SBML Level 3 Package Specification

Groups

Michael Hucka
mhucka@caltech.edu
Computing and Mathematical Sciences
California Institute of Technology
Pasadena, California, US

Lucian P. Smith
lpsmith@u.washington.edu
Computing and Mathematical Sciences
California Institute of Technology
Pasadena, California, US

Version 1, Release 1

01 April 2016

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

This release of the specification is available at
http://co.mbine.org/standards/sbml/level-3/version-1/groups/version-1/release-1

Contents

1 Introduction

The SBML Level 3 Groups package offers a more flexible mechanism for indicating that components of an SBML model are related in some way. The nature of the relationship is left up to the modeler, and can be clarified by means of annotations on model components. Groups may contain either the same or different types of SBML objects, and groups may be nested if desired. There are no predefined behavioral semantics associated with groups.

1.1 Proposal corresponding to this package specification

This specification for Groups in SBML Level 3 Version 1 is based on the proposal located at the following URL:

`https://sbml.svn.sf.net/svnroot/sbml/trunk/specifications/sbml-level-3/version-1/groups/proposal`

The tracking number in the SBML issue tracking system (SBML Team, 2010) for Groups package activities is 2847474. The version of the proposal used as the starting point for this specification is the version of June, 2012 (Hucka, 2012).

1.2 Package dependencies

The Groups package has no dependencies on other SBML Level 3 packages.

1.3 Document conventions

UML 1.0 (Unified Modeling Language; Eriksson and Penker 1998; Oestereich 1999) notation is used in this document to define the constructs provided by this package. Colors in the diagrams carry the following additional information for the benefit of those viewing the document on media that can display color:

- *Black*: Items colored black in the UML diagrams are components taken unchanged from their definition in the SBML Level 3 Core specification document.
- *Green*: Items colored green are components that exist in SBML Level 3 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 Core specification.

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

AbstractClass: Abstract classes are never instantiated directly, but rather serve as parents of other 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 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 Core specification.)

`SomeThing`, `otherThing`: Attributes of classes, data type names, literal XML, and tokens *other* than SBML class names, are printed in an upright typewriter typeface.

For other matters involving the use of UML and XML, this document follows the conventions used in the SBML Level 3 Core specification document.

2 Background and context

SBML Level 2 Versions 2–4 provides two object classes, **CompartmentType** and **SpeciesType**, meant to allow the definition of types of compartments and species. The original motivation for the introduction of these two classes of objects was the expected introduction of a facility for defining generalized reactions, a scheme that would have allowed reactions to be defined on whole classes of entities in a compact format. However, generalized reactions never ended up being introduced in SBML Level 2, and the notion of generalized reactions has been superceded by the effort to support rule-based models using the Level 3 Multistate and Multicomponent Species package. Moreover, the existence of just two types was never satisfactory in Level 2, because it did not satisfy the occasional desire to have other types of objects, such as parameters. For these reasons, when SBML Level 3 was developed, **CompartmentType** and **SpeciesType** were removed, with the expectation that SBML Level 3 packages would be developed to take their place.

The SBML Level 3 Groups package is intended to fill, in at least some capacity, the gaps left by the absence of **CompartmentType** and **SpeciesType** from SBML Level 3 Version 1 Core, while also offering a more flexible mechanism for indicating that components of an SBML model are related. The nature of the relationship is left up to the modeler. The Version 1 definition of this SBML Level 3 package, cannot, unfortunately, entirely take the place of SBML Level 2's **SpeciesType**, since the definition of **SpeciesType** includes usage and validation rules that are not present the final Version 1 of the Groups package. Constructs providing the ability to define such validation rules have been designed, but ultimately were removed from the Version 1 specification because (1) their implementation proved to be a challenge and (2) none of the developers who implemented preliminary support for the Groups specification voiced a need or desire for these additional capabilities. Those constructs have been removed from the final Version 1 of the Groups package, but the potential remains to add them back in a Version 2 specification.

The term *groups* is used in this package rather than *types*, because the latter would imply stronger behavioral constraints on objects than what Groups provides. This package only provides a way of conceptually grouping components of a model. It does not provide a way to define types in the computer science sense; therefore, a different term is appropriate.

2.1 Prior work

The earliest relevant work is the development of the **CompartmentType** and **SpeciesType** object classes in SBML Level 2 beginning with Version 2 (Finney et al., 2006). The original design was based on Andrew Finney's proposal for these object classes, which was made in the context of Finney's proposal for multicomponent species for SBML Level 3 (Finney, 2004). The **SpeciesType** and **CompartmentType** classes were included in SBML Level 2 Version 2; however, a community vote held in 2006 (SBML Editors, 2006b) resulted in the decision that corresponding changes to **Reaction** be postponed to SBML Level 3. Level 2 was thus left only with the typing mechanism for species and compartments. Eventually, further work on generalized reactions and rule-based modeling was moved to an SBML Level 3 package, and **SpeciesType** and **CompartmentType** were removed from the core of SBML Level 3. This left the SBML Level 3 core without an explicit mechanism for defining species types or any other types.

The first version of the Groups proposal was produced by the first author in 2009 (Hucka, 2006), after discussions with the SBML Editors during the 2009 SBML Hackathon held at the European Bioinformatics Institute (Juty et al., 2009). The original proposal differed from the current proposal in several respects. The most notable architectural difference is that membership in groups was done in an inverted fashion compared to the current specification: model entities contained structures that indicated which groups they belonged to, rather than the current scheme, in which group definitions include lists of their members. The current scheme was developed during discussions with the SBML Editors during HARMONY 2011 in New York City (Anwar et al., 2011) and HARMONY 2012 in Maastricht, The Netherlands (Kutmon et al., 2012).

The idea for using an attribute to indicate the type of a group (i.e., a whether it is a classification, partonomy or simply a collection) was put forward by Nicolas Le Novère during the discussions at HARMONY 2012. The resulting attribute `kind` on the **Group** object class solved this problem.

3 Package syntax and semantics

This section contains a definition of the syntax and semantics of the Groups package for SBML Level 3 Version 1 Core. The Groups package involves four new object classes, **Group**, **Member**, **ListOfMembers** and **ListOfGroups**, as well as a simple extension of the existing **Model** object class. Section 5 on page 13 contains complete examples of using the constructs in SBML models.

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 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 Groups package for SBML Level 3 Version 1 Core:

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

In addition, SBML documents using a given package must indicate whether the package can be used to change the mathematical interpretation of a model. This is done using the attribute `required` on the `<sbml>` element in the SBML document. For the Groups package, the value of this attribute must be "`false`", because the use of the Groups package cannot change the mathematical meaning of a model.

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

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

3.2 Primitive data types

The Groups package uses a number of the primitive data types described in Section 3.1 of the SBML Level 3 Version 1 Core specification, including `SId`, clarified below, and adds the `GroupKind` type.

3.2.1 Type `GroupKind`

The `GroupKind` primitive data type is used in the definition of the **Group** class. `GroupKind` is derived from type `string` and its values are restricted to being one of the following possibilities: "`classification`", "`partonomy`", and "`collection`". Attributes of type `GroupKind` cannot take on any other values. The meaning of these three values is discussed in the context of the **Group** class's definition in Section 3.3.

3.2.2 Type `SId`

The `SId` primitive data type is used here unchanged from its description in SBML Level 3 Version 1 Core. When used as the type of a `groups:id` attribute, that identifier is added to the core `SId` namespace, and must continue to follow the same `SId` rules for uniqueness: that is, no `groups:id` may duplicate any other `groups:id`, nor the `id` of any **Model**, **FunctionDefinition**, **Compartment**, **Species**, **Reaction**, **SpeciesReference**, **ModifierSpeciesReference**, **Event**, or **Parameter**, nor the `package:id` of any other SBML Level 3 package element that is also defined as being in the `SId` namespace.

3.3 The Group class

The first and most central class in the Groups package is the **Group** class. Figure 1 on the following page provides the UML diagram of its definition. The **Group** class provides an optional identifier and name, one required attribute (`kind`), and one child: a list of members of the group, described below.

Since **Group** is derived from ***SBase***, and ***SBase*** provides the ability to attach SBO terms as well as MIRIAM annotations, the semantics of a given group in a model can be made more precise by reference to external controlled vocabularies and ontologies. This capability is discussed further in Section 4 on page 11.

3.3.1 The `id` and `name` attributes

The optional `id` attribute on the **Group** object class serves to provide a way to identify a group. The attribute takes a value of type `SId`. Note that the identifier of a group carries no mathematical interpretation and cannot be used in mathematical formulas in a model. **Group** also has an optional `name` attribute, of type `string`. The `name` attribute may be used in the same manner as other `name` attributes on SBML Level 3 Version 1 Core objects; please see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

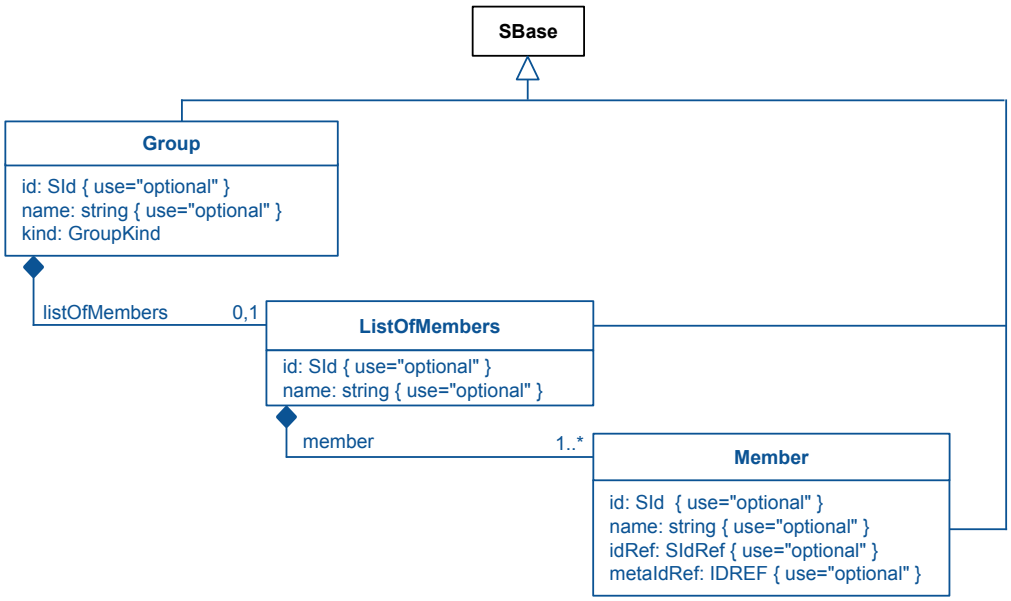

Figure 1: *The definition of the **Group**, **ListOfMembers**, and **Member** classes.*

3.3.2 The `kind` attribute

Group has one required attribute, `kind`, of type `GroupKind`. This attribute is used to indicate the nature of the group defined by a **Group** instance. The `kind` attribute must always have one of the three possible values of `GroupKind`; these values have the following meanings:

`classification`
: The group represents a class, and its members have an *is-a* relationship to the group. For example, the group could represent a type of molecule such as ATP, and the members could be species located in different compartments, thereby establishing that the species are pools of the same molecule in different locations.

`partonomy`
: The group represents a collection of parts, and its members have a *part-of* relationship to the group. For example, the group could represent a cellular structure, and individual compartments could be made members of the group to indicate they represent subparts of that cellular structure.

`collection`
: The grouping is merely a collection for convenience, without an implied relationship between the members. For example, the group could be used to collect together multiple disparate components of a model—species, reactions, events—involved in a particular phenotype, and apply a common annotation rather than having to copy the same annotation to each component individually.

3.3.3 Group membership

If an SBML element is referenced by a **Group**'s child **Member** (directly or indirectly—see below), it is considered to be a member of that **Group**. If the same element is referenced by multiple **Member** objects, this is equivalent to including it just once. It is considered best practice to avoid this, but does not make for an invalid SBML document. Children of referenced elements are not considered to be members of the **Group**: the **KineticLaw** object within a referenced **Reaction** object is not itself a **Group** member. Even the membership of SBML container classes (**ListOfSpecies**, **ListOfCompartments**, etc.) do not imply inclusion of their children as members of the **Group**. The sole exception to this rule is the **ListOfMembers** class, described below.

3.4 The ListOfMembers class

The **ListOfMembers** class is defined in Figure 1 on the preceding page, and must have one or more **Member** children. Since **ListOfMembers** is derived from ***SBase***, it inherits the `sboTerm` and `metaid` attributes, as well as the optional children **Notes** and **Annotation**. Unlike most lists of objects in SBML, however, the `sboTerm` attribute and the **Notes** and **Annotation** children are taken here to apply directly to every SBML element referenced by each child **Member** of this **ListOfMembers**, if that referenced element has no such definition. Thus, if a referenced element has no defined `sboTerm`, child **Notes**, or child **Annotation**, that element should be considered to now have the `sboTerm`, child **Notes**, or child **Annotation** of the **ListOfMembers**.

If multiple **ListOfMembers** have child **Member** elements that reference the same SBML element, and more than one **ListOfMembers** or **Member** has a value for an `sboTerm` attribute, **Notes**, or **Annotation** element, those **Member** elements should be consistent with each other: the `sboTerm` attributes should either be identical, or one should inherit from the other; **Notes** should say the same or similar things; and **Annotation** elements should not conflict. Interpreters may choose to resolve any such conflicts arbitrarily.

3.4.1 The `id` and `name` attributes

The optional `id` attribute of the **ListOfMembers** object class serves to provide a way to reference the collection by its elements instead of as a collection. The `id` attribute on a **Group** is used for the latter, when you need to refer to the group as a group; conversely, the `id` of the **ListOfMembers** is used when you need a short-hand way of referring to all the elements of a group at once. The attribute takes a value of type `SId`. Note that this identifier carries no mathematical interpretation and cannot be used in mathematical formulas in a model.

ListOfMembers also has an optional `name` attribute, of type `string`. The `name` attribute may be used in the same manner as other `name` attributes on SBML Level 3 Version 1 Core objects; please see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

3.5 The Member class

The **Member** class is defined in Figure 1 on the previous page. It has four optional attributes: `id` and `name`, which identify the element, and `idRef` and `metaIdRef` which reference the identifiers of other elements. There must be exactly *one* (and only one) method used to reference another element: either `idRef` or `metaIdRef` may be defined, but not both. (Multiple attributes are needed to account for the different types of identifiers that a given object may have.) The referenced object (including, potentially, another **Group** object) is thus made a member of the group in which the **Member** object is contained.

Since **Member** is derived from ***SBase*** and, as mentioned above, ***SBase*** provides both the ability to attach SBO terms as well as MIRIAM annotations, the semantics of a given member in a model can be made more precise by reference to external controlled vocabularies and ontologies.

The meaning and purpose of the attributes on the object class are described below.

3.5.1 The `id` and `name` attributes

The optional `id` attribute on the **Member** object class serves to provide a way to identify the member reference. The attribute takes a value of type `SId`. Note that the identifier of a member reference carries no mathematical interpretation and cannot be used in mathematical formulas in a model. **Member** also has an optional `name` attribute, of type `string`. The `name` attribute may be used in the same manner as other `name` attributes on SBML Level 3 Version 1 Core objects; please see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

3.5.2 The `idRef` attribute

The attribute `idRef` on the **Member** class has type `SIdRef`, and must be defined if the **Member** has no defined `metaIdRef` attribute, and must not be defined otherwise. The value must be the identifier of an object elsewhere in the **Model**. (Object identifiers are usually set by attributes named `id`; thus, the `idRef` value will usually be the `id` value of an object in the **Model**.) An example value of `idRef` might be the identifier of a species in the model, or the identifier of another group. The namespace in which the `SId` is to be found is the SId namespace of the **Model** to which the **Group** belongs. This may include elements from SBML packages that define elements with `id` values that are part of the SId namespace of the **Model**. A few examples are the **Deletion** elements from the SBML Level 3 Hierarchical Model Composition package, **FluxBound** elements from the Flux Balance Constraints package, and **Group** and **Member** elements from this Groups package.

Conversely, elements with `id` values that are not part of the SId namespace may not be referenced by this `idRef` attribute. In SBML Level 3 Version 1 Core, this includes the **Unit** and **LocalParameter** elements. For any SBML package, the same rule applies, such as for the **Port** elements from the Hierarchical Model Composition package.

3.5.3 The `metaIdRef` attribute

The **Member** attribute `metaIdRef` takes a value of type `IDREF`. The attribute must be defined if the **Member** has no defined `idRef` attribute, and must not be defined otherwise. This attribute is used to refer to a `metaid` attribute value on any other object in the **Model**, for cases where the object being referenced does not have an identifier in the **Model** SId namespace. (This is the case with, for example, rules in SBML Level 3 Version 1 Core.) Since meta identifiers are optional attributes of ***SBase***, all SBML objects have the potential to have a meta identifier value, including most elements from other SBML packages.

Note that even if used in conjunction with the SBML Level 3 Hierarchical Model Composition package, this attribute is not allowed to reference elements that reside within other **Model** objects in the same SBML Document. Referenced elements must be normal members of the parent **Model** containing the **Member** object, and submodel elements may be normally accessed by creating replacements.

3.5.4 Members referencing other Groups and Members

If a **Member** references a **Group**, it is the **Group** itself that is considered to be a member of the parent **Group**, not the corresponding referenced element (or elements). This is also true for elements from other packages that point to other elements, such as **SBaseRef** elements from the SBML Level 3 Hierarchical Model Composition package. However, if instead a **Member** references a **ListOfMembers** object, it is the elements referenced in that list that are considered to be part of the parent **Group**.

The implication of this is that all the rules in this specification that apply to members of a group (such as how the `kind` attribute functions, and the application of `sboterm` attributes on a **ListOfMembers** restrictions, described below) apply to the child group when referenced by the **Group** id, and to the members of the child group when referenced by the **ListOfMembers** id. In an example situation where a parent group includes two **Species** plus a **Group** which itself contains three other **Species**, if the parent group's **ListOfMembers** is given an `sboterm`, that term applies to the two species *and the group, not* to the three child species members of the second group. Note that the parent group's `kind` would also almost certainly be "`collection`" or "`partonomy`", and not "`classification`", as two species and a group are very unlikely to be classified as the same thing. In contrast, in the situation where a parent group

includes two **Species** plus a **ListOfMembers** which contains three other **Species**, the parent group's **ListOfMembers** `sboterm` would apply to the five **Species**, and could be more reasonably marked as a "`classification`".

In a future version of this specification, it may be possible to perform set operations on groups, but for now, this type of union is the only set operation that is possible. For more detail, see Section 4 on page 11.

Finally, groups are not permitted to be circular: no **Member** may reference itself, its parent **ListOfMembers**, nor its parent **Group**. If a **Member** references a **Group**, the same restrictions apply to that subgroup's children: they may not reference the **Member**, its parent **ListOfMembers**, nor its parent **Group**, and if any of those children reference a **Group**, the same restrictions apply to them, etc.

3.5.5 Members referencing other namespaces

If a **Member** has a `idRef` or `metaIdRef` attribute which references an object from a namespace that is not understood by the interpreter of the SBML model, that **Member** must be ignored—the referenced object will not be understood by the interpreter, and therefore has no need to become a member of the group. If an interpreter cannot tell whether a referenced object does not exist or if exists in an unparsed namespace, it may choose to produce a warning.

3.5.6 Example

As mentioned above, exactly one of the attributes `idRef` and `metaIdRef` must have a value in a given **Member** object instance. There are no restrictions on mixing the attributes used across multiple members of the same group, however. The following artificial example illustrates the use of heterogeneous and nested groups. The group "`all_species`" references only species, but the group "`all_entities`" references the compartment, the group of species, and the rule (the last through the use of the `metaIdRef` attribute):

```
...
  <listOfSpecies>
      <species id="S1" compartment="c" initialConcentration="1"/>
      <species id="S2" compartment="c" initialConcentration="2"/>
  </listOfSpecies>
  <listOfCompartments>
      <compartment id="C" size="1"/>
  </listOfCopartments>
  <listOfRules>
      <rateRule variable="S2" metaid="_rule1">
          <math xmlns="http://www.w3.org/1998/Math/MathML">
              <apply>
                  <times/>
                  <ci> S1 </ci>
                  <cn> 0.5 </cn>
              </apply>
          </math>
      </rateRule>
  </listOfRules>
  <listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
      <group kind="collection">
          <listOfMembers id="all_species">
              <member idRef="S1"/>
              <member idRef="S2"/>
          </listOfMembers>
      </group>
      <group id="all_entities" kind="collection">
          <listOfMembers>
              <member idRef="C"/>
              <member idRef="all_species"/>
              <member metaIdRef="_rule1"/>
          </listOfMembers>
      </group>
  </listOfGroups>
...
```

3.6 The extended Model class

The Groups package extends SBML Level 3 Version 1 Core's **Model** class to add one list, **ListOfGroups**, for holding group definitions. Figure 2 provides the UML diagram for the extension.

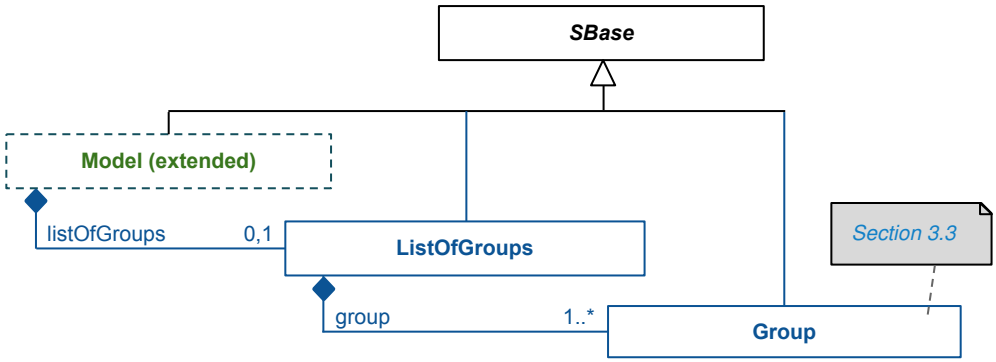

Figure 2: *The extensions of the* ***Model*** *class.* ***Group*** *is defined in Section 3.3 on page 5. In other respects,* ***Model*** *remains defined as in the SBML Level 3 Version 1 Core specification.*

3.6.1 The list of groups

Figure 2 shows that the extension of **Model** by the Groups package involves adding an optional `listOfGroups` subcomponent for holding a **ListOfGroups** container object. If present, the **ListOfGroups** instance must contain at least one **Group** object (Section 3.3 on page 5). In common with other **ListOf**__ classes in SBML, **ListOfGroups** is derived from ***SBase***. It inherits ***SBase***'s attributes `metaid` and `sboTerm`, as well as the subcomponents for **Annotation** and **Notes**, but does not add any new attributes of its own.

4 The semantics of “groups”

A group G is defined by declaring an instance of a **Group** class object within the **ListOfGroups** element of a **Model** object. The group can be given an optional identifier (i.e., a value for its `id` attribute), but even if the group does not have an identifier, the act of declaring a group has the effect of creating it.

An entity X in the model is declared to be part of group G by listing the identifier of X in a **Member** object within the **ListOfGroups** instance of G, or by listing the identifier of a **ListOfMembers** object of a group to which X belongs as a **Member** object of G (thus 'nesting' the subgroup members). The following is an example of an unnested structure:

```
<model id="model_1">
  <listOfSpecies>
    <species id="s1" .../>
    <species id="s2" .../>
    <species id="s3" .../>
    <species id="s4" .../>
  </listOfSpecies>
  ...
  <listOfReactions>
    <reaction id="r1" ...> ... </reaction>
    <reaction id="r2" ...> ... </reaction>
  <listOfReactions>
  ...
  <listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
    <group id="some_species_group" kind="collection">
      <listOfMembers>
        <member idRef="s1"/>
        <member idRef="s3"/>
      </listOfMembers>
    </group>
    <group id="some_reaction_group" kind="collection">
      <listOfMembers>
        <member idRef="r1"/>
        <member idRef="r2"/>
      </listOfMembers>
    </group>
  </listOfGroups>
</model>
```

For any given group, the meaning of group membership is determined by the value of the attribute `kind` on the **Group** object instance. Examples of possible meanings are given in Section 3.3.2 on page 6.

The meaning of a group can be further refined by using annotations (either SBO terms or the **Annotation** element) on the group, or the list of members. This possibility raises the question of how the annotations should be interpreted across the group members. The following are the possibilities defined by the Groups package (summarizing Section 3.3 on page 5, Section 3.4 on page 7, and Section 3.5 on page 7):

- If the annotation or SBO term is on a **Group** object, it is an annotation about the group itself, not the individual members.
- If the annotation or SBO term is on a **Member** object, it is an annotation about the **Member** itself, and not about its referenced element.
- If the annotation or SBO term is on **ListOfMembers**, it is a short-hand that means the annotation applies to each individual referenced member, as if the annotation were put on the referenced SBML elements directly.

Similar rules apply to nested groups:

- If a **Group** object is referenced by a **Member**, it is the group itself that is included in that **Member**'s parent **Group**, not its individual members.

- If a **Member** object is referenced by another **Member**, it is that member that is included in the parent **Group**, and not the referenced SBML element to which the **Member** points.
- If a **ListOfMembers** object is referenced by a **Member**, all of the referenced SBML elements from that **ListOfMembers** are included in the parent **Group**. It is equivalent to including all of the referenced elements directly.

The following example uses the rules described above to nest the members of "`group1`" inside "`group2`", using the "`group1_list`" **ListOfMembers** id:

```
<model id="model_2">
  ...
  <listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
    <group id="group1">
      <listOfMembers id="group1_list">
        <member idRef="..."/>
        <member idRef="..."/>
      </listOfMembers>
    </group>
    <group id="group2">
      <listOfMembers>
        <member idRef="group1_list"/>
        <member idRef="..."/>
        <member idRef="..."/>
      </listOfMembers>
    </group>
  </listOfGroups>
</model>
```

The intended meaning of nested groups can be made more precise by annotating the group and list of members with appropriate annotations using controlled vocabulary terms that describe the meaning.

4.1 Semantic restrictions

The current definition of this package is such that the use of Groups constructs has no impact on the mathematics of a model. As a consequence of this, this package is not required for proper interpretation of a model. The document must use the `required`="`false`" flag on the declaration of the package on the `<sbml>` element in a file.

5 Examples

This section contains examples employing the Groups package for SBML Level 3.

5.1 Simple species typing via annotations

The following is a simple example of using this proposed grouping facility to do something similar to the **SpeciesType** example shown in Section 4.6.3 of the SBML Level 2 Version 4 specification (p. 43).

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
  xmlns:groups="http://www.sbml.org/sbml/level3/version1/groups/version1" groups:required="false">
  <model>
    <listOfSpecies>
      <species id="ATPc" compartment="cytosol" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true"/>
      <species id="ATPm" compartment="mitochon" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true"/>
    </listOfSpecies>
    <listOfCompartments>
      <compartment id="cytosol" spatialDimensions="3" size="1" constant="true"/>
      <compartment id="mitochon" spatialDimensions="3" size="1" constant="true"/>
    </listOfCompartments>
    <groups:listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
      <groups:group groups:id="ATP" groups:kind="classification">
        <groups:listOfMembers sboTerm="SBO:0000248">
          <groups:member groups:idRef="ATPc"/>
          <groups:member groups:idRef="ATPm"/>
        </groups:listOfMembers>
      </groups:group>
    </groups:listOfGroups>
  </model>
</sbml>
```

In this example, both "ATPc" and "ATPm" are intended to be pools of ATP, but located in different compartments. To indicate that they are both conceptually the same kind of molecular entity, the model includes a group definition of the "classification" variety. The two species "ATPc" and "ATPm" are both listed as members of the same group. The **ListOfMembers** is given the sboTerm "SBO:0000248" to indicate that both species are small molecules. This group definition could be enhanced further by including an annotation on the **ListOfMembers** that references the ChEBI database entry for ATP; we omit that detail here in order to concentrate on the Groups constructs.

5.2 Example using meta identifiers

In the next example, two rules both influence a model in the same way, so a group is used to collect the rules and annotate them.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
      xmlns:groups="http://www.sbml.org/sbml/level3/version1/groups/version1"
      groups:required="false">
  <model id="cell" name="cell">
    <listOfCompartments>
      <compartment id="compartment" spatialDimensions="3" size="1" constant="true" />
    </listOfCompartments>
    <listOfSpecies>
      <species id="Dose1" compartment="compartment" initialConcentration="0" constant="false"
               hasOnlySubstanceUnits="false" boundaryCondition="true" />
      <species id="Dose2" compartment="compartment" initialConcentration="0" constant="false"
               hasOnlySubstanceUnits="false" boundaryCondition="true"/>
```

```
    </listOfSpecies>
    <listOfRules>
      <assignmentRule metaid="D1" variable="Dose1">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <piecewise>
            <piece>
              <cn type="integer"> 2 </cn>
              <apply>
                <lt />
                <csymbol encoding="text"
                         definitionURL="http://www.sbml.org/sbml/symbols/time"> time </csymbol>
                <cn type="integer"> 1 </cn>
              </apply>
            </piece>
            <otherwise>
              <cn type="integer"> 0 </cn>
            </otherwise>
          </piecewise>
        </math>
      </assignmentRule>
      <assignmentRule metaid="D2" variable="Dose2">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <piecewise>
            <piece>
              <cn> 1.5 </cn>
              <apply>
                <and />
                <apply>
                  <gt />
                  <csymbol encoding="text"
                           definitionURL="http://www.sbml.org/sbml/symbols/time"> time </csymbol>
                  <cn type="integer"> 5 </cn>
                </apply>
                <apply>
                  <lt />
                  <csymbol encoding="text"
                           definitionURL="http://www.sbml.org/sbml/symbols/time"> time </csymbol>
                  <cn type="integer"> 6 </cn>
                </apply>
              </apply>
            </piece>
            <otherwise>
              <cn type="integer"> 0 </cn>
            </otherwise>
          </piecewise>
        </math>
      </assignmentRule>
    </listOfRules>
    <groups:listOfGroups>
      <groups:group groups:id="effectB" groups:kind="collection">
        <notes>
          <p xmlns="http://www.w3.org/1999/xhtml"> These two rules are in the model as approximation
          for effect B</p>
        </notes>
        <groups:listOfMembers>
          <groups:member groups:metaIdRef="D1"/>
          <groups:member groups:metaIdRef="D2"/>
        </groups:listOfMembers>
      </groups:group>
    </groups:listOfGroups>
  </model>
</sbml>
```

The key point of this example is the use of meta identifiers for SBML entities (in particular, rules) that do not have regular identifiers (i.e., `id` attributes).

5.3 Example using nested groups

In this example, ATP is grouped as ATP in the organelles, and all ATP, with the former being a subgroup of the latter. Because the intent of the group is to capture the ATP elements themselves, a reference to the first group's **ListOfMembers** is used.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
  xmlns:groups="http://www.sbml.org/sbml/level3/version1/groups/version1" groups:required="false">
  <model>
    <listOfSpecies>
      <species id="ATPc" compartment="cytosol" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true"/>
      <species id="ATPm" compartment="mitochon" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true"/>
      <species id="ATPr" compartment="reticulum" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true"/>
      <species id="ATPn" compartment="nucleus" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true"/>
      <species id="ATPe" compartment="extracellular" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true"/>
    </listOfSpecies>
    <listOfCompartments>
      <compartment id="cytosol" spatialDimensions="3" size="1" constant="true"/>
      <compartment id="mitochon" spatialDimensions="3" size="1" constant="true"/>
      <compartment id="reticulum" spatialDimensions="3" size="1" constant="true"/>
      <compartment id="nucleus" spatialDimensions="3" size="1" constant="true"/>
      <compartment id="extracellular" spatialDimensions="3" size="1" constant="true"/>
    </listOfCompartments>
    <groups:listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
      <groups:group groups:id="ATP_organelle" groups:kind="classification">
        <groups:listOfMembers groups:id="ATP_organelle_list">
          <groups:member groups:idRef="ATPm"/>
          <groups:member groups:idRef="ATPr"/>
          <groups:member groups:idRef="ATPn"/>
        </groups:listOfMembers>
      </groups:group>
      <groups:group groups:id="ATP_all" groups:kind="classification">
        <groups:listOfMembers groups:id="ATP_all_list" sboTerm="SBO:0000248">
          <groups:member groups:idRef="ATP_organelle_list"/>
          <groups:member groups:idRef="ATPc"/>
          <groups:member groups:idRef="ATPe"/>
        </groups:listOfMembers>
      </groups:group>
    </groups:listOfGroups>
  </model>
</sbml>
```

In this example, all five ATP **Species** are effectively labeled with SBO term "`SBO:0000248`" to indicate they are all small molecules. An annotation added to "`ATP_all_list`" to additionally specify the ChEBI database entry for ATP would have likewise applied to all five ATP **Species**.

Had the second group referenced "`ATP_organelle`" instead, the SBO term would have been incorrect (since a group is not a small molecule), and the `kind` for the group should have been `collection` instead of `classification`.

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 Groups package. We use the same conventions that 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 Groups package 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 Groups package 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 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 Groups package.

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

General rules about this package

groups-10101 ☑ To conform to the Groups package specification for SBML Level 3 Version 1, an SBML document must declare "`http://www.sbml.org/sbml/level3/version1/groups/version1`" as the XMLNamespaceto use for elements of this package. (Reference: SBML Level 3 Package specification for Groups, Version 1 Section 3.1 on page 5.)

groups-10102 ☑ Wherever they appear in an SBML document, elements and attributes from the Groups package must use the "`http://www.sbml.org/sbml/level3/version1/groups/version1`" namespace, declaring so either explicitly or implicitly. (Reference: SBML Level 3 Package specification for Groups, Version 1 Section 3.1 on page 5.)

General rules about identifiers

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

jects, plus the **Group**, **ListOfMembers**, and **Member** objects defined by the Groups package, plus any objects defined by any other package with `package:id` attributes defined as falling in the 'SId' namespace. (References: SBML Level 3 Package Specification for Groups, Version 1, Section 3.3 on page 5.)

groups-10302 ☑ The value of a `groups:id` must conform to the syntax of the **SBML** data type `SId` (Reference: SBML Level 3 Package specification for Groups, Version 1 Section 3.3.1 on page 6.)

Rules for the extended SBML class

groups-20101 ☑ In all SBML documents using the Groups package, the **SBML** object must have the `groups:required` attribute. (Reference: SBML Level 3 Version 1 Core, Section 4.1.2.)

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

groups-20103 ☑ The value of attribute `groups:required` on the **SBML** object must be set to "`false`". (Reference: SBML Level 3 Package specification for Groups, Version 1 Section 3.1 on page 5.)

Rules for extended Model object

groups-20201 ☑ A **Model** object may contain one and only one instance of the **ListOfGroups** element. No other elements from the SBML Level 3 Groups namespaces are permitted on a **Model** object. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.6 on page 10.)

groups-20202 ☑ The **ListOfGroups** subobject on a **Model** object is optional, but if present, this container object must not be empty. (Reference: SBML Level 3 Specification for Groups Version 1, Section 3.6 on page 10.)

groups-20203 ☑ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfGroups** container object may only contain **Group** objects. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.6 on page 10.)

groups-20204 ☑ A **ListOfGroups** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfGroups** object. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.6 on page 10.)

Rules for Group object

groups-20301 ☑ A **Group** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Group**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

groups-20302 ☑ A **Group** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Group**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

groups-20303 ☑ A **Group** object must have the required attribute `groups:kind`, and may have the optional attributes `groups:id` and `groups:name`. No other attributes from the SBML Level 3 Groups namespaces are permitted on a **Group** object. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.3 on page 5.)

groups-20304 ☑ A **Group** object may contain one and only one instance of the **ListOfMembers** element. No other elements from the SBML Level 3 Groups namespaces are permitted on a **Group** object. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.3 on page 5.)

groups-20305 ☑ The value of the attribute `groups:kind` of a **Group** object must conform to the syntax of SBML data type `groupKind` and may only take on the allowed values of `groupKind` defined in SBML; that is, the value must be one of the following: 'classification', 'partonomy' or 'collection'. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.3 on page 5.)

groups-20306 ☑ The attribute `groups:name` on a **Group** must have a value of data type `string`. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.3 on page 5.)

groups-20307 ☑ The **ListOfMembers** subobject on a **Group** object is optional, but if present, this container object must not be empty. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.3 on page 5.)

groups-20308 ☑ Apart from the general notes and annotations subobjects permitted on all SBML objects, a **ListOfMembers** container object may only contain **Member** objects. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

groups-20309 ☑ A **ListOfMembers** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **ListOfMembers** object. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

groups-20310 ☑ A **ListOfMembers** object may have the optional attributes `groups:id` and `groups:name`. No other attributes from the SBML Level 3 Groups namespaces are permitted on a **ListOfMembers** object. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.4 on page 7.)

groups-20311 ☑ The attribute `groups:name` on a **ListOfMembers** must have a value of data type `string`. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.4 on page 7.)

groups-20312 ▲ If **ListOfMembers** objects from different **Group** objects contain **Member** elements that reference the same SBase object, the `sboterm` and any child **Notes** or **Annotation** elements set for those **ListOfMembers** should be consistent, as they all should apply to the same referenced object. (References: SBML Level 3 Package Specification for Groups, Version 1, Section 3.4 on page 7.)

groups-20313 ☑ Member references may not be circular: no **Member**'s `idRef` or `metaIdRef` may reference itself, its parent **ListOfMembers**, nor its parent **Group**. If a **Member** references a **Group** or a **ListOfMembers**, the same restrictions apply to that subgroup's children: they may not reference the **Member**, its parent **ListOfMembers**, nor its parent **Group**, and if any of those children reference a **Group**, the same restrictions apply to them, etc. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.5.4 on page 8.)

Rules for Member object

groups-20401 ☑ A **Member** object may have the optional SBML Level 3 Core attributes `metaid` and `sboTerm`. No other attributes from the SBML Level 3 Core namespaces are permitted on a **Member**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

groups-20402 ☑ A **Member** object may have the optional SBML Level 3 Core subobjects for notes and annotations. No other elements from the SBML Level 3 Core namespaces are permitted on a **Member**. (Reference: SBML Level 3 Version 1 Core, Section 3.2.)

groups-20403 ☑ A **Member** object may have the optional attributes `groups:id` and `groups:name` and must have a value for one (and exactly one) of the attributes `groups:idRef` and `groups:metaIdRef`. No other attributes from the SBML Level 3 Groups namespaces are permitted on a **Member** object. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.5 on page 7.)

groups-20404 ☑ The attribute `groups:name` on a **Member** must have a value of data type `string`. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.5 on page 7.)

groups-20405 ☑ The value of the attribute `groups:idRef` of a **Member** object must be the identifier of an existing ***SBase*** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.5 on page 7.)

groups-20406 ☑ The value of the attribute `groups:metaIdRef` of a **Member** object must be the `metaid` of an existing ***SBase*** object defined in the enclosing **Model** object. (Reference: SBML Level 3 Package specification for Groups, Version 1, Section 3.5 on page 7.)

groups-20407 ☑ The value of a `groups:idRef` attribute on **Member** objects must conform to the syntax of the SBML data type `SIdRef`. (References: SBML Level 3 Package Specification for Groups, Version 1, Section 3.5 on page 7.)

groups-20408 ☑ The value of a `groups:metaIdRef` attribute on **Member** objects must conform to the syntax of the SBML data type `IDREF`. (References:SBML Level 3 Package Specification for Groups, Version 1, Section 3.5 on page 7.)

Acknowledgments

Work such as this does not take place in a vacuum; many people contributed ideas and discussions that shaped the Groups proposal that you see before you. We particularly thank Nicolas Le Novère, Frank Bergmann, Sarah Keating, Allyson Lister, Robert Phair, Sven Sahle, Stefan Hoops, Chris Myers, and the members of the *sbml-discuss* and *sbml-groups* mailing lists for suggestions and comments.

We would also like to thank Andreas Dräeger, Ben Heavner, Brett Olivier, Kieran Smallbone, and Anna Zhukova for their work in implementing 'Groups' support in their own software tools, with Brett in particular investigating implementaton of the 'nested groups' aspect of the specification.

This work was funded by the National Institutes of Health (USA) under grant R01 GM070923.

References

Anwar, N., Demir, E., Hucka, M., and Novère, N. L. (2011). HARMONY 2011: The HAckathon on Resources for MOdeliNg in biologY. Available via the World Wide Web at http://www.co.mbine.org/events/HARMONY_2011.

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

Finney, A. (2004). Multicomponent Species: A proposal for SBML Level 3. Available via the World Wide Web at http://sbml.org/images/1/19/20041015-finney-multicomponent.pdf.

Finney, A., Hucka, M., and Novère, N. L. (2006). The Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions. Available via the World Wide Web at http://sbml.org/Documents/Specifications.

Hucka, M. (2006). Groups proposal (2009-09). Available via the World Wide Web at http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Groups_Proposal_%282009-09%29.

Hucka, M. (2012). Groups proposal updated (2012-06). Available via the World Wide Web at http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Groups_Proposal_Updated_%282012-06%29.

Juty, N., Novère, N. L., Taddeo, L., and Hucka, M. (2009). Seventh SBML Hackathon. Available via the World Wide Web at http://sbml.org/Events/Hackathons/The_7th_SBML_Hackathon.

Kutmon, M., Evelo, C., Bader, G., Novère, N. L., and Hucka, M. (2012). HARMONY 2012: The HAckathon on Resources for MOdeliNg in biologY. Available via the World Wide Web at http://www.co.mbine.org/events/HARMONY_2012.

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

SBML Editors, T. (2006a). Results of L2v2 specification vote #8: Introducing CompartmentType. Available via the World Wide Web at http://sbml.org/Forums/index.php?t=tree&goto=3057&rid=2.

SBML Editors, T. (2006b). Results of L2v2 specification vote #9: Introducing Generalized Reactions. Available via the World Wide Web at http://sbml.org/Forums/index.php?t=tree&th=764&mid=5567&rid=2.

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