A Model Diagram Layout Extension for SBML

Ralph Gauges,* Ursula Rost, Sven Sahle and Katja Wegner

Bioinformatics and Computational Biochemistry, EML Research, Schloss-Wolfsbrunnen Weg 33,
D-69118 Heidelberg, Germany

ABSTRACT

Motivation: Since the knowledge about processes in living cells is increasing, modeling and simulation techniques are used to get new insights into these complex processes. During the last few years, the SBML file format has gained in popularity and support as a means of exchanging model data between the different modeling and simulation tools. In addition to specifying the model as a set of equations, many modern modeling tools allow the user to create and interact with the model in the form of a reaction graph. Unfortunately, the SBML file format does not provide for the storage of this graph data along with the mathematical description of the model.

Results: Therefore, we developed an extension that makes it possible to store such layout information which describes position and size of objects in the graphical representation.

Availability: The complete specification can be found on [12]. Additionally, a complete implementation exists as part of libSBML [9].

Contact: Ralph.Gauges@eml-r.villa-bosch.de

1 INTRODUCTION

Due to limited resources and technical means, scientists from the early days of biochemistry were mostly investigating singular processes, e.g. single reactions, in order to understand their mechanisms. Nowadays, many of these mechanisms are well understood and scientists turn towards investigating the relations and dependencies between those singular processes to gain a broader picture about the interplay of the different components within an organism. For that reason, scientists study complete reaction networks which can consist of up to several hundred or sometimes even thousands of reactions. Modelling and simulation techniques are being used more and more often to analyze such large networks.

A biochemical model in this context consists of a set of chemical species and a set of reactions that consume and produce some of these species. Often also a mathematical description of the reaction kinetics is given. If this is the case, the behaviour of the model, i.e. the change of concentrations of the various species, can be calculated on a computer. This is called a simulation of the model.

There are several programs that simulate a biochemical reaction network if they are provided with a mathematical description of the model, e.g. a set of ordinary differential equations. Unfortunately, until recently each of those programs had its own way of reading in these descriptions which was not compatible with those of other programs. Users that wanted to exchange models between different programs had to manually convert the model from one format to the other. Therefore, SBML (System Biology Markup Language) [3, 4] was developed five years ago to allow different programs

to exchange biochemical models. Now five years later, close to a hundred programs support the SBML file format [17].

During those five years programs have changed as well and where five years ago most programs happily presented a biochemical model as a set of mathematical equations, many programs today display those models as graphs. In such graphs nodes represent reactants and edges represent biochemical reactions between these reactants. This way of presenting reaction networks allows the user to get a better overview of the network and to recognize relations and interactions within the network. Also many biologists and biochemists are more familiar and comfortable with this way of displaying reaction networks which makes it easier for them to use modeling and simulation software. One drawback of this way of displaying reaction networks is the fact that the user has to invest extra time to create the graphs since mechanisms to do this automatically are still under development [6, 15, 19]. So once the user has created such a graph, he or she would probably like to store it together with the mathematical description and optimally, this would be done in a format that can also be read by other programs. SBML would be such an applicable format but it does not offer any possibility to store such graph information. Therefore we decided to develop an extension to the SBML file format that would allow the storage of this graph information within SBML files

This extension was presented and discussed on several SBML workshops and on the SBML mailing list [11]. It has been accepted by the SBML community and will be included in the upcoming specification for SBML Level 3.

2 DESIGN PRINCIPLES

The scope of this extension to SBML is to store the layout of a reaction graph. That means, we will describe a language that specifies the size and position of graphical representations of biochemical entities like species and reactions and its connections. How the layout elements are actually to be drawn is not specified and is up to the program that reads the layout. As an illustration, let us consider figure 1. All three diagrams are valid renderings of the same layout information and would be described by identical SBML files, because no information about colors, line styles, fonts, etc., is present in our diagram layout extension at the moment.

The design of the extension is based on some underlying design decisions. First of all, the layout of the reaction network should be described in a language that is structurally similar to the language that describes the biochemical model itself. That rules out the use of existing XML languages that can describe general 2D drawings (e.g. SVG [16]) or general graphs (e.g. GraphML [1]).

Since we want to allow for several different graph layouts for the same model, the layout is not stored as annotations to the individual
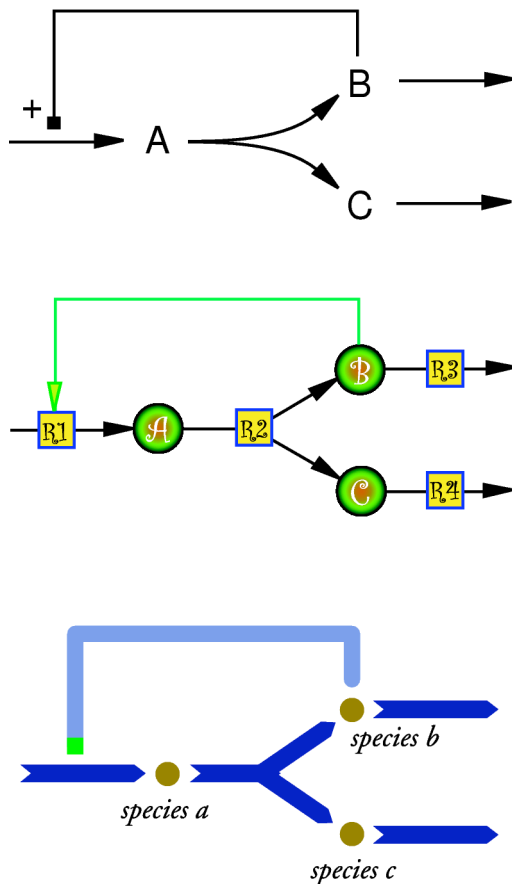
*to whom correspondence should be addressed

Fig. 1. Illustration of different renderings of the same layout information.

model elements but as a separate `Layout` element several of which can be put inside a `ListOfLayouts` structure. The SBML specification already provides some means of extending the standard via `annotation` elements, so we decided to store the layout information as an annotation to the `model` element. The decision to store it there, rather than storing it as an annotation to the `sbml` element, is based on the assumption that later versions of the SBML specification may allow the storage of several models per file. The top element for the layout extension is the `listOfLayouts` which contains one or more layouts. As stated above, this element will be located in the `annotation` tag of the model element for SBML Level 2, but for SBML Level 3 we expect it to be an additional subelement of the `model` element.

Because we primarily describe the layout of elements within a SBML model, the structure of the layout extensions is very similar to the structure of the SBML model. Most elements in the layout have a direct correspondence in the SBML specification. Consider as example, to store the layout information for a compartment we introduced the `compartmentGlyph` or to store the layout information for a species we introduced the `speciesGlyph`.

Also, the connection between the layout and the model has to be described. In many cases this will be a simple one-to-one relation, e.g. a given `species` in the model will be graphically represented by a specific `speciesGlyph` element in the layout. There

are some exceptions however. Sometimes it is advantageous to describe the same model element by several elements in the layout. For example, species that participate in many reactions will end up with many edges going in and out of the species representation. In order to avoid edge crossings those species can have several speciesGlyphs in the layout. There are also cases where a layout element does not correspond to a specific element in the model. This could occur if the layout shows a simplified version of the model where one element in the diagram corresponds to several reactions and intermediate species in the model. Layout elements may also have no counterpart in the SBML `model` at all. This can occur if the layout includes, for example, a legend or some representation of connecting pathways which are not explicitly considered in the model. In our extension the layout elements connect to model elements via references to their ids. The connection is optional.

The layout extension uses a right handed Cartesian coordinate system. The origin of the coordinate system is located in the upper left corner of the screen. The positive x-axis runs from left to right, the positive y-axis runs from top to bottom and the positive z-axis points into the screen. This seems to be the way most graphical tools do it. For printing purposes a point in this coordinate system is presumed to be 1/72 of an inch ($\sim 0.353mm$) as in postscript. Even though we expect most implementations to use 2D space for layout in the near future, the extension is already fit to handle 3D layout data as well.

3 LAYOUT ELEMENTS

Figure 2 shows the relations between the different elements in the layout extension. Inheritance relations are shown as well as which element contains which other elements. All layout elements are derived from the SBML element SBase which is the super element of all elements in SBML according to the SBML Level 2 schema specification [14, 2]. The names of the layout elements mostly consist of the corresponding SBML element name and the postfix `Glyph`.

The layout elements have the following attributes in common. First, the layout elements have an attribute `id`. All glyph elements are derived from `graphicalObject` and must have an id which is unique throughout the model and the layout (with the notable exception of local parameters in reactions). All id attributes are defined to be of type SId as defined in SBML Level 2 [14].

Second, each glyph element that can have a corresponding element in the model has a second attribute that holds the id of this corresponding element in the model. This attribute is optional, so it is possible to create objects that are not connected to elements in the model.

Finally, each glyph element contains a bounding box that defines the size and position of this element without giving additional information about its contents.

Each `boundingBox` element has an subelement called `position` of type Point and a subelement called `dimensions` of type Dimensions. The `position` element has three attributes, called `x`, `y` and `z`, which specify the location of the point in the coordinate system. The `z` attribute is optional and defaults to 0.0 if not specified. The `dimensions` element also has three attributes, called `width`, `height` and `depth`, which specify the size along the positive x, y and z axis, respectively. Again, the `depth` attribute is optional and defaults to 0.0 if not specified. Both elements of type Point and type Dimensions have an attribute called `id` which is
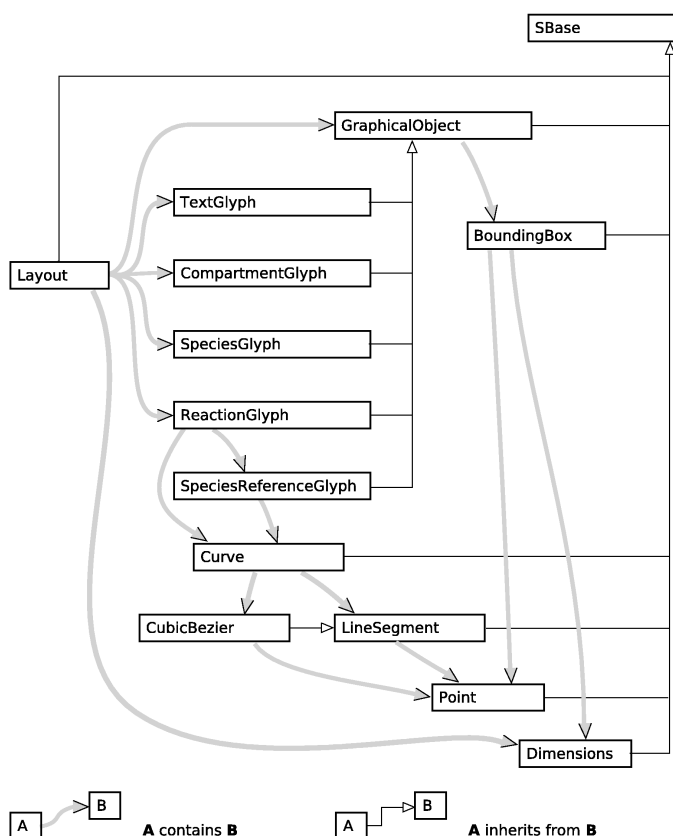
Fig. 2. Containment and inheritance tree for the layout elements.

optional and can be used to reference these elements. All size values for the `dimensions` element must be positive.

In the following sections, we will describe the various elements in detail by giving a short example.

3.1 Layout

All layout information is stored in an element called `listOf-Layouts`. This list can hold one or more `layout` elements which in turn hold layout information for some or all elements of the SBML model, plus additional objects that are not part of the model. The `layout` element has an attribute `id` that uniquely identifies it and an attribute `dimensions` that specifies its size. The actual layout elements are contained in several list elements, namely a `listOfCompartmentGlyphs`, a `listOfSpeciesGlyphs`, a `listOfReactionGlyphs`, a `listOfTextGlyphs`, and a `listOfAdditionalGraphicalObjects`, see the following skeleton:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2"
      level="2" version="1">
 <model id="TestModel with modifiers">
  <annotation>
   <listOfLayouts
   xmlns="http://projects.eml.org/bcb/sbml/level2"
   xmlns:xsi=\
      "http://www.w3.org/2001/XMLSchema-instance">
    <layout id="Layout_1">
```

```
      <dimensions width="400" height="230"/>
      <listOfCompartmentGlyphs>
          ...
      </listOfCompartmentGlyphs>
      <listOfSpeciesGlyph>
          ...
      </listOfSpeciesGlyph>
      <listOfReactionGlyphs>
          ...
      </listOfReactionGlyphs>
      <listOfTextGlyphs>
          ...
      </listOfTextGlyphs>
      <listOfAdditionalGraphicalObjects>
          ...
      </listOfAdditionalGraphicalObjects>
    </layout>
   </listOfLayouts>
  </annotation>
      ...
 </model>
</sbml>
```

3.2 GraphicalObject

The `graphicalObject` element contains a bounding box and an id. All more specific layout elements (`compartmentGlyph`, `speciesGlyph`, `reactionGlyph`, `textGlyph`, and `speciesReferenceGlyph`) are derived from GraphicalObject. Some software tools may want to store information about graphical objects that can not adequately be described by one of the derived elements. This could e.g. be the legend to the layout or some bitmap graphic. Therefore, generic graphical objects can be put into a `listOfAdditionalGraphicalObjects`. Note that there is no default interpretation for these generic graphical objects. However, different software tools can use this mechanism to easily extend the layout description to their special needs (using annotations).

3.3 CompartmentGlyph

The `compartmentGlyph` element is derived from GraphicalObject and has an optional reference to the id of the corresponding compartment in the model.

```
<listOfCompartmentGlyphs>
  <compartmentGlyph id="CG_1" compartment="Yeast">
    <boundingBox>
      <position x="5.0" y="5.0" />
      <dimensions width="390.0" height="220.0" />
    </boundingBox>
  </compartmentGlyph>
      ...
</listOfCompartmentGlyphs>
```

Since the compartment id is optional, the user can also specify compartment glyphs that do not have a corresponding compartment in the model.

3.4 SpeciesGlyph

Species are represented by `speciesGlyph` elements which are grouped in a `listOfSpeciesGlyphs` element. In addition to the attributes from `graphicalObject`, the `speciesGlyph` element has a `species` attribute which can hold the id of the appropriate `species` element in the model. Several species glyphs can refer to the same species in the SBML model which can

be useful to avoid crossing lines. The `species` attribute is optional to allow the program to specify species representations that do not have a direct correspondence in the model. One case where this might be useful is when parts of the network have been collapsed to form a simplified diagram where several model elements are now being represented by a single species glyph.

```
<listOfSpeciesGlyphs>
  <speciesGlyph id="SpeciesGlyph_1" species="NADH">
    <boundingBox>
      <position x="968.816" y="386.91"/>
      <dimensions width="88.8" height="17.4609"/>
    </boundingBox>
  </speciesGlyph>
      ...
</listOfSpeciesGlyphs>
```

3.5 ReactionGlyph

Reactions are represented by `reactionGlyph` elements in the layout extension. Just like the other glyphs, the `reactionGlyph` has an optional attribute that specifies the id of the corresponding reaction in the model. Basically, the graphical representation of a reaction consists of a part that stands for the reaction itself and parts describing the connections between the reaction and the species glyphs. The latter is given by a `listOfSpeciesReferenceGlyphs` which is described below in the next paragraph.

In many cases reaction glyphs and species reference glyphs are represented by curves. For that reason, a reaction glyph or a species reference glyph can contain the specification for a curve object in addition to the bounding box. If both, the bounding box and the curve are specified in a reaction glyph or species reference glyph, the curve takes precedence over the bounding box.

```
<listOfReactionGlyphs>
  <reactionGlyph id="RG_1" reaction="reaction_0">
    <curve>
      <listOfCurveSegments>
        <curveSegment xsi:type="LineSegment">
          <start x="111.364" y="158.972"/>
          <end   x="179.858" y="76.3368"/>
        </curveSegment>
      </listOfCurveSegments>
    </curve>
    <listOfSpeciesReferenceGlyphs>
        ...
    </listOfSpeciesReferenceGlyphs>
  </reactionGlyph>
</listOfReactionGlyphs>
```

The `curve` element contains a `listOfCurveSegments` which contains an arbitrary number of curve segments. For now, we provide the definitions for two types of curve segments (LineSegment and CubicBezier) but this can easily be extended if needed. Since all types of curves can be represented by a series of cubic bezier segments, we think that limiting the curve segment types to only those two types does not pose a problem but rather helps to keep the layout specification simple. CubicBezier is a direct subtype of LineSegment.

The type of the curve segment has to be specified with a `xsi:type` attribute in the `curveSegment` element (`xsi:type =LineSegment` or `xsi:type=CubicBezier`).

The LineSegment type contains two elements of type Point. One is called `start` (the starting point of the line) and the other is called `end` (the endpoint of the line). As mentioned above, the CubicBezier type is derived from LineSegment, so it has the same two Point elements `start` and `end`, which again specify the starting point and the endpoint of the cubic bezier segment. In addition, the two attributes `basePoint1` and `basePoint2` specify the two additional base points that are needed to describe a CubicBezier curve segment. Both base points are optional. If not specified, they are assumed to be in the middle between the two endpoints of the cubic bezier segment thus describing a straight line. If only one base point is given, the other is assumed to be identical to the one specified.

3.6 SpeciesReferenceGlyph

The graphical connection between a species glyph and a reaction glyph, an arrow or some curve in most cases, is represented by the `speciesReferenceGlyph` element. Therefore, a `listOfSpeciesReferenceGlyphs` is contained in a `reactionGlyph`.

The `speciesReferenceGlyph` has a `speciesGlyph` attribute that corresponds to the id of a `speciesGlyph` element that is connected to the `reactionGlyph`.

The `speciesReference` attribute refers to a species reference (or a modifier species reference) in the model. This id defines a relation between a species reference glyph and the appropriate species reference or modifier species reference in the model. Since the specification for SBML Level 2 Version 1 does not include an `id` attribute for elements of type SpeciesReference or ModifierSpeciesReference, we propose to use the annotation of those elements to store an id. This id is of type SId and has to be unique within the model and the layout. SBML Level 2 Version 2 will probably add the id attribute to these element types.

```
<speciesReference species="ATP">
  <annotation>
    <layoutId
    xmlns="http://projects.eml.org/bcb/sbml/level2"
    id="ATP_Reference_1"/>
  </annotation>
</speciesReference>
```

If the `speciesReference` attribute is set, we can deduce the role a certain species plays in the reaction (whether it is a substrate, a product or a modifier) by checking where in the reaction it has been specified. Since this connection from the layout to the model is optional, there are cases where the role of the species can not be derived in that way. For that reason and in the case when the respective information from the model needs to be overridden, we propose an optional `role` attribute. This can be used to provide the role of a metabolite in the reaction as a string. The following values are allowed: *substrate, product, sidesubstrate, sideproduct, modifier, activator, inhibitor* and *undefined.*

The values *substrate* and *product* are used if the species reference is a main product or substrate in the reaction. *sidesubstrate* and *sideproduct* are used for species like ATP, NAD^+, etc. that some renderers might choose to display differently than main substrates and products. *activator* and *inhibitor* are modifiers where their influence on the reaction is known and *modifier* is a more general term if
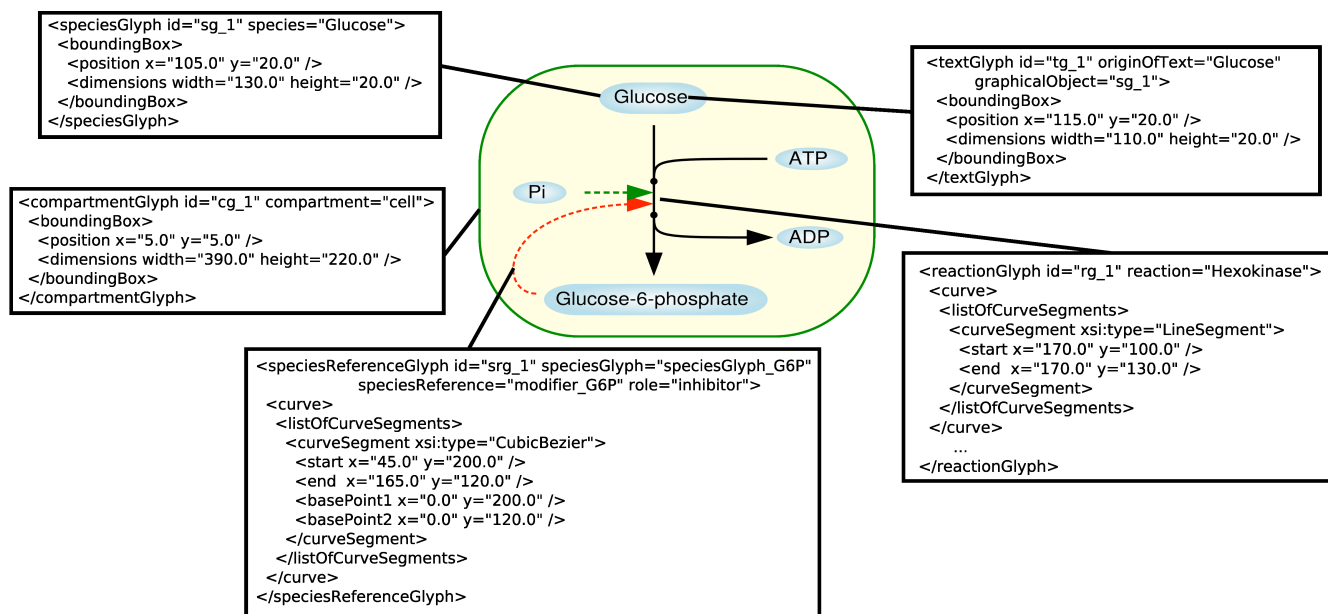
Fig. 3. This figure shows one possible rendering of a simple layout. Layout code is displayed and connected to the corresponding graphical representation via a straight line.

the influence is unknown or changes during the course of the simulation. This list is probably not exhaustive and will be updated as needed.

We are also aware of a proposal [18] that would allow the inclusion of controlled vocabularies based on information from ontologies. If this was possible, the role of a `speciesReference` element could be specified through those means which would make the `role` attribute in `speciesReferenceElements` obsolete.

So far we have defined which graphical objects should be connected to the reaction glyph. This is the minimum information that a render program with biochemical knowledge needs to display the reaction layout. The standard way to render this connection would be a straight line. In most cases, the relation of a species to a reaction will be graphically represented by a curve. In this case the above explained `curve` element can be used. The species reference glyph should either contain a bounding box or a curve specification. If both are given, the bounding box is to be ignored.

```
<listOfSpeciesReferenceGlyphs>
  <speciesReferenceGlyph id="SRG_1"
      speciesReference="SR_1"
      speciesGlyph="SG_8"
      role="substrate">
    <curve>
      <listOfCurveSegments>
        <curveSegment xsi:type="LineSegment">
          <start x="111.364" y="158.972"/>
          <end   x="74.2"    y="138.183"/>
        </curveSegment>
      </listOfCurveSegments>
    </curve>
  </speciesReferenceGlyph>
      ...
</listOfSpeciesReferenceGlyphs>
```

In many cases, the curves will be drawn with an arrowhead (or some other graphical symbol) at one end. Therefore, the direction of the curves matters. We recommend that curves in a species reference glyph are defined so that the arrowhead is at the end of the curve. Usually, this means that for substrates and products the curve starts at the reaction glyph and points towards the species glyph. For species reference glyphs that describe modifiers the curve should point towards the reaction glyph.

3.7 TextLabels

A `listOfTextGlyphs` element in the layout contains an arbitrary number of `textGlyph` elements. Each text glyph describes a text label. The `textGlyph` element has an `id` attribute of type SId which has to be unique within the model and the layout and a bounding box which specifies the size of the text glyph.

```
<listOfTextGlyphs>
   <textGlyph id="TextGlyph_1" graphicalObject="SG_8"
          originOfText="NADH">
     <boundingBox>
       <position x="10" y="120.722"/>
       <dimensions width="128.4" height="17.4609"/>
     </boundingBox>
   </textGlyph>
        ...
</listOfTextGlyphs>
```

The `graphicalObject` attribute is optional and contains the id of another graphical object in the layout. This can be used to bind a text glyph to any graphical object or any object derived from GraphicalObject (e.g. SpeciesGlyph). A text glyph that is bound to a graphical object functions as a label to that object. This means that if the position of the corresponding object in the layout changes, the text label should be moved accordingly. Likewise, if the graphical

object is deleted, the text glyph should be deleted as well. In order to avoid circular dependencies, the `graphicalObject` attribute may only refer to objects that have been defined earlier in the SBML file. This restriction is completely analogous to similar definitions in the SBML specification [4].

The text that is to be displayed by a text glyph can be specified in two ways. It can either be given directly in the `text` attribute, or the text glyph can specify that the text is to come from another object which can be specified by the `originOfText` attribute. The `originOfText` attribute specifies the id of the object that provides the text for the text glyph. This can be any object in the SBML model that has a globally unique id. The text is then either the name of the object or, if it does not have a name, the id. If both `text` and `originOfText` are given in a `textGlyph` element, the `text` attribute takes precedence.

4   CONCLUSION

We presented an SBML extension that can be used to describe a graphical layout of a reaction network in biochemical terms containing placement and size information for the individual network elements. It is very flexible, simple and structurally as well as syntactically fits well into the SBML framework. It has been thoroughly discussed with members of the SBML Forum [11] and has been accepted for inclusion into the upcoming SBML Level 3 specification.

Different implementations for this layout extension already exist. One implementation written by one of the authors is an extension to libSBML, a library for reading, writing, translating and validating SBML files [9]. Another implementation was done by Herbert Sauro and coworkers. It is part of JDesigner [5] and the SBML Layout Viewer [13] which is an independent web based tool for rendering of SBML files with layout information.

An XSLT stylesheet that translates SBML files with layout information into SVG images is also available from our web page.

As mentioned above, this extension to SBML only covers the description of the layout of reaction graphs, i.e. the size and position of the graphical representations of the model elements. However, it is designed to be easily extensible in different ways. On the one hand, it could be extended to contain a semantically richer description of the reaction network, e.g. different graphical symbols for different types of compounds or interactions [7, 8], in connection with a controlled vocabulary [18]. On the other hand, detailed information about how the graphical objects should be rendered can be added (colors, line styles, shapes, etc.). Several modeling tools have already done steps in this direction [10]. We hope that, with the help of the SBML community, we will be able to unite those separate efforts in order to come up with a common way of specifying render information that complements the layout extension.

REFERENCES

[1] GraphML. http://graphml.graphdrawing.org/, 2006.

[2] M. Hucka and A. Finney. Systems biology markup language: Level 2 and beyond. Biochemical Society Transactions, 31:1472–1473, 2003.

[3] M. Hucka, A. Finney, H. Sauro, and H. Bolouri. Systems Biology Markup Language (SBML) Level 1: Structures and Facilities for Basic Model Definitions. http://www.sbml.org/sbml/docs/papers/sbml-level-1/html/sbml-level-1.html, 2005.

[4] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J. H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, Ursula Kummer, N. Le Novere, L. M. Loew, D. Lucio, Pedro Mendes, Eric Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, Jörg Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. Bioinformatics, 1:524–531, 2003.

[5] JDesigner. http://www.cds.caltech.edu/∼hsauro/JDesigner.htm, 2006.

[6] Peter D. Karp and Suzanne Paley. Automated Drawing of Metabolic Pathways. In L. Hunter, D. Searls, and J. Shavlik, editors, Third International Conference on Bioinformatics and Genome Research, pages 207–215. AAAI Press, 1994.

[7] H. Kitano, A. Funahashi, Y. Matsuoka, and K. Oda. Using process diagrams for the graphical representation of biological networks. Nature Biotechnology, 23(8):961–966, 2005.

[8] K. W. Kohn, M. I. Aladjem, J. N. Weinstein, and Y. Pommier. Molecular Interaction Maps of Bioregulatory Networks: A General Rubric for Systems Biology. Mol. Biol. Cell, in Press, 2005.

[9] libSBML. http://www.sbml.org/software/libsbml/, 2006.

[10] H. Sauro. JDesigner SBML Annotation. http://www.cds.caltech.edu/ hsauro/JDSBMLEx.pdf, 2006.

[11] SBML Forums. http://www.sbml.org/forums/, 2006.

[12] SBML Layout Extension documentation. http://projects.villa-bosch.de/bcb/sbml/, 2005.

[13] SBML Layout Viewer. http://134.173.97.95/SBMLLayout.aspx, 2006.

[14] SBML Level 2 schema specification. http://www.sbml.org/sbml/level2/version1/, 2006.

[15] Falk Schreiber. High quality visualization of biochemical pathways in BioPath. In Silico Biology, 6, 2002.

[16] SVG. http://www.w3.org/Graphics/SVG, 2006.

[17] System Biology Markup Language Website. http://www.sbml.org/, 2006.

[18] Systems Biology Ontologies. http://www.biomodels.net/index.php?s=SBO, 2006.

[19] Katja Wegner and Ursula Kummer. A new dynamical layout algorithm for complex biochemical reaction networks. BMC Bioinformatics, 6(212), 2005.