BIOINFORMATICS ORIGINAL PAPER

Vol. 26 no. 7 2010, pages 932–938
doi:10.1093/bioinformatics/btq069

Systems biology

Advance Access publication February 21, 2010

SBRML: a markup language for associating systems biology data with models

Joseph O. Dada[1,2], Irena Spasić[1,2], Norman W. Paton[1,2] and Pedro Mendes[1,2,3,*]

[1]Manchester Centre for Integrative Systems Biology, Manchester Interdisciplinary Biocentre, The University of Manchester, 131 Princess Street, Manchester M1 7DN, [2]School of Computer Science, The University of Manchester, Kilburn Building, Oxford Road, Manchester M13 9PL, UK and [3]Virginia Bioinformatics Institute, Virginia Tech, Washington Street MC 0477, Blacksburg, VA 24061, USA

Associate Editor: Trey Ideker

ABSTRACT

Motivation: Research in systems biology is carried out through a combination of experiments and models. Several data standards have been adopted for representing models (Systems Biology Markup Language) and various types of relevant experimental data (such as FuGE and those of the Proteomics Standards Initiative). However, until now, there has been no standard way to associate a model and its entities to the corresponding datasets, or vice versa. Such a standard would provide a means to represent computational simulation results as well as to frame experimental data in the context of a particular model. Target applications include model-driven data analysis, parameter estimation, and sharing and archiving model simulations.

Results: We propose the Systems Biology Results Markup Language (SBRML), an XML-based language that associates a model with several datasets. Each dataset is represented as a series of values associated with model variables, and their corresponding parameter values. SBRML provides a flexible way of indexing the results to model parameter values, which supports both spreadsheet-like data and multidimensional data cubes. We present and discuss several examples of SBRML usage in applications such as enzyme kinetics, microarray gene expression and various types of simulation results.

Availability and Implementation: The XML Schema file for SBRML is available at http://www.comp-sys-bio.org/SBRML under the Academic Free License (AFL) v3.0.

Contact: pedro.mendes@manchester.ac.uk

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on November 27, 2009; revised on February 8, 2010; accepted on February 16, 2010

1 INTRODUCTION

Systems biology is typically carried out with a solid basis on explicit (computational) models, which are used to guide traditional experimentation as well as data analysis. Computational models are at the core of the systems biology methodology, and therefore an important part of the infrastructure needed for practicing systems biology.

The growing reliance on computational models to support biological research has given rise to several types of modelling software. There are software packages for creation and visualization of models (Funahashi *et al.*, 2003), for their analysis and simulation (Hoops *et al.*, 2006; Moraru *et al.*, 2008) and also for analysing experimental data in their context (Shannon *et al.*, 2003). Additionally, there are several databases for sharing models within the community (Le Novére *et al.*, 2006; Olivier and Snoep, 2004).

The most prominent standard is the Systems Biology Markup Language (SBML; Hucka *et al.*, 2003), which is an XML-based language for representing systems biology models in a way that is largely independent from the means to simulate those models. Models represented in SBML are thus interpretable by a wide range of software, which can manipulate them in different ways: simulate their dynamics through ordinary differential equations, stochastic simulation algorithms, Petri nets or other formalisms; analyse their underlying stoichiometric properties; fit their parameters to experimental data; explore their parameter space by parameter scans or bifurcation analysis and many more computational applications. The existence of a standard way to represent models has been a catalyst for the appearance of these diverse types of software, since they have a common basis in SBML, allowing researchers to share models and effectively use them with these software tools.

Another important standard in the area is MIRIAM (Le Novére *et al.*, 2005), which is a set of guidelines to be followed when communicating models. More specifically, MIRIAM provides a means of unequivocally identifying biological molecules [through the use of resource description framework (RDF) and universal identifiers]. MIRIAM also recommends that models should be encoded in a machine-readable format, and that their authorship and terms of distribution should be specified explicitly. While MIRIAM does not prescribe which machine-readable format should be used to encode the model, SBML is a convenient way to do so, and the combination of these two standards has become a predominant way to specify self-contained models (Herrgård *et al.*, 2008).

The Simulation Experiment Description Markup Language (SED-ML; Köhn and Le Novére, 2008) is another XML-based standardization effort for describing computational simulation experiments.

Despite the increasing popularity of SBML and MIRIAM, which has resulted in many models now being available in electronic form, there is currently no standard way of communicating the actual *results* of the operations carried out on such models

*To whom correspondence should be addressed.

(e.g. simulations). Because of this lack of a standardized way to communicate model-derived data, it is very difficult to share such results between different software applications. Such activities have to be done in ways that require *ad hoc* programs to transform the data formats appropriately.

Here, we propose a new markup language which is intended to specify results from operations carried out on models. We name this format the Systems Biology Results Markup Language (SBRML). While developing SBRML as a means of communicating simulation results, it became obvious that it is equally useful to associate any kind of experimental data to a model, something that seems to be missing in the landscape of computational systems biology. SBRML is therefore a means of specifying any kind of quantitative results in the context of a systems biology model. Some of its major uses are:

(1) associating experimental results with models for passing to analysis tools;
(2) sharing and archiving of model simulations; and
(3) recording the results of analysis for validation, archiving or comparison.

The rest of this article describes the overall structure of SBRML documents and illustrates a number of use cases that are expected to cover the most common applications. Finally, SBRML is discussed in the context of a more complete scenario of computational activities centered on biological models, and therefore showing its relation with other existing and emerging standards.

2 METHODS

SBRML is based on XML (Bray *et al.*, 2008) and is specified through the XML Schema language (Biron and Malhotra, 2004). SBRML Object Model (SBRML-OM) was first developed using the Universal Modelling Language (UML; OMG, 2007) and a Model-Driven Architecture approach was then used to derive the corresponding XML Schema semi-automatically with the help of mapping rules for classes and associations.

In order to test the practicality of data formatting in SBRML, a prototype implementation was created with the software COPASI (Hoops *et al.*, 2006). This was done using COPASI's report definition facilities, which allow one to create output in very flexible ways. We do not foresee that this method will be the best one with which to produce SBRML results from COPASI, and we plan to write a full-fledged SBRML output generator in that package. Indeed it is the authors' expectation that in the future many other systems biology packages will provide means for exporting SBRML.

Since the main objective of SBRML is to associate data with a model, it therefore needs to provide representations of both the model and the data along with mechanisms to associate parts of the dataset with the corresponding elements of the model. Accordingly, SBRML has two major sections, one representing the model, and the other one describing the dataset. Since there is a plethora of very different types of data that may require specialized terms for their description such as concentration, particle numbers, flux, spectrophotometry, etc., it seemed too restrictive to define a priori a controlled vocabulary of terms (this would either be an extremely long and diverse list or otherwise incomplete). Instead, the solution was to provide a third major section in SBRML that lists ontology terms used in the rest of the document and refers to where such ontologies are defined. This allows the language to be extensible and cover any possible type of data.

SBRML-OM top-level classes are shown in Figure 1. The abstract class SBRBase provides a means of attaching arbitrary information on some elements of SBRML through its association with Annotation and Notes classes. The Sbrml class, which is subclass of SBRBase, has three required

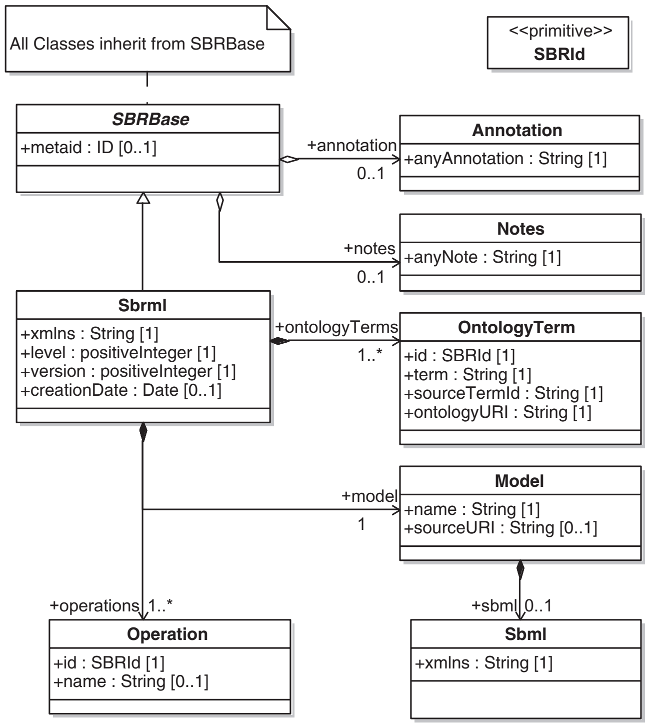


Fig. 1. Top-level classes of SBRML-OM.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbrml xmlns="http://www.sbrml.org/sbrml/level1/version1" level="1" version="1" creationDate="2009-11-27">
  <ontologyTerms>
    <ontologyTerm id="term1" term="..." sourceTermId="..."     ontologyURI="..." />
    <ontologyTerm id="term2" term="..." sourceTermId="..." ontologyURI="..." />
  </ontologyTerms>
  <model name="Curien2003_MetThr_synthesis" sourceURI="urn:miriam:biomodels.db:BIOMD0000000068" />
  <operations>
     <operation>
         ...
     </operation>
        ...
  </operations>
</sbrml>
```

Fig. 2. The skeleton of systems biology results description in SBRML showing only the top-level elements and their subelements.

attributes: the SBRML namespace, level and version, and an optional attribute creation date. An SBRML document will in many cases be used to encode the results of many operations performed on a single model. All operations carried out on the model are defined within the operations element indicated as an association between Sbrml and Operation classes. The ontologyTerms association between Sbrml and OntologyTerm classes as shown in Figure 1 is an element in the instance of Sbrml (SBRML document), which contains instances of OntologyTerm class. Figure 2 shows the skeleton of systems biology results description in SBRML. The top level classes are described in more detail below.

2.1 Ontology section

The use of vocabularies/terms from standard ontology sources to describe various types of data associated with the model is important in order for software tools to correctly interpret the data. There is no single ontology that can provide all the terms needed for the description of the very diverse systems biology data. SBRML provides the OntologyTerm class (Fig. 1) as a mechanism for representing controlled vocabulary terms from different ontologies. The id attribute provides a unique identifier for the ontology term. The term attribute stores the term itself, while the sourceTermId is a

```
<ontologyTerms>
  <ontologyTerm id="term1" term="concentration" sourceTermId="SBO:0000196" ontologyURI="http://www.ebi.ac.uk/sbo/" />
  <ontologyTerm id="term2" term="Spectrophotometry" sourceTermId="C0037805"
      ontologyURI="http://www.nlm.nih.gov/research/umls/" />
</ontologyTerms>
```

Fig. 3. SBRML fragment for encoding ontology terms.

string that is used within the ontology to uniquely identify the concept being referenced. The ontologyURI attribute specifies the unique identifier of the ontology. There will always be some terms that are not yet available in any ontology. Such terms can still be used for data description by using SBRML assigned term identifier and Uniform Resource Name (URN) for the terms. It is highly desirable that the ontologies used are those commonly accepted by the systems biology community, and expressed by their MIRIAM URN (Laibe and Le Novére, 2007). The fragment of SBRML in Figure 3 illustrates how to use terms from the external ontologies in SBRML.

2.2 Model section

SBRML associates data generated from operations with the model variables and their parameter values. The class Model as shown in Figure 1 defines the model used in the operations. The model must have a name attribute. The sourceURI attribute defines the source of the model. If the sourceURI attribute is not specified, the actual SBML representation of the model (see Model association to Sbml class in Fig. 1) must be carried within the instance of Model (inline). Providing a reliable way of accessing the model is important since an SBRML document is intended to be interpreted in the context of a particular model. An SBRML document without an associated model is therefore not a valid SBRML document. The inline representation of the model is more reliable since the model and the data are contained within the same file and therefore will never become separated. Additionally, when the source URI of model is used, there is always a possibility that the model may become incompatible with the data due to external changes to the SBML model, or that the model is no longer available. On the other hand, the inline representation is less practical with respect to the space and time needed to store or exchange the files. We recommend that a URI be used when the model is available in a reliable and strictly regulated repository (such as BioModels); in other circumstances it would be more prudent to include the SBML model inline within the SBRML.

2.3 Results section

As mentioned earlier, an SBRML document will in many cases be used to encode the results of many operations performed on a single model. An operation is defined as an object of the Operation class (Fig. 4). The id attribute is a unique identifier for an instance of Operation. The association to OntologyTerm defines the name of the instance of the Operation in an external ontology source. An operation is characterized by a method, which is associated with a particular piece of software (in the case of simulation this is the simulator software and in the case of experimental data this may be the data acquisition software). The Method and Software classes define the method and software, respectively, that are used by the operation. The actual results of the operation performed on the model are defined in the Result class, and described in detail below. The SBRML fragment in Figure 5 illustrates how to encode the operation performed on model.

The Result class (Fig. 6) provides a flexible structure for associating the data generated by an operation with the model. The actual result is defined by the ResultComponent class. The result in SBRML has two component parts: the description of the result represented by an abstract class DimensionDescription and the result itself defined by the abstract class Dimension. There must be at least one instance of ResultComponent in an instance of Result. The instance of ResultComponent is uniquely identified by an id attribute.

The DimensionDescription describes the structure of the ResultComponent and has three subclasses: CompositeDescription, TupleDescription and AtomicDescription. CompositeDescription describes

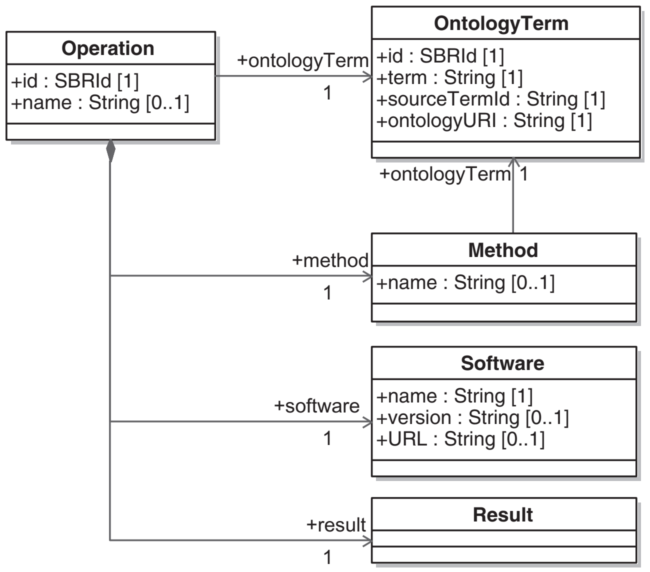


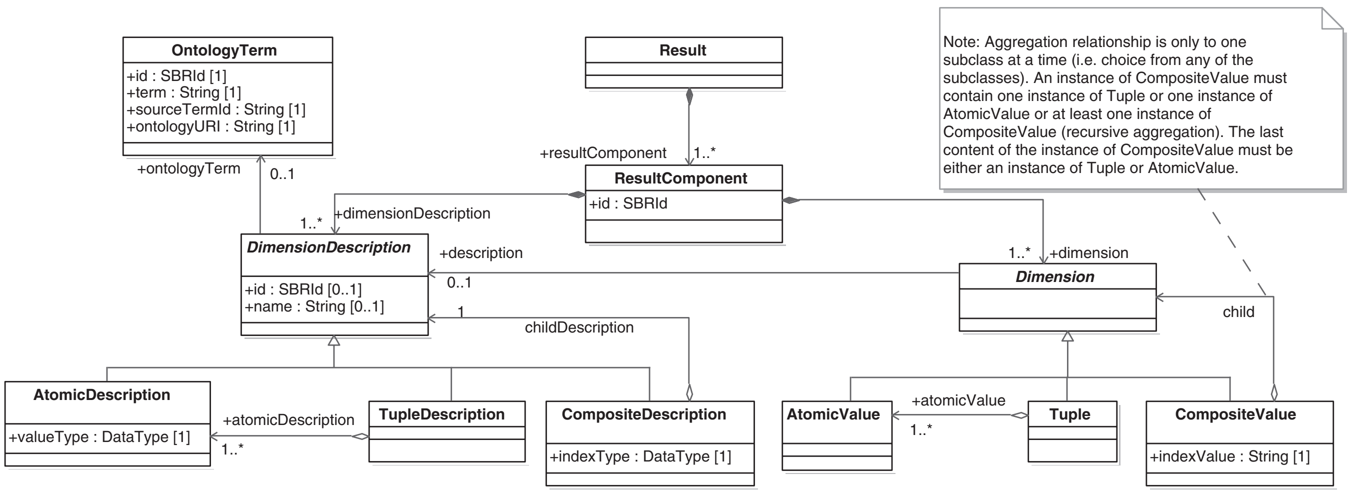
Fig. 4. SBRML-OM—operation class and its associations.

```
<operation id="op1" name="Parameter scan" ontologyTerm="...">
    <method name="Newton method" ontologyTerm="..." />
    <software name=" COPASI" version="COPASI 4.4 Build 26" URL="http://www.copasi.org/download" />
    <result>
      ...
    </result>
</operation>
```

Fig. 5. SBRML fragment for encoding operation performed on biochemical model.

the nesting of dimensions in a result. It has an implicit relationship to CompositeValue, in that any result described by an instance of CompositeDescription must be placed in an instance of CompositeValue. The type of data of the indexValue attribute in CompositeValue is defined by the indexType attribute. Possible values for indexType attribute are standard data types such as string, float, double, integer, etc. An instance of CompositeDescription must contain exactly one instance of any of AtomicDescription, TupleDescription or CompositeDescription. The inherited association to the OntologyTerm allows the ontology term defined by an external ontology source for the name attribute to be referenced. Where results contain structured components that are not represented as distinct dimensions, the structure is described using the TupleDescription, which has an implicit relationship to Tuple. Any result that is described by an instance of TupleDescription must be placed in an instance of Tuple. An instance of TupleDescription must have at least one instance of AtomicDescription, and only one instance of TupleDescrption is allowed within the instance of CompositeDescription. The id and name attributes inherited from the super class are optional for this class. AtomicDescription is used to describe a value in a result that can no longer be subdivided. It has an implicit relationship to AtomicValue. Any result that is described in the instance of AtomicDescription must be placed in the instance of AtomicValue. It has a required valueType attribute that defines the type of data contained in the instance of AtomicValue.

It is sometimes useful to think of data as a set of numbers indexed in an array. A spreadsheet is an instance of such structure in two dimensions. In three or more dimensions we have so-called data cubes. In SBRML, the Dimension class (within a ResultComponent) allows us to describe the model element that is being used to index one of the dimensions of such data structures. For example, a time course is usually a 2D data structure, which associates species concentrations or particle numbers with discrete



OntologyTerm
+id : SBRId [1]
+term : String [1]
+sourceTermId : String [1]
+ontologyURI : String [1]
Result
Note: Aggregation relationship is only to one subclass at a time (i.e. choice from any of the subclasses). An instance of CompositeValue must contain one instance of Tuple or one instance of AtomicValue or at least one instance of CompositeValue (recursive aggregation). The last content of the instance of CompositeValue must be either an instance of Tuple or AtomicValue.
+resultComponent
1..*
+ontologyTerm
0..1
ResultComponent
+id : SBRId
1..*
+dimensionDescription
DimensionDescription
+description
1..*
+dimension
0..1
Dimension
+id : SBRId [0..1]
+name : String [0..1]
1
childDescription
child
AtomicDescription
+valueType : DataType [1]
+atomicDescription
1..*
TupleDescription
CompositeDescription
+indexType : DataType [1]
AtomicValue
+atomicValue
1..*
Tuple
CompositeValue
+indexValue : String [1]

Fig. 6. SBRML-OM—result classes and their associations.

values of time. In this case, the link between the data and the model are the species identifiers. The time course data is thus indexed by time and by species identifiers. Each discrete value of time identifies a set of species concentrations, which correspond to the state of the system at that particular time value; a species identifier provides a further index to the concentration of the corresponding species at that time value. If both concentration and particle number of a species are to be encoded together, the species identifier indexes a tuple that is used to structure those two items, rather than a single number. The indexing mechanism is provided by the indexValue attribute. The value of this attribute in an SBRML document provides an important link between the data, model variables and model parameter values.

The Dimension uses three subclasses: CompositeValue, Tuple and AtomicValue to structure the data as shown in Figure 6. Section 3 provides various examples of how the Result model can be used to encode different types of systems biology data.

It is common to express experimental results associated with some measurement of error. SBRML allows for this by including one, or several, error estimates as a specific AtomicValue entries.

3 RESULTS

In this section, we provide examples of SBRML usage which fall under two main types: output from systems biology modelling software, and input to systems biology data analysis software. The first type includes essentially simulation results, while the second associates experimental data with models. These examples illustrate the breadth of applications that can be addressed by SBRML. The examples also cover the two different ways of structuring data, tuples and data hypercubes, indicating the situations where each one is more appropriate.

3.1 Example 1: a single steady state calculation

This is one of the simplest applications of SBRML as a means of formatting output from a simulation. The SBRML example in Figure 7 encodes the results (species concentration and reaction fluxes) of a steady state solution of model number 68 (Curien *et al.*, 2003) in the BioModels database (Le Novére *et al.*, 2006). The first ontologyTerms section file starts by defining the concepts of ‘steady state’, ‘Newton method’, ‘concentration’ and ‘flux’, which are used subsequently in the file. The model is referenced with the appropriate BioModels URN. Only one operation is specified, which is a steady state calculation, and it is associated with two resultComponent elements. The first resultComponent contains the steady state concentration of the only variable of this model (Phser). The second resultComponent contains values of the fluxes of three reactions (v1, vCys, vThr). Both resultComponents are presented as tuples (name, value). Note that the numerical data is always associated with the identifiers from the SBML model. This allows anyone to interpret these numbers within the appropriate context. Together, the original SBML file and this SBRML file completely specify the model, the simulation carried out and the results obtained.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbrml xmlns="http://www.sbrml.org/sbrml/level1/version1"  level="1" version="1" creationDate="2009-11-27">
 <ontologyTerms>
  <ontologyTerm id="term1" term="steady state" sourceTermId="TEDDY_0000011"
    ontologyURI="http://teddyontology.sourceforge.net/teddy/rel-2007-09-03/ontology/teddy.owl" />
  <ontologyTerm id="term2" term="Newton method" sourceTermId="SBRML:00003"
    ontologyURI="urn:sbrml:ontologyterms" />
  <ontologyTerm id="term3" term="concentration" sourceTermId="SBO:0000196"
    ontologyURI="http://www.ebi.ac.uk/sbo/" />
  <ontologyTerm id="term4" term="flux" sourceTermId="C2348693"
    ontologyURI="http://www.nlm.nih.gov/research/umls/" />
 </ontologyTerms>
 <model name="Curien2003_MetThr_synthesis" sourceURI="urn:miriam:biomodels.db:BIOMD0000000068" />
 <operations>
  <operation id="op1" name="Steady State" ontologyTerm="term1">
   <method name="Newton method" ontologyTerm="term2" />
   <software name=" COPASI" version="COPASI 4.4 Build 26" URL="http://www.copasi.org/download" />
   <result>
    <resultComponent id="component1">
     <dimensionDescription>
      <compositeDescription name="species" indexType="string">
       <atomicDescription name="Concentration" ontologyTerm="term3" valueType="double" />
      </compositeDescription>
     </dimensionDescription>
     <dimension>
      <compositeValue indexValue="Phser">
         <atomicValue>141.063</atomicValue>
      </compositeValue>
     </dimension>
    </resultComponent>
    <resultComponent id="component2">
     <dimensionDescription>
      <compositeDescription name="Reaction" indexType="string">
       <atomicDescription name="Flux" ontologyTerm="term4" valueType="double" />
      </compositeDescription>
     </dimensionDescription>
     <dimension>
      <compositeValue indexValue="v1">
       <atomicValue>1</atomicValue>
      </compositeValue>
      <compositeValue indexValue="vCys">
       <atomicValue>0.152172</atomicValue>
      </compositeValue>
      <compositeValue indexValue="vThr">
       <atomicValue>0.847828</atomicValue>
      </compositeValue>
     </dimension>
    </resultComponent>
   </result>
  </operation>
 </operations>
</sbrml>
```

Fig. 7. Example SBRML file describing results of a steady state solution of model 68 of the BioModels database.

```
<operation id="op1" name="Time Course" ontologyTerm="...">
   <method name="Deterministic (LSODA)" ontologyTerm="..." />
   <software name=" COPASI" version="COPASI 4.4 Build 26" URL="http://www.copasi.org/download" />
   <result>
    <resultComponent id="component1">
     <dimensionDescription>
      <compositeDescription name="Time" ontologyTerm="..." indexType="double">
       <compositeDescription name="species" indexType="string">
        <tupleDescription>
         <atomicDescription name="Concentration" ontologyTerm="..." valueType="double" />
         <atomicDescription name="Particle Numbers" ontologyTerm="..." valueType="Integer" />
        </tupleDescription>
       </compositeDescription>
      </compositeDescription>
     </dimensionDescription>
     <dimension>
      <compositeValue indexValue="0">
       <compositeValue indexValue="Phser">
        <tuple>
         <atomicValue>0</atomicValue>
         <atomicValue>0</atomicValue>
        </tuple>
       </compositeValue>
      </compositeValue>
      <compositeValue indexValue="1">
       <compositeValue indexValue="Phser">
        <tuple>
         <atomicValue>0.996305</atomicValue>
         <atomicValue>5.99989e+17</atomicValue>
        </tuple>
       </compositeValue>
      </compositeValue>
      <compositeValue indexValue="2">... </compositeValue>
     </dimension>
    </resultComponent>
   </result>
  </operation>
```

Fig. 8. SBRML fragment for time course simulation results of model number 68 of the BioModels database.

3.2 Example 2: a time course simulation

A time course simulation is a description of the evolution of the variables of a system (concentrations of chemical species, fluxes, etc.) for increasing values of time. If one considers time to be a parameter of the model, then a time course can be naturally described as a series of states of the system indexed by the parameter time. The SBRML fragment in Figure 8 shows the results of a time course simulation of model number 68 of BioModels. In this case the data are presented as a series of tuples indexed by values of time. Each tuple represents the concentration and the particle number of Phser, the only variable chemical species of this model. The Supplementary File example2.xml gives the complete SBRML document for this example.

3.3 Example 3: enzyme kinetic data

SBRML is not limited to representing results of simulations, it is also useful for connecting experimental results to a systems biology model. This example illustrates the use of SBRML to represent data from an enzyme kinetics experiment which is here associated with a model of the enzymatic reaction carried out in the experiment. Data are from Martins *et al.* (2001), where the enzyme glyoxalase I (EC 4.4.1.5) of *Saccharomyces cerevisiae* was assayed in a progress curve analysis. The data are thus similar to Example 2, whereby time is the varying parameter. In this case, however, eight time courses are represented, each one for a different value of the concentration of substrates and products. Each time course is included as a single resultComponent (see SBRML fragment in Fig. 9). Since the data were obtained through spectrophotometry, the SBML model includes the appropriate rule defining the absorbance at 240 nm as a product of the concentration of the product SDLGSH by its molar absortivity coefficient, making it explicit in the model the assumption made of how light absorption relates to the concentration of one of the chemical species. Note that the complete SBRML file for this example (see Supplementary File example3.xml) includes the SBML file inline as it is not available in a persistent repository.

3.4 Example 4: microarray data

A major data source in functional genomics often used in systems biology (e.g. Castrillo *et al.*, 2007) consists of measurements of RNA with microarray technology. In this example, we illustrate the use of SBRML to represent

```
<operation id="operation1" name="Time Course" ontologyTerm="...">
   <method name="Spectrophotometry" ontologyTerm="..." />
   <result>
    <resultComponent id="experiment1">
     <dimensionDescription>
      <compositeDescription id="exp1_descr1" name="species" indexType="string">
       <atomicDescription name="initial concentration" indexType="double" />
      </compositeDescription>
      <compositeDescription id="exp1_descr2" name="Time" ontologyTerm="..." indexType="double">
       <compositeDescription name="Variable" ontologyTerm="..." indexType="string">
        <atomicDescription name="Value" ontologyTerm="..." valueType="double" />
       </compositeDescription>
      </compositeDescription>
     </dimensionDescription>
     <dimension>
      <compositeValue description="exp1_descr1" indexValue="HTA">
       <atomicValue>0.05</atomicValue>
      </compositeValue>
      <compositeValue description="exp1_descr1" indexValue="MG">
       <atomicValue>0.02</atomicValue>
      </compositeValue>
      <compositeValue description="exp1_descr1" indexValue="GSH">
       <atomicValue>0.76</atomicValue>
      </compositeValue>
      <compositeValue description="exp1_descr2" indexValue="60">
       <compositeValue indexValue="A240">
        <atomicValue>-0.01</atomicValue>
       </compositeValue>
      </compositeValue>
      <compositeValue description="exp1_descr2" indexValue="660">
       <compositeValue indexValue="A240">
        <atomicValue>0.08</atomicValue>
       </compositeValue>
      </compositeValue>
      <compositeValue description="exp1_descr2" indexValue="1260">
       <compositeValue indexValue="A240">
        <atomicValue>0.15</atomicValue>
       </compositeValue>
      </compositeValue>
         ...
      <compositeValue description="exp1_descr2" indexValue="3060">
       <compositeValue indexValue="A240">
        <atomicValue>0.25</atomicValue>
       </compositeValue>
      </compositeValue>
     </dimension>
    </resultComponent>
   <resultComponent id="experiment2">...</resultComponent>
   ...
   </result>
</operation>
```

Fig. 9. SBRML fragment showing how to encode enzyme kinetic data.

microarray data. The example includes the levels of expression of the genes encoding all enzymes of the pentose phosphate pathway in *S.cerevisiae* in different samples of an experiment available from the GEO database (a Pyocyanin dose-response, accession GDS2522, Angell *et al.*, 2006). The data are referenced to an (inline) SBML file of the pentose phosphate metabolic pathway which was obtained as a subset of the metabolic reconstruction of yeast in (Herrgård *et al.*, 2008). This file associates the gene expression levels of the various experiments with the (enzyme) modifiers of each reaction of the metabolic pathway. This connection between RNA levels and enzyme levels is, therefore, an assumption that is made explicitly in this SBRML file. SBRML is therefore a suitable means to declare such assumptions in a machine-readable format. Note that the SBML model in this case is annotated with MIRIAM-compliant (Le Novére *et al.*, 2005) RDF annotations for each molecule represented. Figure 10 shows an SBRML fragment of this example, the entire file is given as Supplementary Material.

3.5 Example 5: complex parameter scan

Parameter scans (or sweeps) are operations where many simulations are carried out where the values of several parameters of a model are changed in a regular pattern. This type of application is suitably represented as a data hypercube (sometimes referred to as a multidimensional spreadsheet). Again model number 68 of BioModels is used and in this case a series of simulations study the effect of the concentrations of cysteine (Cys) and *S*-adenosylmethionine (AdoMet) on the steady state fluxes of cystathieonine gamma-synthase (vCys) and threnine Synthase (vThr) as described in the original work (Curien *et al.*, 2003). This forms a 2D parameter scan, and therefore a 3D table is the most natural way of structuring the results. Alternatives to this would be to provide a single table where some of the columns would repeat the same value many times. The SBRML schema

```
<operation id="operation1" name="Microarray Analysis" ontologyTerm="...">
  <method name="DNA Microarray chips" ontologyTerm="..." />
  <result>
   <resultComponent id="micro1">
    <dimensionDescription>
     <compositeDescription name="sample" ontologyTerm="..." indexType="string">
      <compositeDescription name="species" ontologyTerm="..." indexType="string">
       <tupleDescription>
        <atomicDescription name="pyocyanin" valueType="integer" ontologyTerm="..." />
        <atomicDescription name="expression level" valueType="double" ontologyTerm="..." />
       </tupleDescription>
      </compositeDescription>
     </compositeDescription>
    </dimensionDescription>
    <dimension>
     <compositeValue indexValue="GSM142982">
      <compositeValue indexValue="YNL241C">
       <tuple>
        <atomicValue>0</atomicValue>
        <atomicValue>2155.7</atomicValue>
       </tuple>
      </compositeValue>
      <compositeValue indexValue="YGR248W">
       <tuple>
        <atomicValue>0</atomicValue>
        <atomicValue>724.8</atomicValue>
       </tuple>
      </compositeValue>
      <compositeValue indexValue="YHR163W">
       <tuple>
        <atomicValue>0</atomicValue>
        <atomicValue>1354.3</atomicValue>
       </tuple>
      </compositeValue>
     <compositeValue indexValue="YHR183W">...</compositeValue>
     ...
    </dimension>
   </resultComponent>
  </result>
<operation>
```

Fig. 10. SBRML fragment showing how to encode microarray data.

```
<operation id="op1" name="Parameter scan" ontologyTerm="...">
  <method name="Newton method" ontologyTerm="..." />
  <software name=" COPASI" version="COPASI 4.4 Build 26" URL="http://www.copasi.org/download" />
  <result>
   <resultComponent id="component1">
    <dimensionDescription>
     <compositeDescription name="species" indexType="string">
      <compositeDescription name="Initial Concentration" indexType="double">
       <compositeDescription name="species" indexType="string">
        <compositeDescription name="Initial Concentration" indexType="integer">
         <compositeDescription name="reaction" indexType="string">
          <atomicDescription name="flux" ontologyTerm="..." valueType="double" />
         </compositeDescription>
        </compositeDescription>
       </compositeDescription>
      </compositeDescription>
     </compositeDescription>
    </dimensionDescription>
    <dimension>
     <compositeValue indexValue="Cys">
      <compositeValue indexValue="0.3">
       <compositeValue indexValue="AdoMet">
        <compositeValue indexValue="0">
         <compositeValue indexValue="vCys">
          <atomicValue>0.0136504</atomicValue>
         </compositeValue>
         <compositeValue indexValue="vThr">
          <atomicValue>0.98635</atomicValue>
         </compositeValue>
        </compositeValue>
         ...
        <compositeValue indexValue="100">
         <compositeValue indexValue="vCys">
          <atomicValue>0.010766</atomicValue>
         </compositeValue>
         <compositeValue indexValue="vThr">
          <atomicValue>0.989234</atomicValue>
         </compositeValue>
        </compositeValue>
       </compositeValue>
      </compositeValue>
      <compositeValue indexValue="3">... </compositeValue>
      <compositeValue indexValue="300">... </compositeValue>
     </compositeValue>
    </dimension>
   </resultComponent>
  </result>
 </operation>
```

Fig. 11. SBRML fragment showing how to encode results of complex parameter scan operation on model number 68 of the BioModel database.

provides an easy way to represent this data cube simply by indexing the results (fluxes) with each of the varying parameters as shown in the SBRML fragment in Figure 11. The Supplementary File example5.xml gives the complete SBRML document.

While parameter scans change values of parameters in a regular pattern, parameter sampling changes values of parameters using random distributions. This means that there is no regularity in those values and, therefore, they are not appropriate for indexing results in a data hypercube fashion. For parameter sampling, it is most appropriate to represent the data as tuples.

4 DISCUSSION

It is often emphasized that one of the main characteristics of systems biology is the combined use of experiments and models (Kell, 2006; Kitano, 2002). Several standards already exist to express various aspects of systems biology in computational terms, such as Functional Genomics Experiment (FuGE; Jones *et al.*, 2007) for functional genomics data, the SBML (Hucka *et al.*, 2003) for network and kinetic models or BioPAX (BioPAX Working Group, 2008) for pathways. It is remarkable, however, that until now there have been no attempts at creating a standard way of computationally linking data with models. If systems biology is indeed to succeed as an integrative wet and dry biology exercise there must be a standard way to create associations of data with models. We see two major uses for this: (i) expressing results of computational manipulations of models (e.g. simulations) and (ii) expressing experimental results in the context of a model, mathematical or otherwise.

We have used the COPASI simulator to create the simulation results here, and since we are part of the development team of that software, future versions will provide easy means to export results in SBRML. But for this to be a successful exchange format in systems biology, it is important that other simulators provide similar capabilities, as well as other types of systems biology application including data analysis workflows, databases, etc. On the other hand, it is also important that relevant applications be capable of reading (and interpreting) SBRML.

Given that SBRML is too verbose for being readable by humans (even very dedicated computational systems biologists), we foresee the need for user-friendly SBRML readers. Such applications could format the data from SBRML in tables or data hypercubes or provide graphical displays of the data. Network visualization software such as CellDesigner (Funahashi *et al.*, 2003), CytoScape (Shannon *et al.*, 2003) or Ondex (Kohler *et al.*, 2006) would seem to be particularly appropriate for interpreting and displaying the contents of SBRML files. Example 4 illustrates a type of SBRML file that would benefit from being displayed in such applications. Data analysis software would also benefit from this format, such as applications for parameter estimation demonstrated with Example 3 (in this context, COPASI would also benefit from being able to read SBRML for parameter estimation).

As well as complementing SBML, SBRML can also be seen to complement experimental data standards, such as FuGE (Jones *et al.*, 2007) or MAGE-ML (Spellman *et al.*, 2002). Experimental data standards essentially describe samples, the experimental and analytical processes applied to those samples, and the results of those processes. As such, experimental data standards describe how results are derived from an experimental process, whereas SBRML indicates how results have been derived from an SBML model by inter-relating the model, the analytical process applied to the model and the results of the process.

SBRML and SED-ML are also complementary. While the main purpose of SBRML is to encode the simulation results and/or experimental data and all context in which it was obtained, SED-ML is used for a detailed description of the operations that generate simulation results. This means SED-ML could be used for a detailed description of the specific operations that led to the data in SBRML. One way to achieve this might be the inclusion of an SED-ML container in an SBRML file in similar way to SBML container in 'model' element of SBRML. We will look into this approach and other possible ways to combine SED-ML and SBRML in the future. Note that while SED-ML would be a perfect solution to describing computational operations, it does not provide any support for describing 'wet' experiments and thus SBRML still needs a mechanism for this purpose, which makes their integration not trivial.

It is our conviction that SBRML fills a current need in systems biology. We hope that this document and the SBRML specification stimulate discussion and implementations of this standard among the systems biology community.

ACKNOWLEDGEMENTS

We thank our colleagues in the SBML community who provided feedback about the SBRML specification. We thank Daniel Jameson, Peter Li, Neil Swainston and the rest of the MCISB staff for many helpful comments.

Funding: Biotechnology and Biological Sciences Research Council and Engineering and Physical Sciences Research Council (grant BB/C008129/2, a contribution from The Manchester Centre for Integrative Systems Biology).

Conflict of Interest: none declared.

REFERENCES

Angell,S. *et al.* (2006) Pyocyanin isolated from a marine microbial population: synergistic production between two distinct bacterial species and mode of action. *Chem. Biol.*, **13**, 1349–1359.

BioPAX Working Group (2008) BioPAX—biological pathways exchange language. Level 3, Release Candidate 3 (Version 0.92) Documentation. Available at http://www.biopax.org/release/biopax-level3-documentation.pdf.

Biron,P.V. and Malhotra,A. (2004) XML Schema part 2: Datatypes, W3C recommendation 28 October 2004. Available at http://www.w3.org/TR/xmlschema-2/.

Bray,T. *et al.* (2008) Extensible markup language (XML) 1.0 (third edition), W3C recommendation 26 November 2008. Available at http://www.w3.org/TR/REC-xml.

Castrillo,J.I. *et al.* (2007) Growth control of the eukaryote cell: a systems biology study in yeast. *J. Biol.*, **6**, 4.

Curien,G. *et al.* (2003) A kinetic model of the branch-point between the methionine and threonine biosynthesis pathways in *Arabidopsis thaliana*. *Eur. J. Biochem.*, **270**, 4615–4627.

Funahashi,A. *et al.* (2003) CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. *Biosilico*, **1**, 159–162.

Herrgård,M.J. *et al.* (2008) A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology. *Nat. Biotechnol.*, **26**, 1155–1160.

Hoops,S. *et al.* (2006) COPASI: a COmplex PAthway SImulator. *Bioinformatics*, **22**, 3067–3074.

Hucka,M. *et al.* (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**, 524–531.

Jones,A.R. *et al.* (2007) The Functional Genomics Experiment model (FuGE): an extensible framework for standards in functional genomics. *Nat. Biotechnol.*, **25**, 1127–1133.

Kell,D.B. (2006) Systems biology, metabolic modelling and metabolomics in drug discovery and development. *Drug Discov. Today*, **11**, 1085–1092.

Kitano,H. (2002) Systems biology: a brief overview. *Science*, **295**, 1662–1664.

Kohler,J. *et al.* (2006) Graph-based analysis and visualization of experimental results with ONDEX. *Bioinformatics*, **22**, 1383–1390.

Köhn,D. and Le Novére,N. (2008) SED-ML — An XML Format for the Implementation of the MIASE Guidelines. In Heiner,M. and Uhrmacher,A.M. (eds) Vol. 5307 of *Lecture Notes in Bioinformatics*. Springer, Berlin/Heidelberg, pp. 176–190.

Laibe,C. and Le Novére,N. (2007) MIRIAM Resources: tools to generate and resolve robust cross-references in Systems Biology. *BMC Systems Biol.*, **1**, 58.

Le Novére,N. *et al.* (2006) BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res.*, **34**, D689–D691.

Le Novére,N. *et al.* (2005) Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat. Biotechnol.*, **23**, 1509–1515.

Martins,A.M. *et al.* (2001) *In situ* kinetic analysis of glyoxalase I and glyoxalase II in *Saccharomyces cerevisiae*. *Eur. J. Biochem.*, **268**, 3930–3936.

Moraru,I.I. *et al.* (2008) Virtual cell modelling and simulation software environment. *IET Syst. Biol.*, **2**, 352–362.

Olivier,B.G. and Snoep,J.L. (2004) Web-based kinetic modelling using JWS online. *Bioinformatics*, **20**, 2143–2144.

Object Management Group (2007) Unified Modeling Language, Infrastructure, V2.1.2. Available at http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF.

Shannon,P. *et al.* (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, **13**, 2498–2504.

Spellman,P.T. *et al.* (2002) Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biol.*, **3**, research0046.1–0046.9.