SBML Level 3 Package: Layout ('layout')

Ralph Gauges
gauges@hs-albsig.de
Hochschule Albstadt-Sigmaringen
Sigmaringen, Germany

Ursula Rost
ursula.rost@bioquant.uni-heidelberg.de
Modelling of Biological Processes
University of Heidelberg
Heidelberg, Germany

Sven Sahle
sven.sahle@bioquant.uni-heidelberg.de
Modelling of Biological Processes
University of Heidelberg
Heidelberg, Germany

Katja Wengler
wengler@dhbw-karlsruhe.de
Biological and neural computation
Baden-Wuerttemberg Cooperate State University
Karlsruhe, Germany

Frank T. Bergmann
fbergmann@caltech.edu
Computing and Mathematical Sciences
California Institute of Technology
Pasadena, CA, US

Version 1 (Draft)

May 17, 2013

This is a draft specification for the package 'layout' and not a normative document. Please send feedback to the Package Working Group mailing list at sbml-layout@lists.sourceforge.net

The latest release, past releases, and other materials related to this specification are available at
http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/Layout_(layout)

This release of the specification is available at
http://sbml.org/Documents/Specifications/Layout_Level_1_Version_1

Contents

1 Introduction

With the Systems Biology Markup Language (SBML) there now is a common standard for the exchange of dynamical systems data which has already been adopted by many applications in this field Hucka et al. (2011, 2003); SBML Team (2012).

In 2002 we worked on an automatic layout algorithm (Wegner and Kummer (2005)). Based on a given SBML file species and reactions should be placed automatically as a network. Since there was no way to save the final layout of the network, we started developing the layout extension. In 2003 the first draft was done and presented on the SBML workshop in St. Louis in 2004. Based on the discussions in St. Louis Ralph Gauges finalized a first specification and implementation for libSBML which was published in Gauges et al. (2006). During the next SBML meetings the layout extension was discussed heavily and also challenged by other proposals but due to the constant support from the community (e.g.: Deckard et al. (2006)), it got finally accepted as a package of SBML Level 3.

1.1 Proposal corresponding to this package specification

This specification for Layout in SBML Level 3 Version 1 is based on the proposal by the same authors, located at the following URL:

http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Layout

The tracking number in the SBML issue tracking system (SBML Team, 2010) for the Layout package activities is 3594234. The version of the proposal used as the starting point for this specification is the version of April 2005. Previous versions of the current proposal are available from:

http://otto.bioquant.uni-heidelberg.de/sbml/

Details of earlier independent proposals are provided in Section 2.

1.2 Tracking number

As initially listed in the SBML issue tracking system under:
http://sourceforge.net/tracker/?func=detail&aid=3594234&group_id=71971&atid=894711.

1.3 Package dependencies

The Layout package adds additional classes to SBML Level 3 Version 1 Core and has no dependency on any other SBML Level 3 package.

1.4 Document conventions

Following the precedent set by the SBML Level 3 Version 1 Core specification document, we use UML 1.0 (Unified Modeling Language; Eriksson and Penker 1998; Oestereich 1999) class diagram notation to define the constructs provided by this package. We also use color in the diagrams to carry additional information for the benefit of those viewing the document on media that can display color. The following are the colors we use and what they represent:

- *Black*: Items colored black in the UML diagrams are components taken unchanged from their definition in the SBML Level 3 Version 1 Core specification document.
- *Green*: Items colored green are components that exist in SBML Level 3 Version 1 Core, but are extended by this package. Class boxes are also drawn with dashed lines to further distinguish them.
- *Blue*: Items colored blue are new components introduced in this package specification. They have no equivalent in the SBML Level 3 Version 1 Core specification.

We also use the following typographical conventions to distinguish the names of objects and data types from other entities; these conventions are identical to the conventions used in the SBML Level 3 Version 1 Core specification document:

AbstractClass: Abstract classes are classes that are never instantiated directly, but rather serve as parents of other object classes. Their names begin with a capital letter and they are printed in a slanted, bold, sans-serif typeface. In electronic document formats, the class names defined within this document are also hyperlinked to their definitions; clicking on these items will, given appropriate software, switch the view to the section in this document containing the definition of that class. (However, for classes that are unchanged from their definitions in SBML Level 3 Version 1 Core, the class names are not hyperlinked because they are not defined within this document.)

Class: Names of ordinary (concrete) classes begin with a capital letter and are printed in an upright, bold, sans-serif typeface. In electronic document formats, the class names are also hyperlinked to their definitions in this specification document. (However, as in the previous case, class names are not hyperlinked if they are for classes that are unchanged from their definitions in the SBML Level 3 Version 1 Core specification.)

`SomeThing`, `otherThing`: Attributes of classes, data type names, literal XML, and generally all tokens *other* than SBML UML class names, are printed in an upright typewriter typeface. Primitive types defined by SBML begin with a capital letter; SBML also makes use of primitive types defined by XML Schema 1.0 (Biron and Malhotra, 2000; Fallside, 2000; Thompson et al., 2000), but unfortunately, XML Schema does not follow any capitalization convention and primitive types drawn from the XML Schema language may or may not start with a capital letter.

The UML diagrams in this document show the name of the class on top. Below are the attributes specific to that class. Optional attributes have some default value which may be NULL. Arrays are written in square brackets where with the valid array length within those brackets. So an array [2..] would mean that it can hold from 2 to ∞ number of objects.

For other matters involving the use of UML and XML, we follow the conventions used in the SBML Level 3 Version 1 Core specification document.

2 Background and context

Currently, there is no official way of encoding the layout of computational models in SBML. Software tools wishing to share this information have been using the SBML annotation scheme to store this information in proprietary form.

The layout proposal was made in early 2003, since then it has been incorporated into libSBML (Gauges et al. (2006)) and has been used by software applications (e.g.: Sahle et al. (2006), Bergmann and Sauro (2006)) for SBML Level 2 and Level 3.

The overall structure of this proposal reflects design decisions that will be detailed in this section. These decisions are mainly based on the discussion on the mailing list and during the workshop in St. Louis. It was requested that several layouts should be stored in one SBML file. And so the layout is stored in a **ListOfLayouts** as child of the the **Model** element instead of direct annotations to the model constituents.

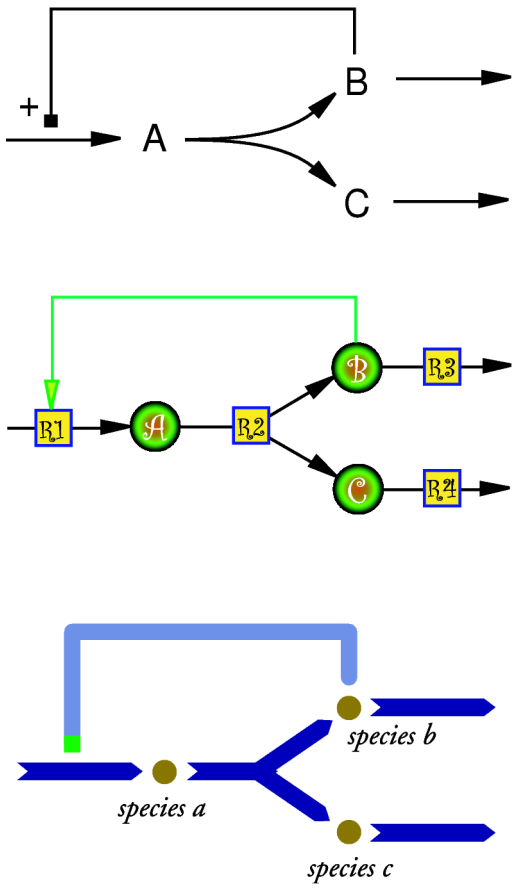

Figure 1: *Illustration of different renderings of the same layout.*

The layout of a reaction network diagram should be described as graphical representations of species and reactions (and not as arbitrary drawing or graph). This means that existing languages for the description of vector drawings (SVG) or general graphs cannot be used. While it may seem unnecessary to invent a new language when an existing one like SVG could in principle be used to describe the layout of a reaction network, there are good reasons to have a language tailored specifically for the layout of SBML models.

Presumably, most programs that will use this SBML extension are primarily programs dealing with biochemi-

cal models. Internally, they will have data structures for species and reactions, so it will be natural for them to describe the layout of the reaction network also in terms of species and reactions (and not in terms of polygons or splines). Thus the `layout` object has a similar structure like the SBML `model` object and contains lists of graphical representations of compartments, species, and reactions (called `compartmentGlyph, speciesGlyph,` and `reactionGlyph` respectively). Additional layout elements and relationships can be represented by using the `graphicalObject` and `generalGlyph` elements.

Another important question is the level of detail that the description should provide. For simplicity, only the layout (i.e., the position of the different graphical objects) of the diagram is encoded, not the details of how it should be rendered. That is left to the SBML Level 3 Render package.

Figure 1 illustrates this distinction. All three diagrams could be renderings of the same layout and would be described by identical SBML files. No information about colors, line styles, fonts, etc., is present in the layout description.

The next question is how the relation between the model and the layout should be established. There seems to be consensus that one model element can be represented by several layout elements. For example, it can be useful to have several representations of one species in the layout to avoid crossing lines. This can be accomplished if every layout element has a field that refers to the id of a model element.

There are also cases where a layout element does not correspondent to exactly one model element. One example would be if a layout shows a simplified version of the model where one reaction in the layout corresponds to several reactions and intermediate species in the model. This is the reason why the field in the layout elements that refers to the model elements is optional, allowing layout objects that do not have a specific counterpart in the SBML model.

The result of all this is a way to describe a graphical layout of a reaction network in biochemical terms. This layout can be closely tied to the biochemical model. A graphical model editor for example would typically create a layout that is closely connected (by a one-to-several relation from the model elements to the layout elements) to the model.

A more general layout design program could also create a layout that is not so closely tied to the model, for example, it could create a layout that shows a simplified version of the model.

3 Package syntax and semantics

In this section, we define the syntax and semantics of the Layout package for SBML Level 3 Version 1. We expound on the various data types and constructs defined in this package, then in Section 4 on page 22,we provide complete examples of using the constructs in an example SBML model.

3.1 Namespace URI and other declarations necessary for using this package

Every SBML Level 3 package is identified uniquely by an XML namespace URI. For an SBML document to be able to use a given SBML Level 3 package, it must declare the use of that package by referencing its URI. The following is the namespace URI for this version of the Layout package for SBML Level 3 Version 1:

"`http://www.sbml.org/sbml/level3/version1/layout/version1`"

In addition, SBML documents using a given package must indicate whether understanding the package is required for complete mathematical interpretation of a model, or whether the package is optional. This is done using the attribute `required` on the `<sbml>` element in the SBML document. For the Layout Package, the value of this attribute must be set to "`false`".

The following fragment illustrates the beginning of a typical SBML model using SBML Level 3 Version 1 and this version of the Layout package:

```
<?xml version="1.0" encoding="UTF-8"?>
 <sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
   xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1" layout:required="false">
```

Historically, the layout package has also been used to encode the layout of SBML Level 2 models, there the following namespace is to be used within an annotation:

"`http://projects.eml.org/bcb/sbml/level2`"

A minimal example would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
 <sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
   <model>
    <annotation>
      <listOfLayouts xmlns="http://projects.eml.org/bcb/sbml/level2">
       [...]
      </listOfLayouts>
    </annotation>
   </model>
  </sbml>
```

3.2 Coordinate System and Size

All size information given for layout objects are understood to be in Point (pt), which is defined to be 1/72 of an inch (0.3527777778 mm) as in postscript.

The layout extension uses a Cartesian coordinate system. The origin of the coordinate system will be in the upper left corner of the screen. The positive x-axis runs from left to right, the positive y-axis run from top to bottom and the positive z-axis points into the screen. The reason for having the origin in the upper left corner of the screen is that most 2D packages do it that way. This coordinate system is also right handed just like the ones in OpenGL and Java3D, which should facilitate 3D implementations as well.

3.3 Primitive data types

Section 3.1 of the SBML Level 3 Version 1 Core specification defines a number of primitive data types and also uses a number of XML Schema 1.0 data types (Biron and Malhotra, 2000). Of the data types defined and referenced in SBML Level 3 Version 1 Core, we make use of `integer`, `double`, `string`, `SId`, `SIdRef` and `IDREF`. Additionally, the Layout package defines the `SpeciesReferenceRole` and uses the `xsi:type` from the XML Schema instance (XSI) namespace.

Figure 8 illustrates the interrelation between these entities.

The `SId` type is used as the data type for the identifiers of all layout classes. As such they have to be unique within the model and are not allowed to conflict with `SId`s from SBML elements or any package that use the same `SId` namespace.

3.3.1 Type SpeciesReferenceRole

The Layout package defines a new enumerated type `SpeciesReferenceRole` which represents the role that a species takes within a reaction. It can take only one of the following values:

`substrate`,	`product`,
`sidesubstrate`,	`sideproduct`,
`modifier`,	`activator`,
`inhibitor`	`undefined`.

3.3.2 Type `xsi:type`

In order to easily differentiate between **LineSegment** and **CubicBezier** elements, the `xsi:type` attribute from the XML Schema instance (XSI) namespace is used. The namespace is:

"`http://www.w3.org/2001/XMLSchema-instance`"

For this purpose the `xsi:type` is set to the following fixed values: "`LineSegment`" for line segments and "`CubicBezier`" for splines.

3.4 General features

This section describes the common classes **Point** , **Dimensions** and **BoundingBox** that are used by all other classes of this package. It also describes the representation of curves.

3.4.1 The Point class

A point is specified via the required attributes `x`, `y` and an optional attribute `z`, all of which are of type `double`. If the attribute `z` is not specified, the object is a two dimensional object.

The **Point** class also has an optional attribute `id` of type `SId`. While not used in the Layout package, it can be used by programs to refer to the elements.

3.4.2 The Dimensions class

A dimension is specified via the required attributes `width`, `height` and an optional attribute `depth`, all of which are of type `double`. If the attribute `depth` is not specified, the object is a two dimensional object.

The `width` specifies the size of the object in the direction of the positive x axis, the `height` attribute specifies the size of the object along the positive y axis and the `depth` attribute specifies the size of the object along the positive z axis. All sizes for **Dimensions** objects are positive values, and so the attributes are not allowed to take negative values.

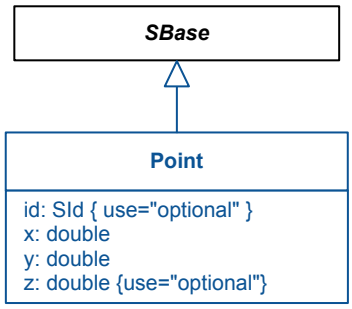

Figure 2: *The definitions of the* ***Point*** *class.*

The **Dimensions** class also has an optional attribute `id` of type `SId`. While not used in the Layout package, it can be used by programs to refer to the elements.

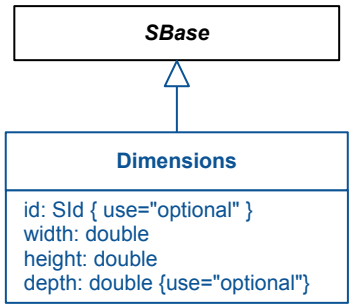

Figure 3: *The definitions of the* ***Dimensions*** *class.*

3.4.3 The BoundingBox class

A **BoundingBox** consists of the required elements `position` and `dimensions`.

The **BoundingBox** class also has an optional attribute `id` of type `SId`. While not used in the Layout package, it can be used by programs to refer to the elements.

The `position` *element*

The `position` always specifies the upper left corner of the bounding box. The `position` is of type **Point** .

The `dimensions` *element*

The `dimensions` element is required and of type **Dimensions**. It represents the size of the bounding box.

3.4.4 The Curve class

The curve class describes how to connect elements of the layout package. It is fully specified by a mandatory `listOfCurveSegments` element and is used in four places:

- **SpeciesReferenceGlyph**: Here it describes a curve from/to the center piece of the parent **ReactionGlyph** to/from the **SpeciesGlyph** it represents.
- **ReactionGlyph**: Here it describes a curve for the center piece of a reaction.

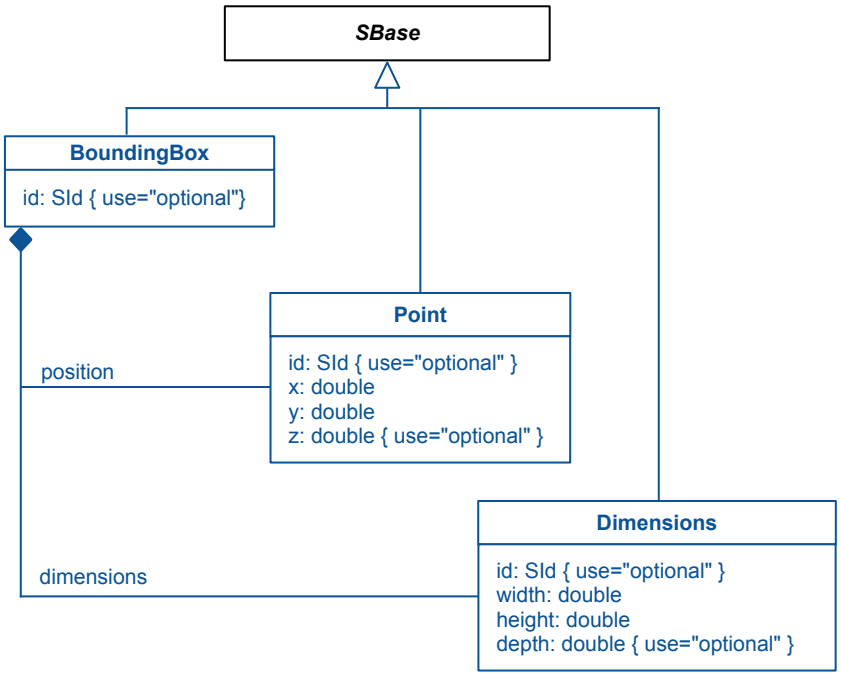

Figure 4: *The definitions of the* ***BoundingBox*** *class.*

- **ReferenceGlyph**: Here it describes a curve from/to the center piece of the parent **GeneralGlyph** to/from the glyph it represents.
- **GeneralGlyph**: Here it describes a curve for the center piece of an additional relationship.

In the above the *center piece* refers to either the **Curve** element of a **ReactionGlyph**, or its **BoundingBox**.

The `listOfCurveSegments` *element*

The `listOfCurveSegments` contains an arbitrary number of curve segments that can be either of type **LineSegment** or of type **CubicBezier**.

3.4.5 The LineSegment class

The **LineSegment** class consists of the mandatory attribute `xsi:type` and two elements of type **Point** . One is called `start` and represents the starting point of the line, the other is called `end` and represents the endpoint of the line.

The `xsi:type` *attribute*

For straight line segments, the attribute `xsi:type` must have the value set to:

"`LineSegment`"

Note that the `xsi:type` is from the `xsi` namespace:

"`http://www.w3.org/2001/XMLSchema-instance`"

3.4.6 The CubicBezier class

In order to be able to represent smooth curves the Layout package defines the class **CubicBezier**. It represents a Bezier curve (Wikipedia (2013)), as is readily available in most Graphic APIs.

The class **CubicBezier** is derived from **LineSegment**. It consists of four elements, the two inherited elements `start`

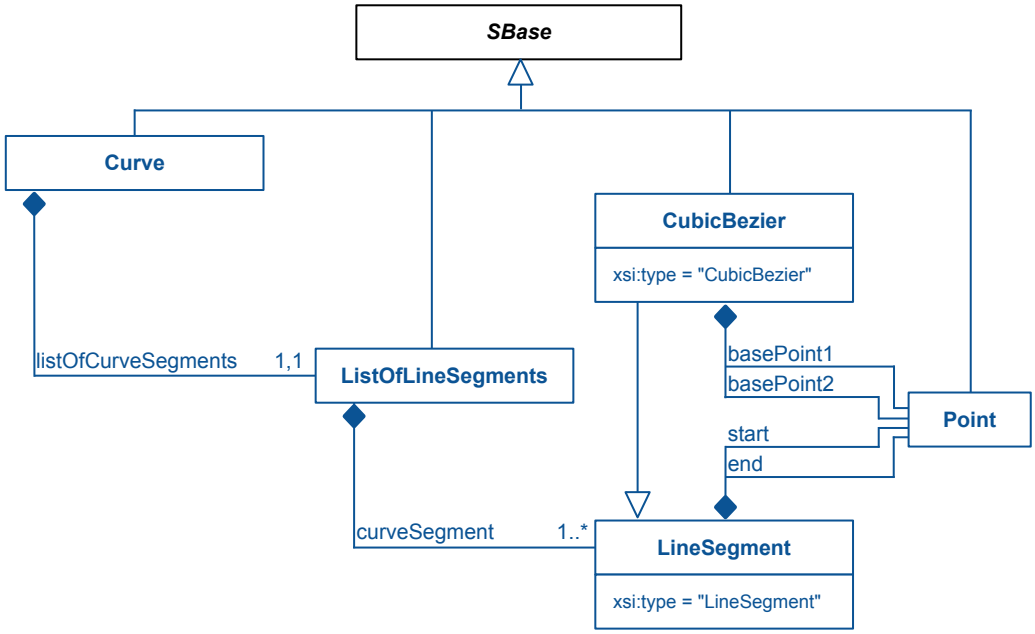

Figure 5: *The definitions of the* ***Curve*** *class.*

and `end`, that specify the starting point and the endpoint of the cubic bezier curve, and two elements `basePoint1` and `basePoint2` which specify the two additional base points that are needed to describe a cubic bezier curve.

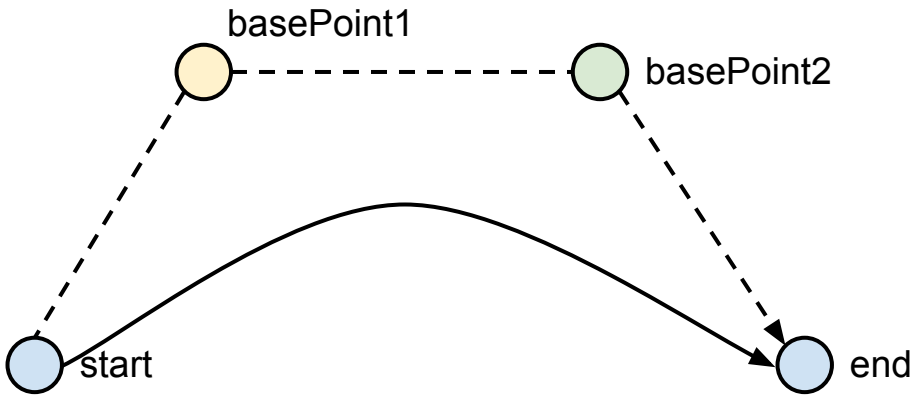

Figure 6: *A depiction of a cubic bezier curve, including the points* `start`, `end` *as well as* `basePoint1` *and* `basePoint2`.

The `basePoint1` element represents the base point closer to the `start` point and `basePoint2` the base point closer to `end` point. This allows tools not able to render bezier curves to approximate them by directly connecting the four points (as visible in the dashed line in Figure 6).

The `xsi:type` *attribute*

For cubic bezier curves, the attribute `xsi:type` must have the value set to:

"`CubicBezier`"

Note that the `xsi:type` is from the `xsi` namespace:

"`http://www.w3.org/2001/XMLSchema-instance`"

3.5 The extended Model class

The **SBML Model** class is extended with the addition of one child the `listOfLayouts`. A **Model** may contain at most one such list.

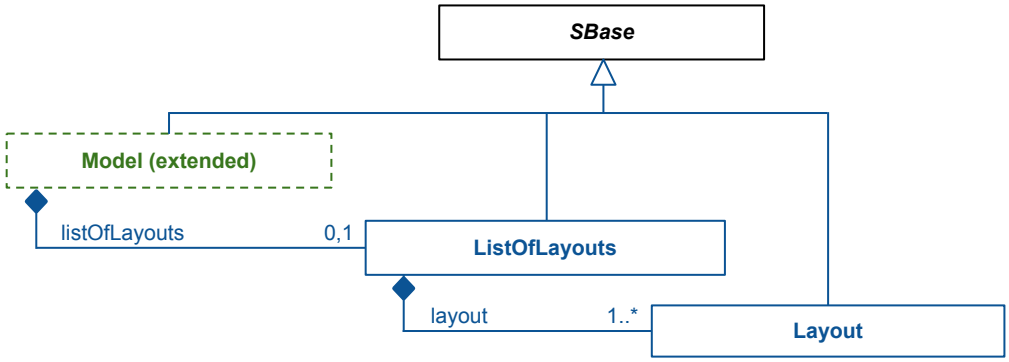

Figure 7: *The definitions of the extended* ***Model*** *class.*

The listOfLayouts class

As shown in Figure 7 the **ListOfLayouts** is derived from ***SBase*** and inherits the attributes `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes**. The **ListOfLayouts** must contain at least one Layout (defined in Section 3.6).

Should the Layout package be used in an **SBML** Level 2 model, the **ListOfLayouts** needs to be placed in an **Annotation** on the **Model** element. For a complete example that uses SBML Level 2 see Section 4.6 on page 30.

3.6 The Layout class

The layout class stores layout information for some or all elements of the SBML model as well as additional objects that need not be connected to the model.

The **Layout** has two attributes: `id` and `name`. Additionally a **Dimensions** element specifies the size of the layout. The actual layout elements are contained in several lists namely: a **ListOfCompartmentGlyphs**, a **ListOfSpeciesGlyphs**, a **ListOfReactionGlyphs**, a **ListOfTextGlyphs**, and a **ListOfAdditionalGraphicalObjects**. Each of these lists can only occur once, and, if present, is not allowed to be empty.

The `id` *attribute*

The `id` attributes takes a required value of type `SId`. The `id` which uniquely identifies the layout.

The `name` *attribute*

The `name` attributes takes an optional value of type `String`. It allows to specify a human readable name for the **Layout**.

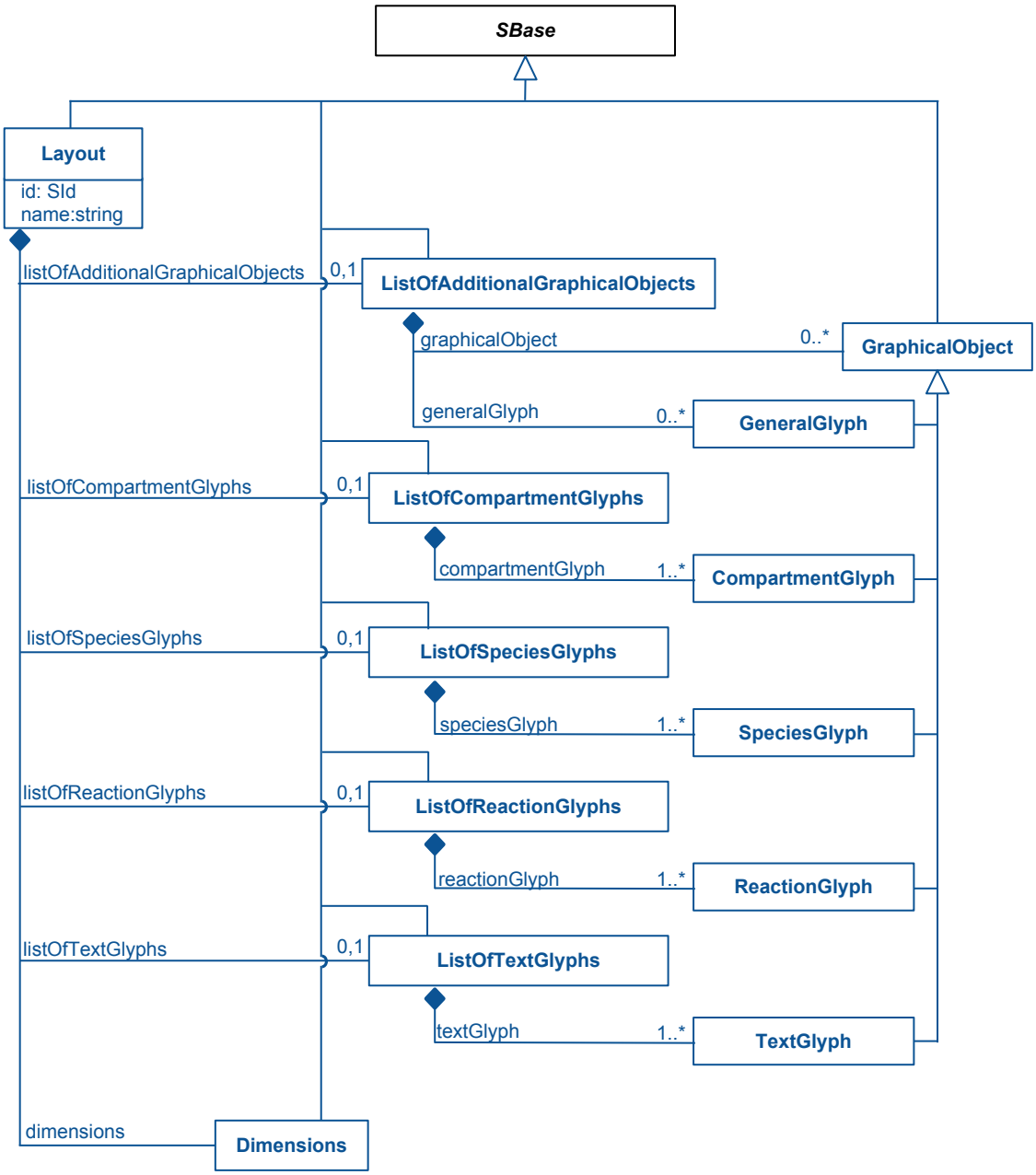

Figure 8: *A UML representation of the Layout package. Derived from* ***SBase****, the Layout classes inherit support for constructs such as SBML* ***Notes*** *and* ***Annotation****s. See Section 1.4 for conventions related to this figure. The individual classes are further discussed in the text.*

The `dimensions` *element*

The `dimensions` element of type **Dimensions** specifies the dimensions of this layout. This element is required. It holds the dimensions of all layout elements (care should be taken when using cubic beziers (see Section 3.4.6), that the described curve also lies within the given dimensions).

The `listOfCompartmentGlyphs` *element*

The `listOfCompartmentGlyphs`, when present must contain one or more **CompartmentGlyph** elements.

Note that not all **Compartment** elements have to be represented by a **CompartmentGlyph**. In fact quite often the compartment is omitted in a layout for a uni-compartmental network. (In which case the **ListOfCompartmentGlyphs** would be ommitted altogether.)

The `listOfSpeciesGlyphs` *element*

The `listOfSpeciesGlyphs`, when present must contain one or more **SpeciesGlyph** elements. Just as with **CompartmentGlyph** elements, not every **Species** of the model has to have a representation in this list.

The `listOfReactionGlyphs` *element*

The `listOfReactionGlyphs`, when present must contain one or more **ReactionGlyph** elements. Again, not all reactions of the containing SBML model have to be included in the containing layout.

The `listOfTextGlyphs` *element*

The `listOfTextGlyphs`, when present, must contain one or more **TextGlyph** elements.

The `listOfAdditionalGraphicalObjects` *element*

Most objects for which layout information is to be included in an SBML file have a corresponding object in the SBML model. As there might be cases where the user wants to include object types in the layout that do fall in any of the other categories described below, we include a `listOfAdditionalGraphicalObjects` in each Layout object. This list holds an arbitrary number of `graphicalObject` elements. The `graphicalObject` only defines a bounding box in a specific place in the layout without giving additional information about its contents.

The `listOfAdditionalGraphicalObjects`, when present must contain one or more of the following elements: **GraphicalObject**, **GeneralGlyph**.

When using a **GraphicalObject** it is recommended that some form of meta information is provided. For additional relationships such as SBML events or rules, the **GeneralGlyph** can be used, see Section 3.11.

3.7 The GraphicalObject class

All the more specific layout elements (**CompartmentGlyph**, **GeneralGlyph**, **SpeciesGlyph**, **ReactionGlyph**, **ReferenceGlyph**, **TextGlyph**, and **SpeciesReferenceGlyph**) are derived from **GraphicalObject**.

Each GraphicalObject has a mandatory **BoundingBox**, which specifies the position and the size of the object.

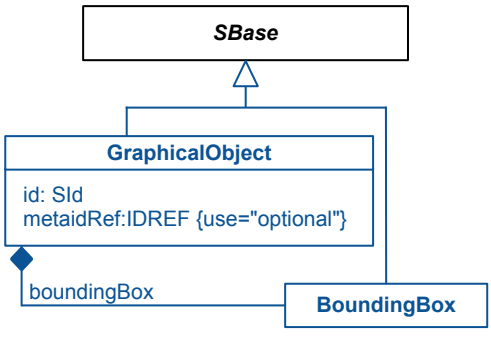

Figure 9: *The definitions of the* ***GraphicalObject*** *class.*

While **GraphicalObject** is the base class for most elements in the Layout package, it is **not** an abstract class. It can be instantiated when used in the `listOfAdditionalGraphicalObjects` to describe additional elements and relationships. Since it only describes a **BoundingBox**, programs are encouraged to add **Annotation** objects that

describe program specific graphical information.

The `id` *attribute*

The **GraphicalObject** has a mandatory `id` attribute of type `SId` through which it can be identified.

The `metaidRef` *attribute*

The **GraphicalObject** has an optional `metaidRef` attribute of type `IDREF` that allows the object to uniquely reference elements in the **Model**. Wherever possible it is preferred that the more specific reference mechanisms are used.

3.8 The CompartmentGlyph class

The **CompartmentGlyph** class is derived from **GraphicalObject** and inherits its attributes. Additionally, it has two optional attributes `compartment` and `order`. For an example see Section 4.1.

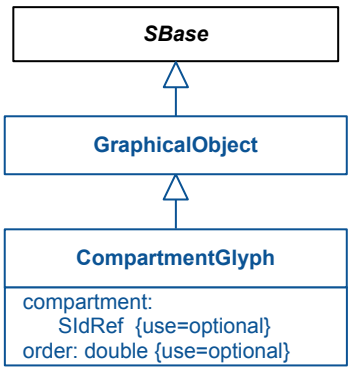

Figure 10: *The definitions of the* ***CompartmentGlyph*** *class.*

The `compartment` *attribute*

The `compartment` attribute is of type `SIdRef`. It is used to add a reference to the `id` of the corresponding compartment in the model. Since the `compartment` is optional, the user can specify compartments in the layout that are not part of the model.

If the `compartment` attribute is used together with the `metaidRef` they need to refer to the same object in the **Model**.

The `order` *attribute*

The `order` attribute is an optional attribute of type `double`. It is there to handle the case where compartments in a layout are overlapping, and tools would want to clearly disambiguate which **CompartmentGlyph** is on top of another one.

The `order` attribute follows the coordinate system. There the `z` dimension is pointing into the screen, thus an element with a **lower** `order` value will be **in front** of elements with a **higher** value.

If not specified, the `order` is undefined and tools are free to display the compartment glyphs in the order that best fits their needs.

Note: Usually the `order` attribute is only used for two dimensional layouts since in three dimensional layouts the `z` coordinate can be used to easily distinguish CompartmentGlyphs. However, there may be uses for having the same `z` coordinate in three dimensional layouts, in which case the `order` attribute will help to disambiguate the drawing order.

3.9 The SpeciesGlyph class

In addition to the attributes from **GraphicalObject**, the **SpeciesGlyph** object has an optional `species` attribute. For an example see Section 4.2.

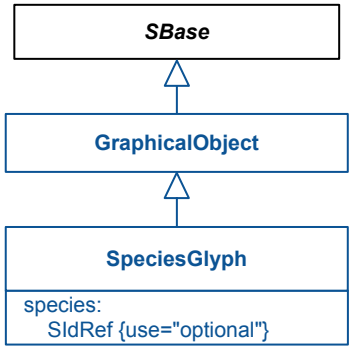

Figure 11: *The definitions of the **SpeciesGlyph** class.*

The `species` *attribute*

The `species` attribute of type `SIdRef` allows modelers to link the **SpeciesGlyph** to the `id` of the corresponding species object in the **Model**. The `species` attribute is optional to allow the program to specify species representations that do not have a direct correspondence in the model. This might be useful if some pathway has been collapsed, but is still treated by layout programs.

If the `species` attribute is used together with the `metaidRef` they need to refer to the same object in the **Model**.

3.10 The ReactionGlyph class

The **ReactionGlyph** is used to represent **Reaction** elements in the layout. Analogous to how a **Reaction** object has to at least have one reactant or product, the **ReactionGlyph** has to at least have one **SpeciesReferenceGlyph** stored in the **ListOfSpeciesReferenceGlyphs**.

The **ReactionGlyph** inherits from **GraphicalObject**. In addition to the attributes inherited from **GraphicalObject**, the **ReactionGlyph** is described by an attribute `reaction`, a **Curve** element and a `listOfSpeciesReferenceGlyphs` element.

The **Curve** describes the center section of a **ReactionGlyph**. The center section is frequently used by tools to separate the point where substrates arcs come together, from the point where product arcs split off. The **Curve** is optional, and when not present the dimensions of the inherited **BoundingBox** describes the center section, by storing its position and dimension.

For an example see Section 4.3.

The `reaction` *attribute*

The `reaction` attribute of type `SIdRef` is used to specify the `id` of the corresponding **Reaction** in the model. This reference is optional.

If the `reaction` attribute is used together with the `metaidRef` they need to refer to the same object in the **Model**.

The `listOfSpeciesReferenceGlyphs` *element*

Since a **Species** element can have several graphical representations in the layout there must be a way to specify which **SpeciesGlyph** should be connected to the **ReactionGlyph**. This is done using the `listOfSpeciesReference-`

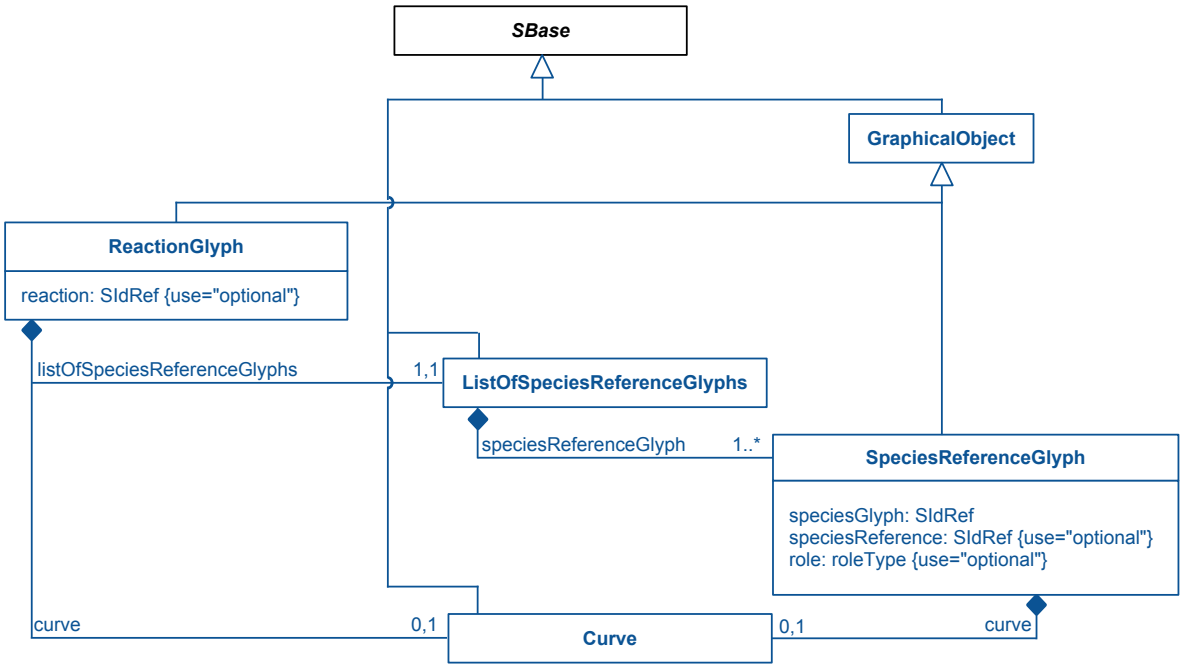

Figure 12: *The definitions of the* ***ReactionGlyph*** *class.*

Glyphs.

The **ListOfSpeciesReferenceGlyphs** is mandatory, since every **Reaction** has to have at least one reactant or product.

The `curve` *element*

The optional **Curve** element (Section 3.4.4) can be used to describe a curve representation for the **ReactionGlyph**.

If a **ReactionGlyph** specifies a curve, the bounding box is to be ignored.

3.10.1 The SpeciesReferenceGlyph class

The `speciesReferenceGlyph` element describes the graphical connection between a **SpeciesGlyph** and a **ReactionGlyph** (which would be an arrow or some curve in most cases).

As can be seen in the diagram Section 3.10, a **SpeciesReferenceGlyph** inherits from **GraphicalObject**. Additionally, it has a mandatory attribute `speciesGlyph` and two optional attributes `speciesReference` and `role`. Optionally, the **SpeciesReferenceGlyph** also has an element `curve`.

If the `curve` is specified, it overrides the inherited bounding box.

The `speciesGlyph` *attribute*

The `speciesGlyph` is of type `SIdRef`. It contains a reference to the `id` of a **SpeciesGlyph** object that is to be connected to the **ReactionGlyph**. This attribute is mandatory so as to ensure unambiguity about which SpeciesGlyph has to be connected with this ReactionGlyph.

The `speciesReference` *attribute*

The `speciesReference` is an optional attribute of type `SIdRef` that allows modelers to connect the **SpeciesReferenceGlyph** with a particular **SpeciesReference** (or **ModifierSpeciesReference**) of the containing model.

If the `speciesReference` attribute is used together with the `metaidRef` they need to refer to the same object in

the **Model**.

NOTE: The `id` on a **SpeciesReference** and **ModifierSpeciesReference** was only introduced in SBML Level 3 Version 1 Core, in order to successfully use the `speciesReference` in prior Levels of SBML, the corresponding species reference has to be annotated with an identifier. The annotation would look like this:

```
<annotation>
  <layoutId xmlns="http://projects.eml.org/bcb/sbml/level2" id="theId"/>
</annotation>
```

This `id` has to be unique within the global `SId` namespace of the SBML model and can thus be used to reference a given species reference. For a complete example see also Section 4.3.

The `role` *attribute*

The `role` attribute is of `SpeciesReferenceRole` (see Section 3.3.1) and is used to specify how the species reference should be displayed. Allowed values are `substrate`, `product`, `sidesubstrate`, `sideproduct`, `modifier`, `activator`, `inhibitor` and `undefined`.

This attribute is optional and should only be necessary if the optional `speciesReference` attribute is not given or if the respective information from the model needs to be overridden.

- 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 simple chemicals like ATP, NAD+, etc. This allows programs to render them as side reactions.
- `activator` and `inhibitor` are modifiers where their influence on the reaction is known and `modifier` is a more general term if the influence is unknown or changes during the course of the simulation.

In order to specify more specific types of interactions it is recommended to use the `sboTerm` on the **SpeciesReference**. If both `role` and `sboTerm` are specified and conflicting, it is the `role` that takes precedence.

The `curve` *element*

The `curve` is an optional element of type **Curve**. When present the glyphs bounding box (as inherited from the **GraphicalObject**) is to be disregarded.

So as to make the drawing of these curves as easy as possible the line segments should be ordered depending on the role of the **SpeciesReferenceGlyph**. If no `role` attribute is defined the role to be assumed is taken from the role that the **SpeciesReference** referenced via the attribute `speciesReference` has, otherwise it is `undefined`.

- `product`, `sideproduct`, `substrate`, `sidesubstrate`, `undefined`: The line segments have their `start` element at the **ReactionGlyph** and their `end` element at the **SpeciesGlyph**.
- `activator`, `inhibitor`, `modifier`: The line segments have their `start` element at the **SpeciesGlyph** and their `end` element at the **ReactionGlyph**.

3.11 The GeneralGlyph class

The **GeneralGlyph** is used to facilitate the representation of elements other than **Compartment**, **Species** and **Reaction** and thus can be used for the display of relationships of **Rule** or elements defined by other SBML packages. It closely follows the structure of the **ReactionGlyph**.

The **GeneralGlyph** is defined via an optional attribute `reference` as well as the elements `curve`, `listOfReferenceGlyphs` and `listOfSubGlyphs`.

For an example that uses the **GeneralGlyph** to represents a basic influence see Section 4.7.

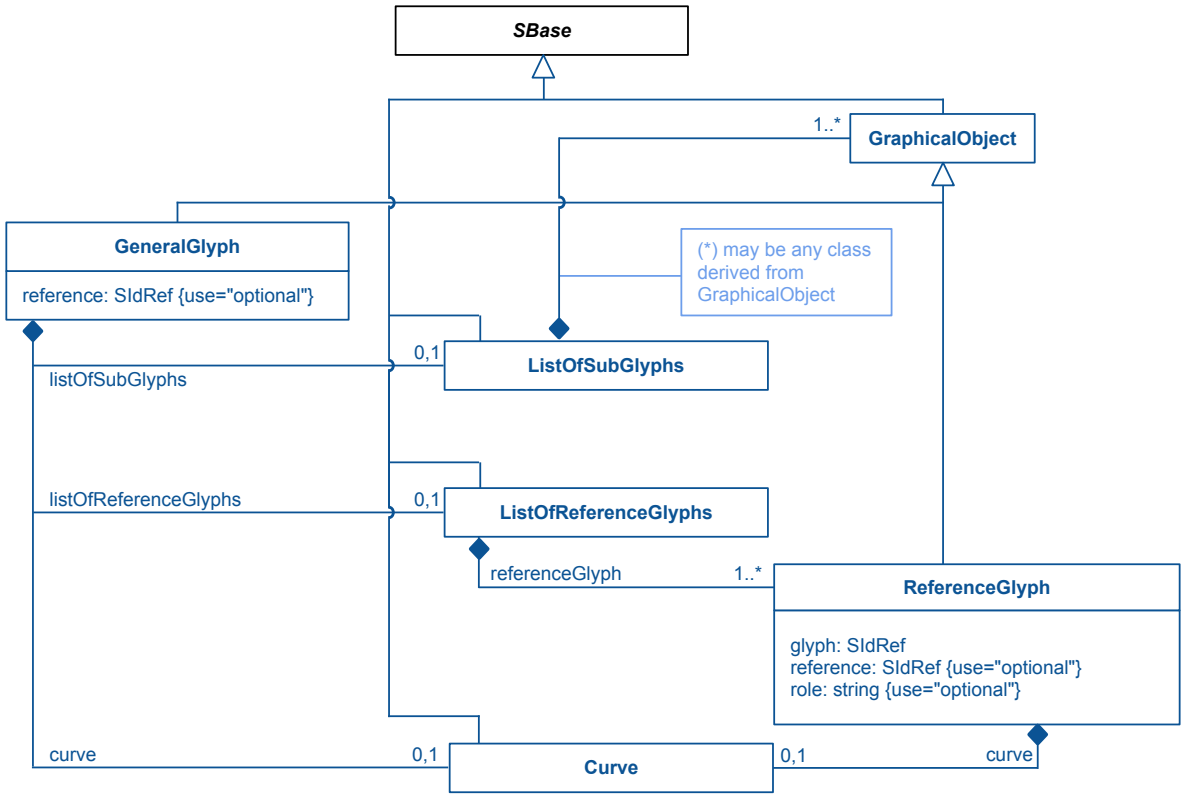

Figure 13: *The definitions of the* ***GeneralGlyph*** *class.*

The `reference` *attribute*

The optional `reference` attribute of type `SIdRef` that can be used to specify the `id` of the corresponding element in the model that is represented.

If the `reference` attribute is used together with the `metaidRef` they need to refer to the same object in the **Model**.

The `listOfSubGlyphs` *element*

The **ListOfSubGlyphs** is an optional list that can contain sub glyphs of the **GeneralGlyph**. One example of its use could be a sub-module, that contains species glyphs and reaction glyph, that are not necessarily part of the containing **Model**. Another example would be an **Event**, visualized with its **Trigger** and additional **GeneralGlyph**s for its **EventAssignment**s. The **ListOfSubGlyphs** consists of **GraphicalObject** or derived classes. Thus, unlike the **ListOfAdditionalGraphicalObjects** (which may only contain **GraphicalObject**s or **GeneralGlyph**s), the `listOfSubGlyphs` may contain any derived class, such as for example **TextGlyph** elements.

When present the `listOfSubGlyphs` must contain at least one such element.

The `listOfReferenceGlyphs` *element*

The **ListOfReferenceGlyphs** is optional, since conceivably the **GeneralGlyph** could just contain a number of sub glyphs. When present, it must include at least one **ReferenceGlyph**.

The `curve` *element*

The optional **Curve** element (Section 3.4.4) can be used to describe a curve representation for the **GeneralGlyph**.

If a **GeneralGlyph** specifies a curve, the bounding box is to be ignored.

3.11.1 The ReferenceGlyph class

The `referenceGlyph` element describes the graphical connection between an arbitrary **GraphicalObject** (or derived element) and a **GeneralGlyph** (which would be an arrow or some curve in most cases).

As can be seen in the diagram Section 3.11, a **ReferenceGlyph** inherits from **GraphicalObject**. Additionally it has a mandatory attribute `glyph` and two optional attributes `reference` and `role`. Optionally the **ReferenceGlyph** also has an element `curve`.

The ReferenceGlyph should either contain a bounding box or a curve specification, if both are given, the bounding box should be ignored.

The `glyph` *attribute*

The `glyph` is of type `SIdRef`. It contains a reference to the `id` of a **GraphicalObject** (or derived) object that is to be connected to the **GeneralGlyph**.

This attribute is mandatory so as to ensure unambiguously which glyph has to be connected with this GeneralGlyph.

The `reference` *attribute*

The `reference` is an optional attribute of type `SIdRef` that is used to connect the **ReferenceGlyph** with a element of the containing SBML model.

The `role` *attribute*

The `role` attribute is of type `string` and is used to specify how the reference should be displayed.

While as `string`, the value of the `role` attribute is unconstraint, current implementations use the same values as specified in Section 3.3.1.

The `curve` *element*

The `curve` is an optional element of type **Curve**. When present the glyph's bounding box (as inherited from the **GraphicalObject**) is to be disregarded.

So as to make the drawing of these curves as easy as possible the line segments should be ordered depending on the role of the **ReferenceGlyph**:

- If the glyph represents a modification it should start at the glyph and end at the center of the **GeneralGlyph**.
- otherwise it should begin at the center section of the **GeneralGlyph** and end at the reference glyph.

3.12 The TextGlyph class

The **TextGlyph** class describes the position and dimension of text labels. It inherits from **GraphicalObject** and adds the attributes `graphicalObject`, `text` and `originOfText`.

For an example see Section 4.4.

The `graphicalObject` *attribute*

The attribute `graphicalObject` is of type `SIdRef`. It contains the `id` of any **GraphicalObject** and specifies that the **TextGlyph** should be considered to be a label to that object. This allows modelers to indicate that the label is to be moved together with the object.

If the `graphicalObject` attribute is used together with the `metaidRef` they need to refer to the same object in the **Layout**.

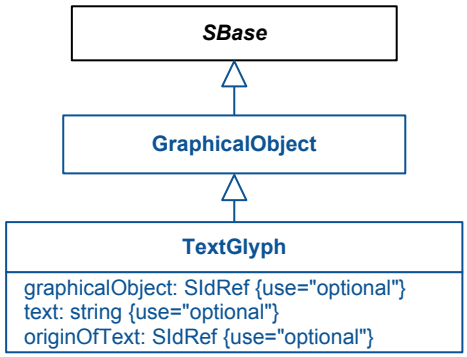

Figure 14: *The definitions of the* ***TextGlyph*** *class.*

The `text` *attribute*

The optional `text` attribute is of type `string`. It facilitates adding of independent text, like a title or a comment to the diagram.

The `originOfText` *attribute*

Additionally the optional attribute `originOfText` of type `SIdRef` can hold the `id` of an entity in the SBML model. If it is specified, the text to be displayed is taken from the `name` attribute of the referenced object.

If both attributes `originOfText` and `text` are specified, the `text` attribute overrides the `originOfText`.

4 Examples

4.1 CompartmentGlyph Example

Below a **CompartmentGlyph** is defined for the compartment `Yeast`. It is located at position `x=5, y=5`and has dimensions `width=390, height=220`.

```
<?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>
          <compartmentGlyph id="CompartmentGlyph_1" compartment="Yeast">
            <boundingBox id="bb1">
              <position x="5" y="5"/>
              <dimensions width="390" height="220"/>
            </boundingBox>
          </compartmentGlyph>
        </listOfCompartmentGlyphs>
                .
                .
                .
      </layout>
     </listOfLayouts>
    </annotation>
    <listOfCompartments>
      <compartment id="Yeast"/>
    </listOfCompartments>
        .
        .
        .
  </model>
</sbml>
```

4.2 SpeciesGlyph Example

Below a **SpeciesGlyph** is defined for the species `Glucose` at position `x=105, y=20` with dimensions `width=130, height=20`.

The following defines a basic **SpeciesGlyph**.

```
<?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"/>
                .
                .
                .
        <listOfSpeciesGlyphs>
          <speciesGlyph id="SpeciesGlyph_Glucose" species="Glucose">
            <boundingBox id="bb2">
              <position x="105" y="20"/>
              <dimensions width="130" height="20"/>
```

```
            </boundingBox>
          </speciesGlyph>
        </listOfSpeciesGlyphs>
            .
            .
            .
       </layout>
      </listOfLayouts>
    </annotions>
         .
         .
         .
    <listOfSpecies>
      <species id="Glucose" compartment="Yeast"/>
    </listOfSpecies>
       .
       .
       .
  </model>
</sbml>
```

4.3 ReactionGlyph Example

The following defines a **ReactionGlyph** for reaction `Hexokinase` containing only a center segment represented as straight line.

```
<?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"/>
              .
              .
              .
        <listOfReactionGlyphs>
          <reactionGlyph id="glyph_Hexokinase" reaction="Hexokinase">
            <curve>
              <listOfCurveSegments>
                <curveSegment xsi:type="LineSegment">
                  <start x="170" y="100"/>
                  <end x="170" y="130"/>
                </curveSegment>
              </listOfCurveSegments>
            </curve>
            <listOfSpeciesReferenceGlyphs>
                    .
                    .
                    .
            </listOfSpeciesReferenceGlyphs>
          </reactionGlyph>
        </listOfReactionGlyphs>
             .
             .
             .
      </layout>
     </listOfLayouts>
    </annotation>
          .
          .
          .
    <listOfReactions>
```

```
      <reaction id="Hexokinase" reversible="false">
                    .
                    .
                    .
      </reaction>
    </listOfReactions>
        .
        .
        .
  </model>
</sbml>
```

The following adds a **SpeciesReferenceGlyph** and demonstrates adding an `id` to a **SpeciesReference** in SBML Level 2 Version 1.

```
<?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">
        </dimensions>
              .
              .
              .
        <listOfReactionGlyphs>
          <reactionGlyph id="glyph_Hexokinase" reaction="Hexokinase">
                    .
                    .
                    .
            <listOfSpeciesReferenceGlyphs>
              <speciesReferenceGlyph id="SpeciesReferenceGlyph_Glucose"
                   speciesReference="SpeciesReference_Glucose"
                   speciesGlyph="SpeciesGlyph_Glucose" role="substrate">
                <curve>
                  <listOfCurveSegments>
                    <curveSegment xsi:type="LineSegment">
                      <start x="170" y="100">
                      </start>
                      <end x="170" y="50">
                      </end>
                    </curveSegment>
                  </listOfCurveSegments>
                </curve>
              </speciesReferenceGlyph>
                        .
                        .
                        .
            </listOfSpeciesReferenceGlyphs>
          </reactionGlyph>
        </listOfReactionGlyphs>
              .
              .
              .
      </layout>
     </listOfLayouts>
    </annotation>
        .
        .
        .
    <listOfReactions>
      <reaction id="Hexokinase" reversible="false">
        <listOfReactants>
```

```
          <speciesReference species="Glucose">
            <annotation>
              <layoutId xmlns="http://projects.eml.org/bcb/sbml/level2"
                        id="SpeciesReference_Glucose"/>
            </annotation>
          </speciesReference>
                .
                .
                .
        </listOfReactants>
      </reaction>
    </listOfReactions>
          .
          .
          .
  </model>
</sbml>
```

4.4 TextGlyph Example

The following defines a **TextGlyph** with `id=TextGlyph_Glucose` that is connected to a **SpeciesGlyph** `SpeciesGlyph_Glucose` and retrieves the text to display from the **Species** with `id="Glucose"`.

```
<?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"/>
                .
                .
                .
        <listOfSpeciesGlyphs>
          <speciesGlyph id="SpeciesGlyph_Glucose" species="Glucose">
            <boundingBox id="bb2">
              <position x="105" y="20"/>
              <dimensions width="130" height="20"/>
            </boundingBox>
          </speciesGlyph>
        </listOfSpeciesGlyphs>
                .
                .
                .
        <listOfTextGlyphs>
          <textGlyph id="TextGlyph_Glucose" graphicalObject="SpeciesGlyph_Glucose"
                     originOfText="Glucose">
            <boundingBox id="bbA">
              <position x="115" y="20">
              </position>
              <dimensions width="110" height="20">
              </dimensions>
            </boundingBox>
          </textGlyph>
        </listOfTextGlyphs>
            .
            .
            .
      </layout>
     </listOfLayouts>
    </annotation>
         .
         .
```

```
        .
    <listOfSpecies>
      <species id="Glucose" name="Glucose" compartment="Yeast"/>
    </listOfSpecies>
       .
       .
       .
  </model>
</sbml>
```

4.5 Example using SBML Level 3 Version 1

Here a small complete example to illustrate and complement the paragraphs above. The model consists of the Hexokinase reaction.

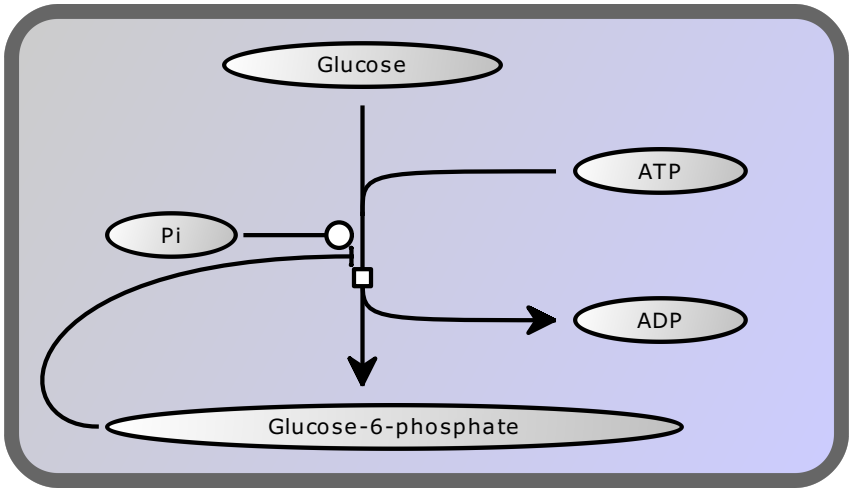

Figure 15: *One possible rendering of the example layout.*

This reaction has a feedback inhibition by glucose-6-phosphate and it is activated by free organic phosphate. These two relations are represented by species reference glyphs and a corresponding role attribute. We did not include any coordinates in the third dimension.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core"
    xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
    level="3" version="1" layout:required="false" >
  <model id="TestModel_with_modifiers" timeUnits="time">
    <listOfUnitDefinitions>
      <unitDefinition id="volume">
        <listOfUnits>
          <unit kind="litre" exponent="1" scale="0" multiplier="1"/>
        </listOfUnits>
      </unitDefinition>
      <unitDefinition id="substance">
        <listOfUnits>
          <unit kind="mole" exponent="1" scale="0" multiplier="1"/>
```

```
      </listOfUnits>
    </unitDefinition>
    <unitDefinition id="time">
      <listOfUnits>
        <unit kind="second" exponent="1" scale="0" multiplier="1"/>
      </listOfUnits>
    </unitDefinition>
  </listOfUnitDefinitions>
  <listOfCompartments>
    <compartment id="Yeast" spatialDimensions="3" units="volume" constant="true"/>
  </listOfCompartments>
  <listOfSpecies>
    <species id="Glucose" compartment="Yeast" substanceUnits="substance"
    hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
    <species id="G6P" name="Glucose-6-phosphate" compartment="Yeast"
    substanceUnits="substance" hasOnlySubstanceUnits="false"
    boundaryCondition="false" constant="false"/>
    <species id="ATP" compartment="Yeast" substanceUnits="substance"
    hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
    <species id="ADP" compartment="Yeast" substanceUnits="substance"
    hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
    <species id="Pi" compartment="Yeast" substanceUnits="substance"
    hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
  </listOfSpecies>
  <listOfReactions>
    <reaction id="Hexokinase" reversible="false" fast="false">
      <listOfReactants>
        <speciesReference id="SpeciesReference_Glucose" species="Glucose"
    stoichiometry="1" constant="true"/>
        <speciesReference id="SpeciesReference_ATP" species="ATP"
    stoichiometry="1" constant="true"/>
      </listOfReactants>
      <listOfProducts>
        <speciesReference id="SpeciesReference_G6P" species="G6P"
    stoichiometry="1" constant="true"/>
        <speciesReference id="SpeciesReference_ADP" species="ADP"
    stoichiometry="1" constant="true"/>
      </listOfProducts>
      <listOfModifiers>
        <modifierSpeciesReference id="ModifierSpeciesReference_G6P" species="G6P"/>
        <modifierSpeciesReference id="ModifierSpeciesReference_Pi" species="Pi"/>
      </listOfModifiers>
    </reaction>
  </listOfReactions>
  <layout:listOfLayouts xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1">
    <layout:layout layout:id="Layout_1">
      <layout:dimensions layout:width="400" layout:height="230"/>
      <layout:listOfCompartmentGlyphs>
        <layout:compartmentGlyph layout:id="CompartmentGlyph_1" layout:compartment="Yeast">
          <layout:boundingBox layout:id="bb1">
            <layout:position layout:x="5" layout:y="5"/>
            <layout:dimensions layout:width="390" layout:height="220"/>
          </layout:boundingBox>
        </layout:compartmentGlyph>
      </layout:listOfCompartmentGlyphs>
      <layout:listOfSpeciesGlyphs>
        <layout:speciesGlyph layout:id="SpeciesGlyph_Glucose"  layout:species="Glucose">
          <layout:boundingBox layout:id="bb2">
            <layout:position layout:x="105" layout:y="20"/>
            <layout:dimensions layout:width="130" layout:height="20"/>
          </layout:boundingBox>
        </layout:speciesGlyph>
        <layout:speciesGlyph layout:id="SpeciesGlyph_G6P" layout:species="G6P">
          <layout:boundingBox layout:id="bb5">
            <layout:position layout:x="50" layout:y="190"/>
```

```
        <layout:dimensions layout:width="270" layout:height="20"/>
      </layout:boundingBox>
    </layout:speciesGlyph>
    <layout:speciesGlyph layout:id="SpeciesGlyph_ATP" layout:species="ATP">
      <layout:boundingBox layout:id="bb3">
        <layout:position layout:x="270" layout:y="70"/>
        <layout:dimensions layout:width="80" layout:height="20"/>
      </layout:boundingBox>
    </layout:speciesGlyph>
    <layout:speciesGlyph layout:id="glyph_ADP" layout:species="ADP">
      <layout:boundingBox layout:id="bb4">
        <layout:position layout:x="270" layout:y="140"/>
        <layout:dimensions layout:width="80" layout:height="20"/>
      </layout:boundingBox>
    </layout:speciesGlyph>
    <layout:speciesGlyph layout:id="SpeciesGlyph_Pi" layout:species="Pi">
      <layout:boundingBox layout:id="bb6">
        <layout:position layout:x="50" layout:y="100"/>
        <layout:dimensions layout:width="60" layout:height="20"/>
      </layout:boundingBox>
    </layout:speciesGlyph>
  </layout:listOfSpeciesGlyphs>
  <layout:listOfReactionGlyphs>
    <layout:reactionGlyph layout:id="glyph_Hexokinase" layout:reaction="Hexokinase">
      <layout:curve>
        <layout:listOfCurveSegments>
          <layout:curveSegment
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="LineSegment">
            <layout:start layout:x="170" layout:y="100"/>
            <layout:end layout:x="170" layout:y="130"/>
          </layout:curveSegment>
        </layout:listOfCurveSegments>
      </layout:curve>
      <layout:listOfSpeciesReferenceGlyphs>
        <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_Glucose"
  layout:speciesReference="SpeciesReference_Glucose"
  layout:speciesGlyph="SpeciesGlyph_Glucose"
  layout:role="substrate">
          <layout:curve>
            <layout:listOfCurveSegments>
              <layout:curveSegment
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="LineSegment">
                <layout:start layout:x="170" layout:y="100"/>
                <layout:end layout:x="170" layout:y="50"/>
              </layout:curveSegment>
            </layout:listOfCurveSegments>
          </layout:curve>
        </layout:speciesReferenceGlyph>
        <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_ATP"
  layout:speciesReference="SpeciesReference_ATP"
  layout:speciesGlyph="SpeciesGlyph_ATP"
  layout:role="sidesubstrate">
          <layout:curve>
            <layout:listOfCurveSegments>
              <layout:curveSegment
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="CubicBezier">
                <layout:start layout:x="170" layout:y="100"/>
                <layout:end layout:x="260" layout:y="80"/>
                <layout:basePoint1 layout:x="170" layout:y="80"/>
                <layout:basePoint2 layout:x="170" layout:y="80"/>
              </layout:curveSegment>
            </layout:listOfCurveSegments>
          </layout:curve>
```

```
      </layout:speciesReferenceGlyph>
      <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_G6P_1"
layout:speciesReference="SpeciesReference_G6P"
layout:speciesGlyph="SpeciesGlyph_G6P"
layout:role="product">
        <layout:curve>
          <layout:listOfCurveSegments>
            <layout:curveSegment
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="LineSegment">
              <layout:start layout:x="170" layout:y="130"/>
              <layout:end layout:x="170" layout:y="180"/>
            </layout:curveSegment>
          </layout:listOfCurveSegments>
        </layout:curve>
      </layout:speciesReferenceGlyph>
      <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_ADP"
layout:speciesReference="SpeciesReference_ADP"
layout:speciesGlyph="glyph_ADP"
layout:role="sideproduct">
        <layout:curve>
          <layout:listOfCurveSegments>
            <layout:curveSegment
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="CubicBezier">
              <layout:start layout:x="170" layout:y="130"/>
              <layout:end layout:x="260" layout:y="150"/>
              <layout:basePoint1 layout:x="170" layout:y="150"/>
              <layout:basePoint2 layout:x="170" layout:y="150"/>
            </layout:curveSegment>
          </layout:listOfCurveSegments>
        </layout:curve>
      </layout:speciesReferenceGlyph>
      <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_G6P_2"
layout:speciesReference="ModifierSpeciesReference_G6P"
layout:speciesGlyph="SpeciesGlyph_G6P"
layout:role="inhibitor">
        <layout:curve>
          <layout:listOfCurveSegments>
            <layout:curveSegment
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="CubicBezier">
              <layout:start layout:x="45" layout:y="200"/>
              <layout:end layout:x="165" layout:y="120"/>
              <layout:basePoint1 layout:x="0" layout:y="200"/>
              <layout:basePoint2 layout:x="0" layout:y="120"/>
            </layout:curveSegment>
          </layout:listOfCurveSegments>
        </layout:curve>
      </layout:speciesReferenceGlyph>
      <layout:speciesReferenceGlyph layout:id="SpeciesReferenceGlyph_PI"
layout:speciesReference="ModifierSpeciesReference_Pi"
layout:speciesGlyph="SpeciesGlyph_Pi"
layout:role="activator">
        <layout:curve>
          <layout:listOfCurveSegments>
            <layout:curveSegment
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="CubicBezier">
              <layout:start layout:x="115" layout:y="110"/>
              <layout:end layout:x="165" layout:y="110"/>
              <layout:basePoint1 layout:x="140" layout:y="110"/>
              <layout:basePoint2 layout:x="140" layout:y="110"/>
            </layout:curveSegment>
          </layout:listOfCurveSegments>
        </layout:curve>
```

```
            </layout:speciesReferenceGlyph>
          </layout:listOfSpeciesReferenceGlyphs>
        </layout:reactionGlyph>
      </layout:listOfReactionGlyphs>
      <layout:listOfTextGlyphs>
        <layout:textGlyph layout:id="TextGlyph_Glucose"
    layout:originOfText="Glucose"
    layout:graphicalObject="SpeciesGlyph_Glucose">
          <layout:boundingBox layout:id="bbA">
            <layout:position layout:x="115" layout:y="20"/>
            <layout:dimensions layout:width="110" layout:height="20"/>
          </layout:boundingBox>
        </layout:textGlyph>
        <layout:textGlyph layout:id="TextGlyph_G6P"
    layout:originOfText="G6P"
    layout:graphicalObject="SpeciesGlyph_G6P">
          <layout:boundingBox layout:id="bbD">
            <layout:position layout:x="60" layout:y="190"/>
            <layout:dimensions layout:width="250" layout:height="20"/>
          </layout:boundingBox>
        </layout:textGlyph>
        <layout:textGlyph layout:id="TextGlyph_ATP"
    layout:originOfText="ATP"
    layout:graphicalObject="SpeciesGlyph_ATP">
          <layout:boundingBox layout:id="bbB">
            <layout:position layout:x="280" layout:y="70"/>
            <layout:dimensions layout:width="60" layout:height="20"/>
          </layout:boundingBox>
        </layout:textGlyph>
        <layout:textGlyph layout:id="TextGlyph_ADP"
    layout:originOfText="ADP"
    layout:graphicalObject="glyph_ADP">
          <layout:boundingBox layout:id="bbC">
            <layout:position layout:x="280" layout:y="140"/>
            <layout:dimensions layout:width="60" layout:height="20"/>
          </layout:boundingBox>
        </layout:textGlyph>
        <layout:textGlyph layout:id="TextGlyph_PI"
    layout:originOfText="Pi"
    layout:graphicalObject="SpeciesGlyph_Pi">
          <layout:boundingBox layout:id="bbE">
            <layout:position layout:x="60" layout:y="100"/>
            <layout:dimensions layout:width="40" layout:height="20"/>
          </layout:boundingBox>
        </layout:textGlyph>
      </layout:listOfTextGlyphs>
    </layout:layout>
  </layout:listOfLayouts>
 </model>
</sbml>
```

4.6 Example using SBML Level 2 Version 1

As mentioned before, the Layout package has been used since 2003 in **SBML** Level 2 documents. Wheras the previous example used the Level 3 package, here Here we repeat the previous example again using the **SBML** annotations.

In order to use the Layout package in **SBML** Level 2 documents, all that is needed is to use the Level 2 Namespace (see Section 3.1) and move the **ListOfLayouts** into the **Model** `annotation` element.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" level="2" version="1">
```

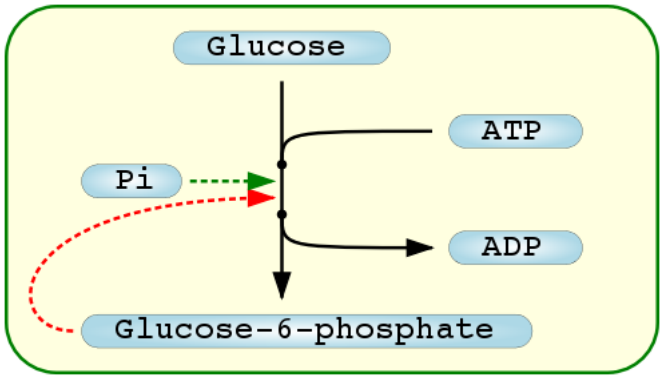

Figure 16: *Another possible rendering of the example layout.*

```
<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>
        <compartmentGlyph id="CompartmentGlyph_1" compartment="Yeast">
          <boundingBox id="bb1">
            <position x="5" y="5"/>
            <dimensions width="390" height="220"/>
          </boundingBox>
        </compartmentGlyph>
      </listOfCompartmentGlyphs>
      <listOfSpeciesGlyphs>
        <speciesGlyph id="SpeciesGlyph_Glucose" species="Glucose">
          <boundingBox id="bb2">
            <position x="105" y="20"/>
            <dimensions width="130" height="20"/>
          </boundingBox>
        </speciesGlyph>
        <speciesGlyph id="SpeciesGlyph_G6P" species="G6P">
          <boundingBox id="bb5">
            <position x="50" y="190"/>
            <dimensions width="270" height="20"/>
          </boundingBox>
        </speciesGlyph>
        <speciesGlyph id="SpeciesGlyph_ATP" species="ATP">
          <boundingBox id="bb3">
            <position x="270" y="70"/>
            <dimensions width="80" height="20"/>
          </boundingBox>
        </speciesGlyph>
        <speciesGlyph id="glyph_ADP" species="ADP">
          <boundingBox id="bb4">
            <position x="270" y="140"/>
            <dimensions width="80" height="20"/>
          </boundingBox>
        </speciesGlyph>
        <speciesGlyph id="SpeciesGlyph_Pi" species="Pi">
          <boundingBox id="bb6">
            <position x="50" y="100"/>
            <dimensions width="60" height="20"/>
          </boundingBox>
        </speciesGlyph>
      </listOfSpeciesGlyphs>
      <listOfReactionGlyphs>
```

```
          <reactionGlyph id="glyph_Hexokinase" reaction="Hexokinase">
            <curve>
              <listOfCurveSegments>
                <curveSegment xsi:type="LineSegment">
                  <start x="170" y="100"/>
                  <end x="170" y="130"/>
                </curveSegment>
              </listOfCurveSegments>
            </curve>
            <listOfSpeciesReferenceGlyphs>
              <speciesReferenceGlyph id="SpeciesReferenceGlyph_Glucose"
                            speciesReference="SpeciesReference_Glucose"
                            speciesGlyph="SpeciesGlyph_Glucose" role="substrate">
                <curve>
                  <listOfCurveSegments>
                    <curveSegment xsi:type="LineSegment">
                      <start x="170" y="100"/>
                      <end x="170" y="50"/>
                    </curveSegment>
                  </listOfCurveSegments>
                </curve>
              </speciesReferenceGlyph>
              <speciesReferenceGlyph id="SpeciesReferenceGlyph_ATP"
                            speciesReference="SpeciesReference_ATP"
                            speciesGlyph="SpeciesGlyph_ATP" role="sidesubstrate">
                <curve>
                  <listOfCurveSegments>
                    <curveSegment xsi:type="CubicBezier">
                      <start x="170" y="100"/>
                      <end x="260" y="80"/>
                      <basePoint1 x="170" y="80"/>
                      <basePoint2 x="170" y="80"/>
                    </curveSegment>
                  </listOfCurveSegments>
                </curve>
              </speciesReferenceGlyph>
              <speciesReferenceGlyph id="SpeciesReferenceGlyph_G6P_1"
                              speciesReference="SpeciesReference_G6P"
                              speciesGlyph="SpeciesGlyph_G6P" role="product">
                <curve>
                  <listOfCurveSegments>
                    <curveSegment xsi:type="LineSegment">
                      <start x="170" y="130"/>
                      <end x="170" y="180"/>
                    </curveSegment>
                  </listOfCurveSegments>
                </curve>
              </speciesReferenceGlyph>
              <speciesReferenceGlyph id="SpeciesReferenceGlyph_ADP"
                            speciesReference="SpeciesReference_ADP"
                            speciesGlyph="glyph_ADP" role="sideproduct">
                <curve>
                  <listOfCurveSegments>
                    <curveSegment xsi:type="CubicBezier">
                      <start x="170" y="130"/>
                      <end x="260" y="150"/>
                      <basePoint1 x="170" y="150"/>
                      <basePoint2 x="170" y="150"/>
                    </curveSegment>
                  </listOfCurveSegments>
                </curve>
              </speciesReferenceGlyph>
              <speciesReferenceGlyph id="SpeciesReferenceGlyph_G6P_2"
                      speciesReference="ModifierSpeciesReference_G6P"
                      speciesGlyph="SpeciesGlyph_G6P" role="inhibitor">
                <curve>
```

```
                <listOfCurveSegments>
                  <curveSegment xsi:type="CubicBezier">
                    <start x="45" y="200"/>
                    <end x="165" y="120"/>
                    <basePoint1 x="0" y="200"/>
                    <basePoint2 x="0" y="120"/>
                  </curveSegment>
                </listOfCurveSegments>
              </curve>
            </speciesReferenceGlyph>
            <speciesReferenceGlyph id="SpeciesReferenceGlyph_PI"
                  speciesReference="ModifierSpeciesReference_Pi"
                  speciesGlyph="SpeciesGlyph_Pi" role="activator">
              <curve>
                <listOfCurveSegments>
                  <curveSegment xsi:type="CubicBezier">
                    <start x="115" y="110"/>
                    <end x="165" y="110"/>
                    <basePoint1 x="140" y="110"/>
                    <basePoint2 x="140" y="110"/>
                  </curveSegment>
                </listOfCurveSegments>
              </curve>
            </speciesReferenceGlyph>
          </listOfSpeciesReferenceGlyphs>
        </reactionGlyph>
      </listOfReactionGlyphs>
      <listOfTextGlyphs>
        <textGlyph id="TextGlyph_Glucose" graphicalObject="SpeciesGlyph_Glucose"
                   originOfText="Glucose">
          <boundingBox id="bbA">
            <position x="115" y="20"/>
            <dimensions width="110" height="20"/>
          </boundingBox>
        </textGlyph>
        <textGlyph id="TextGlyph_G6P" graphicalObject="SpeciesGlyph_G6P"
                   originOfText="G6P">
          <boundingBox id="bbD">
            <position x="60" y="190"/>
            <dimensions width="250" height="20"/>
          </boundingBox>
        </textGlyph>
        <textGlyph id="TextGlyph_ATP" graphicalObject="SpeciesGlyph_ATP"
                   originOfText="ATP">
          <boundingBox id="bbB">
            <position x="280" y="70"/>
            <dimensions width="60" height="20"/>
          </boundingBox>
        </textGlyph>
        <textGlyph id="TextGlyph_ADP" graphicalObject="glyph_ADP"
                   originOfText="ADP">
          <boundingBox id="bbC">
            <position x="280" y="140"/>
            <dimensions width="60" height="20"/>
          </boundingBox>
        </textGlyph>
        <textGlyph id="TextGlyph_PI" graphicalObject="SpeciesGlyph_Pi"
                   originOfText="Pi">
          <boundingBox id="bbE">
            <position x="60" y="100"/>
            <dimensions width="40" height="20"/>
          </boundingBox>
        </textGlyph>
      </listOfTextGlyphs>
    </layout>
  </listOfLayouts>
```

```
    </annotation>
    <listOfCompartments>
      <compartment id="Yeast"/>
    </listOfCompartments>
    <listOfSpecies>
      <species id="Glucose" compartment="Yeast"/>
      <species id="G6P" name="Glucose-6-phosphate" compartment="Yeast"/>
      <species id="ATP" compartment="Yeast"/>
      <species id="ADP" compartment="Yeast"/>
      <species id="Pi" compartment="Yeast"/>
    </listOfSpecies>
    <listOfReactions>
      <reaction id="Hexokinase" reversible="false">
        <listOfReactants>
          <speciesReference species="Glucose">
            <annotation>
              <layoutId xmlns="http://projects.eml.org/bcb/sbml/level2"
                        id="SpeciesReference_Glucose"/>
            </annotation>
          </speciesReference>
          <speciesReference species="ATP">
            <annotation>
              <layoutId xmlns="http://projects.eml.org/bcb/sbml/level2"
                        id="SpeciesReference_ATP"/>
            </annotation>
          </speciesReference>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="G6P">
            <annotation>
              <layoutId xmlns="http://projects.eml.org/bcb/sbml/level2"
                        id="SpeciesReference_G6P"/>
            </annotation>
          </speciesReference>
          <speciesReference species="ADP">
            <annotation>
              <layoutId xmlns="http://projects.eml.org/bcb/sbml/level2"
                        id="SpeciesReference_ADP"/>
            </annotation>
          </speciesReference>
        </listOfProducts>
        <listOfModifiers>
          <modifierSpeciesReference species="G6P">
            <annotation>
              <layoutId xmlns="http://projects.eml.org/bcb/sbml/level2"
                        id="ModifierSpeciesReference_G6P"/>
            </annotation>
          </modifierSpeciesReference>
          <modifierSpeciesReference species="Pi">
            <annotation>
              <layoutId xmlns="http://projects.eml.org/bcb/sbml/level2"
                        id="ModifierSpeciesReference_Pi"/>
            </annotation>
          </modifierSpeciesReference>
        </listOfModifiers>
      </reaction>
    </listOfReactions>
  </model>
</sbml>
```

4.7 Example using the GeneralGlyph

Here a small complete example that shows how to draw a basic influence diagram, where `Node0` negatively influences `Node1`.

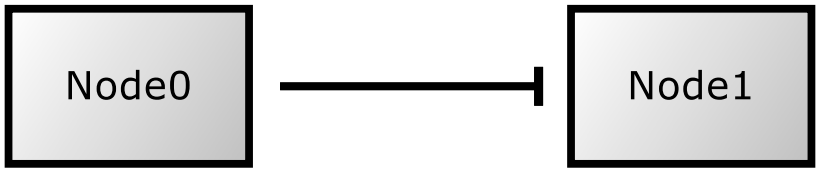

Figure 17: *One possible rendering of an influence diagram.*

In this example we have a simple SBML model that contains two species, `Node0` and `Node1`. For both of them there are species glyph elements and associated Text labels. A **GeneralGlyph** then represents the influence.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core"
      xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1"
    level="3" version="1" layout:required="false" >
  <model id="GeneralGlyphExample" >

  <listOfCompartments>
      <compartment id="compartment"
    spatialDimensions="3" size="1" constant="true"/>
    </listOfCompartments>

  <listOfSpecies>
      <species sboTerm="SBO:0000395" id="Node0"
      compartment="compartment" initialConcentration="0"
      hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
      <species  sboTerm="SBO:0000395" id="Node1"
      compartment="compartment" initialConcentration="0"
      hasOnlySubstanceUnits="false" boundaryCondition="false" constant="false"/>
    </listOfSpecies>

    <layout:listOfLayouts
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:layout="http://www.sbml.org/sbml/level3/version1/layout/version1">
      <layout:layout layout:id="example">
        <layout:dimensions layout:width="239.037328720093" layout:height="76.5"/>

    <layout:listOfSpeciesGlyphs>
          <layout:speciesGlyph layout:id="sGlyph_0" layout:species="Node0" >
            <layout:boundingBox>
              <layout:position layout:x="16" layout:y="18"/>
              <layout:dimensions layout:width="62" layout:height="40"/>
            </layout:boundingBox>
          </layout:speciesGlyph>
          <layout:speciesGlyph layout:id="sGlyph_1" layout:species="Node1" >
            <layout:boundingBox>
              <layout:position layout:x="161" layout:y="18"/>
              <layout:dimensions layout:width="62" layout:height="40"/>
            </layout:boundingBox>
          </layout:speciesGlyph>
        </layout:listOfSpeciesGlyphs>

    <layout:listOfAdditionalGraphicalObjects>
          <layout:generalGlyph layout:id="rGlyph_0" reference="sGlyph_0">
        <!-- unused bounding box -->
            <layout:boundingBox>
              <layout:position layout:x="0" layout:y="0"/>
              <layout:dimensions layout:width="0" layout:height="0"/>
            </layout:boundingBox>
            <layout:listOfReferenceGlyphs>
```

```
        <!-- reference glyph representing the inhibited species -->
              <layout:referenceGlyph
            layout:id="SpeciesReference_J0_0"
          layout:reference="Node1"
          layout:glyph="sGlyph_1"
          layout:role="inhibitor">

                <layout:curve>
                  <layout:listOfCurveSegments>
                    <layout:curveSegment
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="LineSegment">
                      <layout:start layout:x="86" layout:y="38"/>
                      <layout:end layout:x="153" layout:y="38"/>
                    </layout:curveSegment>
                  </layout:listOfCurveSegments>
                </layout:curve>

              </layout:referenceGlyph>
            </layout:listOfReferenceGlyphs>
          </layout:generalGlyph>
        </layout:listOfAdditionalGraphicalObjects>

    <layout:listOfTextGlyphs>

          <layout:textGlyph layout:id="tGlyph_0" layout:text="Node0"
           layout:graphicalObject="sGlyph_0">
            <layout:boundingBox>
              <layout:position layout:x="16" layout:y="18"/>
              <layout:dimensions layout:width="62" layout:height="40"/>
            </layout:boundingBox>
          </layout:textGlyph>

          <layout:textGlyph layout:id="tGlyph_1" layout:text="Node1"
                        layout:graphicalObject="sGlyph_1">
            <layout:boundingBox>
              <layout:position layout:x="161" layout:y="18"/>
              <layout:dimensions layout:width="62" layout:height="40"/>
            </layout:boundingBox>
          </layout:textGlyph>

        </layout:listOfTextGlyphs>
      </layout:layout>
    </layout:listOfLayouts>
  </model>
</sbml>
```

A Validation of SBML documents

A.1 Validation and consistency rules

This section summarizes all the conditions that must (or in some cases, at least *should*) be true of an SBML Level 3 Version 1 model that uses the Layout package. We use the same conventions as are used in the SBML Level 3 Version 1 Core specification document. In particular, there are different degrees of rule strictness. Formally, the differences are expressed in the statement of a rule: either a rule states that a condition *must* be true, or a rule states that it *should* be true. Rules of the former kind are strict SBML validation rules—a model encoded in SBML must conform to all of them in order to be considered valid. Rules of the latter kind are consistency rules. To help highlight these differences, we use the following three symbols next to the rule numbers:

☑ A checked box indicates a *requirement* for SBML conformance. If a model does not follow this rule, it does not conform to the Layout specification. (Mnemonic intention behind the choice of symbol: "This must be checked.")

▲ A triangle indicates a *recommendation* for model consistency. If a model does not follow this rule, it is not considered strictly invalid as far as the Layout specification is concerned; however, it indicates that the model contains a physical or conceptual inconsistency. (Mnemonic intention behind the choice of symbol: "This is a cause for warning.")

★ A star indicates a strong recommendation for good modeling practice. This rule is not strictly a matter of SBML encoding, but the recommendation comes from logical reasoning. As in the previous case, if a model does not follow this rule, it is not strictly considered an invalid SBML encoding. (Mnemonic intention behind the choice of symbol: "You're a star if you heed this.")

The validation rules listed in the following subsections are all stated or implied in the rest of this specification document. They are enumerated here for convenience. Unless explicitly stated, all validation rules concern objects and attributes specifically defined in the Layout package.

☞ For convenience and brevity, we use the shorthand "`layout:x`" to stand for an attribute or element name `x` in the namespace for the Layout package, using the namespace prefix `layout`. In reality, the prefix string may be different from the literal "`layout`" used here (and indeed, it can be any valid XML namespace prefix that the modeler or software chooses). We use "`layout:x`" because it is shorter than to write a full explanation everywhere we refer to an attribute or element in the Layout package namespace.

General rules about this package

layout-10101 ☑ To conform to the Layout package specification for SBML Level 3 Version 1, an SBML document must declare the use of the following XML Namespace: "`http://www.sbml.org/sbml/level3/version1/layout/version1`". (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.1 on page 7.)

layout-10102 ☑ Wherever they appear in an SBML document, elements and attributes from the Layout package must be declared either implicitly or explicitly to be in the XML namespace "`http://www.sbml.org/sbml/level3/version1/layout/version1`". (References: SBML Level 3 Package Specification for Layout , Version 1, Section 3.1 on page 7.)

General rules about identifiers

layout-10301 ☑ (Extends validation rule #10301 in the SBML Level 3 Version 1 Core specification.) Within a **Model** the values of the attributes `id` and `layout:id` on every instance of the following classes of objects must be unique across the set of all `id` and `layout:id` attribute values of all such objects in a model: the **Model** itself, plus all contained **FunctionDefinition**, **Compartment**, **Species**, **Reaction**, **SpeciesReference**, **ModifierSpeciesReference**, **Event**, and **Parame-**

ter objects, plus the **BoundingBox**, **CompartmentGlyph**, **GeneralGlyph**, **GraphicalObject**, **Layout**, **SpeciesGlyph**, **SpeciesReferenceGlyph**, **ReactionGlyph**, **ReferenceGlyph**, and **TextGlyph** objects defined by the Layout package. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.3 on page 8.)

Rules for the extended SBML class

layout-20101 ☑ In all SBML documents using the Layout package, the **SBML** object must include a value for the attribute `layout:required` attribute. (References: SBML Level 3 Version 1 Core, Section 4.1.2.)

layout-20102 ☑ The value of attribute `layout:required` on the **SBML** object must be of the data type `boolean`. (References: SBML Level 3 Version 1 Core, Section 4.1.2.)

layout-20103 ☑ The value of attribute `layout:required` on the **SBML** object must be set to "`false`". (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.1 on page 7.)

Rules for extended Model object

layout-20201 ☑ There may be at most one instance of **ListOfLayouts** element within a **Model** object using Layout. No other elements from the Layout package are allowed. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.5 on page 12.)

layout-20202 ☑ The **ListOfLayouts** within a **Model** object is optional, but if present, these it object must not be empty. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.5 on page 12.)

layout-20203 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfLayouts** container object may only contain **Layout** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.5 on page 12.)

layout-20204 ☑ A **ListOfLayouts** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfLayouts** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.5 on page 12.)

Rules for the Layout object

layout-20301 ☑ A Layout object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a Layout. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20302 ☑ A Layout object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a Layout. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20303 ☑ There may be at most one instance of each of the following kinds of objects within a Layout object: **Dimensions**, **ListOfCompartmentGlyphs**, **ListOfSpeciesGlyphs**, **ListOfReactionGlyphs**, **ListOfTextGlyphs**, **ListOfAdditionalGraphicalObjects**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 12.)

layout-20304 ☑ The various **ListOf**___ subobjects within a Layout object are optional, but if present, these container object must not be empty. Specifically, if any of the following classes of objects are present on the Layout, it must not be empty: **ListOfCompartmentGlyphs**, **ListOfSpeciesGlyphs**, **ListOfReactionGlyphs**, **ListOfTextGlyphs**, **ListOfAdditionalGraphicalObjects**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 12.)

layout-20305 ☑ A Layout object must have the required attribute `layout:id` and may have the optional attribute `layout:name`. No other attributes from the SBML Level 3 Layout namespace are permitted on a Layout object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 12.)

layout-20306 ☑ The attribute `layout:name` of a Layout must be of the data type `string`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 12.)

layout-20307 ☑ A **ListOfCompartmentGlyphs** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfCompartmentGlyphs** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 13.)

layout-20308 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfCompartmentGlyphs** container object may only contain **CompartmentGlyph** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 13.)

layout-20309 ☑ A **ListOfSpeciesGlyphs** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfSpeciesGlyphs** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 14.)

layout-20310 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfSpeciesGlyphs** container object may only contain **SpeciesGlyph** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 14.)

layout-20311 ☑ A **ListOfReactionGlyphs** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfReactionGlyphs** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 14.)

layout-20312 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfReactionGlyphs** container object may only contain **ReactionGlyph** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 14.)

layout-20313 ☑ A **ListOfAdditionalGraphicalObjects** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfAdditionalGraphicalObjects** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 14.)

layout-20314 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfAdditionalGraphicalObjects** container object may only contain **GeneralGlyph** and **GraphicalObject** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 14.)

layout-20315 ☑ A Layout object must contain exactly one **Dimensions** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.6 on page 12.)

Rules for the GraphicalObject class

layout-20401 ☑ A **GraphicalObject** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **GraphicalObject**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20402 ☑ A **GraphicalObject** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **GraphicalObject**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20403 ☑ There may be at most one instance of an **BoundingBox** object on a **GraphicalObject**, no other elements from the Layout namespace are permitted on a **GraphicalObject**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20404 ☑ A **GraphicalObject** object must have the required attribute `layout:id` and may have the optional attribute `layout:metaidRef`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **GraphicalObject** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20405 ☑ The attribute `layout:metaidRef` of a **GraphicalObject** must be of the data type `IDREF`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20406 ☑ The value of a `layout:metaidRef` of a **GraphicalObject** must be of the `metaid` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20407 ☑ A **GraphicalObject** object must contain exactly one **BoundingBox** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

Rules for the CompartmentGlyph object

layout-20501 ☑ A **CompartmentGlyph** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **CompartmentGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20502 ☑ A **CompartmentGlyph** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **CompartmentGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20503 ☑ There may be at most one instance of an **BoundingBox** object on a **CompartmentGlyph**, no other elements from the Layout namespace are permitted on a **CompartmentGlyph**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.8 on page 15.)

layout-20504 ☑ A **CompartmentGlyph** object must have the required attribute `layout:id` and may have the optional attributes `layout:metaidRef`, `layout:compartment` or `layout:order`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **CompartmentGlyph** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.8 on page 15.)

layout-20505 ☑ The attribute `layout:metaidRef` of a **CompartmentGlyph** must be of the data type `IDREF`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20506 ☑ The value of a `layout:metaidRef` of a **CompartmentGlyph** must be of the `metaid` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20507 ☑ The attribute `layout:compartment` of a **CompartmentGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.8 on page 15.)

layout-20508 ☑ The value of a `layout:compartment` of a **CompartmentGlyph** must be of the `id` of an existing **Compartment** in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.8 on page 15.)

layout-20509 ☑ If both attributes `layout:compartment` and `layout:metaidRef` are specified on a **CompartmentGlyph** they have to reference the same **Compartment** of the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.8 on page 15.)

layout-20510 ☑ The attribute `layout:order` of a **CompartmentGlyph** must be of the data type `double`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.8 on page 15.)

Rules for the SpeciesGlyph object

layout-20601 ☑ A **SpeciesGlyph** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **SpeciesGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20602 ☑ A **SpeciesGlyph** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **SpeciesGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20603 ☑ There may be at most one instance of an **BoundingBox** object on a **SpeciesGlyph**, no other elements from the Layout namespace are permitted on a **SpeciesGlyph**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.9 on page 16.)

layout-20604 ☑ A **SpeciesGlyph** object must have the required attribute `layout:id` and may have the optional attribute `layout:metaidRef` or `layout:species`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **SpeciesGlyph** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.9 on page 16.)

layout-20605 ☑ The attribute `layout:metaidRef` of a **SpeciesGlyph** must be of the data type `IDREF`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20606 ☑ The value of a `layout:metaidRef` of a **SpeciesGlyph** must be of the `metaid` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20607 ☑ The attribute `layout:species` of a **SpeciesGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.9 on page 16.)

layout-20608 ☑ The value of a `layout:species` of a **SpeciesGlyph** must be of the `id` of an existing **Species** in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.9 on page 16.)

layout-20609 ☑ If both attributes `layout:species` and `layout:metaidRef` are specified on a **SpeciesGlyph** they have to reference the same **Species** of the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.9 on page 16.)

Rules for the ReactionGlyph object

layout-20701 ☑ A **ReactionGlyph** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **ReactionGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20702 ☑ A **ReactionGlyph** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **ReactionGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20703 ☑ There may be at most one instance of each of the following kinds of objects within a **ReactionGlyph** object: **BoundingBox**, **Curve**, **ListOfSpeciesReferenceGlyphs**, no other elements from the Layout namespace are permitted on a **ReactionGlyph**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10 on page 16.)

layout-20704 ☑ A **ReactionGlyph** object must have the required attribute `layout:id` and may have the optional attribute `layout:metaidRef` or `layout:reaction`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **ReactionGlyph** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10 on page 16.)

layout-20705 ☑ The attribute `layout:metaidRef` of a **ReactionGlyph** must be of the data type `IDREF`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20706 ☑ The value of a `layout:metaidRef` of a **ReactionGlyph** must be of the `metaid` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20707 ☑ The attribute `layout:reaction` of a **ReactionGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10 on page 16.)

layout-20708 ☑ The value of a `layout:reaction` of a **ReactionGlyph** must be of the `id` of an existing **Reaction** in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10 on page 16.)

layout-20709 ☑ If both attributes `layout:reaction` and `layout:metaidRef` are specified on a **ReactionGlyph** they have to reference the same **Reaction** of the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10 on page 16.)

layout-20710 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfSpeciesReferenceGlyphs** container object may only contain **SpeciesReferenceGlyph** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10 on page 16.)

layout-20711 ☑ A **ListOfSpeciesReferenceGlyphs** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfSpeciesReferenceGlyphs** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10 on page 16.)

Rules for the GeneralGlyph object

layout-20801 ☑ A **GeneralGlyph** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **GeneralGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20802 ☑ A **GeneralGlyph** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **GeneralGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20803 ☑ There may be at most one instance of each of the following kinds of objects within a **GeneralGlyph** object: **BoundingBox**, **Curve**, **ListOfReferenceGlyphs** and **ListOfSubGlyphs**, no other elements from the Layout namespace are permitted on a **GeneralGlyph**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

layout-20804 ☑ A **GeneralGlyph** object must have the required attribute `layout:id` and may have the optional attribute `layout:metaidRef` or `layout:reference`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **GeneralGlyph** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

layout-20805 ☑ The attribute `layout:metaidRef` of a **GeneralGlyph** must be of the data type `IDREF`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20806 ☑ The value of a `layout:metaidRef` of a **GeneralGlyph** must be of the `metaid` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20807 ☑ The attribute `layout:reference` of a **GeneralGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

layout-20708 ☑ The value of a `layout:reference` of a **GeneralGlyph** must be of the `id` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

layout-20809 ☑ If both attributes `layout:reference` and `layout:metaidRef` are specified on a **GeneralGlyph** they have to reference the same element of the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

layout-20810 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfReferenceGlyphs** container object may only contain **ReferenceGlyph** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

layout-20811 ☑ A **ListOfReferenceGlyphs** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfReferenceGlyphs** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

layout-20812 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfSubGlyphs** container object may only contain **CompartmentGlyph**, **SpeciesGlyph**, **ReactionGlyph**, **GeneralGlyph**, **GraphicalObject**, **TextGlyph**, **SpeciesReferenceGlyph** and **ReferenceGlyph** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

layout-20813 ☑ A **ListOfSubGlyphs** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfSubGlyphs** object.

(References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11 on page 18.)

Rules for the TextGlyph object

layout-20901 ☑ A **TextGlyph** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **TextGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20902 ☑ A **TextGlyph** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **TextGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-20903 ☑ A **TextGlyph** object must contain exactly one **BoundingBox** object, no other elements from the Layout namespace are permitted on a **TextGlyph**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.12 on page 20.)

layout-20904 ☑ A **TextGlyph** object must have the required attribute `layout:id` and may have the optional attributes `layout:metaidRef`, `layout:graphicalObject`, `layout:text` and `layout:originOfText`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **TextGlyph** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.12 on page 20.)

layout-20905 ☑ The attribute `layout:metaidRef` of a **TextGlyph** must be of the data type `IDREF`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20906 ☑ The value of a `layout:metaidRef` of a **TextGlyph** must be of the `metaid` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-20907 ☑ The attribute `layout:originOfText` of a **TextGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.12 on page 20.)

layout-20908 ☑ The value of a `layout:originOfText` of a **TextGlyph** must be of the `id` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.12 on page 20.)

layout-20909 ☑ If both attributes `layout:originOfText` and `layout:metaidRef` are specified on a TextGlyph they have to reference the same element of the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.12 on page 20.)

layout-20910 ☑ The attribute `layout:graphicalObject` of a **TextGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.12 on page 20.)

layout-20911 ☑ The value of a `layout:graphicalObject` of a **TextGlyph** must be of the `id` of an existing **GraphicalObject** (or derived) element in the **Layout**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.12 on page 20.)

layout-20912 ☑ The attribute `layout:text` of a **TextGlyph** must be of the data type `string`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.12 on page 20.)

Rules for the SpeciesReferenceGlyph object

layout-21001 ☑ A **SpeciesReferenceGlyph** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **SpeciesReferenceGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21002 ☑ A **SpeciesReferenceGlyph** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **SpeciesReferenceGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21003 ☑ A **SpeciesReferenceGlyph** may have at most one instance of a **BoundingBox** and **Curve** object, no other elements from the Layout namespace are permitted on a **SpeciesReferenceGlyph**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

layout-21004 ☑ A **SpeciesReferenceGlyph** object must have the required attributes `layout:id` and `layout:speciesGlyph` and may have the optional attribute `layout:metaidRef` or `layout:speciesReference` and `layout:role`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **SpeciesReferenceGlyph** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

layout-21005 ☑ The attribute `layout:metaidRef` of a **SpeciesReferenceGlyph** must be of the data type `IDREF`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-21006 ☑ The value of a `layout:metaidRef` of a **SpeciesReferenceGlyph** must be of the `metaid` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-21007 ☑ The attribute `layout:speciesReference` of a **SpeciesReferenceGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

layout-21008 ☑ The value of a `layout:speciesReference` of a **SpeciesReferenceGlyph** must be of the `id` of an existing **SpeciesReference** in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

layout-21009 ☑ If both attributes `layout:speciesReference` and `layout:metaidRef` are specified on a **SpeciesReferenceGlyph** they have to reference the same **SpeciesReference** of the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

layout-21010 ☑ The attribute `layout:speciesGlyph` of a **SpeciesReferenceGlyph** must be of the data type `SIdRef`.
(References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

layout-21011 ☑ The value of a `layout:speciesGlyph` of a **SpeciesReferenceGlyph** must be of the `id` of an existing **SpeciesGlyph** in the **Layout**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

layout-21012 ☑ The attribute `layout:role` of a **SpeciesReferenceGlyph** must be of the data type `SpeciesReferenceRole`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

layout-21013 ☑ The value of a `layout:role` of a **SpeciesReferenceGlyph** must be one of: `substrate`, `product`, `sidesubstrate`, `sideproduct`, `modifier`, `activator`, `inhibitor` or `undefined`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.10.1 on page 17.)

Rules for the ReferenceGlyph object

layout-21101 ☑ A **ReferenceGlyph** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **ReferenceGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21102 ☑ A **ReferenceGlyph** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **ReferenceGlyph**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21103 ☑ There may be at most one instance of an **BoundingBox** and **Curve** object on a **ReferenceGlyph**, no other elements from the Layout namespace are permitted on a **ReferenceGlyph**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11.1 on page 20.)

layout-21104 ☑ A **ReferenceGlyph** object must have the required attributes `layout:id` and `layout:glyph` and may have the optional attribute `layout:metaidRef` or `layout:reference` and `layout:role`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **ReferenceGlyph** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11.1 on page 20.)

layout-21105 ☑ The attribute `layout:metaidRef` of a **ReferenceGlyph** must be of the data type `IDREF`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-21106 ☑ The value of a `layout:metaidRef` of a **ReferenceGlyph** must be of the `metaid` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

layout-21107 ☑ The attribute `layout:reference` of a **ReferenceGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11.1 on page 20.)

layout-21108 ☑ The value of a `layout:reference` of a **ReferenceGlyph** must be of the `id` of an existing element in the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11.1 on page 20.)

layout-21109 ☑ If both attributes `layout:reference` and `layout:metaidRef` are specified on a **ReferenceGlyph** they have to reference the same element of the **Model**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11.1 on page 20.)

layout-21110 ☑ The attribute `layout:glyph` of a **ReferenceGlyph** must be of the data type `SIdRef`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11.1 on page 20.)

layout-21111 ☑ The value of a `layout:glyph` of a **ReferenceGlyph** must be of the `id` of an existing **GraphicalObject** (or derived) element in the **Layout**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11.1 on page 20.)

layout-21112 ☑ The attribute `layout:role` of a **ReferenceGlyph** must be of the data type `string`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.11.1 on page 20.)

Rules for the Point class

layout-21201 ☑ A **Point** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core or Layout namespace are permitted on a **Point** . (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21202 ☑ A **Point** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **Point** . (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21203 ☑ A **Point** object must have the required attributes `layout:x` and `layout:y` and may have the optional attributes `layout:id` and `layout:z`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **Point** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.1 on page 8.)

layout-21204 ☑ The attributes `layout:x`, `layout:y` and `layout:z` of a **Point** element must be of the data type `double`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.1 on page 8.)

Rules for the Dimensions class

layout-21201 ☑ A **Dimensions** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core or Layout namespace are permitted on a **Dimensions**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21202 ☑ A **Dimensions** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **Dimensions**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21203 ☑ A **Dimensions** object must have the required attributes `layout:width` and `layout:height` and may have the optional attributes `layout:id` and `layout:depth`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **Dimensions** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.2 on page 8.)

layout-21204 ☑ The attributes `layout:width`, `layout:height` and `layout:depth` of a **Dimensions** object must be of the data type `double`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.2 on page 8.)

Rules for the BoundingBox class

layout-21301 ☑ A **BoundingBox** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **BoundingBox**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21302 ☑ A **BoundingBox** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **BoundingBox**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21303 ☑ There must be exactly one instance of a **Point** and a **Dimensions** object on a **BoundingBox**, no other elements from the Layout namespace are permitted on a **BoundingBox**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.3 on page 9.)

layout-21304 ☑ A **BoundingBox** object must have the optional attribute `layout:id`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **BoundingBox** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.3 on page 9.)

layout-21305 ☑ If the `layout:z` attribute of a **Point** element on a **BoundingBox** is not specified, the attribute `layout:depth` of its **Dimensions** object must also not be specified. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.7 on page 14.)

Rules for the Curve class

layout-21401 ☑ A **Curve** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **Curve**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21402 ☑ A **Curve** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **Curve**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21403 ☑ There must be exactly one instance of a **ListOfCurveSegments** object on a **Curve**, no other elements from the Layout namespace are permitted on a **Curve**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.4 on page 9.)

layout-21407 ☑ A **ListOfCurveSegments** object may have the optional `metaid` and `sboTerm` defined by SBML Level 3 Core. No other attributes from the SBML Level 3 Core namespace or the Layout namespace are permitted on a **ListOfCurveSegments** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.4 on page 10.)

layout-21408 ☑ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfCurveSegments** container object may only contain **LineSegment** and **CubicBezier** objects. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.4 on page 10.)

layout-21409 ☑ The **ListOfCurveSegments** container object may not be emtpy. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.4 on page 10.)

Rules for the LineSegment class

layout-21501 ☑ A **LineSegment** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **LineSegment**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21502 ☑ A **LineSegment** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **LineSegment**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21503 ☑ A **LineSegment** must specify the two **Point** elements `start` and `end`. No other elements from the Layout namespace are permitted on a **LineSegment**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.5 on page 10.)

layout-21504 ☑ A **LineSegment** object must have the required attribute `xsi:type`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **LineSegment** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.5 on page 10.)

layout-21505 ☑ The attribute `xsi:type` of a **LineSegment** must be of the data type `string`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.5 on page 10.)

layout-21506 ☑ The value of a `xsi:type` of a **LineSegment** must be `LineSegment`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.5 on page 10.)

Rules for the CubicBezier class

layout-21601 ☑ A **CubicBezier** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespace are permitted on a **CubicBezier**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21602 ☑ A **CubicBezier** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespace are permitted on a **CubicBezier**. (References: SBML Level 3 Version 1 Core, Section 3.2.)

layout-21603 ☑ A **CubicBezier** must specify the four **Point** elements `start`, `basePoint1`, `basePoint2` and `end`. No other elements from the Layout namespace are permitted on a **CubicBezier**. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.6 on page 10.)

layout-21604 ☑ A **CubicBezier** object must have the required attribute `xsi:type`. No other attributes from the SBML Level 3 Layout namespace are permitted on a **CubicBezier** object. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.6 on page 10.)

layout-21605 ☑ The attribute `xsi:type` of a **CubicBezier** must be of the data type `string`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.6 on page 10.)

layout-21606 ☑ The value of a `xsi:type` of a **CubicBezier** must be `CubicBezier`. (References: SBML Level 3 Package Specification for Layout, Version 1, Section 3.4.6 on page 10.)

B Acknowledgments

We would like to thank the following people who contributed in various ways to the development of both the original proposal and this document.

For financial/travel/technical/moral and general support we thank especially Ursula Kummer (Heidelberg University, Germany), Michael Hucka (CalTech, USA) and Herbert Sauro (University of Washington, USA).

A special work of thanks goes to Sarah Keating for her invaluable work with libSBML and help with the specification document and critical reading of this document.

We also would like to thank Andreas Dräger, Lucian P. Smith, Chris J. Myers, Nicolas Rodriguez, Nicolas Le Novère, the Layout PWG and all those who contributed ideas and held discussions with us on various occasions.

References

Bergmann, F. T. and Sauro, H. M. (2006). SBW - a modular framework for systems biology. In *Proceedings of the 38th conference on Winter simulation*, WSC '06, pages 1637–1645. Winter Simulation Conference.

Biron, P. V. and Malhotra, A. (2000). XML Schema part 2: Datatypes (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at http://www.w3.org/TR/xmlschema-2/.

Deckard, A., Bergmann, F. T., and Sauro, H. M. (2006). Supporting the sbml layout extension. *Bioinformatics*, 22(23):2966–2967.

Eriksson, H.-E. and Penker, M. (1998). *UML Toolkit*. John Wiley & Sons, New York.

Fallside, D. C. (2000). XML Schema part 0: Primer (W3C candidate recommendation 24 October 2000). Available via the World Wide Web at http://www.w3.org/TR/xmlschema-0/.

Gauges, R., Rost, U., Sahle, S., and Wegner, K. (2006). A model diagram layout extension for sbml. *Bioinformatics*, 22(15):1879–1885.

Hucka, M., Bergmann, F. T., Hoops, S., Keating, S. M., Sahle, S., and Wilkinson, D. J. (2011). Available via the World Wide Web at http://sbml.org/Documents/Specifications.

Hucka, M., Finney, A., Sauro, H., Bolouri, H., Doyle, J., Kitano, H., Arkin, A., Bornstein, B., Bray, D., Cornish-Bowden, A., Cuellar, A., Dronov, S., Gilles, E., Ginkel, M., Gor, V., Goryanin, I., Hedley, W., Hodgman, T., Hofmeyr, J., Hunter, P., Juty, N., Kasberger, J., Kremling, A., Kummer, U., Novère, N. L., Loew, L., Lucio, D., Mendes, P., Minch, E., Mjolsness, E., Nakayama, Y., Nelson, M., Nielsen, P., Sakurada, T., Schaff, J., Shapiro, B., shimizu, T., Spence, H., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., and Wang, J. (2003). The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–31.

Oestereich, B. (1999). *Developing Software with UML: Object-Oriented Analysis and Design in Practice*. Addison-Wesley.

Sahle, S., Gauges, R., Pahle, J., Simus, N., Kummer, U., Hoops, S., Lee, C., Singhal, M., Xu, L., and Mendes, P. (2006). Simulation of biochemical networks using copasi: a complex pathway simulator. In *Proceedings of the 38th conference on Winter simulation*, WSC '06, pages 1698–1706. Winter Simulation Conference.

SBML Team (2010). The SBML Issue Tracker. Available via the World Wide Web at http://sbml.org/issue-tracker.

SBML Team (2012). System biology markup language. Available via the World Wide Web at http://www.sbml.org/.

Thompson, H. S., Beech, D., Maloney, M., and Mendelsohn, N. (2000). XML Schema part 1: Structures (W3C candidate recommendation 24 October 2000). Available online via the World Wide Web at the address http://www.w3.org/TR/xmlschema-1/.

Wegner, K. and Kummer, U. (2005). A new dynamical layout algorithm for complex biochemical reaction networks. *BMC Bioinformatics*, 6(1):212.

Wikipedia (2013). Bèzier curve — Wikipedia, the free encyclopedia. Available via the World Wide Web at http://en.wikipedia.org/wiki/B%C3%A9zier_curve. [Online; accessed 5-May-2013].