SBML Level 3 Package Specification

Required Elements

Lucian P. Smith
lpsmith@u.washington.edu
Department of Bioengineering
University of Washington
Seattle, Washington, US

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

Version 1 (Draft)

10 December 2012

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

This release of the specification is available at
TBD

Contents

1 Introduction

The SBML Level 3 Required Elements package is a small package that allows models to declare exactly which components of a model have the meaning of their mathematical semantics changed by another package. It does this simply by defining two optional attributes on ***SBase***; these attributes can then be attached to any component of a model. The rest of this specification explains the package constructs and how they can be used.

1.1 Proposal corresponding to this package specification

This specification for Required Elements in SBML Level 3 Version 1 is based on the proposal located at this URL:

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

The tracking number in the SBML issue tracking system (SBML Team, 2010) for Required Elements package activities is 3594695. The version of the proposal used as the starting point for this specification is the version of August, 2011 (Smith, 2011).

1.2 Package dependencies

The Required Elements 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

At the heart of many SBML models is mathematics. When present, the mathematics is considered to be of principle importance to understanding the model. Many SBML Level 3 packages extend the mathematical concepts present in SBML Level 3 Version 1 Core, and when they do, a model that includes these concepts must add a `required`="`true`" flag on the definition of that namespace. However, for software tools that wish to manipulate and augment SBML models with extended mathematics which they do not understand, a blanket `required`="`true`" flag on the namespace is not granular enough. The tools need to know which *specific* model elements have had their mathematics changed, so that they can understand which parts of the model are safe to manipulate freely and which must be treated with care.

The Required Elements package is an exceedingly small package that allows model writers to declare specifically which components of the model have had their mathematics changed, by which package, and whether the components have SBML Level 3 Version 1 Core alternatives present. The Required Elements package accomplishes this by defining two optional attributes on SBML's ***SBase*** class: `mathOverriden` and `coreHasAlternativeMath`. These attributes are described in more detail in Section 3 on the following page.

3 Package syntax and semantics

This section contains a definition of the syntax and semantics of the Required Element package for SBML Level 3 Version 1 Core.

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 Required Elements 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 understanding the package is required for complete mathematical interpretation of a model. This is done using the attribute `required` on the `<sbml>` element in the SBML document. For the Required Elements package, the value of this attribute must be "`false`", because the use of the Required Elements 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 Required Elements package:

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

3.2 Primitive data types

The Required Elements package uses two of the primitive data types described in Section 3.1 of the SBML Level 3 Version 1 Core specification, specifically `string` and `boolean`. The package does not define any new types.

3.3 The extended SBase class

As mentioned above, this package's reason for being is to define two optional attributes on the SBML Level 3 ***SBase*** class. Figure 1 provides the definition of this extended ***SBase*** class. These attributes are added to ***SBase*** instead of to those elements that have math in them (such as **Parameter**, **Species**, etc.) because this approach allows other SBML Level 3 packages to define new elements with mathematics which may, in turn be overridden by other packages.

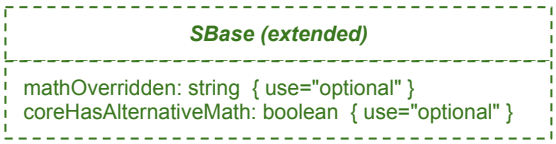

Figure 1: *The definition of the extended* ***SBase*** *class.*

3.3.1 The `mathOverridden` *attribute*

The first attribute added to ***SBase*** is `mathOverriden`. Its value type is `string`. When set, the attribute must be set to the namespace URI of the SBML Level 3 package that redefines the mathematical semantics of the object on which the attribute is set. In other words, if the mathematical semantics of a given component *C* in a model is changed by the use of a Level 3 package *P*, then *C* can be given the attribute `mathOverridden` and the value of that attribute should be *P*'s namespace URI. The examples below will make this more clear.

Note that the `mathOverridden` attribute should be placed on the model component that is directly affected, and not on a component that is indirectly affected by the changed mathematics. For example, if a model contains a **FunctionDefinition** object whose mathematics are extended by the SBML Level 3 package for Distributions, and in addition, elsewhere in the model there is an **InitialAssignment** object that uses the function, the `mathOverridden` attribute must be added only on the **FunctionDefinition** object and not the **InitialAssignment**, because the latter is only indirectly affected.

3.3.2 The `coreHasAlternativeMath` *attribute*

The second attribute added to ***SBase*** is `coreHasAlternativeMath`. Its value type is `boolean`. The attribute should be set to "`true`" or "`false`" depending on whether an interpreter that *only* understands SBML Level 3 Version 1 Core would have a workable and complete (if perhaps different) version of the mathematics for a given component in a model. "Complete understanding" is an objective requirement for setting this attribute to "`true`"—the Level 3 Core parts of the component must not leave any of the mathematics undefined. "Workable" requires a judgement call on the part of the modeler: if the modeler feels that the alternative version makes sense in an alternative context, they may set the attribute value to "`true`"; conversely, if they feel that the resulting model component makes no sense, even if technically "complete", then they should set the attribute value to "`false`".

3.4 Additional considerations

The Required Elements package is not, itself, required (that is, models that use it must set `required`="`false`" for its namespace when using it) because it, in and of itself, does not change any mathematical semantics. However, it is intended to be used by other required packages, which may in turn make the use of one or both attributes from this package required in certain contexts defined in those package specifications. For example, the Spatial Processes package could require that modelers set the attribute `mathOverridden` to the value "`http://www.sbml.org/sbml/level3/version1/distrib/version1`" for any reactions whose kinetic laws were redefined by that package. This would provide a clean way of ensuring that only one package overrode the math for any single SBML component: if two packages required the `mathOverridden` attribute to be set to their own namespace, it would be impossible to doubly-override a single element without failing to comply with at least one package specification.

It is possible that future versions of SBML Level 3 may incorporate the constructs of the Required Elements package directly into the Core specification itself, should this prove generally useful.

4 Examples

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

4.1 Function whose meaning is adequately captured by Core mathematics

In this example, there is a **FunctionDefinition** object that has had its mathematics changed by a (as-yet hypothetical) Distributions package.

```
<listOfFunctionDefinitions>
  <functionDefinition id="unitGaussian"
                      xmlns:req="http://www.sbml.org/sbml/level3/version1/req/version1"
                      req:mathOverridden="http://www.sbml.org/sbml/level3/version1/distrib/version1"
                      req:coreHasAlternateMath="true">
    <math xmlns="http://www.w3.org/1998/Math/MathML"
          xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
       <lambda>
         <cn> 0.5 </cn>
       </lambda>
    </math>
    <distrib:call xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"
                  distrib:function="gaussian" distrib:mean="0.5" distrib:variance="0.25"
    </distrib:call>
  </functionDefinition>
</listOfFunctionDefinitions>
```

The function `unitGaussian` above has both a straight SBML Level 3 Core definition (whose MathML formula has the effect of returning the numerical value 0.5), and a package-defined version (in effect, "select a random number from a Gaussian distribution with mean 0.5 and variance 0.25"). The modeler has elected to claim that the SBML Core definition's version of the function is adequate for the model in which it is being used, although different results are to be expected.

4.2 Function whose meaning is *not* adequately captured by Core mathematics

In this example, the modeler has decided that the plain SBML Level 3 Core version of the mathematics will not make an adequate substitution for using the mathematics defined by the Distributions package:

```
<listOfFunctionDefinitions>
  <functionDefinition id="gaussian"
                      xmlns:req="http://www.sbml.org/sbml/level3/version1/req/version1"
                      req:mathOverridden="http://www.sbml.org/sbml/level3/version1/distrib/version1"
                      req:coreHasAlternateMath="false">
    <math xmlns="http://www.w3.org/1998/Math/MathML"
          xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
       <lambda>
         <bvar> <ci> mean </ci> </bvar>
         <bvar> <ci> variance </ci> </bvar>
         <cn> 0 </cn>
       </lambda>
    </math>
    <distrib:call xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"
                  distrib:function="gaussian" distrib:mean="mean" distrib:variance="variance"
    </distrib:call>
  </functionDefinition>
</listOfFunctionDefinitions>
```

4.3 Function whose definition is incomplete

In the following example, the function definition is marked as having `coreHasAlternateMath`="`false`" because the lambda function is actually incomplete.

```
<listOfFunctionDefinitions>
  <functionDefinition id="gaussian"
                      xmlns:req="http://www.sbml.org/sbml/level3/version1/req/version1"
                      req:mathOverridden="http://www.sbml.org/sbml/level3/version1/distrib/version1"
                      req:coreHasAlternateMath="false">
    <math xmlns="http://www.w3.org/1998/Math/MathML"
          xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
      <lambda>
        <bvar> <ci> mean </ci> </bvar>
        <bvar> <ci> variance </ci> </bvar>
      </lambda>
    </math>
    <distrib:call xmlns:distrib="http://www.sbml.org/sbml/level3/version1/distrib/version1"
                  distrib:function="gaussian" distrib:mean="mean" distrib:variance="variance"
    </distrib:call>
  </functionDefinition>
</listOfFunctionDefinitions>
```

A Validation of SBML documents using Required Elements constructs

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 Required Elements 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 Required Elements 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 Required Elements 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 Required Elements package.

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

General rules about the Required Elements package

req-10101 ☑ To conform to Version 1 of the Required Elements package specification for SBML Level 3, an SBML document must declare the use of the following XML Namespace: "`http://www.sbml.org/sbml/level3/version1/req/version1`". (References: SBML Level 3 Package Specification for Required Elements, Version 1, Section 3.1 on page 5.)

req-10102 ☑ Wherever they appear in an SBML document, elements and attributes from the Required Elements package must be declared either implicitly or explicitly to be in the XML namespace "`http://www.sbml.org/sbml/level3/version1/req/version1`". (References: SBML Level 3 Package Specification for Required Elements, Version 1, Section 3.1 on page 5.)

req-10103 ☑ The value of attribute `req:required` on the **SBML** object must be set to "`false`". (References: SBML Level 3 Package Specification for Required Elements, Version 1, Section 3.1 on page 5.)

Rules for Required Elements attributes

req-20101 ☑ The attribute `mathOverridden`, if present, must have a value of type `string`. (References: SBML Level 3 Package Specification for Required Elements, Version 1, Section 3.3 on page 5.)

req-20102 ☑ The attribute `mathOverridden`, if present, must be the label of an SBML Level 3 package present in the same document. (References: SBML Level 3 Package Specification for Required Elements, Version 1, Section 3.3 on page 5.)

req-20103 ☑ The attribute `coreHasAlternativeMath`, if present, must have a value of type `boolean`. (References: SBML Level 3 Package Specification for Required Elements, Version 1, Section 3.3 on page 5.)

req-20104 ☑ If the value of the attribute `coreHasAlternativeMath` is set `true`, any `math` child of the parent element must be present and complete. (References: SBML Level 3 Package Specification for Required Elements, Version 1, Section 3.3 on page 5.)

Acknowledgments

Work such as this does not take place in a vacuum; many people contributed ideas and discussions that shaped the Required Elements proposal that you see before you. We particularly thank Sarah Keating, Frank Bergmann, James Schaff, and the members of the *sbml-discuss* mailing list for suggestions and comments.

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

References

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

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

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

Smith, L. P. (2011). Required Elements Proposal. Available via the World Wide Web at http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Required_Elements_Proposal.