Physiological Hierarchy Parameter List markup language (PHPL)

Language Specification Version 1.0

PhysioDesigner Project

July 30, 2014

Table of Contents

Abstract

Physiological Hierarchy Parameter List markup language (PHPL) is an XML based language describing a set of parameters (values of static-parameter type physical quantities and initial values of state type physical quantities) to store them apart from PHML model itself. Then a PHPL file can be used to set multiple parameter values in a PHML model at once. This will help to switch over a model representing a certain physiological function into a model of another kind of function. PHPL can be used also for changing a set of parameter values at once in mid-course of a simulation.

1. Motivation

Sometimes a single model can represent different physiological phenomena by working with different parameter values. In such a case, if it is possible to switch those parameter values from one model to another model at once, it would be convenient. There is also another kind of demand. During a simulation, sometimes we want to change parameter values when a certain event happens.
To extract the parameter values from a PHML model, store them, and re-set them to the model could be a meaningful protocol for modeling and simulation. We can also store a couple of PHPL files in a database in association with a model, so that users can select one of sets of parameter values when they download the model.

2. PHPL top level element: phpl

phpl has `header` and `module-set` as its child elements.
This takes attributes `version` and `xmlns`. Currently at April 6st, 2013, `version` is `1.0`. The name space `xmlns` is `http://www.physiodesigner.org/2013/ns/phpl/1.0`.

Table 2: *Attributes and nested elements for phpl*

Element	Used-by	Description
phpl	None	PHPL top element. complex type

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

Sub-elements	Occurrence	Description
header	1..1	complex type
module-set	1..∞	complex type

2.1. header

See the section 3. PHPL components: header.

2.2. module-set

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

3. PHPL components: header

This contains general information for the model and this parameter set.

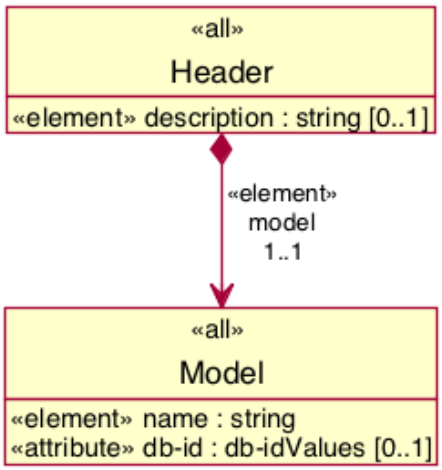

Fig. 3: Subclasses of the Header class

Table 3: *Attributes and nested elements for header*

Element	Used-by	Description
header	phpl	complex type sub-elements must occur in sequential order as listed in Sub-elements

Attribute	type	Value
None		

Sub-elements	Occurrence	Description
model	1..1	complex type
description	0..1	simple type

3.1. model

This is information of the model from which this parameter set was extracted. If the model has db-id, the db-id is given as an attribute `db-id`. This is optional. The name of the model must be given in the sub-element.

Table 3.1: *Attributes and nested elements for model*

Element	Used-by	Description
model	header	complex type

Attribute	type	Value

db-id	xs:string	

Sub-elements	Occurrence	Description
name	1..1	simple type

3.1.1. name

The name of the model.

Table 3.1.1: *Simple type element name*

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

3.2. description

This is optional. This explains what this parameter set represents, means or whatever else.

Table 3.2: *Simple type element description*

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

4. PHPL components: module-set / module

This is a container of multiple `module` elements.

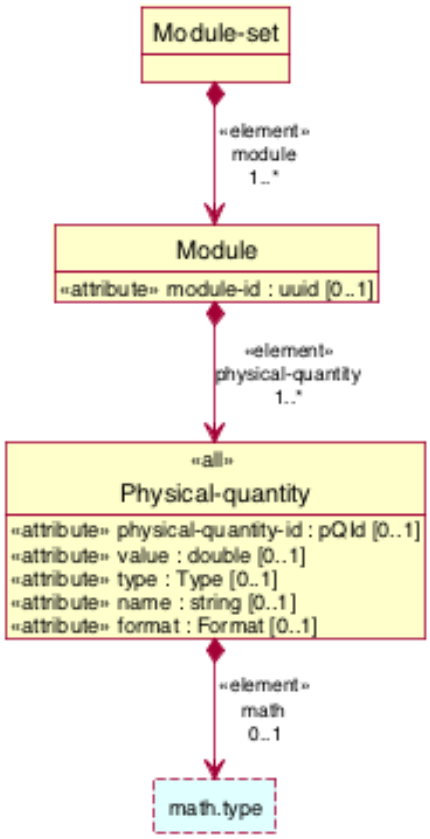

Fig. 4: Subclasses of the Module-set class

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

Element	Used-by	Description
module-set	phpl	complex type

Attribute	type	Value
None		

Sub-elements	Occurrence	Description
module	1..∞	

There are two targets in PHML to which this `module` corresponds to. That is a `module` of functional type and `instance`.
This module is characterized by an attribute `module-id`. In a case of that the target is a functional module in the PHML model, this `module-id` corresponds to

the module-id attribute in the module element of PHML. In a case that the target is an instance, this module-id corresponds to the alias-module-id attribute in the instance element of PHML.

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

Element	Used-by	Description
module	module-set	complex type

Attribute	type	Value
module-id	UUID	

Sub-elements	Occurrence	Description
physical-quantity	1..∞	complex type

4.1. physical-quantity

The physical-quantity is specified by physical-quantity-id and value as mandatory attributes. The physical-quantity-id is corresponds to the physical-quantity-id in the PHML model. The value of the value attribute must be numeric.
Besides, optionally type and name attributes can be given. The type can take either static-parameter or initial-value, corresponding to the target physical-quantity in the PHML model. The name corresponds to the name of the target physical-quantity in the PHML model.
In a case that the value is a vector or a matrix, the value cannot be fit in the value attribute. The format attribute specifies "math", and the value must be given by using MathML in the sub-element m:math followed by m:matrix and m:matrixrow.

Table 4.1: *Attributes and nested elements for physical-quantity*

Element	Used-by	Description
physical-quantity	module	complex type

Attribute	type	Value
physical-quantity-id	xs:integer	
value	xs:double	
type	xs:string	choose from static-parameter / initial-value
name	xs:string	
format	xs:string	math

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

Appendix A : PHPL UML Overview

PHPL Class Diagram

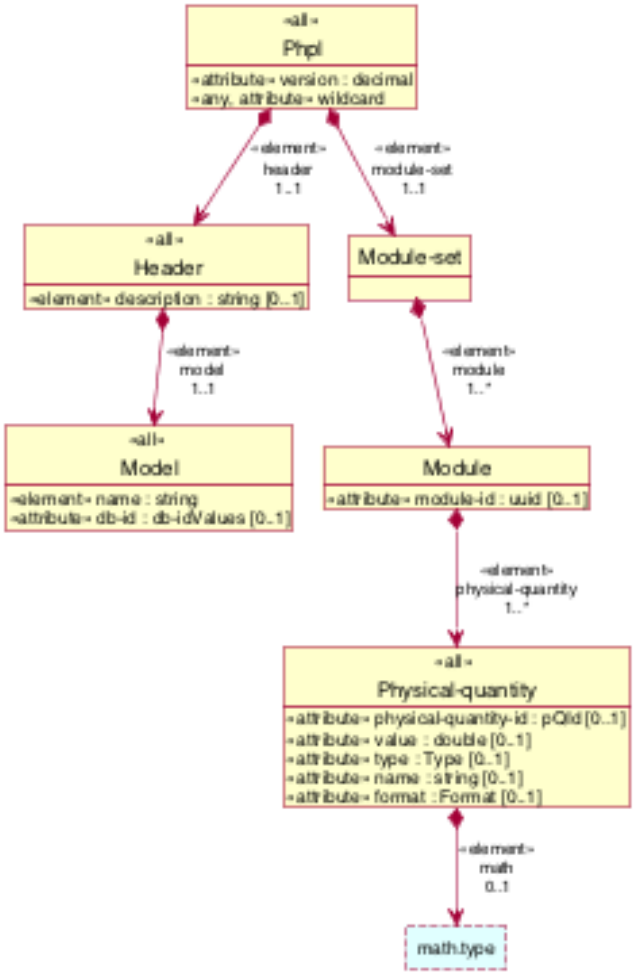

Fig. appendix-A: *PHPL all classes*

Appendix B : Example

```
1.  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2.  <phpl xmlns="http://www.physiodesigner.org/2013/ns/phpl/1.0" xmlns:m="http://
    www.w3.org/1998/Math/MathML">
3.    <header>
4.      <model db-id= "">
5.        <name>my model</name>
6.      </model>
7.      <description>This is description.</description>
8.    </header>
9.    <module-set>
10.     <module module-id="ea3b84ed-3cad-4758-90d9-d53fd7fa1292">
11.       <physical-quantity type="static-parameter" physical-quantiy-id="3" name="c"
    value="1.75"/>
12.       <physical-quantity type="initial-value" physical-quantiy-id="2" name="y"
    value="0.2"/>
13.     </module>
14.     <module module-id="f69720e0-d7bb-4a80-93c4-c02339d58823">
15.       <physical-quantity type="static-parameter" physical-quantiy-id="2"
    name="pulse_onset_time" value="20"/>
16.     </module>
17.     <module module-id="56f05090-c974-4d26-8f2e-debae430acfc">
18.       <physical-quantity type="static-parameter" physical-quantiy-id="3" name="c"
    value="1.75"/>
19.       <physical-quantity type="initial-value" physical-quantiy-id="2" name="x"
    value="0.2"/>
20.       <physical-quantity type="static-parameter" physical-quantiy-id="4" name="M"
    format="math">
21.         <m:math><m:matrix>
22.           <m:matrixrow><m:cn>1</m:cn><m:cn>2</m:cn></m:matrixrow>
23.           <m:matrixrow><m:cn>3</m:cn><m:cn>4</m:cn></m:matrixrow>
24.         </m:matrix></m:math>
25.       </physical-quantity>
26.     </module>
27.   </module-set>
28. </phpl>
```