JDesigner SBML Annotation

Herbert M. Sauro, Keck Graduate Institute

January 8, 2003

1 Introduction

Version 1.5+ of JDesigner adds support to save and load graphical models in the form of SBML. SBML Level 1 has no support for layout information, instead it supports arbitrary extensions through the annotations tag (described in the SBML Level 1 documentation). Each extension requires a specific name space so that names used in one extension do not interfere with other 3rd-party extensions. To support the JDesigner layout format I have designated the following name space

```
xmlns:jd="http://www.sys-bio.org/sbml"
```

In practice, the header for a SBML file which supports the JDesigner layout extension will be of the following form:

```
<?xml version="1.0" encoding="UTF-8"?>
  <sbml xmlns "http://www.sbml.org/sbml/level1" level="1"
  version="1" xmlns:jd="http://www.sys-bio.org/sbml">
```

The extensions themselves are remarkably simple and few in number.

2 Coordinates

We do not specify the size of the overall canvas, this is because JDesigner employs a virtual canvas of indefinite size, hence a specified size for the canvas makes no sense. All coordinate units are in pixels. The pixel dimensions are generated by the JDesigner application and correspond roughly to 92dpi.

The exact dimensions depend on the rendering device. Note that the size of the layout diagram is not fixed by these coordinates because it is a simple matter to scale all pixel dimensions and font sizes as appropriate.

3 Compartment Annotation

Compartments other than the default universal compartment (unit volume compartment) are rendered as rectangular areas on the drawing area. Layout information on compartments is stored as an annotation in the `compartment` element, for example:

```
<listOfCompartments>
  <compartment name = "uVol" volume = "1.0">
    <annotations>
      <jd:display>
        <boundingBox x="230" y="120" w="60" h="89"/>
      <jd:display/>
    </annotations>
  </compartment>
</listOfCompartments.
```

A compartment annotation has only one property which is its bounding box. The bounding box is defined in terms of the x, y coordinates of the top left corner and the width and height of the compartment. A bounding box of 0,0,0,0 is reserved for the universal volume element in JDesigner. This volume element represents the entire canvas area and is assumed to be of unit volume. There is no need to specify a bounding box for the Universal Volume. By default, nodes which have no explicitly specified compartment are assumed to reside in the universal compartment.

4 Species Annotation

Species nodes obviously require layout information, in particular to indicate position and visual attribute data. Layout annotation for species elements is held in the species element. Each specie will have it's own annotation. For example:

```
<specie name = "ATP"
```

```
        boundaryCondition = "false"
        initialAmount = "0.1" compartment = "uVol">

   <annotations>
     <jd:display x="234" y="123"
                edgeThickness = "1"
                edgeColor = "0"
                selectedEdgeColor = "255"
                fillColor = "0"
                borderType = "ntRound"
                iconIndex="4">

          <font fontName = "Arial"
                fontSize = "8"
                fontStyle = ""
                fontColor = "0"/>
     </jd:display>
  </annotations>
</specie>
```

The `jd:display` tag has eight attributes, these are described in the table below:

Attribute	Meaning
x	Top left hand x coordinate of node
y	Top left hand y coordinate of node
edgeThickness	If the node has a border, the thickness of the border is indicated by this attribute.
edgeColor	Indicates the color of the border
selectedEdgeColor	This is the color of the border when the the user selects the node.
fillColor	This is the node fill color.
borderType	Indicates the type of border to draw around the node, these currently include: "ntRound", "ntSquare", "ntCircle" or "ntNone"
iconIndex	Set to a value >= 0 if the node should be replaced by an icon, this attribute indicates by index, the icon that the renderer should use.

Some of the attributes indicate a color. Colors are encoded using a decimal encoded integer. This value corresponds to a four byte long word, the low three bytes represent RGB color intensities for blue, green, and red, respectively. The value `$00FF0000` represents full-intensity, pure blue, `$0000FF00`

is pure green, and `$000000FF` is pure red. `$00000000` is black and `$00FFFFFF` is white. Thus a color value of `255` represents red.

Within the `jd:display` element one can also specify a font. The font element has four attributes, these are described in the table below:

Attribute	Meaning
fontName	The name of the font
fontSize	The font size of pt
fontStyle	This indicates the font style to use, this can include any combination of "U", "B" or "I", these codes represent underline, bold and italics respectively.
fontColor	This indicates the color of the font

5 Reaction Annotation

Reaction arcs are described by lines joining node to node. The simplest type of line is a straight line, JDesigner can also render bezier curves between nodes. In cases where there are multiple nodes connecting multiple nodes, such as in the case of bibi reactions, arcs are constructed using more than one line. Thus a bibi reaction is drawn using four lines, one line emanating from each node. The four lines join by default at the centroid of the nodes, called the arc center (See Figure 1).

The minimum amount of information required to render arcs joining node to node is the arc center, that is a single x/y coordinate. Even this information may be omitted since the arc center can be assumed to be the centroid of the joining arcs, however this means that the arc center is then a fixed point. For maximum flexibility both the arc center and the bezier control points are stored in the SBML extension. Note that UniUni reactions only have two bezier control points and no arc center.

Reaction annotation is written immediately after the end of the listOfProducts element. An example of a reaction annotation is given below:

```
<annotations>
   <jd:arcSeg lineThickness = "1"
              lineColor = "6587523"
              fillColor = "6587523"
              selectedLineColor = "255">
```

```
        <pt x="0" y="0" />
        <pt x="159" y="97" />
        <pt x="179" y="116" />
    </jd:arcSeg>
</annotations>
```

The arc layout tag is arcSeg, this has four attributes and one or more point elements.

The arcSeg attributes include the line thickness, the line color, the fill color and the selected line color.

Attribute	Meaning
lineThickness	The thickness of line to use when drawing the arcs
lineColor	The line color.
fillColor	If the line has a thickness greater than one then this indicates the fill color to use.
selectedLineColor	This is the color to use when the arc is selected

The coordinate information is held in the point elements. The first point element is always assumed to be the arc center coordinate. Because this example illustrates a UniUni reaction, the arc center has no value and is nominally set to zero. The remaining point elements indicate control points on bezier curves. These point elements are optional since a layout engine may not be able, or has no desire, to render bezier curves and instead only needs to draw straight line segments between the nodes. In addition some thought needs to be given in future versions to other curve types which layout engines may wish to implement, for example cubics, parametric curves, etc.

The control points for the arc emanating from the reactant are given in the order closest to the reactant, thus the first point refers to the control point emanating from the reactant node, and the second control points refers to the control point emanating from the product node. For reactions with multiple arcs (eg UniBi etc), the order on the product side is arc center nearest control point first followed by the product control points, see figure 1 below for details of this ordering.

Other reaction types, such as BiUni, BiBI etc will have additional bezier control points in addition to a non-zero arc center.

This is an example of a UniBi reaction

```
<annotations>
```

```
   <jd:arcSeg lineThickness = "1" lineColor = "6587523"
              fillColor = "6587523" selectedLineColor = "255">
     <pt x="329" y="139" />

     <pt x="329" y="140" />
     <pt x="303" y="138" />

     <pt x="355" y="140" />
     <pt x="361" y="135" />

     <pt x="355" y="140" />
     <pt x="368" y="149" />
  </jd:arcSeg>
</annotations>
```

A UniBI reaction has one arc center (the first point) followed by three pairs of control points. As before the bezier control points may be ignored especially if the renderer simply draws straight lines between the nodes. A UniBi reaction is assumed to be made from three bezier curves which intersect at the arc center. The three pairs of control points refer to three bezier curves. The first pair refers to the bezier emanating from the reactant, and the remaining two pairs, refer to the two beziers directed at the first and second products respectively, emanating from the arc center. Note that p21 and p31 coincide (See Figure 1), this is an implementation issue so that we get nice looking beziers and is not a constraint on the annotation.

The annotations described here are probably the bear minimum required to render a biochemical network. As mentioned, one could get away with simply storing the coordinates of the nodes, all other information can be computed.

6 Reaction Built-in Rate Law Specifiers

The reaction section contains the stoichiometry and the rate laws. The rate laws are essentially free format in order to be able to cater to the widest possible range of simulators. To support applications which have built-in rate laws, I have added an annotation called 'jd:builtin'. Note that even though built-in information is supplied by this annotation, SBML writers must still output the kinetic law element as described in the SBML specification so

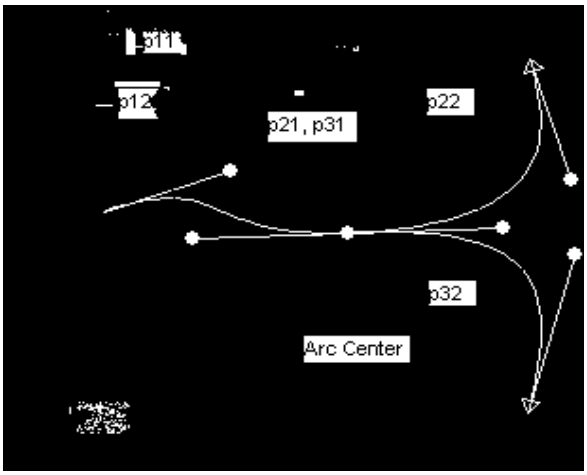

Figure 1: Control Points and Arc Center

that those tools which do not support built-in rate laws can still read the model.

The format for the `jd:builtin` element is quite simple, the `jd:builtin` tag itself has a name attribute, which is the name of the built-in rate law. These names are described in the built-in rate law configuration file that is installed with JDesigner (ratelaws.xml). The example below shows the name `immec` which represents an irreversible Michaelis-Menten rate law with a competitive inhibitor. The remainder of the annotation is a list of symbols, these symbols include two types, parameter symbols and effector symbols. The reason for including this information is so that JDesigner can dynamically build an edit panel for the user to edit the parameter values and select the effector for the reaction.

```
<jd:builtIn name = "immec">
  <listOfSymbols>
    <parameter name = "J0_Vmax"/>
    <parameter name = "J0_Km1"/>
    <parameter name = "J0_Ki"/>
    <effector name = "Node2"/>
  </listOfSymbols>
</jd:builtIn>
```

An entry in the list of symbols simply includes the type of symbol in the

tag name and the name of the symbol in the name attribute.

7 Header Information

SBML JDesigner annotation has a header element which comes immediately after the `<sbml>` element. In the current version the header has two sub-elements, these are the version header and the model header. The version header, indicated by the tag `VersionHeader` has a single attribute, `SBMLVersion` indicating the version of the JDesigner annotation, currently set to "1.0". The second element, `ModelHeader` is a simple place holder for author and model information. Currently it only has three attributes, the author's name, the model title and the model version. These may be augmented with additional attributes in later versions.

```
<annotations>
   <jd:header>
      <VersionHeader SBMLVersion = "1.0"/>
      <ModelHeader Author = "Mr Smith" ModelVersion = "0.1" ModelTitle = "My big model"/>
   </jd:header>
```

Immediately after the header but within the same annotation element there is also a display graphics header. Currently this only has a single attribute with the tag `BackGroundColor` and is used by JDesigner to set the background colour of the canvas.

```
<jd:display>
   <SBMLGraphicsHeader BackGroundColor = "15728639"/>
</jd:display>
```

8 Notes

The notes element in SBML Level 1 is restricted to XHTML and is also an integral part of the XML. This means that parsers must explicitly parse the XHTML in order to extract the notes. To make life easier for code writers and to broaden the utility of notes, JDesigner supports an additional note section as an annotation. The annotation node is tagged with the name `jd:Notes` and has a single attribute to indicate the format of the notes. In the current version, only `ASCII` and `XHTML` is supported. Within the notes element there can be one or more note subelements. These contain a single

attribute called `value` which contains the notes themselves. The reason for permitting multiple note subelements is that memo type GUI controls tend to store text in the form of discrete lines rather than one block of text, the use of multiple notes allows a GUI application to easily store the notes and read them back in.

```
<annotations>
   <jd:notes type = "ASCII">
        <note value = "Some notes"/>
   </jd:notes>
</annotations>
```

```
<annotations>
   <jd:notes type = "XHTML">
      <note value = " "/>
   </jd:notes>
</annotations>
```

JDesigner notes can be currently associated with the model, individual nodes and individual reactions.

9 Shadow Node Annotation

In the graphical layout of complex reaction networks it is sometimes the case that nodes receive or emit multiple reaction arcs, this results in a tangle of arcs which makes the network look overly complex. One strategy to use, to simplify such networks, is to create 'shadow' or copies of nodes. Thus rather than try to extend an arc a great distance from one node to another, a copy (or shadow) of the destination node is created and positioned close to the source node. ATP and ADP are prime examples of nodes which tend to take part in many reactions and creating shadow nodes is a great convenience. The figures 2 and 3 below illustrate the effect of employing shadow nodes on a model of glycolysis.

To support shadow nodes in SBML two sets of annotation elements has been devised. This annotation does not effect the model in any way and is only of use to visual layout tools which may choose to ignore the annotation if they so wish. The first annotation is specified in the specie element. This annotation is used to list any shadow (or copies) nodes associated with the

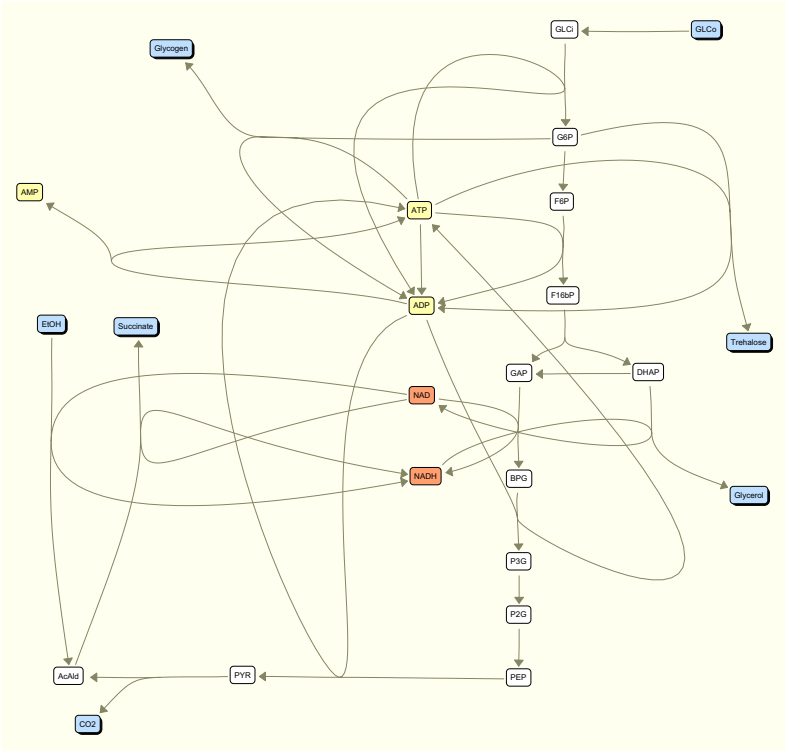

Figure 2: Glycolysis without Shadow Nodes

node. The annotation uses the tag `jd:listOfShadows` and contains the list of shadow nodes. Each shadow node listed has three attributes, a name, and it's x/y coordinate. The name is a unique reference for this node and is constructed by adding two underscore characters to the start of the root name of the node followed by an integer indicating it's number. Thus in the example below, there is a single shadow with the name `__Node1_0`. The root name for this shadow is `Node1`. Since it is the first shadow it has the shadow number `_0`. If this specie had other shadow nodes they would be named as, `__Node1_1, __Node1_2, etc`.

```
<specie name = "Node1" boundaryCondition = "false"
        initialAmount = "0" compartment = "uVol">

   <annotations>
      <jd:listOfShadows>
```

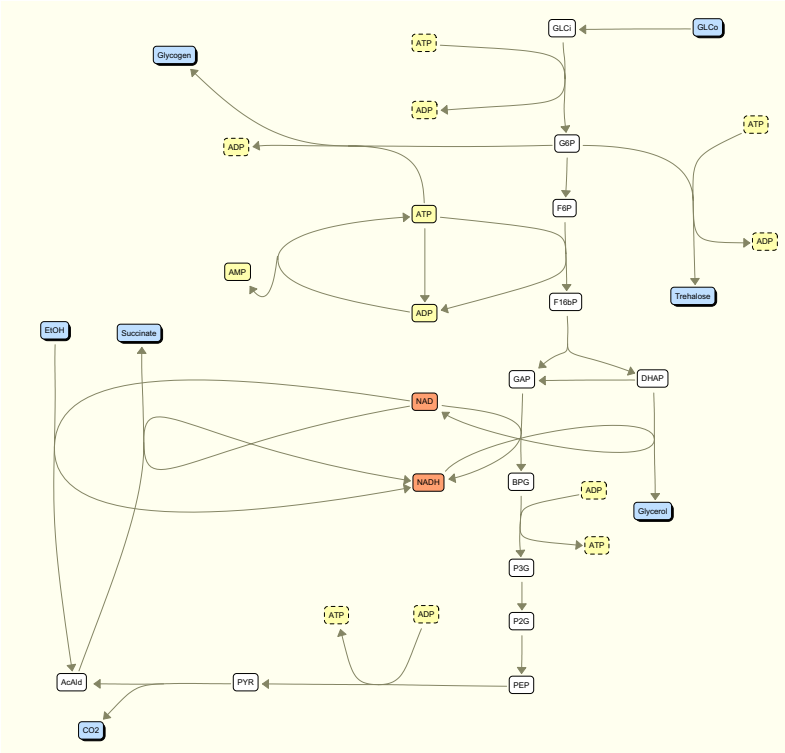

Figure 3: Glycolysis after Employing Shadow Nodes

```
            <shadow name = "__Node1_0" x = "246" y = "312"/>
        </jd:listOfShadows>
    </annotations>

</specie>
```

Note that SBML generators should include all specie annotation in a single annotation element.

The second annotation for supporting shadow nodes is located in a specieReference element. When a reaction refers to reactants and products, we use the specieReference to indicate the actual reactants and products that are associated with the reaction. However, when a reaction is connected to a shadow node we must indicate the shadow node. The root name of a specieReference is indicated by a specie attribute. If the species that is ac-

tually connected to the reaction is a shadow node, then we indicate this with an annotation with tag name `jd:shadowRef`. This element has a single attribute, the `shadowRef` name which indicates the particular shadow to use. This in the example below, `Node1` is visually represented by a shadow node, called `__Node1_0`. A layout tool can determine the coordinates of this shadow node by searching for the shadow name `__Node1_0` in the specie shadow list for node `Node1`.

```
<specieReference specie = "Node1" stoichiometry = "1">
    <annotations>
       <jd:shadowRef shadowRef = "__Node1_0"/>
    </annotations>
</specieReference>
```

10 Non-Model Display Elements

Apart from laying out networks, JDesigner also allows a user to annotate their model with other visual elements. In the current version of JDesigner (1.7), only two such elements are supported, these are text and ellipse objects.

These non-model display elements are recorded in the SBML just after the graphics header element. The following SBML is an example:

```
<jd:otherDisplayObjects>
   <textObject name = "Some Text">
      <font fontName = "Arial" fontSize = "8" fontStyle = "" fontColor = "0"/>
      <boundingBox x = "103" y = "116" w = "51" h = "14"/>
   </textObject>
   <circleObject>
      <shapeProperties fillColor = "15728639" outlineColor = "0"/>
      <boundingBox x = "277" y = "158" w = "75" h = "70"/>
   </circleObject>
   <circleObject>
      <shapeProperties fillColor = "15728639" outlineColor = "0"/>
      <boundingBox x = "256" y = "247" w = "-1" h = "0"/>
   </circleObject>
   <circleObject>
      <shapeProperties fillColor = "15728639" outlineColor = "0"/>
      <boundingBox x = "117" y = "229" w = "80" h = "82"/>
   </circleObject>
   <circleObject>
      <shapeProperties fillColor = "15728639" outlineColor = "0"/>
```

```
            <boundingBox x = "217" y = "314" w = "-2" h = "-1"/>
         </circleObject>
      </jd:otherDisplayObjects>
```

The section starts with the element tag `otherDisplayObjects`. Within this element are the display elements themselves. The text object has two subelements, one to indicate the font characteristics and the other to indicate a bounding box.

The circle element also has two subelements, one to indicate the characteristics of the circle and another to indicate the bounding box.

11 Exmmaple SBML with JDesigner Annotation

The following SBML illustrates a complete SBML file with JDesigner annotation.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Created by XMLPrettyPrinter on 11/26/2002 -->

<sbml xmlns = "http://www.sbml.org/sbml/level1" level = "1"
   version = "1" xmlns:jd = "http://www.sys-bio.org/sbml">
   <annotations>
      <jd:header>
         <VersionHeader SBMLVersion = "1.0"/>
         <ModelHeader Author = "Mr Smith" ModelVersion = "0.1" ModelTitle = "My big model"/>
      </jd:header>
      <jd:display>
         <SBMLGraphicsHeader BackGroundColor = "15728639"/>
      </jd:display>
      <jd:otherDisplayObjects>
         <textObject name = "Example Model">
            <font fontName = "Arial" fontSize = "8" fontStyle = "" fontColor = "0"/>
            <boundingBox x = "75" y = "58" w = "71" h = "14"/>
         </textObject>
         <circleObject>
            <shapeProperties fillColor = "16777215" outlineColor = "0"/>
            <boundingBox x = "177" y = "187" w = "51" h = "52"/>
         </circleObject>
         <circleObject>
            <shapeProperties fillColor = "16777215" outlineColor = "0"/>
            <boundingBox x = "259" y = "246" w = "0" h = "0"/>
         </circleObject>
      </jd:otherDisplayObjects>
   </annotations>
   <!--                    -->
   <!--  Model Starts Here  -->
   <!--                    -->
   <model name = "untitled">
      <listOfCompartments>
         <compartment name = "uVol" volume = "1">
            <annotations>
               <jd:display>
                  <boundingBox x = "0" y = "0" w = "0" h = "0"/>
               </jd:display>
            </annotations>
         </compartment>
      </listOfCompartments>
      <listOfSpecies>
         <specie name = "Node0" boundaryCondition = "false" initialAmount = "0" compartment = "uVol">
```

```
            <annotations>
               <jd:display x = "123" y = "123" iconIndex = "-1" edgeThickness = "1"
                           edgeColor = "0" selectedEdgeColor = "255" fillColor = "15728639" borderType = "ntRound">
                  <font fontName = "Arial" fontSize = "8" fontStyle = "" fontColor = "0"/>
               </jd:display>
            </annotations>
         </specie>
         <specie name = "Node1" boundaryCondition = "false" initialAmount = "0" compartment = "uVol">
            <annotations>
               <jd:display x = "298" y = "174" iconIndex = "-1" edgeThickness = "1"
                           edgeColor = "0" selectedEdgeColor = "255" fillColor = "15728639" borderType = "ntRound">
                  <font fontName = "Arial" fontSize = "8" fontStyle = "" fontColor = "0"/>
               </jd:display>
            </annotations>
         </specie>
      </listOfSpecies>
      <listOfReactions>
         <reaction name = "J0" reversible = "false">
            <listOfReactants>
               <specieReference specie = "Node0" stoichiometry = "1"/>
            </listOfReactants>
            <listOfProducts>
               <specieReference specie = "Node1" stoichiometry = "1"/>
            </listOfProducts>
            <annotations>
               <jd:arcSeg lineThickness = "1" lineColor = "6587523" fillColor = "6587523" selectedLineColor = "255">
                  <pt x = "224" y = "154"/>
                  <pt x = "226" y = "155"/>
                  <pt x = "226" y = "155"/>
               </jd:arcSeg>
            </annotations>
            <kineticLaw formula = "kJ0*Node0">
            </kineticLaw>
         </reaction>
      </listOfReactions>
   </model>
</sbml>
```


Acknowledgement

I am grateful to Hiroaki Kitano and Akira Funahashi (author of SBEdit) for useful discussions.