Physiological Hierarchy Simulation Parameter markup language (PHSP)

Language Specification Version 1.0

PhysioDesigner Project

July 30, 2014

Table of Contents

Abstract

Physiological Hierarchy Simulation Parameter markup language (PHSP) is an XML based language describing a configuration for performing multiple simulations scanning different parameter values and initial values in certain ranges on Flint. In a PHSP document, there are a section to define parameters specifying ranges for scanning, and a section to define targeted entities in a model, i.e. physical quantities for PHML and species and parameters for SBML.

1. Motivation

A function to performing multiple simulations on a model with different parameter values or initial values is one of highly demanded functions by users for Flint. Such simulations are useful to analyze the dependency or stability of the model in terms of the concerned parameters. For the function, we need to describe the ranges of values in which parameters take values for multiple simulations.
SED-ML could be a candidate language to describe such information. Actually it is possible, but so far it is not convenient. So now we define and use PHSP. However when some standard such as SED-ML come to be matured to describe necessary information, we can think of using such standard.

2. PHSP top level element: phsp

PHSP is a top level element has a model as its child element.
This takes attributes version and xmlns. Currently at April 1st, 2013, version is 1.0. The name space xmlns is http://www.physiodesigner.org/2013/ns/phsp/1.0.
This is a container of multiple model elements.

Table 2: *Attributes and nested elements for phsp*

Element	Used-by	Description
phsp	None	PHSP top element. complex type

Attribute	type	Value
version	xs:decimal	
xmlns	namespace URI	http://www.physiodesigner.org/2013/ns/phsp/1.0

Sub-elements	Occurrence	Description
model	1..∞	complex type

2.1. model

model specifies the location of the model by the iref attribute. The language describing the model must be specified by format attribute. The format can take either one of two values, “phml” or “sbml”.
This has two child elements, i.e. parameter-set and target-set.

Table 2.1: *Attributes and nested elements for model*

Element	Used-by	Description
model	phsp	complex type

Attribute	type	Value
iref	xs:string	
format	xs:string	choose from phml / sbml

Sub-elements	Occurrence	Description
parameter-set	1.1	complex type
target-set	1.1	complex type
description	0..1	simple type

2.1.1. parameter-set

See the section 3. PHSP components: parameter-set / parameter.

2.1.2. target-set

See the section 4. PHSP components: target-set / target.

2.1.3. description

Table 2.1.3: *Simple type element description*

Element	Used-by	Description
description	model	the value must be xs:string

3. PHSP components: parameter-set / parameter

The `parameter-set` element is a container of parameter.

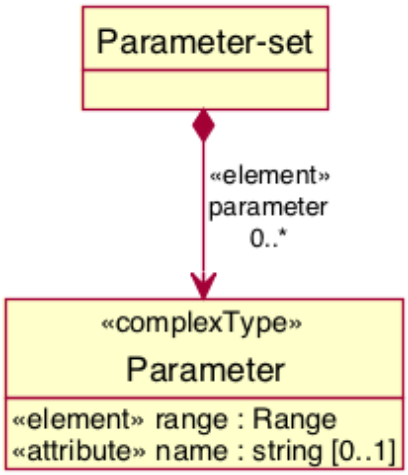

Fig. 3: *Subclasses of the Parameter-set class*

A `parameter` is defined for representing a value which will be used as a static-parameter (constant) or initial value in a model. The `parameter` is defined with a `range` in which the parameter takes one value for single simulation run, so that multiple simulations can be performed with several parameter values within the `range`.
An attribute `name` must be given in `parameter` element. This `name` is used to refer this parameter's value in the `target` section. The range is defined in the child element `range`.

Table 3-i: *Attributes and nested elements for parameter-set*

Element	Used-by	Description
parameter-set	model	complex type sub-elements must occur in sequential order as listed in Sub-elements

Attribute	type	Value
None		

Sub-elements	Occurrence	Description
parameter	0..∞	complex type

Table 3-ii: *Attributes and nested elements for parameter*

Element	Used-by	Description

parameter	paremeter-set	complex type sub-elements must occur in sequential order as listed in Sub-elements

Attribute	type	Value
name	xs:string	

Sub-elements	Occurrence	Description
range	1..∞	complex type

3.1. range

The range element is used to define the range in which the parameter value can be.

Table 3.1: *Attributes and nested elements for range*

Element	Used-by	Description
range	parameter	complex type

Attribute	type	Value
type	xs:string	choose from interval / enum
lower	xs:integer	
upper	xs:integer	
step	xs:integer	

When the type="interval", the range is defined by specifying the lower and upper limits. For numerical reason, it is also possible to define a step so that a set of discrete values between the lower and upper limits can be generated at the simulator.
If the lower, upper and step values are given numerically, these values are specified by attributes lower, upper and step, respectively.

```
<range type="interval" lower="1" upper="10" step="1">
```

When the type="enum", the range is represented by a set of numbers which the parameter can take, instead of specifying a range with lower, upper and step. These numbers must be written as comma-delimited text.

```
<range type="enum">1,2,3,4,5,6,10,9,8,7</range>
```

4. PHSP components: target-set / target

The `target-set` element is a container of `target`.

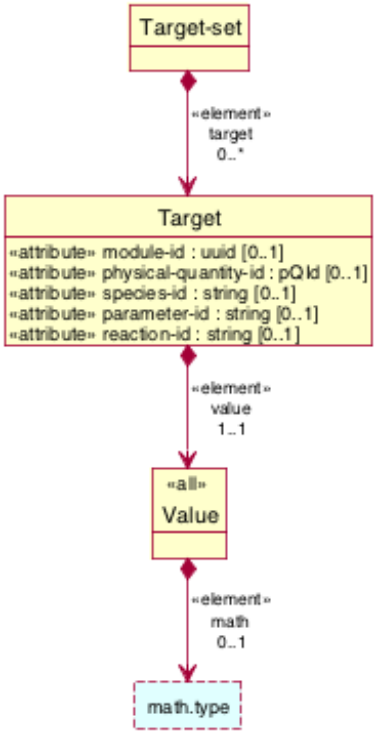

Fig. 4: *Subclasses of the Target-set class*

Table 4-i: *Attributes and nested elements for target-set*

Element	Used-by	Description
target-set	model	complex type

Attribute	type	Value
None		

Sub-elements	Occurrence	Description
target	0..∞	

Table 4-ii: *Attributes and nested elements for target*

Element	Used-by	Description
target	target-set	complex type

Attribute	type	Value
module-id	UUID	only when format=”phml” in model element
physical-quantity-id	xs:integer	only when format=”phml” in model element

species-id	xs:string	only when format=”sbml” in model element
parameter-id	xs:string	only when format=”sbml” in model element
reaction-id	xs:string	only when format=”sbml” in model element

Sub-elements	Occurrence	Description
value	1..1	complex type

The target specifies the targeted model to which modification of parameter values applies.

When format of the model element is "phml", attributes module-id, physical-quantity-id must be given. In a PHML model, functional modules and instances are the target to set values of physical-quantities. In a case of functional modules, the module-id attribute of the target element corresponds to the module-id attribute of the module element of PHML. However in a case of instance modules, the module-id attribute of the target element must correspond to the alias-module-id of the instance element of PHML.

```
1.  <target-set>
2.    <target module-id="c1db88ee-e2fc-40fc-a6f1-91e5a23d0c3b" physical-quantity-id="2">
3.      <value>
4.        <m:math><m:ci>x1</m:ci></m:math>
5.      </value>
6.    </target>
7.  </target-set>
```

When format of the model element is "sbml", attributes either species-id or parameter-id, and reaction-id if necessary to specify local parameter, must be given.

```
1.  <target-set>
2.    <target species-id="s1">
3.      <value>
4.        <m:math><m:ci>x1</m:ci></m:math>
5.      </value>
6.    </target>
7.  </target-set>
```

4.1. value

The value is defined by using MathML. If the value is defined as <m:math><m:ci>x1</m:ci></m:math>, it means that the physical quantity with physical-quantity-id="2" in a module with module-id="1211...." will be x1 in a simulation. Here “x1” is a name of a parameter, hence it must be defined as a parameter in parameter-set section.

In the definition of the value, it is possible to use pre-defined functions to generate random (probabilistic) value. Available functions are listed in the Appendix C.

Table 4.1: *Attributes and nested elements for value*

Element	Used-by	Description
value	target	complex type

Attribute	type	Value
None		

Sub-elements	Occurrence	Description
m:math	0..∞	

Appendix A : PHSP UML Overview

PHSP Class Diagram

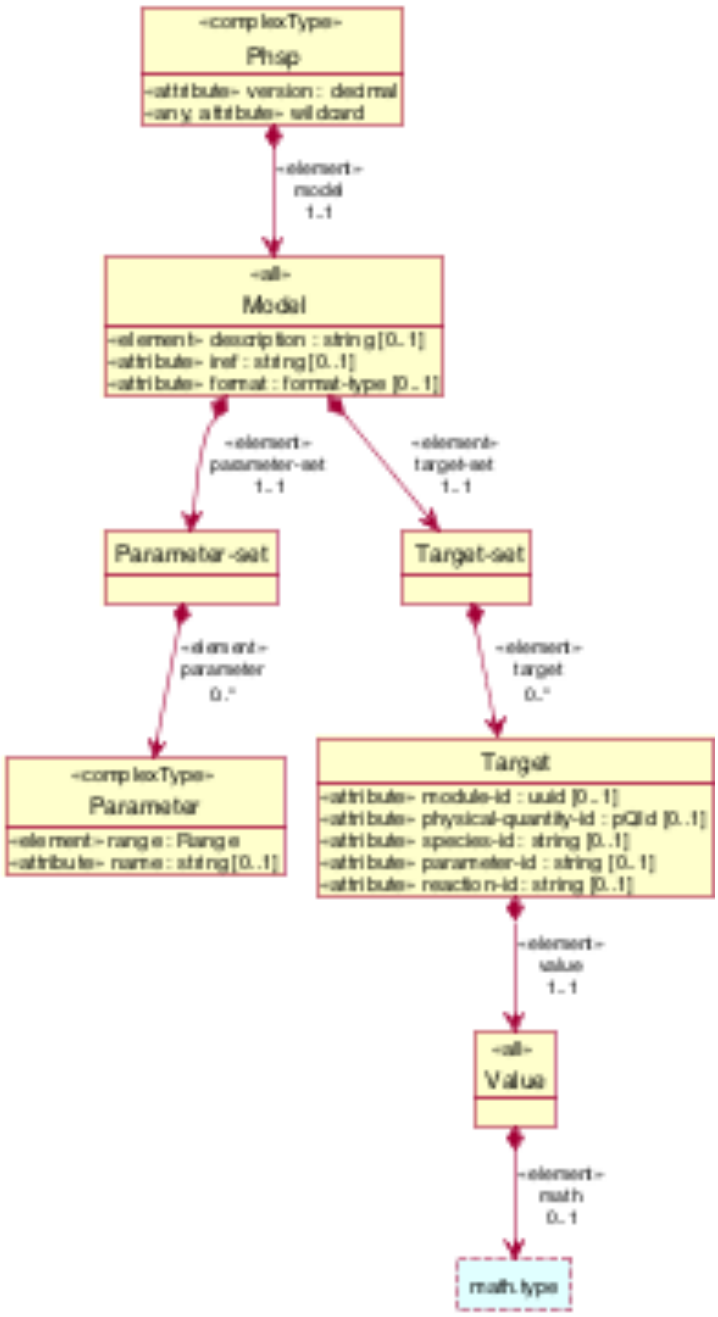

Fig. appendix-A: *PHSP all classes*

Appendix B : Example

```
1.  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2.  <phsp version="1.0" xmlns="http://www.physiodesigner.org/2013/ns/phsp/1.0"
                        xmlns:m="http://www.w3.org/1998/Math/MathML">
3.    <model iref="/xxx/yyy/ddd/model.phml" format="phml">
4.      <parameter-set>
5.        <parameter name="x1">
6.          <range type="interval" lower="1" upper="10" step="1"/>
7.        </parameter>
8.        <parameter name="y1">
9.          <range type="enum">1,2,3,4,5,6,10,9,8,7</range>
10.       </parameter>
11.     </parameter-set>
12.     <target-set>
13.       <target module-id="1211DC07-291E-43FC-AEB5-A621656BC120"
                  physical-quantity-id="2">
14.         <value>
15.           <m:math><m:ci>x1</m:ci></m:math>
16.         </value>
17.       </target>
18.       <target module-id="1211DC07-291E-43FC-AEB5-A621656BC120"
                   physical-quantity-id="4">
19.         <value>
20.           <m:math><m:ci>y1</m:ci></m:math>
21.         </value>
22.       </target>
23.       <target module-id="B60A8767-92C3-4F0F-9BE4-E5B150BB3C7A"
                   physical-quantity-id="1">
24.         <value>
25.           <m:math><m:apply><m:plus/><m:ci>x1</m:ci><m:ci>y1</m:ci></m:apply></m:math>
26.         </value>
27.       </target>
28.     </target-set>
29.       <description>Here is description.</description>
30.   </model>
31.   <model>
32.     .....
33.   </model>
34. </phsp>
```

Appendix C : Functions

Random number generator functions which are available in definition for the `target` element are listed below.

- **uniform_variate**(a, b, left=0/1, right=0/1)
  The uniform distribution. Return a real value N which follows the uniform distribution satisfying a [<|<=] N [<|<=] b.
  Lower end point is included if left=0, not included if left=1. The same for right side end point.
  For example, `uniform_variate(a,b,0,1)` corresponds to a value in the interval [a,b).

- **poisson_variate**(lambda)
  The Poisson distribution. This function takes one parameter lambda. The mean and variance are equal to lambda.

- **exponential_variate**(lambda)
  The exponential distribution. This function takes one parameter lambda. The mean of the distribution is 1/lambda, and the variance is 1/(lambda)^2. This returns a value between non-negative value (i.e. 0 to infinity).

- **gamma_variate**(alpha, beta)
  The gamma distribution. Parameters alpha>0 and beta>0.

- **gauss_variate**(mu, sigma)
  The Gauss distribution. The parameter mu is the mean, and sigma is the standard deviation. This function is a little bit faster than normalvariate().