Parameter Sets: a standard technology for use with SBML

Andrew Finney, Michael Hucka
{afinney,mhucka}@cds.caltech.edu
Systems Biology Workbench Development Group
ERATO Kitano Systems Biology Project
Control and Dynamical Systems, MC 107-81
California Institute of Technology, Pasadena, CA 91125, USA
http://www.cds.caltech.edu/erato

Principal Investigators: John Doyle and Hiroaki Kitano

March 4, 2003

Contents

1 Introduction

This white paper discusses Parameter Sets, a possible new standard interchange format or technology for use with SBML (1). Parameter sets are designed to facilitate the separation of the initial condition and parameter values of a model from the model structure itself. In its most basic form a parameter set is a collection of *key value pairs* where the key refers to an object attribute in an SBML data source and the value is a new value for the referenced attribute. A parameter set can be applied to an existing model: a new model is created in which the attributes of the original model are substituted according to the key value pairs in the parameter set. In this document the original document is called the *source* and the new model is called the *target* model.

1.1 About this document

This document outlines a set of requirements for parameter sets in Section 2 (similar to a formal "request for proposals") as well as to describe possible implementations of these requirements in Section 3.

The primary purpose of this document is to present options which the SBML community can discuss and then use as a framework for constructing a final proposal for a parameter sets standard. This document is meant to provoke the discussion of real requirement and practical implementations rather than providing a definitive statement on either.

1.2 Brief History of Parameter Sets

The idea of parameter sets in the context of SBML was first raised as a requirement publicly by Cliff Schaffer at the BioSPICE MDL meeting in Cambridge, MA in September 2002. The requirement was discussed on the *sbml-discuss* mailing list. By the SBML Forum meeting at ICSB 2002 in December in Stockholm there was a consensus for some standard development in this area. Many of the requirements described here are taken from discussions on the mailing list and at that meeting. The authors would like to acknowledge Ben Bornstein's suggestion of using XSLT as an implementation technology

2 Requirements

2.1 General Motivation

At the start of biochemical network model development many of the parameter values of the modelled system are unknown. As a result a key step of the model development process is finding parameter values which enable the model exhibit the same behaviors as the biological system under study. In addition many researchers argue that one of the key tasks in model development is establishing the robustness and fragilities of a given model to specific parameters values. Thus in the majority of cases during model development the values of parameters and initial conditions are generally more volatile than the model structure as either human or automated processes explore the parameter space of a model. The result is one or more parameter sets none of which can be described as definitive. Modelling tools operate on these parameter sets independently of the model to which they are applied.

A project may build several models of the same organism for which it has a large set of experimental data. The experimental data is stored or made available in the form of a parameter set which is applied to the models constructed by the project. Managing the parameter data in this way ensures that experimental data can (a) be used consistently across several models and (b) be updated without requiring model data to be changed.

The following sections describe various features of parameter sets that may (or perhaps not) be required:

2.2 Overloading

The attribute values contained in a parameter set supply values for missing attributes but also replace existing values in a model that the set is applied to.

The attribute values in a model are typically viewed as more definitive than those in a parameter set however the values in the parameter set may contain a reasonable set of alternative values given incomplete experimental results. A sets of one or more mutations can be modelled as a set of alternative parameter values.

2.3 Incomplete

A parameter set can be incomplete: a parameter set may not necessarily contain values for all relevant attributes in a given model.

Only a subset of parameter values may be explored during the construction of a model as some attributes values are more definitive than others e.g. they are the result of accurate experimental techniques. In most cases a given mutation only affects a small subset of model attributes thus a mutation would be best represented using a parameter set that has value for a subset of attributes.

2.4 Attributes not in Source Model

A parameter set may contain values for attributes that are not in the source model, in which case these values are ignored.

If a parameter set is used as a "database" of values taken form experiments that are to be applied to several models then it is unlikely that all models will use all the attribute values stored in the parameter set.

2.5 Cascading

A target model can be described by a sequence of parameters sets combined with a source model. Each parameter set overloads the attribute values defined by the previous parameter set or the original model

This enables, for example, a parameter set describing a mutant to be combined with a parameter set derived by optimization for the same model.

2.6 Model specificity

A parameter set may or may not be specific to a given source model. A parameter set should have an optional field that refers to a specific model. It is not clear whether the standard should or can enforce the correct use of this field. The use of this field enables a target model to be assembled starting from just a parameter set. When a parameter set is part of a cascade this field refers to the previous parameter set in the sequence starting from the source model.

A parameter set may have attribute values that have been fitted to a specific model in other cases a parameter set may be a "database" of values taken form experiments that are to be applied to several models.

2.7 Parameter Sets as Independent Data Stores

Whilst parameter sets primarily are used as transformations on source models, they should support reading and writing operations on the parameter set. Ideally these operations should be possible without reference to a source model. For example given the identifier for a species it should be possible to read and the change the species initial concentration in the parameter set without referring to a model.

There will probably be a tension between this requirement and the requirement for extendability (see Section 2.8) because a generic format may be difficult to parse without the source model.

2.8 Extendability

The parameter set form should ideally use a form which ensures that it can be applied to all current and future forms of SBML. Such a form is necessarily generic.

Having a generic form will ensure that parameter sets can easily be applied to the potentially complex forms that could arise in SBML Level 3.

2.9 Scope of Transformation

It is not clear what operations a parameter set could apply to a source model to generate a target model. Should a parameter set be restricted just to setting alternative attributes values? For example should a parameter set insert or remove elements? If we restrict changes just to overloading attribute values do we need to limit the set of attributes that can be changed?

For the moment the consensus appears to be that at a minimum parameter values and initial conditions (species concentration and compartment volumes) can be overloaded by parameter sets.

2.10 Reference to Expanded Form and Interface Conformance

It is possible that a parameter set could refer to objects that are implied by the source model but that do not exist explicitly in the source model's XML encoding. A parameter set may refer to attributes in some expanded form derived from the source model.

For example a SBML Level 3 source model may contain several instances of a sub-model. The sub-model may contain a number of parameters. A parameter set for this source model may contain attribute values for the parameters in specific instances of the sub-models even though there are no explicit structures in the XML for those parameters.

In the use case described above the overloaded parameters are either not part of the declared interface for the sub-model or the source model assumes default values for the parameters. The consensus appears to support the use of parameter sets that ignore or bypass any explicit sub-model interfaces.

[It is possible that the definition of complex species will make it possible to refer to reactions and species that are not explicitly listed in a source model. For example consider a source model containing a template reaction: a reaction definition, that can be applied to set of configurations of complex species. A template reaction represents, in SBML Level 2 terms, a set of reactions. A parameter set entry could be applied to the rate of a specific reaction instance (or set of reaction instances) of a template reaction.]

2.11 Simultaneous Support for Multiple levels

Given the requirement that a parameter set can be used with more than one model then it should be possible for a given parameter set to be applied to models encoded in different levels of SBML.

2.12 Abstraction

It is conceivable that a parameter set may wish to derive the values initial conditions and parameters of a given model from each other i.e. one value is calculated via another.

This capability is not yet available in SBML itself. It is not clear that parameter sets are the best place for these math constructs. For the purposes of this document we assume that this requirement is best addressed in the context of SBML Level 3.

2.13 Best Practice

A best practice document for the use of parameter sets will be required. For exchange between software applications target models can be potentially exchanged as a single reference to the last parameter set in a cascade of parameter set (each parameter set refers the previous set through to the source model). Alternatively target models could be exchanged with an explicit parameter set sequence (a set of references).

If model has one definitive set of parameter values then that set should be contained within the model. If a model is published in a context where more than one set of parameters described, in for example text, then those sets should be supplied in parameter sets as opposed to separate models. Parameter sets should be used to describe alternative parameter sets for defining mutations.

3 Implementation

3.1 XSLT

As parameter sets operate as transforms of source models it is natural to consider the generic XML transformation technology: the Extensible Stylesheet Language Transformations (XSLT) (cla). This standard is

well documented so rather than give an detailed explanation of this technology I will simply give a simple example.

Consider the following Level 2 model fragment:

```
...
<species id="P0" boundaryCondition="false" initialAmount="0" compartment="compartment"/>
<species id="T0" boundaryCondition="false" initialAmount="0" compartment="compartment"/>
<species id="P1" boundaryCondition="false" initialAmount="0" compartment="compartment"/>
<species id="T1" boundaryCondition="false" initialAmount="0" compartment="compartment"/>
<species id="P2" boundaryCondition="false" initialAmount="0" compartment="compartment"/>
<species id="T2" boundaryCondition="false" initialAmount="0" compartment="compartment"/>
<species id="C" boundaryCondition="false" initialAmount="0" compartment="compartment"/>
<species id="Cn" boundaryCondition="false" initialAmount="0" compartment="compartment"/>
...
<reaction id="J23" reversible="false">
<listOfReactants>
  <speciesReference species="Mt" stoichiometry="1" />
</listOfReactants>
<listOfProducts>
  <speciesReference species="NN" stoichiometry="1" />
</listOfProducts>
<kineticLaw>
<math xmlns="http://www.w3.org/1998/MathML">
  ...
</math>
<listOfParameters>
  <parameter id="kd_24" value="0.01" />
  <parameter id="V_24" value="1.2" />
  <parameter id="Km_24" value="0.2" />
</listOfParameters>
</kineticLaw>
</reaction>
...
```

If we apply the following XSLT to this model:

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

<xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.1"
    xmlns:sbml="http://www.sbml.org/sbml/level2" exclude-result-prefixes="sbml">

<xsl:template match="sbml:species[@id='P0']/@initialAmount">
<xsl:attribute name="initialAmount">5</xsl:attribute>
</xsl:template>

<xsl:template match="sbml:species[@id='C']/@initialAmount">
<xsl:attribute name="initialAmount">10</xsl:attribute>
</xsl:template>

<xsl:template match="@*|node()" >
  <xsl:copy>
    <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

The first two `template` elements overload the initial concentrations of species `P0` and `C` substituting values of 5 and 10 respectively. The final `template` element simply copies all other structures to the output.

We can cascade this parameter set with the following parameter set:

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

<xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.1"
    xmlns:sbml="http://www.sbml.org/sbml/level2" exclude-result-prefixes="sbml">
```

```
<xsl:import href="justa.xsl"/>

<xsl:template match="sbml:species[@id='C']/@initialAmount">
<xsl:attribute name="initialAmount">20</xsl:attribute>
</xsl:template>

<xsl:template match="sbml:reaction[@id='J23']//sbml:parameter[@id='V_24']/@value">
<xsl:attribute name="value">1.2</xsl:attribute>
</xsl:template>

</xsl:stylesheet>
```

The first `template` element overloads the previous `template` element to set the initial concentration of `C` to 20. The second `template` element replaces the value of the `V_24` parameter with 1.2.

The `import` element doesn't implement cascading in exactly the form described in section 2.5. The cascading overloads the matching of `template` elements rather than creating a pipeline of transformations. The effect is roughly the same however.

To enable the XSLT to easily parsed independently of a model the form of the XSLT to be used is restricted. The last template element is always the recursive copy `template` element unless the XSLT imports that element as shown in the examples above. All other `template` elements will simply contain `attribute` elements. The value of the `template match` attribute will contain the abbreviated XPATH form shown in the examples. These XPATH strings can contain only references to SBML `reaction`, `parameter`, `species`, and `compartment` elements. Matches to these elements must be qualified with an equality match to their `id` attributes. The last part of the XPATH string must consist of a match to `value`, `initialAmount` or `volume` attributes.

The disadvantages of XSLT are:

- can't support transformation of implied structures but only those structures explicitly in the source model. This can be only overcome by defining the form of the implied structures or expanded form. See section 2.10.
- parsing XSLT as a parameter set independently of a source model is difficult and requires a restricted form of XSLT to be defined as the parameter set format standard.
- one parameter set can't be applied to both Level 1 and Level 2 without duplicating information (by duplicating templates).
- no way to optionally apply the XSLT to a specific model
- cascading will require the careful application of `import` elements

The advantages of using XSLT are:

- existing well documented and well supported standard
- very wide scope of transformation
- will support all future levels of SBML
- all other requirements of parameter sets are met by XSLT.

3.2 Non-Generic Format

This section describes a new non-generic format for parameter sets as a clear alternative to XSLT. There seems to be little point in developing another generic form like XSLT.

The scope of the format described here allows the initial values of parameters, species concentration and compartment volume to be set. To support modularity the values of these parameters in sub-models and specific instances of sub-models can be specified.

The UML form of this format is given in Figure 1.

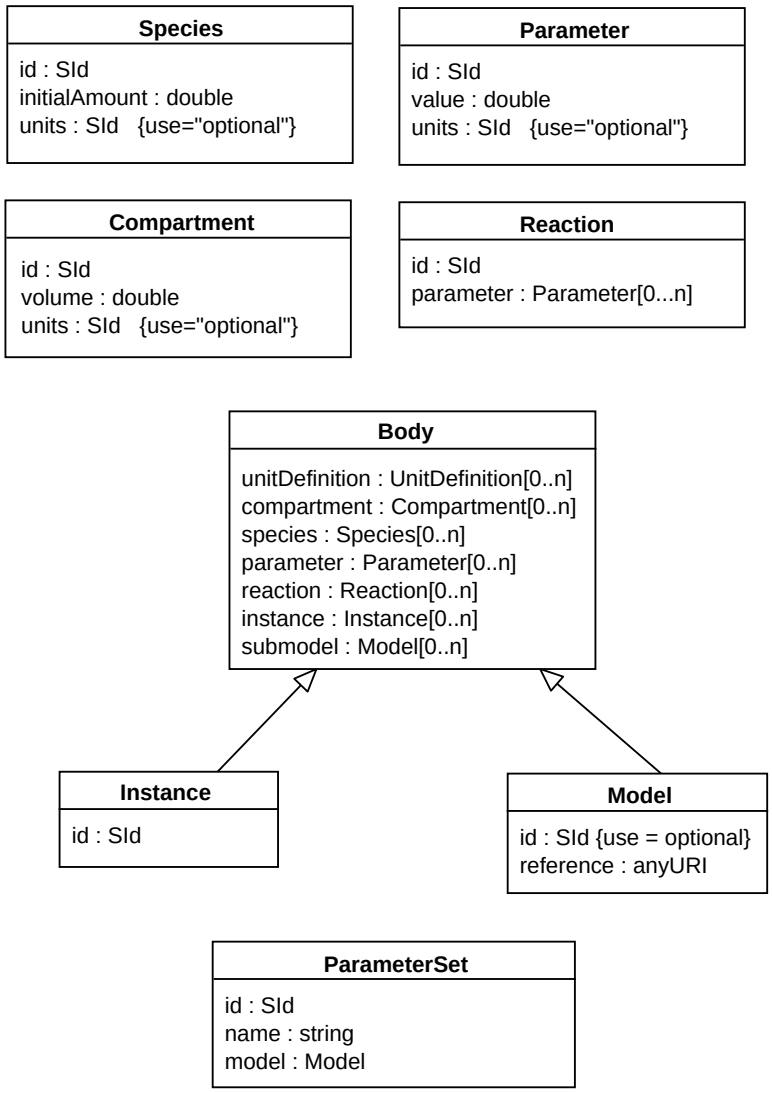

Figure 1: *The UML for a proposed format for parameter sets. Types without definitions are defined in SBML or XML Schema.* `ParameterSet` *is the top level element.*

The top-level element in this format is `parameterSet` element. Any element in this format, apart from a `parameterSet` element, refers to an element in SBML with the same class name and `id` attribute value. For example a `species` element with an `id` attribute `foo` refers to an SBML `species` element with an `id` attribute value `foo`. The `double` type attributes on the `species`, `parameter` and `compartment` elements overload the attributes, with same name, on the referenced SBML elements.

The `reaction`, `instance` and `model` elements can enclose other parameter set elements (to any depth) and define the context in which those enclosed elements have effect. For example a `parameter` element enclosed in a `reaction` element refers specifically to the SBML `parameter` element enclosed in the referenced SBML `reaction` element.

The `parameterSet` element contains a single `model` element which refers to the SBML model that the parameter set can be applied to. This top most `model` element has an optional `id` attribute indicating the model that the parameter set can be applied to. If this attribute is missing then the parameter set can be applied to an SBML model with an arbitrary `id` value. The optional `reference` attribute contains the URL

of the referenced model or the previous parameter set in a cascade.

As an example here are the parameter sets corresponding the XSLT transformations given above. First a parameter set overloading the initial concentrations of species `P0` and `C`.

```
<?xml version="1.0" encoding="UTF-8"?>
<parameterSet xmls="/http://www.sbml.org/parameterSet">
  <model>
    <listOfSpecies>
      <species id="P0" initialAmount="5"/>
      <species id="C" initialAmount="10"/>
    </listOfSpecies>
  </model>
</parameterSet>
```

Secondly a parameter set, overloading the previous set, setting the initial concentration of species `C` and the value of parameter `V_24` of reaction `J23`.

```
<?xml version="1.0" encoding="UTF-8"?>
<parameterSet xmls="/http://www.sbml.org/parameterSet">
  <model reference="justa.xml">
    <listOfSpecies>
      <species id="C" initialAmount="20"/>
    </listOfSpecies>
    <listOfReactions>
      <reaction id="J23">
        <listOfParameters>
          <parameter id="V_24" value="1.2"/>
        </listOfParameters>
      </reaction>
    </listOfReactions>
  </model>
</parameterSet>
```

The `model` element, as well as containing lists of `species`, `parameter` and `compartment` elements, contains lists of `model` and `instance` elements to support SBML Level 3 model composition. Following the pattern outlined above `model` elements contained inside other `model` elements refer to SBML sub-models and must have an `id` attribute value. `instance` elements follow a similar pattern however the elements enclosed inside `instance` elements refer to SBML elements in the sub-model that the SBML `instance` is an instance of. These sub-elements only apply to the specific instance of the sub-model i.e. the structures "implied" by the SBML `instance` element. Whereas the elements enclosed inside `model` elements refer to actual SBML elements within the referenced SBML `model` element those inside `instance` elements do not. At a given hierarchical level the values enclosed in `instance` elements overload those enclosed in a `model` element.

The values of `units` attributes should be either the `id` of a `unitDefinition` in either the enclosing `model` element or referenced SBML `model` element or a built-in unit such as `time`, `volume` or `substance`. The units simply define the units of the double value on the given element The unit definition elements must not overload the `time`, `volume` and `substance` built-ins.

The format described in this section addresses the majority of requirements outlined in section 2 it is not a generic format and thus cannot be immediately be used to define a wider range of transformations or be applied to later SBML Levels without defining more elements for the format.

References

[cla] XSL transformations (xslt) version 1.0.

[1] Hucka, M., Finney, A., Sauro, H. M., and Bolouri, H. (2001). Systems Biology Markup Language (SBML) Level 1: Structures and facilities for basic model definitions. Available via the World Wide Web at `http://www.cds.caltech.edu/erato`.