SED-ML – An XML Format for the Implementation of the MIASE Guidelines

Dagmar Köhn[1] and Nicolas Le Novère[2]

[1] Research Training School dIEM oSiRiS, University of Rostock, Germany
[2] European Bioinformatics Institute, Hinxton, CB10 1SD, UK

Abstract. Share and reuse of biochemical models have become two of the main issues in the field of Computational Systems Biology. There already exist widely-accepted formats to encode the structure of models. However, the problem of describing the simulations to be run using those models has not yet been tackled in a satisfactory way. The community believes that providing detailed information about simulation recipes will highly improve the efficient use of existing models. Accordingly a set of guidelines called the Minimum Information About a Simulation Experiment (MIASE) is currently under development. It covers information about the simulation settings, including information about the models, changes on them, simulation settings applied to the models and output definitions. Here we present the Simulation Experiment Description Markup Language (SED-ML), an XML format that enables the storage and exchange of part of the information required to implement the MIASE guidelines. SED-ML is independent of the formats used to encode the models – as long as they are expressed in XML –, and it is independent of the software tools used to run the simulations. Several test implementations are being developed to benchmark SED-ML on simple cases, and pave the way to a more complete support of MIASE.

1 Introduction

As Systems Biology transforms into one of the main fields in life sciences, the number of available computational models is growing at an ever increasing pace. At the same time, their size and complexity are also increasing. The need to build on existing studies by reusing models therefore becomes more imperative. It is now generally accepted that one needs to be able to exchange the biochemical and mathematical structure of models. Guidelines, such as the *Minimum Information Requested in the Annotation of Models* (MIRIAM [1]), describe the information that needs to be exchanged to properly understand a model; computer formats, such as SBML [2] or CellML [3], allow people to implement those guidelines and exchange models between a large diversity of tools.

However, the computational modeling procedure is not limited to the definition of the model structure. According to the MIRIAM specification, "the model, when instantiated within a suitable simulation environment, must be able to reproduce all relevant results given in the reference description that can readily

M. Heiner and A.M. Uhrmacher (Eds.): CMSB 2008, LNBI 5307, pp. 176–190, 2008.

be simulated" [1]. MIRIAM does not impose to list those relevant results, or to describe how to obtain them. It became nevertheless clear that the description of simulation experiments was mandatory to correctly exchange, re-use and interpret models. This led to the development of the *Minimum Information About a Simulation Experiment* (MIASE). Obtaining a desired numerical result often requires to run complex simulation tasks on original and perturbed models. Furthermore, the same model can provide various results when simulated using different approaches. Well-known examples are systems that exhibit steady-state when simulated with deterministic approaches, and oscillation or multistationarity when simulated with stochastic methods. MIASE addresses exactly these problems by providing a list of mandatory information required for the production – or reproduction – of a given set of simulation results. This information can be split into the following four categories:

Information about the models simulated
MIASE recommends to explicitly define all models used in a simulation by providing a specific name and the source of each model. The use of a model as such is often not sufficient to get a desired simulation result, therefore changes that have to be applied to the model before the simulation must be described in detail. Examples are the assignment of a new value (e. g. constant, initial concentration), or the change of a mathematical expression (e. g. using different enzyme kinetics).

Information about the simulation methods used
Each simulation can be characterized by certain types of simulation procedures to be run (e. g. steady-state, time course) and the simulation algorithms used to perform them. The information has to be sufficiently detailed so that no arbitrary choices have to be made when setting up the simulations.

Information about the tasks performed
Once simulation settings and changes on the models have been defined, the simulation tasks undertaken to complete the simulation experiment need to be specified. Typically, that will involve describing how a simulation procedure has to be applied to a specific model, and in which order.

Information about the outputs produced
It is often necessary to define the transformations that have to be performed on the raw output of the simulation tasks, and how to provide the final results. These results can be numerical or graphical. For instance, a model of a periodic process can provide just time courses showing oscillations; or it can, on the contrary, provide phase diagrams, which are more explicit in describing the relationship between variables. An even more striking example of the necessity for output definitions is the bifurcation diagram.

The adoption of MIASE will be greatly fastened, both on the generation and the reuse sides, if the required information is encoded in a standard format – produced and understood by simulation software. The object model (SED-OM) presented in this paper is a platform independent prototype model encoding MIASE guidelines for simple simulation experiments. We also present an XML based implementation of that model (SED-ML) which is introduced with

a detailed example in section 3. Related efforts are compared and discussed in section 4.

2 The Simulation Experiment Description Object Model

The *Simulation Experiment Description Object Model* (SED-OM) is a formal representation of the MIASE guidelines using the Unified Modeling Language (UML [4]). The top-level classes of the SED-OM can be seen in Figure 1 and will be described in more detail in the following section. For clarification, the SED-OM class names are put in brackets using `typewriter` font.

2.1 Information on the Model and Model Changes

A MIASE based simulation description will in many cases make use of more than just one model. That is why all models have to be defined clearly for later reference. In SED-OM, all models involved in the simulation experiment are in a list of models (see Figure 2). Each model (`Model`) has its own unambiguous identifier (`id`). Additionally, it may have a name (`name`) and it may hold information about its encoding (`type`). As most simulation tools support only particular formats, it is strongly recommended to provide the type of model encoding (e. g. CellML). Information about the model format helps simulation tools to decide whether the model can be loaded directly or has to be converted into another format first. SED-OM also requires the source of the model to be defined (`source`). It is not within the scope of SED-OM to store model representations, but to provide a secure way of accessing them. The source should be reliable, meaning it should point to a repository of curated models in order to ensure the correctness and validity of the model.

Models often need to be modified before subjected to a simulation task. Those changes can be direct atomic changes on simple attributes of a model, such as changes on a parameter or on the initial concentration (`ChangeAttribute`). Complex changes, depending on other values, can be described using mathematical expressions (`ChangeMath`). Those expressions are mathML [5] constructs that are made up of parameters defined in a list of parameters (`Parameter`) and variables defined in a list of variables (`Variable`). A variable object holds a reference to an already defined model and targets a certain XML element within that model using XPath. Finally, a general class (`ChangeXML`) allows to replace any piece of XML code by another valid one, including void which amounts to a deletion.

The XML Path Language (XPath [6]) has been chosen to target model elements. Apart from being a natural choice when working with XML files, XPath expressions allow to unambiguously identify any (syntactical) part of a model that can be altered. XPath offers a very convenient way of describing changes on the model independently of the actual model representation format: An XPath expression defines a path through an XML document and points to a particular XML element or XML attribute within the document. The only restriction imposed upon the model by the usage of XPath is that it has to be available in an

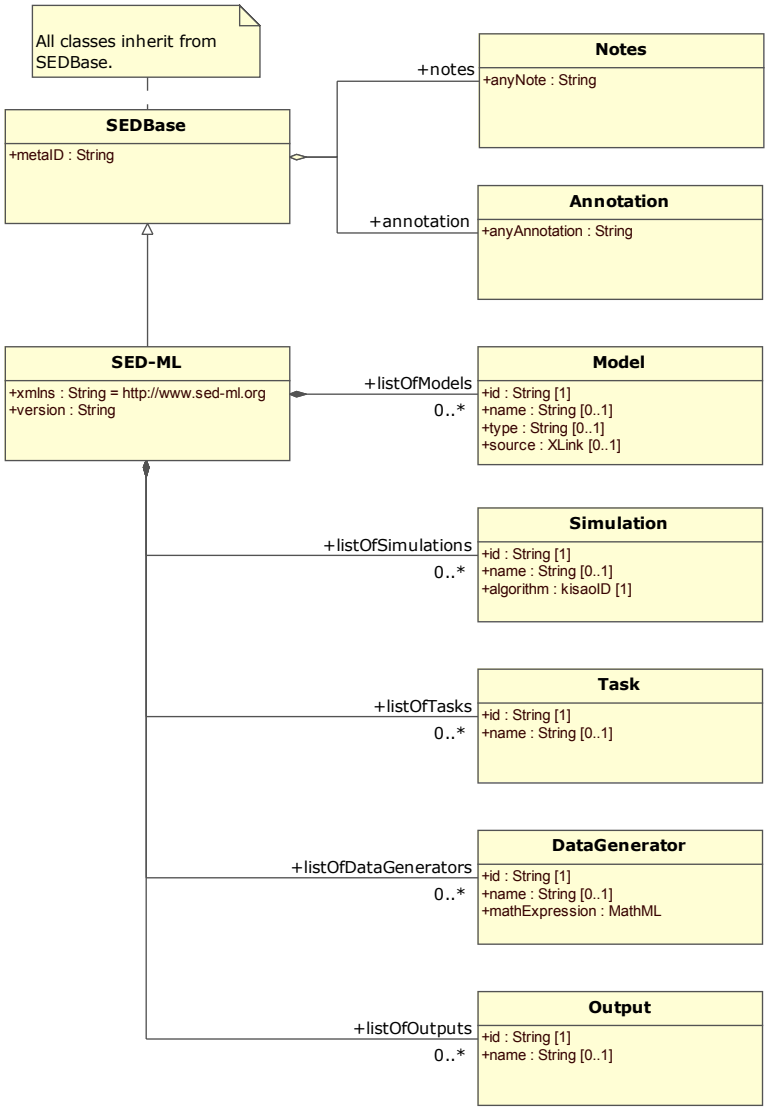

Fig. 1. SED-OM – Top level classes

XML based format. The addressed XML element can be a leaf element, or an element containing a whole mathematical expression. In principle, everything that can be addressed by an XPath expression can be modified. Other solutions that were considered for the definition of changes on a model would have involved the creation of change classes for each supported language format, depending on the current version of the standard and its syntactical naming of the model elements.

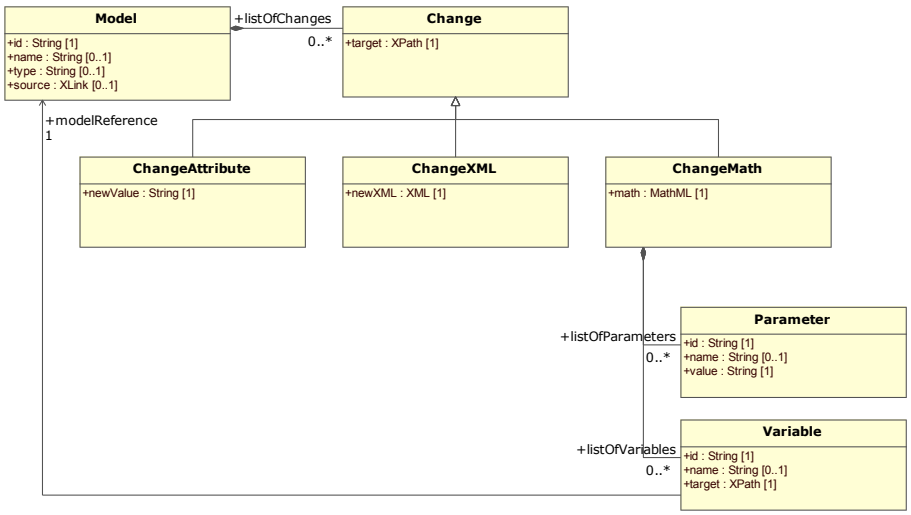

Fig. 2. SED-OM – The Model class

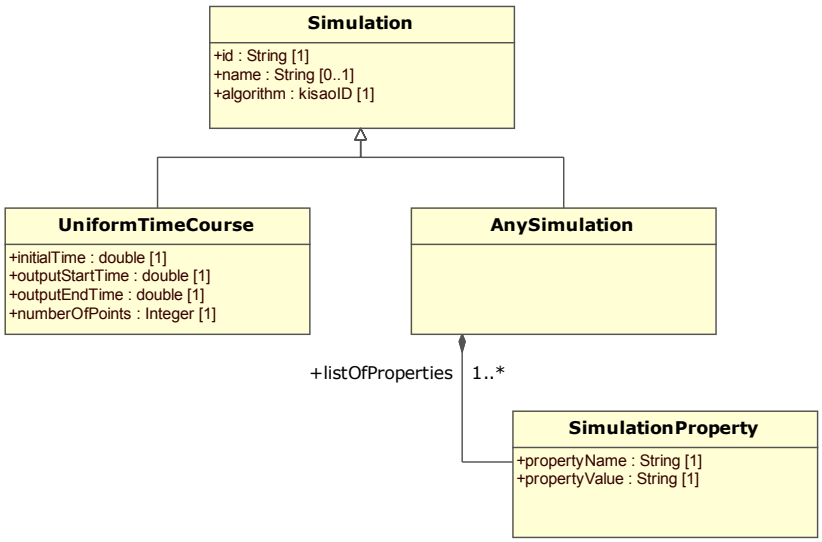

Fig. 3. SED-OM – The Simulation class

2.2 Information on the Simulation Settings

A simulation is typically characterized by the simulation algorithm used, the settings applied to the simulation algorithm, and the simulation type.

Each simulation (`Simulation`, see Figure 3) can be referred to by an identifier (`id`). It might also contain a name (`name`) and a reference to the simulation algorithm used to run the experiment (`algorithm`). This algorithm reference

is an identifier corresponding to a KiSAO term. The *Kinetic Simulation Algorithm Ontology* (KiSAO [7]) is an effort to characterize and categorize existing algorithms for the simulation of quantitative models within the field of Systems Biology. Using terms from an ontology rather than agreed-upon strings allows for reasoning. The simplest reasoning procedure is to find that algorithm available from KiSAO which is the closest to the one described in the simulation description, if the latter is not available for the user.

Depending on the chosen simulation algorithm different settings have to be applied. The necessary information which settings that are can be retrieved from the KiSA ontology which will provide the according information about additional settings for each simulation algorithm covered by the ontology. At the current state of development, KiSAO does not allow for extracting the mandatory simulation algorithm settings. As a consequence, the storage of simulation algorithm settings are not yet possible.

Very important is the type of simulation that should be launched. SED-OM defines the different types of simulations as sub-classes of the `Simulation` class. For the time being, `UniformTimeCourse` simulations are supported. The inclusion of further simulation types has been postponed to future versions of SED-OM as the integration of classes with different but overlapping attributes is not trivial. Until then, the `AnySimulation` class functions as a generic place holder for all additional simulation types. Depending on the type of simulation, different additional information has to be provided, such as the initial simulation time for uniform time courses. For the `AnySimulation` class, those simulation properties can explicitly be defined in the `SimulationProperty` class through the name and the value of the property (`propertyName`, `propertyValue`). For particular simulation types derived from the general simulation class, those attributes are already defined in the SED-OM, e. g. `initialTime` in the `UniformTimeCourse` class.

2.3 Information on the Simulation Task

In a simulation experiment, simulation approaches described in a `Simulation` object are combined with specific models described in a `Model` object. In SED-OM, the association between those two objects is supported through the definition of tasks (`Task`, see Figure 4). Each task contains one reference to a model and one reference to a simulation. The task itself can be referenced by its own identifier (`id`) and might have an additional name (`name`).

By providing the opportunity of explicitly linking models to simulations, a redundant definition of models as well as of simulation settings is avoided – a single model can easily be used with several different simulations and vice versa.

2.4 Information on the Output

One important part of SED-OM is the description of a simulation experiment based on particular (changed or unchanged) models. However, just as important is the definition of the results to be produced and the way to provide them (see Figure 5).

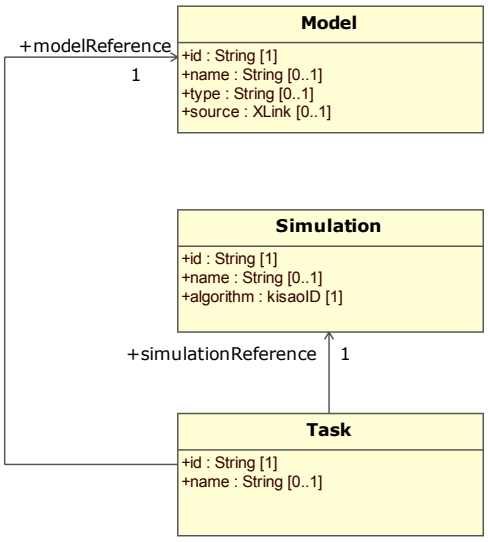

Fig. 4. SED-OM – The Task class

The output class (`Output`) can be referred to by an identifier (`id`) and an optional name (`name`). The SED-OM allows for the definition of different kinds of outputs, which can either be specified as simple data tables (i. e. reports) or as plots. Reports (`Report`) consist of a number of columns (`Column`); a formula defines how to generate the data written in each column (see the `DataGenerator` class further down). In addition, the SED-OM provides structures for two dimensional plots (`Plot2D`) and three dimensional plots (`Plot3D`). A two dimensional plot displays a number of curves (`Curve`) and a three dimensional plot displays a number of surfaces (`Surface`). Curves and surfaces refer to the data to be mapped on the according axes, and precise if the mapping is logarithmic or not. The aim of the output class is to define concisely the procedure leading to a certain output rather than to define *how* it should be presented to the user. Nonetheless, since all classes may have notes attached, it is always possible to store meta data such as information on the output shape or labels for curves.

The formulas used to generate the data are described in the `DataGenerator` class (see Figure 6). All types of output reference an instance of that class. In doing so it does not matter whether the data is calculated for plots or for the columns of a report. One example for such a calculation is the definition of the x-axis of a plot (referenced through the `Curve` class by an `xDataReference`). Each data generator has an identifier (`id`) and might have a name (`name`). A single data generator consists of a list of variables (`Variable`), a list of parameters (`Parameter`) and a mathematical expression (`Math`). A variable definition is a reference to an existing variable in one of the defined models. However, instead of referencing an element in a particular model using the model id, the variable definition refers to the the task that simulates the model. As every task uses only one model and one simulation setting description, this reference is

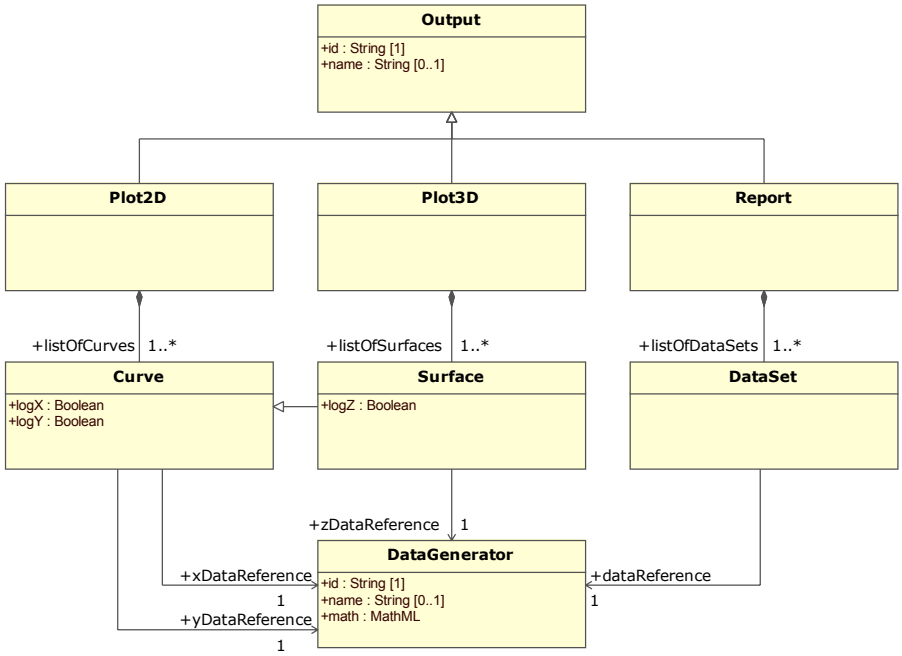

Fig. 5. SED-OM – The Output class

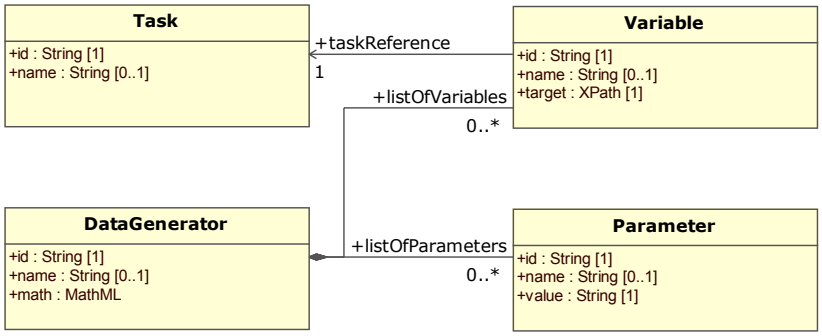

Fig. 6. SED-OM – The DataGenerator class

unambiguous. The variable inside the model is addressed via XPath expressions for the same reasons justifying the use of XPath in the description of model changes. Parameters are values introduced additionally to be used in the mathematical post-processing of the variable's values. To facilitate calculations based on the defined parameters and variables, mathematical expressions (`Math`) can be constructed using mathML. The use of a data generator could, for instance, lead to the following definition of a plot: "Take variable v1 of model m02 and multiply its values by 2. Use the result as the abcissa x-axis of a 2D plot".

3 A Simple Example for a Simulation Description

As an example for a simulation description in the *Simulation Experiment Description Markup Language* (SED-ML) we will use a model of circadian oscillations of PER and TIM proteins in Drosophila published by Leloup and Goldbeter [8]. The SED-ML file describes a uniform time course simulation run on the original model, as well as on a perturbed version of it. As has been shown in [8], the system changes its behavior from oscillation to chaos depending on the values of two parameters (maximal velocity of TIM messenger degradation V_mT and maximal velocity of degradation of the bi-phosphorylated TIM V_dT). In order to show the difference between both behaviors, a simulation experiment with two different parameter settings is described in the following simulation experiment in listing 1.1.

```
<?xml version="1.0" encoding="utf-8"?>
<sedML version="1.0" xmlns="http://www.miase.org/">
  <notes>Changing a system from oscillation to chaos</notes>
  <listOfSimulations>
    <uniformTimeCourse id="simulation1"
    algorithm="KiSAO:0000071" initialTime="0" outputStartTime="50"
    outputEndTime="1000" numberOfPoints="1000" />
  </listOfSimulations>
  <listOfModels>
    <model id="model1" name="Circadian Oscillations" type="SBML"
   source="urn:miriam:biomodels.db:BIOMD0000000021" />
   <model id="model2" name="Circadian Chaos" type="SBML" source="model1">
      <listOfChanges>
        <changeAttribute target="/sbml/model/listOfParameters/
        parameter[@id='V_mT']/@value" newValue="0.28">
        </changeAttribute>
        <changeAttribute target="/sbml/model/listOfParameters/
        parameter[@id='V_dT']/@value" newValue="4.8">
        </changeAttribute>
      </listOfChanges>
    </model>
  </listOfModels>
  <listOfTasks>
    <task id="task1" name="Baseline" modelReference="model1"
    simulationReference="simulation1">
    </task>
    <task id="task2" name="Modified parameters" modelReference="model2"
    simulationReference="simulation1">
    </task>
  </listOfTasks>
  <listOfDataGenerators>
    <dataGenerator id="time" name="Time">
      <mathExpression>
        <math>
          <apply>
            <plus />
            <csymbol encoding="text"
            definitionURL="http://www.sbml.org/sbml/symbols/time">time
            </csymbol>
          </apply>
        </math>
      </mathExpression>
    </dataGenerator>
    <dataGenerator id="tim1" name="tim mRNA (total)">
      <listOfVariables>
        <variable id="v1" taskReference="task1"
        target="/sbml/model/listOfSpecies/species[@id='Mt']" />
      </listOfVariables>
```

```
49        <mathExpression>
50          <math>
51            <apply>
52              <plus />
53              <ci>v1</ci>
54            </apply>
55          </math>
56        </mathExpression>
57      </dataGenerator>
58      <dataGenerator id="tim2" name="tim mRNA (changed parameters)">
59        <listOfVariables>
60          <variable id="v2" taskReference="task2"
61          target="/sbml/model/listOfSpecies/species[@id='Mt']" />
62        </listOfVariables>
63        <mathExpression>
64          <math>
65            <apply>
66              <plus />
67              <ci>v2</ci>
68            </apply>
69          </math>
70        </mathExpression>
71      </dataGenerator>
72    </listOfDataGenerators>
73    <listOfOutputs>
74      <plot2D id="plot1" name="tim mRNA with Oscillation and Chaos">
75        <listOfCurves>
76          <curve logX="false" logY="false" xDataReference="time"
77          yDataReference="tim1" />
78          <curve logX="false" logY="false" xDataReference="time"
79          yDataReference="tim2" />
80        </listOfCurves>
81      </plot2D>
82    </listOfOutputs>
83  </sedML>
```

Listing 1.1. Encoding of simulation settings using SED-ML

The original model used for the simulation experiment is model number 21 in BioModels database [9]. This is specified by the `source` attribute of the first model entry in the list of models (l. 11). The second model defined in the SED-ML file references the first one (`source="model1"`, l. 12). Contrary to the first model definition, this XML element contains a sub-element `listOfChanges` (ll. 13-20) that has two `changeAttribute` elements, each defining one change in the XML model representation. Both changes apply new values to existing parameters: The parameter `V_mT` is adapted in the first change definition (`newValue="0.28"`, l. 15), and the parameter `V_dT` is adapted in the second change definition (`newValue="4.8"`, l. 18).

In lines four to eight, the simulation settings are stored: The simulation has been characterized as a uniform time course (ll. 5-7) running from timepoint zero to 1000, but starting the output at timepoint 50. The simulation algorithm is specified by a KiSAO id (`KiSAO:0000071`, l. 6) which corresponds to the ontology entry "livermore solver for ordinary differential equations" (LSODE).

After the models have been defined and the simulation settings have been stored, the next step is to combine both of them by creating simulation tasks (`listOfTasks`, ll. 23-30). The first task runs the original model with the (only) simulation setting defined (`model1` with `simulation1`, ll. 24-26). The second task

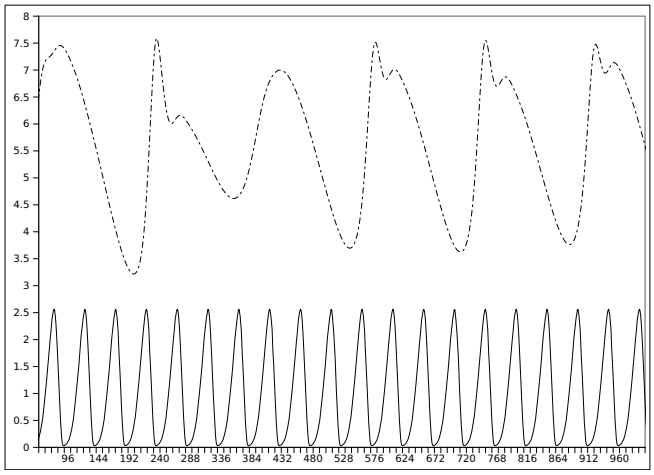

Fig. 7. Simulation result gained from the SED-ML description (created using COPASI 4.2 (Build 22) and Gnumeric Spreadsheet 1.7.11): tim mRNA concentration (line) and tim mRNA concentration with updated parameters V_mT and V_dT (dotted line)

then runs model2 with modified parameters using the same simulation setting (`model2` with `simulation1`, ll. 27-28).

The information given so far is sufficient for the design of valid and repeatable simulation experiments. SED-ML also offers structures for the specification of desired outputs. The example in listing 1.1 creates three different data generator elements: The first data generator (ll. 32-43) is a simple specification of time using the time construct available from SBML. The second data generator element (ll. 44-56) points to the species with identifier `Mt` which is the total amount of tim mRNA (as can be gained from the SBML model description file) that comes out of the task `task1`. The third data generator again points to the total amount of tim mRNA, but it is now using the values coming out of task `task2` (ll. 57-69). Note that `task2` – in contrast to `task1` – is performed on the changed model (`model2`).

The last part of the SED-ML file describes a plot (`listOfOutputs`, ll. 70-79). It consists of two curves (ll. 75-76). Both curves plot time on their x-axis; however, the first curve plots the total amount of tim mRNA using the original model (`yDataReference="tim1"`), and the second curve plots the total amount of tim mRNA applying the parameter changes (`yDataReference="tim2"`). The result of the simulation following the specifications in the SED-ML file is shown in Figure 7.

4 Related Work

The problem of simulation experiment descriptions is not new to Systems Biology and has been addressed by several groups before. Of course, each simulation tool that is capable of storing simulation settings uses its own internal storage format.

For example, COPASI [10] uses an XML based format for encoding the selected simulation algorithm and the task definitions. However, those formats can only be used with a specific simulation tool, and therefore simulation experiment descriptions cannot be exchanged with others.

Standardization communities face the problem of describing simulation experiments as well. One example is the ongoing discussion about the CellML Metadata Specification [11] in the CellML community. The authors mention the need to not only describe a model but also to describe "details of any particular simulation being run". The proposed solution is to extend the CellML meta data concept by additional simulation description concepts. CellML meta data, and thus also the simulation meta data, are specified using the Resource Description Framework (RDF [12]). With help of the CellML Metadata Specification, one or more simulation runs can be associated and described in one model specification. The description covers information about the type of simulation and about the simulation algorithm used (including the name of the linear solver, the specification of the iteration method and the multistep method used). Apart from that, the specification of step size and starting values for the simulation are supported. The CellML Metadata Specification is currently in the state of a discussion draft. Unlike SED-ML, the approach chosen by the CellML community will store simulation specification details inside the model definition and thus be restricted to the use of CellML models. By suggesting to refer to a model rather than being part of it, SED-ML enhances reusability of simulation descriptions and supports the description of simulation experiments using not only a single model, but a number of models – which could even be encoded in different description formats.

A specification to characterize simulation experiments has been proposed in the SBML community as well [13]. Along with the development of SBML Level 3 extensions, the author proposes the description of simulation settings. Although part of Level 3 extensions, the description of simulation runs is suggested to be included inside the SBML model. So as to define simulation runs, the proposal consists of several parts: (1) the definition of changes on the model, such as updates on initial values, model parameters and others; (2) the specification of simulation parameters and the storage of (time,value) pairs to maintain simulation results; and (3) the definition of plots through specification of the axes and the data that should be shown in the output. The proposal is currently available as a DTD [14] draft. Again, the approach as it has been introduced in [13] does not follow the ideas of a simulation description format independent of software tools and model description languages. Additionally, the inclusion of simulation results is proposed. This is not considered to be part of SED-ML, but in our opinion should be covered by other efforts.

5 Discussion and Future Work

In this paper, an approach for the description of simulation experiments has been introduced. The novel idea is to define a set of minimal guidelines detailed

enough to unambiguously define a simulation experiment, independent of specific simulation tools and particular model description languages. A first version of a model for the realization of those guidelines has been proposed (SED-OM) and has been encoded using XML (SED-ML). A sample simulation experiment in the SED-ML format has been described in detail. It showed that the SED-OM can be applied to existing models. The use is restricted to simple simulation experiments though.

SED-ML can encode simulation experiments being run with several models, which can even exist in different formats (e. g. comparing simulation results of a CellML model and an SBML model). SED-ML can specify different simulation settings applicable to the same model (e. g. running a model with a stochastic and a deterministic simulation algorithm). Combinations of both are also possible, it is easily conceivable to set up a simulation experiment that results in an output comparing a parameter of a CellML model to a parameter of an SBML model, depending on different simulation algorithms.

However, there are a number of important issues in simulation experiments that are currently not covered by the SED-OM. The description of more complex simulation tasks, e. g. parameter scans, is not yet supported. The difficulty here is to decide how to describe the range of parameter changes that have to be applied to a model. One option is to do that in the `Task` class, another option is to extend the functionality of the `Change` class. Furthermore, the current SED-OM allows to freely combine variables from different tasks in one output – although the combination is depending on integrity restrictions. For example, the output of variables from different simulation settings in one plot is only possible as long as all participating simulations produce the same time points. Another complex task that is not yet supported is the linear execution of simulation experiments, meaning that the result of one simulation is used as the input for another simulation task. For example, the result of a steady state analysis will lead to a model with changed parameters. If that model then should be simulated using a time course simulation, the results of the steady state analysis have to be applied to the original model before. The definition of such sequences is not yet supported by the SED-OM.

For all those reasons, the SED-OM must be further discussed. The use of SED-ML and test implementations in different simulation tools will help enhancing the coverage and robustness of the format.

6 Resources

If you want to contribute to the SED-OM and SED-ML development, or should you have any questions or comments, please contact the authors or visit the website on `http://www.ebi.ac.uk/compneur-srv/sed-ml`. The current SED-OM and sample SED-ML instances can be downloaded from the MIASE project homepage on sourceforge `http://www.sourceforge.net/miase`. For discussions of the MIASE guidelines, please join the mailing list `miase-discuss@lists.sourceforge.net` or visit the web site on `http://www.ebi.ac.uk/compneur-srv/miase`.

Acknowledgements

The authors would like to thank everybody from the community involved in the discussion of the MIASE guidelines. Special thanks goes to all members of the `miase-discuss` mailing list who have been very active in discussing both the MIASE guidelines and the SED-OM format, in particular Frank Bergman (roadRunner), Ion Moraru (VCell), Sven Sahle (COPASI) and Henning Schmidt (SBToolbox). Exchanges with Sven Sahle enriched the discussion part. Part of the work was funded by the Marie Curie program and by the German Research Association (DFG Research Training School "dIEM oSiRiS" 1387/1).

References

1. Le Novère, N., Finney, A., Hucka, M., Bhalla, U.S., Campagne, F., Collado-Vides, J., Crampin, E.J., Halstead, M., Klipp, E., Mendes, P., Nielsen, P., Sauro, H., Shapiro, B., Snoep, J., Spence, H., Wanner, B.: Minimum Information Requested In the Annotation of biochemical Models (MIRIAM). Nature Biotechnology 23(12), 1509–1515 (2005)
2. Hucka, M., Bolouri, H., Finney, A., Sauro, H., Doyle, J., Kitano, H., Arkin, A., Bornstein, B., Bray, D., Cuellar, A., Dronov, S., Ginkel, M., Gor, V., Goryanin, I., Hedley, W., Hodgman, T., Hunter, P., Juty, N., Kasberger, J., Kremling, A., Kummer, U., Le Novère, N., Loew, L., Lucio, D., Mendes, P., Mjolsness, E., Nakayama, Y., Nelson, M., Nielsen, P., Sakurada, T., Schaff, J., Shapiro, B., Shimizu, T., Spence, H., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., Wang, J.: The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. Bioinformatics 19, 524–531 (2003)
3. Lloyd, C., Halstead, M., Nielsen, P.: CellML: its future, present and past. Progress in Biophysics & Molecular Biology 85, 433–450 (2004)
4. Object Management Group (OMG): Unified Modeling Language (UML), Version 2.1.2 (2007), `http://www.omg.org/spec/UML/2.1.2/`
5. Ausbrooks, R., Buswell, S., Carlisle, D., Dalmas, S., Devitt, S., Diaz, A., Froumentin, M., Hunter, R., Ion, P., Kohlhase, M., Miner, R., Poppelier, N., Smith, B., Soiffer, N., Sutor, R., Watt, S.: Mathematical Markup Language (MathML) Version 2.0, 2nd edn. (2003)
6. World Wide Web Consortium (W3C) Recommendation: XML Path Language (XPath) (1999), `http://www.w3.org/TR/xpath`
7. Köhn, D., Le Novère, N.: The Kinetic Simulation Algorithm Ontology (KiSAO). Website (2007), `http://www.ebi.ac.uk/compneur-srv/kisao/`
8. Leloup, J., Goldbeter, A.: Chaos and birhythmicity in a model for circadian oscillations of the per and tim proteins in drosophila. Journal of theoretical biology 198(3), 445–459 (1999)
9. Le Novère, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., Li, L., Sauro, H., Schilstra, M., Shapiro, B., Snoep, J., Hucka, M.: Biomodels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic Acids Research 34(Database issue) (January 2006)
10. Hoops, S., Sahle, S., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U.: Copasi a complex pathway simulator. Bioinformatics 22(24), 3067–3074 (2006)

11. Miller, A.: CellML simulation metadata specification – a specification for simulation metadata (2007), http://www.cellml.org/specifications/metadata/simulations
12. World Wide Web Consortium (W3C) Recommendation: Resource Description Framework (RDF) (1997), http://www.w3.org/RDF
13. Kopalov, F.: SBML extensions for level 3: Experiments, simulation, parameters, results and plots (March 2008) (unpublished proposal)
14. World Wide Web Consortium (W3C) Specification: Document Type Definition (DTD) (1998), http://www.w3.org/XML/1998/06/xmlspec-report