Version 3

Nested Tasks Proposal

A proposal for extending SED-ML L1V1

Frank T. Bergmann <fbergman@caltech.edu>
6/9/2012

Contents

About this document

This document describes a third version of the proposal for encoding nested simulation experiments [1] in SED-ML [2]. The proposal has been updated to reflect all discussions made in the list and HARMONY 2012. This proposal also provides explicit examples that go beyond the ones previously detailed on the author's blog [3].

Changes in this Version

The proposal has been updated specifically in the following points:

- Reintroduction of the steady state primitive (and with that the step attribute of the one step class becomes a required attribute again)
- Instead of a nested simulation class the concept of a repeated task is introduced. It allows encoding all simulation experiments that could previously encoded with the nested simulation class.
- Previous proposals inconsistently used grouping elements '<ranges>' or '<changes>' when the appropriate naming in SED-ML would have been '<listOfRanges>' and '<listOfChanges>'. This has been corrected
- The uniform range class received an additional attribute type with values 'linear' / 'log'. This makes it more convenient rather than to use the combination of ranges to achieve arbitrary ranges.

Summary

Instead of redefining a large number of simulation types to suite various simulation tasks, which will make it very difficult for tool developers to support SED-ML, this proposal suggests a nested simulation task that breaks down complex tasks into separate steps. This is achieved by introducing two new simulation classes: "one step", a simulation class that performs a specified simulation algorithm until a given output step and "steady state" that when invoked attempts to bring the model to the next steady state. At the heart of the proposal stands the "repeated task" a class that performs a specified simulation multiple times (where the exact number is specified through a "range" construct while allowing specific aspects of the model to change (in the simplest case that would be by advancing the simulation time). Alternatively the "repeated task" can define a list of subtasks to be carried out.

This approach is easy to implement for tool developers and covers a large number of commonly carried out simulation experiments: parameter scans or collecting stochastic traces by running a simulation multiple times.

The Proposal

This section discusses the two new simulation classes.

New Simulation Class: oneStep

This calculates one further output step for the model from its current state. Note that this does NOT necessarily equate to one integration step. The simulator is allowed to take as many steps as needed, all that has to be fulfilled with this simulation class, is that, at the end, the desired output time is reached.

```
<listOfSimulations>
  <oneStep id="s1" step="0.1">
    <algorithm kisaoID="KISAO:0000019" />
  </oneStep>
</listOfSimulations>
```

This example defines a simulation step of a deterministic simulation which advances the associated model from its current time to current time + 0.1, as the next output step.

step

The oneStep class has one required attribute “step”. It defines the next output step that should be reached by the algorithm. In the previous version of the proposal the step variable was mandatory. However, due to the suggestions to remove the “steady state” simulation class and combine it with “one step” the “step” attribute becomes meaningless in some scenarios.

New Simulation Class: steadyState

The steady state class represents a steady state computation (as for example implemented by NLEQ or Kinsolve).

```
<listOfSimulations>
  <steadyState id="steady">
    <algorithm kisaoID="KISAO:0000282" />
  </steadyState >
</listOfSimulations>
```

New Task Class: repeatedTask

Nested simulation experiments are the core of this proposal. In previous proposals this class was implemented as a simulation class, however the community felt that this class was blurring the line between simulation classes and tasks. Based on Nicolas Le Novère’s rephrasing of the simulation class as task the repeatedTask was born. It basically consists of the following steps:

1. Compute the current value of ranges
2. Apply the changes defined
3. Run the specified *simulationReference* on the model **OR** run all tasks defined in the list of subtasks
4. If *resetModel* is defined, reset the model to initial conditions
5. If we have more values within the range specified in the *range* attribute go to 1, otherwise end.

The overall structure of the *repeatedTask* class is as follows:

```
<repeatedTask id="task3" resetModel="false"
              modelReference="model1" simulationReference="simulation1" range="current">
  <listOfRanges>
    <vectorRange id="current">
        <value> 1 </value>
        <value> 4 </value>
        <value> 10 </value>
    </vectorRange>
  </listOfRanges>
  <listofChanges>
    <setValue target="/sbml/model/listOfParameters/parameter[@id='w']">
     <listOfVariables>
         <variable id="val" name="current range value" target="#current" />
     </listOfVariables>
     <math>
         <ci> val </ci>
     </math>
    </setValue>
  </listOfChanges>
</repeatedTask>
```

For the case that the repeated task is meant to only sweep over one type of simulation. Otherwise the list of sub tasks would be used to list all individual simulations to be performed. For example:

```
<task id="task1" modelReference="model1" simulationReference="simulation1" />
<repeatedTask id="task3" modelReference="model1" resetModel="false" range="current">
  <listOfRanges>
    <vectorRange id="current">
        <value> 1 </value>
        <value> 4 </value>
        <value> 10 </value>
    </vectorRange>
  </listOfRanges>
  <listofChanges>
    <setValue target="/sbml/model/listOfParameters/parameter[@id='w']">
     <listOfVariables>
         <variable id="val" name="current range value" target="#current" />
     </listOfVariables>
     <math>
         <ci> val </ci>
     </math>
    </setValue>
  </listOfChanges>
  <listOfSubTasks>
    <subTask task="task1" />
  </listOfSubTasks>
</repeatedTask>
```

Below the individual elements are described.

range attribute

The *range* attribute determines the range that this nested simulation experiment iterates over. This is akin to loop variables in programming languages. In the example above the *repeatedTask* will iterate over three values: 1, 4 and 10.

resetModel attribute

The *resetModel* attribute indicates whether the model should be reset after completing a iteration of the nested simulation experiment. Consider the case where a piecewise simulation is performed. In this

case you would not want the model to be reset to initial conditions after each piece of the simulation. However, in the case of collecting several stochastic traces you might want to reset the model each time.

listOfRanges

Ranges represent the iterative element of the nested simulation experiment that provides the collection of values to iterate over. In order to be able to refer to the current value of a range element, we add an id attribute. When the value of the id attribute is used in a listOfVariables within the repeated task class it is to be extended with the current value.

There are three different range types:

UniformRange: this range is defined through a *start* value, an *end* value and a *number of points*. An additional *type* attribute can take the values *linear* or *log* and determines whether to draw the values logarithmically or linearly. This is quite similar to what is used today in the UniformTimecourse simulation experiment. For example:

```
<uniformRange id="current" start="0.0" end="10.0" numberOfPoints="100" type="linear" />
```

VectorRange: this range can be seen as simply a collection of values. For example:

```
<vectorRange id="current">
    <value> 1 </value>
    <value> 4 </value>
    <value> 10 </value>
</vectorRange>
```

FunctionalRange: defines a function to be evaluated in order to determine the next value. Previously the functional range was simply stated as a MathML element. However, this will not allow us to reference model elements, and so it needs to look like the computeChange element. That is we need a listOfVariables, along with the Math description. Additionally we need another attribute that specifies an index variable “index” so that we can evaluate the function at different points. This variable can be used in the MathML expression. The value to be used for the index is provided by another range when the functionalRange is invoked. For example:

```
<functionalRange id="current" index="index">
  <listOfVariables>
     <variable id="w" name="current parameter value"
                      target="/sbml/model/listOfParameters/parameter[@id='w']" />
  </listOfVariables>
  <function>
    <math>
       <apply>
       <times/>
       <ci> w </ci>
       <ci> index </ci>
       </apply>
    </math>
  </function>
</functionalRange>
```

Here another example of using the values in a piecewise expression:

```
<uniformRange id="index" start="0" end="10" numberOfPoints="100" />
<functionalRange id="current" index="index">
  <function>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <piecewise>
        <piece>
          <cn> 8 </cn>
          <apply>
            <lt />
            <ci> index </ci>
            <cn> 1 </cn>
          </apply>
        </piece>
        <piece>
          <cn> 0.1 </cn>
          <apply>
            <and />
            <apply>
              <geq />
              <ci> index </ci>
              <cn> 4 </cn>
            </apply>
            <apply>
              <lt />
              <ci> index </ci>
              <cn> 6 </cn>
            </apply>
          </apply>
        </piece>
        <otherwise>
          <cn> 8 </cn>
        </otherwise>
      </piecewise>
    </math>
  </function>
</functionalRange>
```

listOfChanges

Multiple changes to the model are allowed:

```
<listOfChanges>
  <setValue target="/sbml/model/listOfParameters/parameter[@id='w']" range="current" >
   <math>
       <ci> current </ci>
   </math>
  </setValue>
</listOfChanges>
```

In order to specify the object that will receive the new value we use an xpath query in form of the target attribute. The setValue class inherits all properties from the SED-ML Variable class, and so instead of the 'target' attribute it is also possible to refer to SED-ML implicit symbols. The additional attribute 'range' describes the range object that provides the current value (see also the functionalRange examples).

listOfSubTasks

When not using the *simulationReference* all operations to be carried out are specified in the list of subtasks. The subTask itself has one required attribute *task* that references the id of another task defined in the listOfTasks. The subtasks in the list are un-ordered. This allows the subTasks to be carried

out in parallel. In order to establish that one subtask should be carried out before another it can contain a listOfDependTasks where the ids of other tasks can be referenced.

```
<listOfSubTasks>
  <subTask task="task1" >
   <listOfDependTasks>
       <dependendTask task="task2" />
   </listOfDependTasks>
  </subTask>
  <subTask task="task2" />
</listOfSubTasks>
```

In the example above the task *task2* has to be carried out before *task1*.

NOTE: if parallel execution is not required at this point we could very well use an attribute 'order' that

Discussion / Known Issues

The nested task construct explained above is easy to implement and versatile. It can be implemented to construct arbitrarily complex datasets. This section details the known issues this brings along.

Further subclasses of Task

At HARMONY it was suggested to add further subclasses of tasks. In Nicolas's proposal [4] the example would be the 'singleTask' that would work just like a 'repeatedTask' except that it would not have a 'listOfRanges'. This proposal chose the easier to implement route of simply on task class. But obviously the introduction of a 'singleTask' would work just as well.

Another suggestion was centered on the fact that the repeatedTask could only have **EITHER** a simulationReference **OR** a listOfSubTasks. While this technicality does not impose any difficulty in implementation, it was seen as inelegant (even though there would be precedence in SBML, where one could have either a double stoichiometry attribute on a SpeciesReference, or a StoichiometryMath object). The proposal instead was to change the inheritance and separate all things nicely (see Figure 1). And while it is possible to create this hierarchy, this proposal chose not to go that route as it would have changed the structure of SED-ML L1V1 and would be a bit more difficult to support.

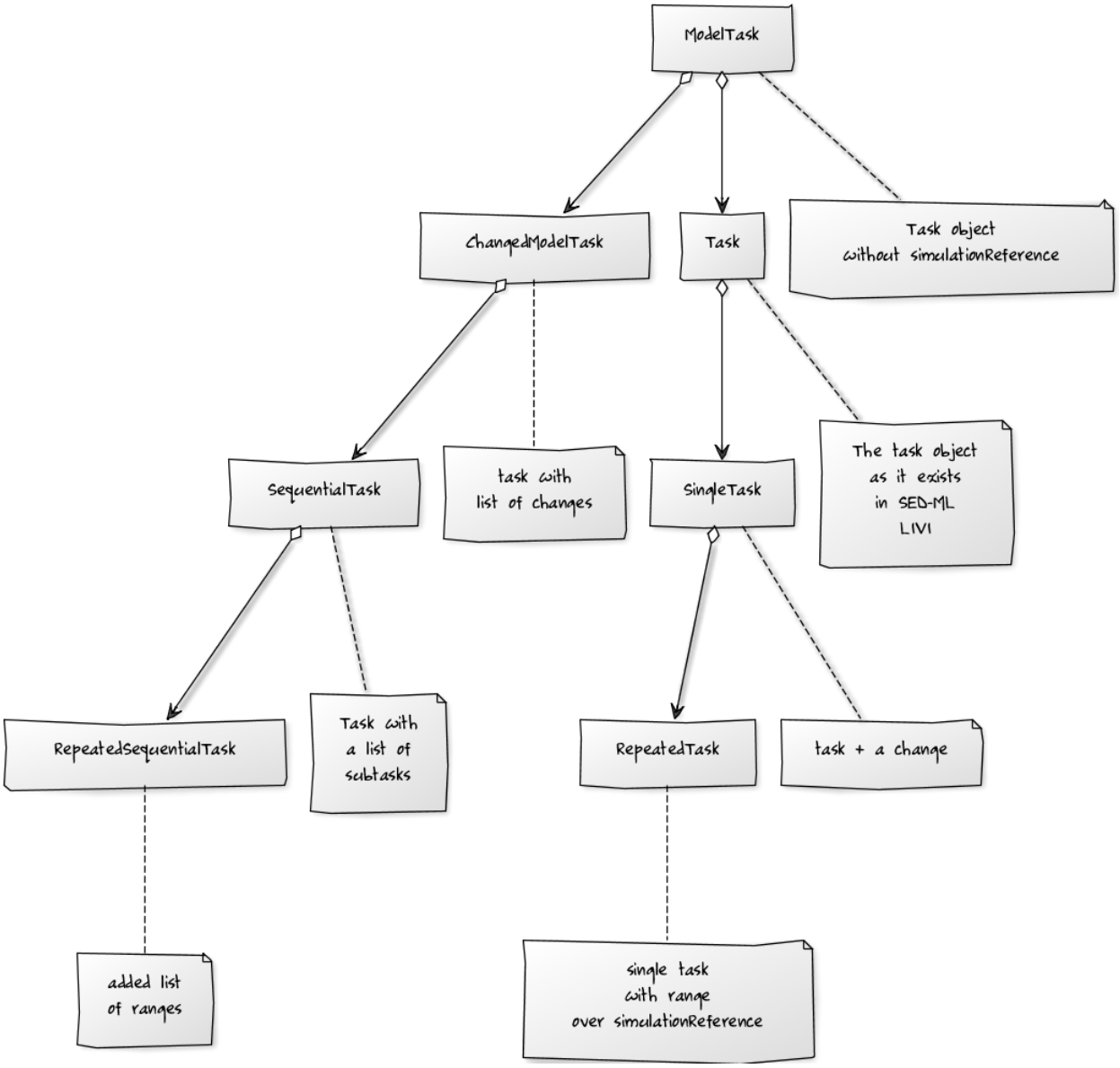


Figure 1: Alternative to the one repeatedTask presented in this document that was discussed at HARMONY: By creating a new super class of Task and creating a number of subclasses it would be possible to get rid of the simulationReference vs. listOfSubTasks situation without loss of functionality.

Collecting data from a nested simulation

This proposal only describes the *task* aspect, that is how to generate the data. What is not described is how to describe its visualization. When implementing the nested simulation task it would be best to have it generate a dataset of arbitrary dimensions, and present them to the user to let them slice the data as desired. The SED-ML community is discussing several alternative ways on how to deal with higher dimensional DataGenerators (one proposal involved using NuML [4, 5], and another to allow arbitrary MathML expressions [6]).

No matter which direction is chosen on how to describe the plots it the necessity of generating the results exists, and this proposal addresses them.

Semantics of a Nested Simulation Task

This proposal favors the approach of defining a general nested simulation task. Once implemented, it will allow a tool to perform a large number of different simulations. To add semantic information to the task it suffices to register a KISAO term for the specific task used, or to annotate it using MIRIAM RDF annotations on the task itself.

The author strongly believes this is preferable to the situation where we would need to define a vast number of individual simulation tasks like: 1D ParameterScan, 2D ParameterScan, 1D Logarithmic-ParameterScan or a RepeatedSimulation.

Acknowledgements

Thanks go to the SED-ML Editors as well as the SED-ML community for suggestions on the previous proposal and to Brett Olivier for implementing the previous proposal. Finally thanks to the Saurolab, University of Washington that is hosting the SED-ML Web Tools application.

Examples

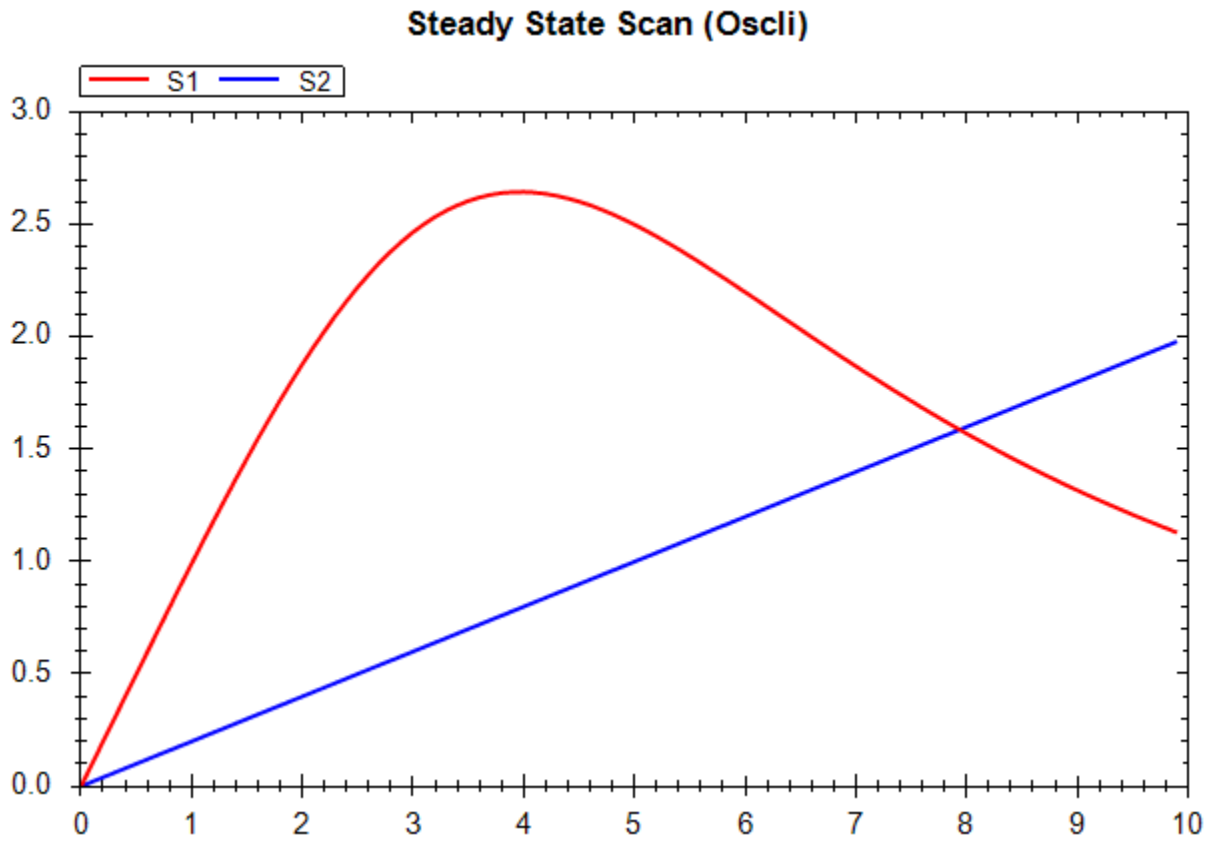
The following examples have been produced by libSedML [7]. The current version has prototypical support for the nested simulation task outlined above. The web application SED-ML Web Tools [8] have also been upgraded to be able to simulate them.

1D Steady State Parameter Scan

In this simple task one nested simulation task calls out to a oneStep task (performing a steady state computation) and varies a parameter to get different responses.

In the description below the range to be used in the setValue construct uses the Xpath variant as described above. To see the use of the 'range' attribute see the next example.

The Results



The Description

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Written by libSedML v1.1.4536.26450 see http://libsedml.sf.net -->
<sedML level="1" version="1" xmlns="http://sed-ml.org/">
  <listOfSimulations>
    <steadyState id="steady1">
      <algorithm kisaoID="KISAO:0000282" />
    </steadyState>
  </listOfSimulations>
  <listOfModels>
    <model id="model1" language="urn:sedml:language:sbml"
source="http://libsedml.svn.sourceforge.net/viewvc/libsedml/trunk/Samples/models/oscli.xml" />
```

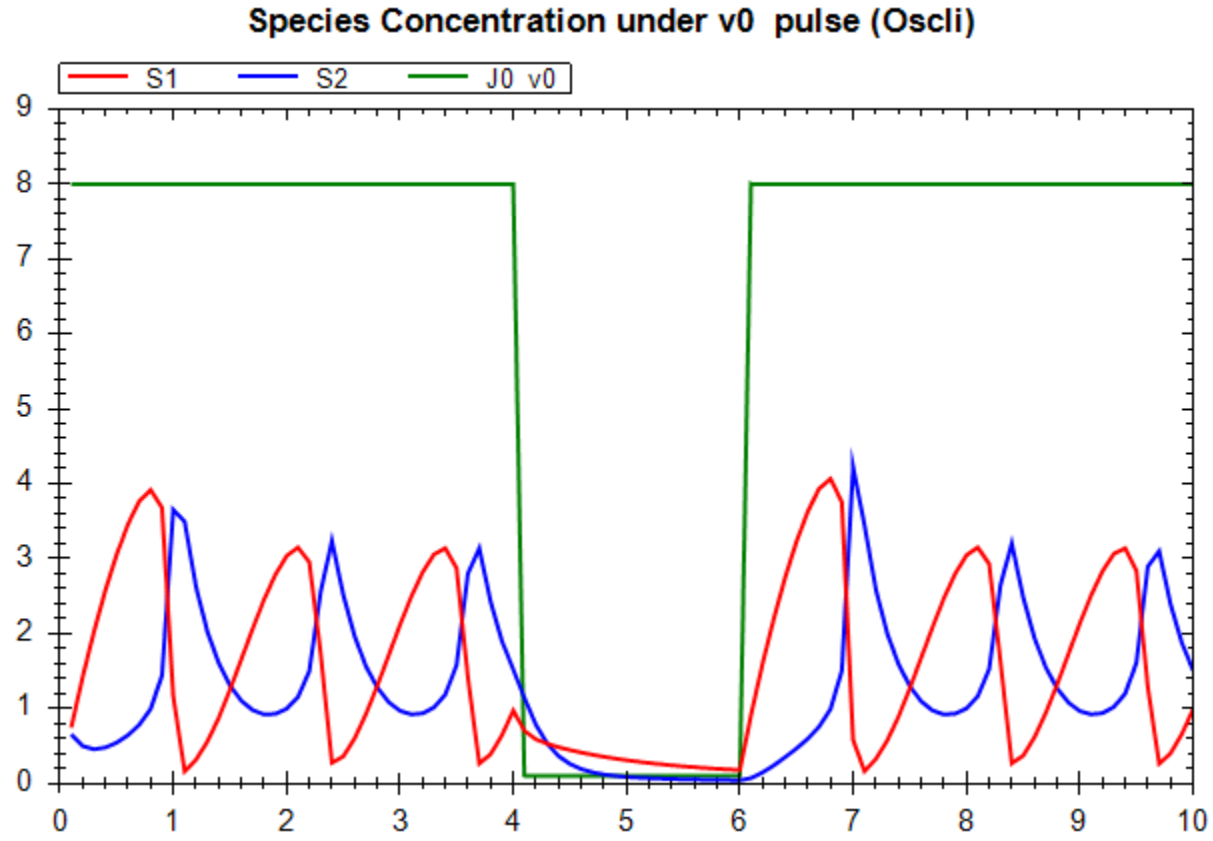
```
  </listOfModels>
  <listOfTasks>
    <repeatedTask id="task1" modelReference="model1" simulationReference="steady1" resetModel="true"
range="">
      <listOfChanges>
        <setValue target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J0_v0']"
range="current">
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> current </ci>
          </math>
        </setValue>
      </listOfChanges>
      <listOfRanges>
        <uniformRange id="current" start="0" end="10" numberOfPoints="100" type="linear" />
      </listOfRanges>
    </repeatedTask>
  </listOfTasks>
  <listOfDataGenerators>
    <dataGenerator id="J0_v0_1" name="J0_v0">
      <listOfVariables>
        <variable id="J0_v0" name="J0_v0" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J0_v0']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> J0_v0 </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="S1_1" name="S1">
      <listOfVariables>
        <variable id="S1" name="S1" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='S1']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> S1 </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="S2_1" name="S2">
      <listOfVariables>
        <variable id="S2" name="S2" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='S2']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> S2 </ci>
      </math>
    </dataGenerator>
  </listOfDataGenerators>
  <listOfOutputs>
    <plot2D id="plot1" name="Steady State Scan (Oscli)">
      <listOfCurves>
        <curve id="curve1" logX="false" logY="false" xDataReference="J0_v0_1" yDataReference="S1_1" />
        <curve id="curve2" logX="false" logY="false" xDataReference="J0_v0_1" yDataReference="S2_1" />
      </listOfCurves>
    </plot2D>
    <report id="report1" name="Steady State Values">
      <listOfDataSets>
        <dataSet id="col1" dataReference="J0_v0_1" label="J0_v0" />
        <dataSet id="col2" dataReference="S1_1" label="S1" />
        <dataSet id="col3" dataReference="S2_1" label="S2" />
      </listOfDataSets>
    </report>
  </listOfOutputs>
</sedML>
```

Perturbing a Simulation

Often it is interesting to see how the dynamic behavior of a model changes when some perturbations are applied to the model. In this example we include one nested simulation task that makes repeated use of a oneStep task (that advances an ODE integration to the next output step). During the steps one parameter is modified effectively killing the oscillations in a model. Once the value is reset the oscillations recover.

Note: In the example below we use a functionalRange, although the same result could also be achieved using the setValue element directly.

The Results



The Description

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Written by libSedML v1.1.4536.26450 see http://libsedml.sf.net -->
<sedML level="1" version="1" xmlns="http://sed-ml.org/">
  <listOfSimulations>
    <oneStep id="stepper" step="0.1">
      <algorithm kisaoID="KISAO:0000019" />
    </oneStep>
  </listOfSimulations>
  <listOfModels>
    <model id="model1" language="urn:sedml:language:sbml"
source="http://libsedml.svn.sourceforge.net/viewvc/libsedml/trunk/Samples/models/oscli.xml" />
  </listOfModels>
  <listOfTasks>
```

```
    <repeatedTask id="task1" modelReference="model1" simulationReference="stepper" resetModel="false"
range="index">
      <listOfChanges>
        <setValue target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J0_v0']"
range="current">
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> current </ci>
          </math>
        </setValue>
      </listOfChanges>
      <listOfRanges>
        <uniformRange id="index" start="0" end="10" numberOfPoints="100" type="linear" />
        <functionalRange id="current" index="index">
          <listOfVariables>
            <variable id="val" name="current loop value" target="//*[@id='current']" />
          </listOfVariables>
          <function>
            <math xmlns="http://www.w3.org/1998/Math/MathML">
              <piecewise>
                <piece>
                  <cn> 8 </cn>
                  <apply>
                    <lt />
                    <ci> index </ci>
                    <cn> 1 </cn>
                  </apply>
                </piece>
                <piece>
                  <cn> 0.1 </cn>
                  <apply>
                    <and />
                    <apply>
                      <geq />
                      <ci> index </ci>
                      <cn> 4 </cn>
                    </apply>
                    <apply>
                      <lt />
                      <ci> index </ci>
                      <cn> 6 </cn>
                    </apply>
                  </apply>
                </piece>
                <otherwise>
                  <cn> 8 </cn>
                </otherwise>
              </piecewise>
            </math>
          </function>
        </functionalRange>
      </listOfRanges>
    </repeatedTask>
  </listOfTasks>
  <listOfDataGenerators>
    <dataGenerator id="time_1" name="time">
      <listOfVariables>
        <variable id="time" name="time" taskReference="task1" target="time" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> time </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="J0_v0_1" name="J0_v0">
      <listOfVariables>
        <variable id="J0_v0" name="J0_v0" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J0_v0']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> J0_v0 </ci>
```

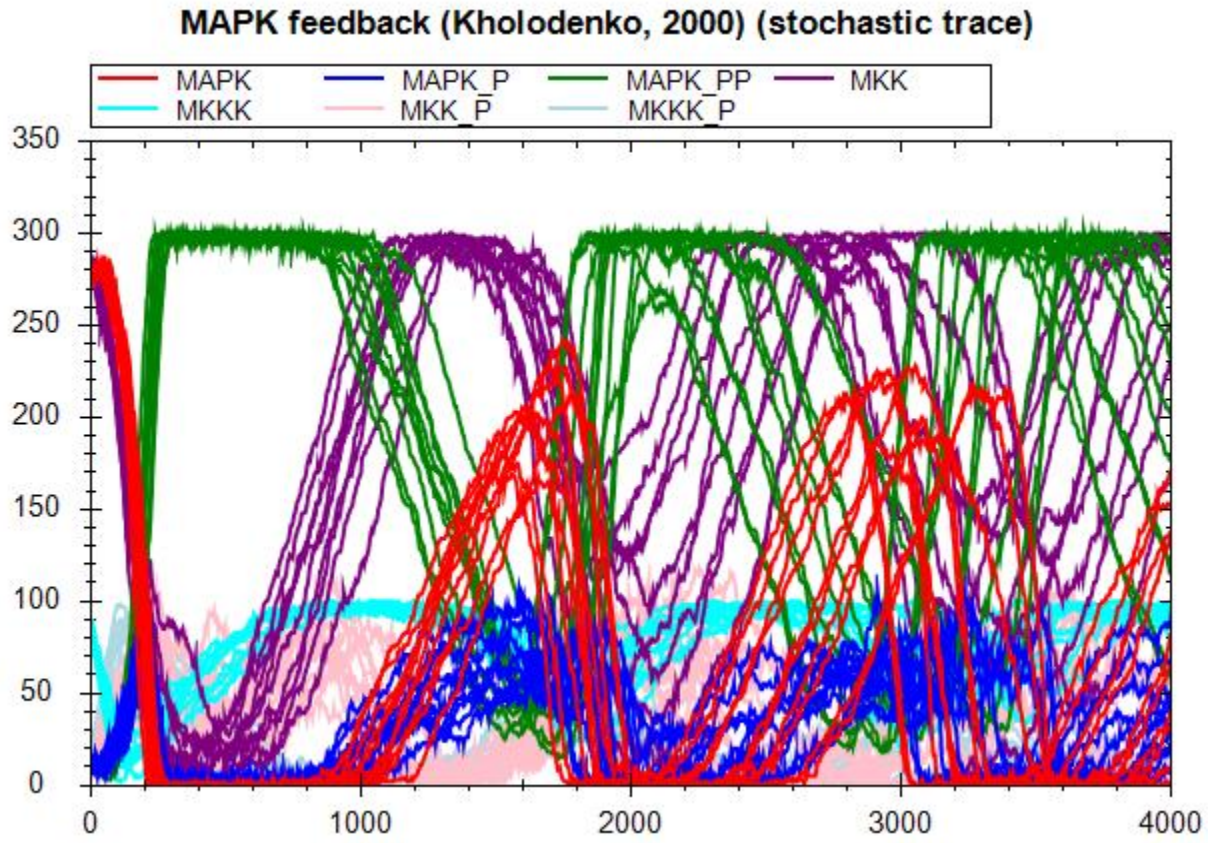
```
      </math>
    </dataGenerator>
    <dataGenerator id="S1_1" name="S1">
      <listOfVariables>
        <variable id="S1" name="S1" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='S1']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> S1 </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="S2_1" name="S2">
      <listOfVariables>
        <variable id="S2" name="S2" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='S2']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> S2 </ci>
      </math>
    </dataGenerator>
  </listOfDataGenerators>
  <listOfOutputs>
    <plot2D id="plot1" name="Species Concentration under v0  pulse (Oscli)">
      <listOfCurves>
        <curve id="curve1" logX="false" logY="false" xDataReference="time_1" yDataReference="S1_1" />
        <curve id="curve2" logX="false" logY="false" xDataReference="time_1" yDataReference="S2_1" />
        <curve id="curve3" logX="false" logY="false" xDataReference="time_1" yDataReference="J0_v0_1" />
      </listOfCurves>
    </plot2D>
    <report id="report1" name="Species Concentration under v0  pulse (Oscli)">
      <listOfDataSets>
        <dataSet id="col0" dataReference="time_1" label="" />
        <dataSet id="col1" dataReference="J0_v0_1" label="" />
        <dataSet id="col2" dataReference="S1_1" label="" />
        <dataSet id="col3" dataReference="S2_1" label="" />
      </listOfDataSets>
    </report>
  </listOfOutputs>
</sedML>
```

Repeated Stochastic Simulation

NOTE: Since this example produces three dimensional results (time, species concentration, multiple repeats). Currently we do not have a way for post-processing these values. In my implementation of plots I allow higher dimensional values and flatten them by overlaying them onto the desired plot.

Running just one stochastic trace does not give you a complete picture of the behavior of a system. A large number of traces are needed to provide a result. In this example we perform just ten traces of a simulation to give the general idea. Here we have one nested simulation task running over a repeated uniform time course simulation (performing a stochastic simulation run).

The Results



The Description

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Written by libSedML v1.1.4536.26450 see http://libsedml.sf.net -->
<sedML level="1" version="1" xmlns="http://sed-ml.org/">
  <listOfSimulations>
    <uniformTimeCourse id="timecourse1" initialTime="0" outputStartTime="0" outputEndTime="4000"
numberOfPoints="1000">
      <algorithm kisaoID="KISAO:0000241" />
    </uniformTimeCourse>
  </listOfSimulations>
  <listOfModels>
    <model id="model1" language="urn:sedml:language:sbml" source="c:\users\fbergmann\documents\sbml
models\BorisEjb.xml" />
  </listOfModels>
  <listOfTasks>
```

```
    <repeatedTask id="task1" modelReference="model1" simulationReference="timecourse1" resetModel="true"
range="">
      <listOfRanges>
        <uniformRange id="current" start="0" end="10" numberOfPoints="10" type="linear" />
      </listOfRanges>
    </repeatedTask>
  </listOfTasks>
  <listOfDataGenerators>
    <dataGenerator id="time1" name="time">
      <listOfVariables>
        <variable id="time" taskReference="task1" symbol="urn:sedml:symbol:time" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> time </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="MAPK1" name="MAPK">
      <listOfVariables>
        <variable id="MAPK" name="MAPK" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MAPK']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MAPK </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="MAPK_P1" name="MAPK_P">
      <listOfVariables>
        <variable id="MAPK_P" name="MAPK_P" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MAPK_P']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MAPK_P </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="MAPK_PP1" name="MAPK_PP">
      <listOfVariables>
        <variable id="MAPK_PP" name="MAPK_PP" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MAPK_PP']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MAPK_PP </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="MKK1" name="MKK">
      <listOfVariables>
        <variable id="MKK" name="MKK" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MKK']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MKK </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="MKK_P1" name="MKK_P">
      <listOfVariables>
        <variable id="MKK_P" name="MKK_P" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MKK_P']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MKK_P </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="MKKK1" name="MKKK">
      <listOfVariables>
        <variable id="MKKK" name="MKKK" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MKKK']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MKKK </ci>
      </math>
```

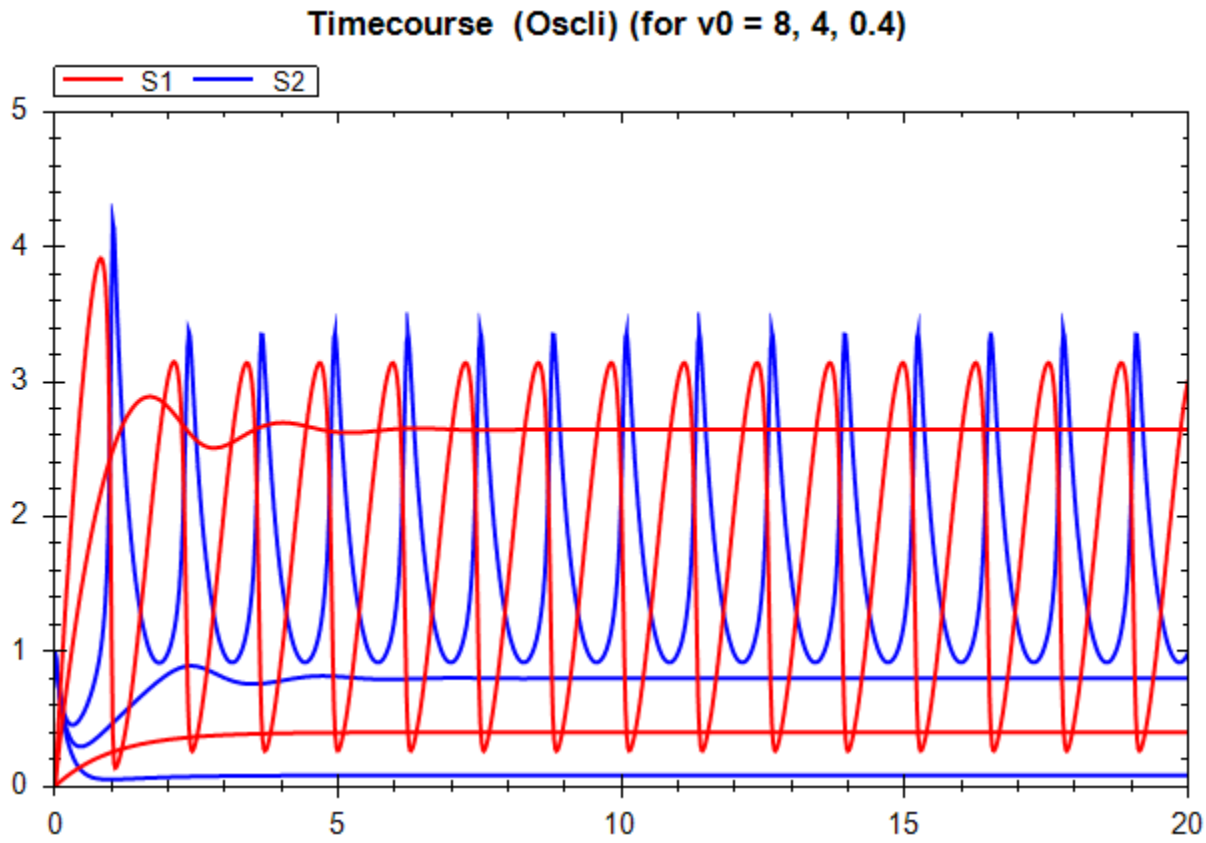
```
    </dataGenerator>
    <dataGenerator id="MKKK_P1" name="MKKK_P">
      <listOfVariables>
        <variable id="MKKK_P" name="MKKK_P" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MKKK_P']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MKKK_P </ci>
      </math>
    </dataGenerator>
  </listOfDataGenerators>
  <listOfOutputs>
    <plot2D id="plot1" name="MAPK feedback (Kholodenko, 2000) (stochastic trace)">
      <listOfCurves>
        <curve id="curve1" logX="false" logY="false" xDataReference="time1" yDataReference="MAPK1" />
        <curve id="curve2" logX="false" logY="false" xDataReference="time1" yDataReference="MAPK_P1" />
        <curve id="curve3" logX="false" logY="false" xDataReference="time1" yDataReference="MAPK_PP1" />
        <curve id="curve4" logX="false" logY="false" xDataReference="time1" yDataReference="MKK1" />
        <curve id="curve5" logX="false" logY="false" xDataReference="time1" yDataReference="MKKK1" />
        <curve id="curve6" logX="false" logY="false" xDataReference="time1" yDataReference="MKK_P1" />
        <curve id="curve7" logX="false" logY="false" xDataReference="time1" yDataReference="MKKK_P1" />
      </listOfCurves>
    </plot2D>
  </listOfOutputs>
</sedML>
```

1D Time course Parameter Scan

NOTE: Since this example produces three dimensional results (time, species concentration, multiple parameter values). Currently we do not have a way for post-processing these values. In my implementation of plots I allow higher dimensional values and flatten them by overlaying them onto the desired plot.

Here we have one nested simulation task running over a repeated uniform time course simulation (performing a deterministic simulation run) after each run the parameter value is changed.

The Results



The Description

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Written by libSedML v1.1.4536.26450 see http://libsedml.sf.net -->
<sedML level="1" version="1" xmlns="http://sed-ml.org/">
  <listOfSimulations>
    <uniformTimeCourse id="timecourse1" initialTime="0" outputStartTime="0" outputEndTime="20"
numberOfPoints="1000">
      <algorithm kisaoID="KISAO:0000019" />
    </uniformTimeCourse>
  </listOfSimulations>
  <listOfModels>
    <model id="model1" language="urn:sedml:language:sbml"
source="http://libsedml.svn.sourceforge.net/viewvc/libsedml/trunk/Samples/models/oscli.xml" />
  </listOfModels>
  <listOfTasks>
    <repeatedTask id="task1" modelReference="model1" simulationReference="timecourse1" resetModel="true"
range="">
```

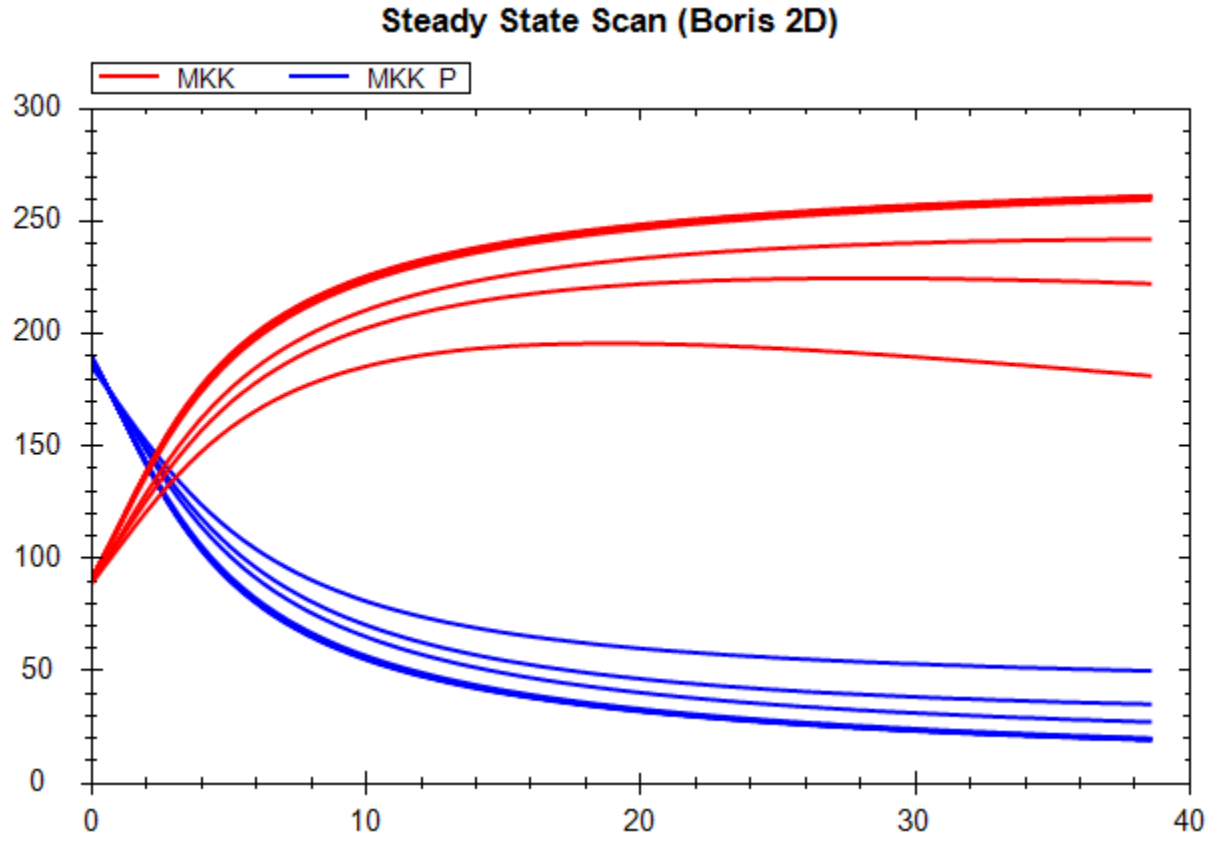
```
      <listOfChanges>
        <setValue target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J0_v0']"
range="current">
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> current </ci>
          </math>
        </setValue>
      </listOfChanges>
      <listOfRanges>
        <vectorRange id="current">
          <value>8</value>
          <value>4</value>
          <value>0.4</value>
        </vectorRange>
      </listOfRanges>
    </repeatedTask>
  </listOfTasks>
  <listOfDataGenerators>
    <dataGenerator id="time1" name="time">
      <listOfVariables>
        <variable id="time" name="time" taskReference="task1" target="time" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> time </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="J0_v0_1" name="J0_v0">
      <listOfVariables>
        <variable id="J0_v0" name="J0_v0" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J0_v0']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> J0_v0 </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="S1_1" name="S1">
      <listOfVariables>
        <variable id="S1" name="S1" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='S1']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> S1 </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="S2_1" name="S2">
      <listOfVariables>
        <variable id="S2" name="S2" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='S2']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> S2 </ci>
      </math>
    </dataGenerator>
  </listOfDataGenerators>
  <listOfOutputs>
    <plot2D id="plot1" name="Timecourse  (Oscli) (for v0 = 8, 4, 0.4)">
      <listOfCurves>
        <curve id="curve1" logX="false" logY="false" xDataReference="time1" yDataReference="S1_1" />
        <curve id="curve2" logX="false" logY="false" xDataReference="time1" yDataReference="S2_1" />
      </listOfCurves>
    </plot2D>
  </listOfOutputs>
</sedML>
```

2D Steady State Parameter Scan

NOTE: Since this example produces three dimensional results (species concentration, parameter1 values, parameter2 values). Currently we do not have a way for post-processing these values. In my implementation of plots I allow higher dimensional values and flatten them by overlaying them onto the desired plot.

Here we have one nested simulation task running over another nested simulation which runs over a oneStep task (performing a steady state computation). Each nested simulation task modifies a different parameter.

The Results



The Description

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Written by libSedML v1.1.4541.15753 see http://libsedml.sf.net -->
<sedML level="1" version="1" xmlns="http://sed-ml.org/">
  <listOfSimulations>
    <steadyState id="steady1">
      <algorithm kisaoID="KISAO:0000282" />
    </steadyState>
  </listOfSimulations>
  <listOfModels>
    <model id="model1" language="urn:sedml:language:sbml" source="c:\users\fbergmann\documents\sbml
models\boris.xml" />
  </listOfModels>
  <listOfTasks>
```

```
    <repeatedTask id="task1" modelReference="model1" simulationReference="" resetModel="false"
range="current">
      <listOfSubTasks>
        <subTask order="1" task="task2" />
      </listOfSubTasks>
      <listOfChanges>
        <setValue target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J1_KK2']"
range="current">
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> current </ci>
          </math>
        </setValue>
      </listOfChanges>
      <listOfRanges>
        <vectorRange id="current">
          <value>1</value>
          <value>5</value>
          <value>10</value>
          <value>50</value>
          <value>60</value>
          <value>70</value>
          <value>80</value>
          <value>90</value>
          <value>100</value>
        </vectorRange>
      </listOfRanges>
    </repeatedTask>
    <repeatedTask id="task2" modelReference="model1" simulationReference="steady1" resetModel="false"
range="current1">
      <listOfChanges>
        <setValue target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J4_KK5']"
range="current1">
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <ci> current1 </ci>
          </math>
        </setValue>
      </listOfChanges>
      <listOfRanges>
        <uniformRange id="current1" start="1" end="40" numberOfPoints="100" type="linear" />
      </listOfRanges>
    </repeatedTask>
  </listOfTasks>
  <listOfDataGenerators>
    <dataGenerator id="J4_KK5_1" name="J4_KK5">
      <listOfVariables>
        <variable id="J4_KK5" name="J4_KK5" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J4_KK5']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> J4_KK5 </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="J1_KK2_1" name="J1_KK2">
      <listOfVariables>
        <variable id="J1_KK2" name="J1_KK2" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfParameters/sbml:parameter[@id='J1_KK2']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> J1_KK2 </ci>
      </math>
    </dataGenerator>
    <dataGenerator id="MKK_1" name="MKK">
      <listOfVariables>
        <variable id="MKK" name="MKK" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MKK']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MKK </ci>
      </math>
```

```
    </dataGenerator>
    <dataGenerator id="MKK_P_1" name="MKK_P">
      <listOfVariables>
        <variable id="MKK_P" name="MKK_P" taskReference="task1"
target="/sbml:sbml/sbml:model/sbml:listOfSpecies/sbml:species[@id='MKK_P']" />
      </listOfVariables>
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> MKK_P </ci>
      </math>
    </dataGenerator>
  </listOfDataGenerators>
  <listOfOutputs>
    <plot2D id="plot1" name="Steady State Scan (Boris 2D)">
      <listOfCurves>
        <curve id="curve1" logX="false" logY="false" xDataReference="J4_KK5_1" yDataReference="MKK_1" />
        <curve id="curve2" logX="false" logY="false" xDataReference="J4_KK5_1" yDataReference="MKK_P_1" />
      </listOfCurves>
    </plot2D>
    <report id="report1" name="Steady State Values (Boris2D)">
      <listOfDataSets>
        <dataSet id="col0" dataReference="J4_KK5_1" label="" />
        <dataSet id="col1" dataReference="J1_KK2_1" label="" />
        <dataSet id="col2" dataReference="MKK_1" label="" />
        <dataSet id="col3" dataReference="MKK_P_1" label="" />
      </listOfDataSets>
    </report>
  </listOfOutputs>
</sedML>
```

References

[1] F. T. Bergmann, "A Simple Nested Simulation for SED-ML," 2010. [Online]. Available: http://dx.doi.org/10.1038/npre.2010.4257.1.

[2] D. Waltemath, F. T. Bergmann, R. Adams and N. Le Novère, "Simulation Experiment Description Markup Language," 2011. [Online]. Available: http://sed-ml.org/documents/sed-ml-L1V1.pdf.

[3] F. T. Bergmann, "Nested Simulation Experiments," 8 March 2010. [Online]. Available: http://frank-fbergmann.blogspot.com/2010/03/nested-simulation-experiments.html.

[4] N. Le Novère, "Reply to Nested Proposal v2," 2012. [Online]. Available: http://tinyurl.com/7zq5zkg.

[5] F. T. Bergmann, "SBRML Interoperability," 2011. [Online]. Available: http://precedings.nature.com/documents/6329/version/1.

[6] "Numerical Markup Language," 2012. [Online]. Available: http://code.google.com/p/numl/.

[7] J. Cooper and G. Mirams, "Functional Curation: Potential Future Directions for SED-ML," 2011. [Online]. Available: http://dx.doi.org/10.1038/npre.2011.6327.1.

[8] F. T. Bergmann, "libSedML - Libraries and Tools supporting SED-ML," 2012. [Online]. Available: http://libsedml.sourceforge.net/libSedML/Welcome.html.

[9] F. T. Bergmann, "SED-ML Web Tools," 2012. [Online]. Available: http://sysbioapps.dyndns.org/SED-ML_Web_Tools.

[10] F. Bergmann, "SBRML Interoperability," 2011. [Online]. Available: http://precedings.nature.com/documents/6329/version/1.