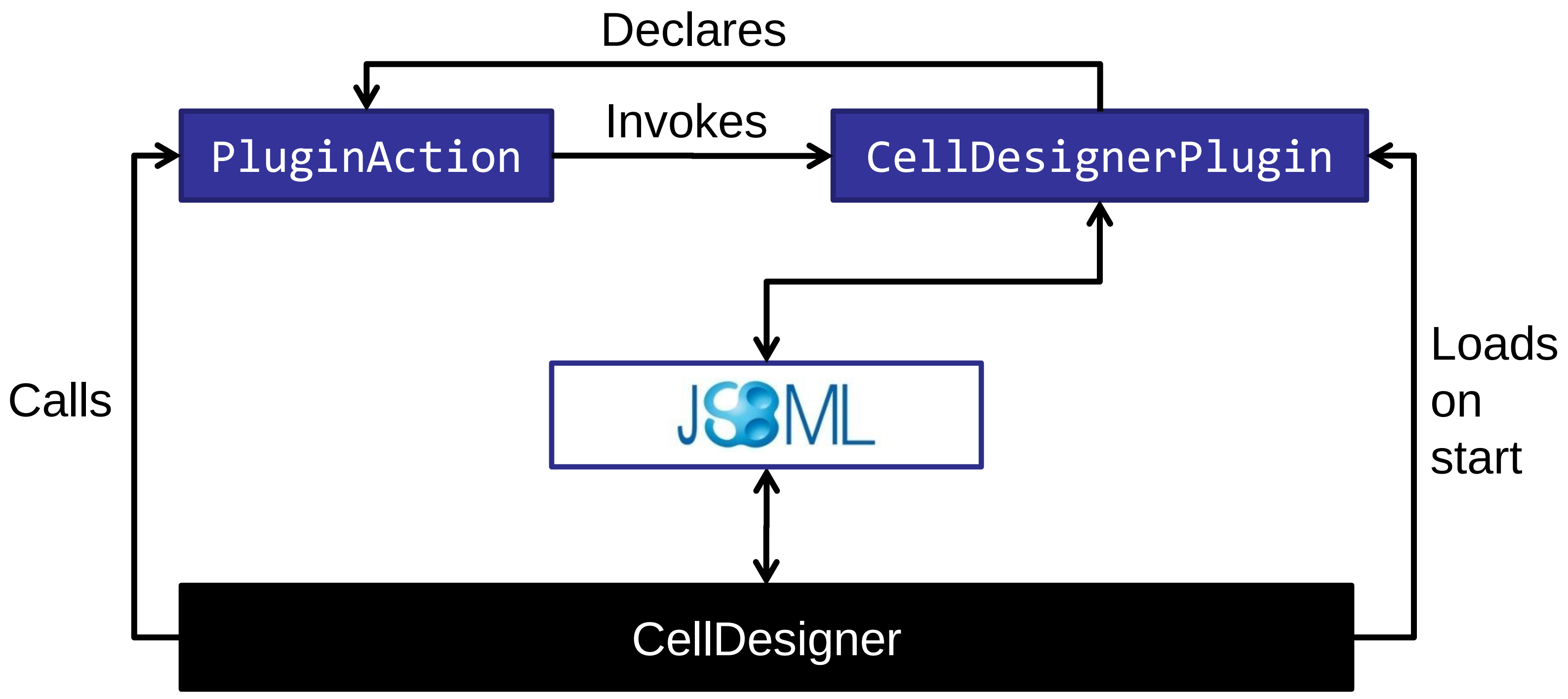
- LibSBML compatibility module for switching between libSBML and JSBML still under development

LibSBML module

```java
/** @param args the path to a valid SBML file. */
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:
    LibSBMLReader reader = new LibSBMLReader();
    SBMLDocument doc = reader.convertSBMLDocument(args[0]);

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

  } catch (Throwable e) {
    e.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`

```java
package org.sbml.jsbml.cdplugin;

import java.awt.event.ActionEvent;
import javax.swing.JMenuItem;
import jp.sbi.celldesigner.plugin.PluginAction;

/** A simple implementation of an action for a CellDesigner plug-in */
public class SimpleCellDesignerPluginAction extends PluginAction {

  private SimpleCellDesignerPlugin plugin;

  /** Constructor memorizes the plug-in data structure. */
  public SimpleCellDesignerPluginAction(SimpleCellDesignerPlugin plugin) {
    this.plugin = plugin;
  }

  /** Executes an action if the given commant occurs. */
  public void myActionPerformed(ActionEvent ae) {
    if (ae.getSource() instanceof JMenuItem) {
      String itemText = ((JMenuItem) ae.getSource()).getText();
      if (itemText.equals(SimpleCellDesignerPlugin.ACTION)) {
        plugin.startPlugin();
      }
    } else {
      System.err.printf("Unsupported␣source␣of␣action␣%s\n", ae
          .getSource().getClass().getName());
    }
  }

}
```

CellDesigner module: Example for a `CellDesignerPlugin`

```java
/** A very simple implementation of a plugin for CellDesigner. */
public class SimpleCellDesignerPlugin extends CellDesignerPlugin {

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

  /** Creates a new CellDesigner plugin with an entry in the menu bar. */
  public SimpleCellDesignerPlugin() {
    super();
    try {
      System.out.printf("\n\nLoading␣%s\n\n", APPLICATION_NAME);
      SimpleCellDesignerPluginAction action = new
          SimpleCellDesignerPluginAction(this);
      PluginMenu menu = new PluginMenu(APPLICATION_NAME);
      PluginMenuItem menuItem = new PluginMenuItem(ACTION, action);
      menu.add(menuItem);
      addCellDesignerPluginMenu(menu);
    } catch (Exception exc) {
      exc.printStackTrace();
    }
  }

  /** This method is to be called by our CellDesignerPluginAction. */
  public void startPlugin() {
    PluginSBMLReader reader = new PluginSBMLReader(getSelectedModel(), SBO
        .getDefaultPossibleEnzymes());
    Model model = reader.getModel();
    SBMLDocument doc = new SBMLDocument(model.getLevel(), model
        .getVersion());
    doc.setModel(model);
    new JSBMLvisualizer(doc);
  }
```