Enterprise Architect

User Guide Series

Parametric Simulation using OpenModelica

Author: Sparx Systems

Date: 6/06/2016

Version: 1.0

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

Parametric Simulation using OpenModelica

SysML Parametric models support the engineering analysis of critical system parameters, including the evaluation of key metrics such as performance, reliability and other physical characteristics. These models unite requirements models with system design models by capturing executable constraints based on complex mathematical relationships. Parametric diagrams are specialized Internal Block diagrams that help you, the modeler, to combine behavior and structure models with engineering analysis models such as performance, reliability, and mass property models. Enterprise Architect helps you to develop and simulate SysML Parametric models quickly and simply.

This text is derived from the SysML entry in the online Wikipedia.

The advantages of SysML over UML for systems engineering become obvious if you consider a concrete example, such as modeling an automotive system. With SysML you can use Parametric diagrams to precisely define performance and mechanical constraints such as maximum acceleration, curb weight, air conditioning capacity, and interior cabin noise management.

For further information on the concepts of SysML Parametric models, refer to the official OMG SysML website and its linked sources.

Enterprise Architect allows you to extend the utility of your SysML parametric models by annotating them with extra information that allows the model to be simulated. The resulting model is then generated to as a Modelica model and can be solved using OpenModelica.

SysMLSimConfiguration Artifact

The simulation properties for your model are stored against a simulation artifact. This preserves your original model and allows multiple simulations to be configured against a single SysML model. The simulation artifact can be found on the Artifacts toolbox page.

User Interface Reference

The user interface for the SysML simulation is described in the following topic.

SysML Simulation Configuration Window

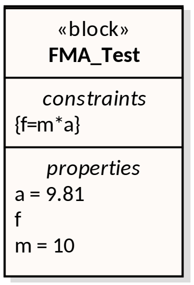
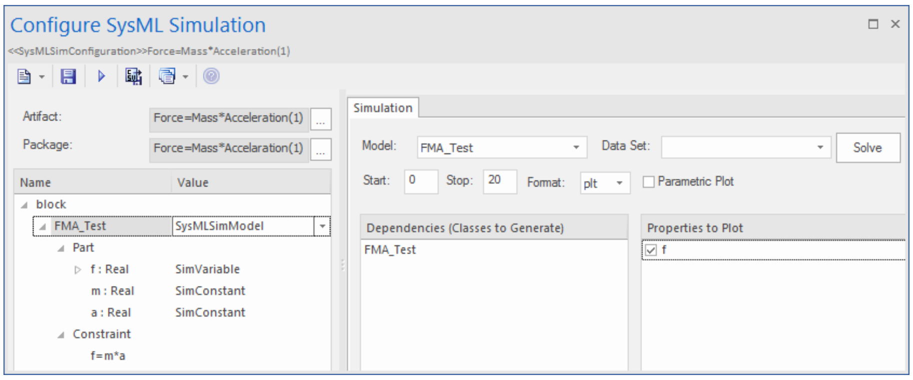
OpenModelica Example

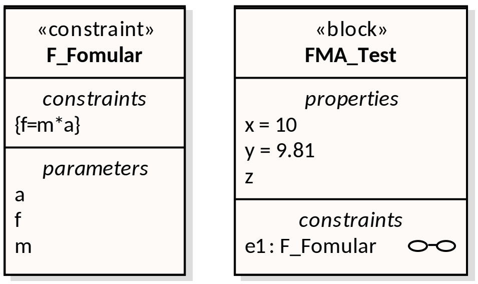
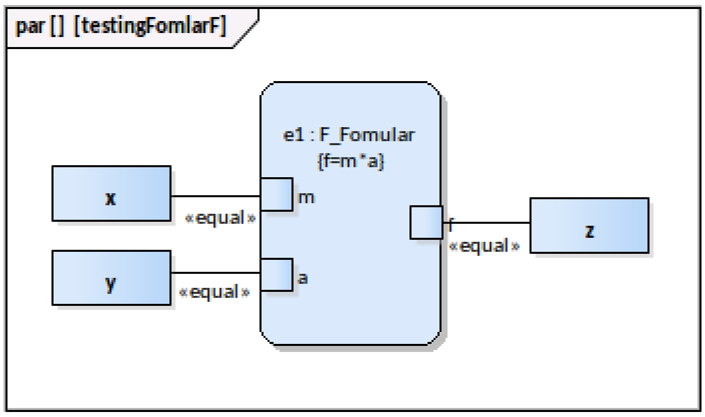
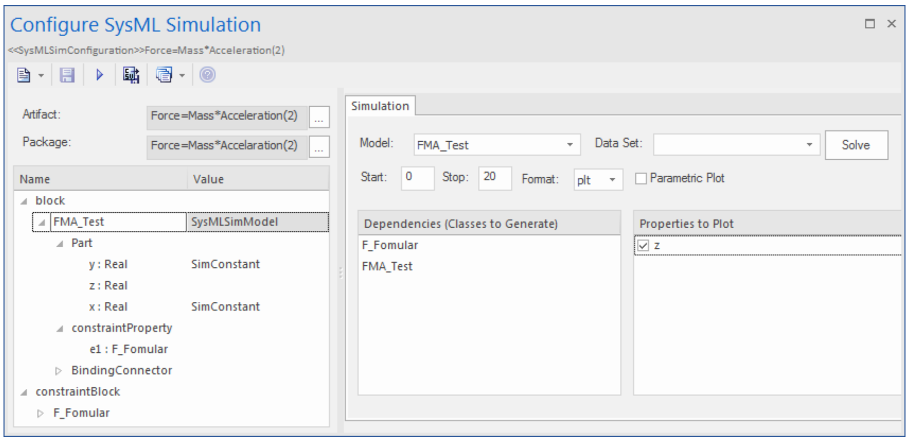
The following topic demonstrates how to create and simulate a model of a simple electrical circuit using the integration with OpenModelica.

SysML Simulation Example

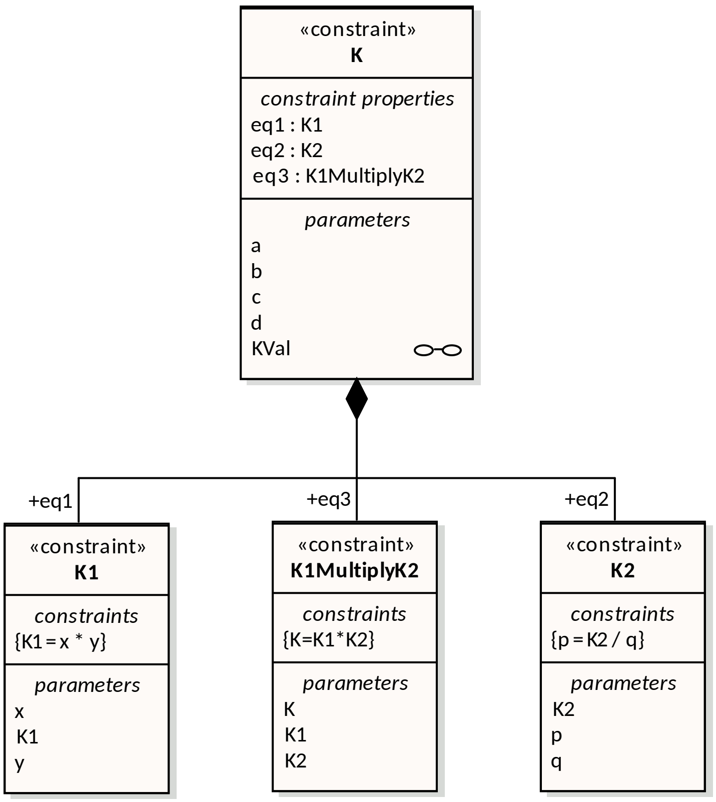
Creating a Parametric Model

Constraint Equations

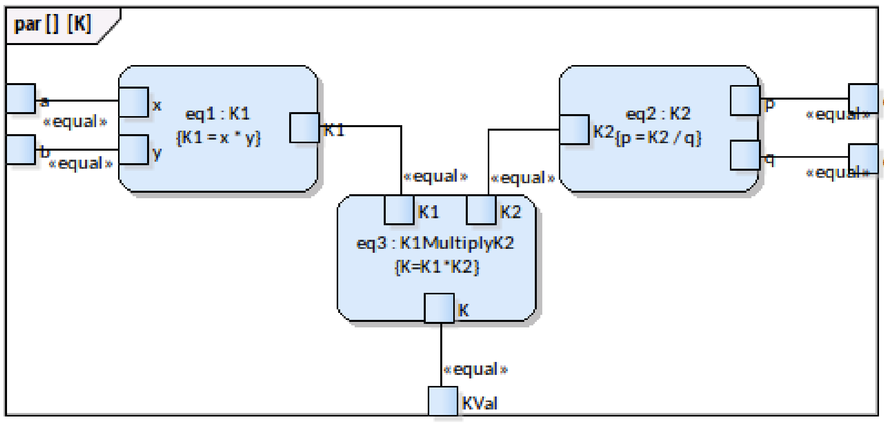
Inline constraint equations on a Block	Define constraints directly in the block is straightforward and is the easiest way to define constraint equations. As the following figure shows, constraint " f = m * a " is defined in a block element.  «block» FMA_Test constraints {f=m*a} properties a = 9.81 f m = 10 *Tips: You can define multiple constraints in one block.* • Create a SysMLSimConfiguration element "Force=Mass*Acceleration(1)", point it to the Package of "FMA_Test"; • Set "FMA_Test" to be SysMLSimModel; • Set part "a" and "m" to be "SimConstant"; optionally set part "f" to be SimVariable • Plot for "f" and simulate.  Configure SysML Simulation <<SysMLSimConfiguration>>Force=Mass*Acceleration(1) Artifact: Force=Mass*Acceleration(1) Package: Force=Mass*Accelaration(1) Name Value block; FMA_Test SysMLSimModel; Part; f : Real SimVariable; m : Real SimConstant; a : Real SimConstant; Constraint; f=m*a. Simulation: Model: FMA_Test Data Set: Solve Start: 0 Stop: 20 Format: plt ☐ Parametric Plot. Dependencies (Classes to Generate): FMA_Test. Properties to Plot: ☑ f. A chart should be plotted with f = 98.1 (which comes from 10 * 9.81)
Reuseable Constraint Blocks	If one equation is commonly used in many blocks, a constraint block can be created and used as constraint property in each blocks. Here is the changes we made based on the previous example: • Create a Constraint Block element "F_Formular" with 3 parameters a, m, f and a constraint "f = m * a";

	 «constraint» F_Fomular constraints {f=m*a} parameters a f m «block» FMA_Test properties x = 10 y = 9.81 z constraints e1 : F_Fomular *Tips: Primitive type "Real" will be applied if property types are empty.* • Create a Block "FMA_Test" with 3 properties x, y, z and give x and y default vlaues 10, 9.81 respectively. • Create a Parametric Diagram in FMA_Test, show the properties x, y, z; • Create a "Constraint Property" "e1" typed to "F_Fomular" and show the parameters; • Draw binding connectors between "x—m; y—a; f—z" as the following figure shows:  par [] [testingFomlarF] e1 : F_Fomular {f=m*a} x «equal» m y «equal» a f «equal» z • Create a SysMLSimConfiguration artifact element and configure as this: - Set FMA_Test to be SysMLSimModel; - Set x and y to be SimConstant; - Plot for "z" and simulate;  Configure SysML Simulation <<SysMLSimConfiguration>>Force=Mass*Acceleration(2) Artifact: Force=Mass*Acceleration(2) Package: Force=Mass*Accelaration(2) Name Value block FMA_Test SysMLSimModel Part y : Real SimConstant z : Real x : Real SimConstant constraintProperty e1 : F_Fomular BindingConnector constraintBlock F_Fomular Simulation Model: FMA_Test Data Set: Solve Start: 0 Stop: 20 Format: plt ☐ Parametric Plot Dependencies (Classes to Generate) F_Fomular FMA_Test Properties to Plot ☑ z A chart should be plotted with f = 98.1 (which comes from 10 * 9.81)
Connected Constraint Properties	SysML allows constraint properties exist in constraint blocks to define constraints in a more flexible way. In the following figure, constraint block "K" defines parameters a, b, c, d and

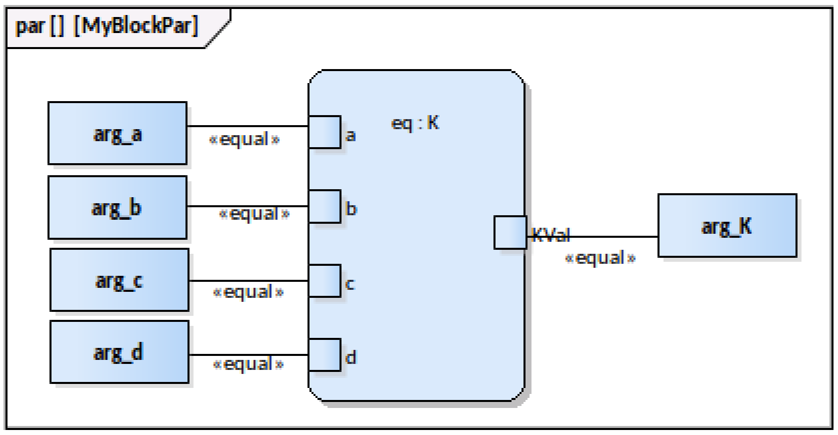
KVal; 3 constraint properties eq1, eq2 and eq3, typed to K1, K2, K3 and K1MultiplyK2 respectively.



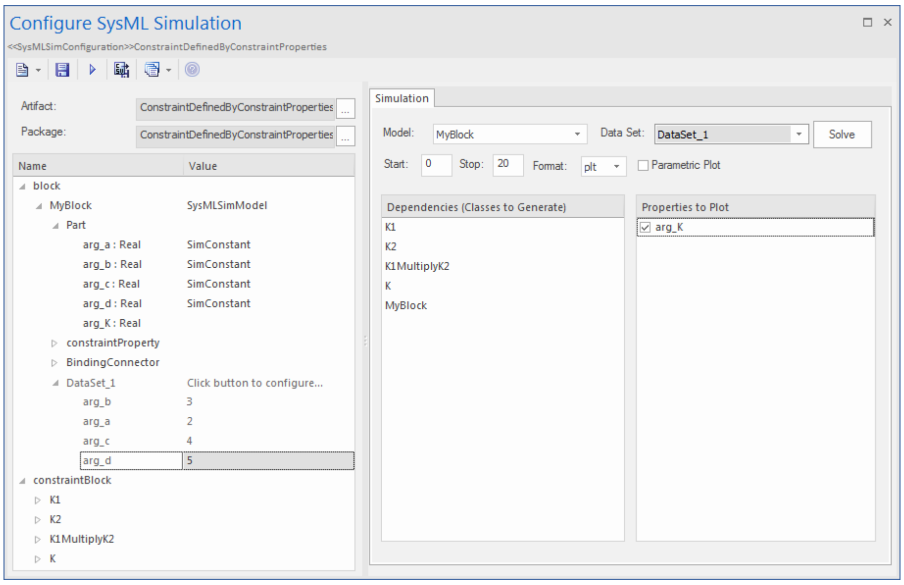
Create a parametric diagram in constraint block K and connect the parameters to the cnstraint properties with binding connectors, as below:



- Create a model MyBlock with 5 properties
- Create a constraint property eq for MyBlock and show the parameters
- Bind the properties to the parameters



- Provide values (arg_a = 2, arg_b = 3, arg_c = 4, arg_d = 5) in a data set
- In the simulation page, Select MyBlock, DataSet_1 and plot for arg_K, Simulate


Configure SysML Simulation

<<SysMLSimConfiguration>>ConstraintDefinedByConstraintProperties

Artifact: ConstraintDefinedByConstraintProperties

Package: ConstraintDefinedByConstraintProperties

Name	Value
block	
MyBlock	SysMLSimModel
Part	
arg_a : Real	SimConstant
arg_b : Real	SimConstant
arg_c : Real	SimConstant
arg_d : Real	SimConstant
arg_K : Real	
constraintProperty	
BindingConnector	
DataSet_1	Click button to configure...
arg_b	3
arg_a	2
arg_c	4
arg_d	5
constraintBlock	
K1	
K2	
K1MultiplyK2	
K	

Simulation

Model: MyBlock Data Set: DataSet_1

Start: 0 Stop: 20 Format: plt ☐ Parametric Plot

Dependencies (Classes to Generate)
K1
K2
K1MultiplyK2
K
MyBlock

Properties to Plot
✓ arg_K

The result 120 (calculated as 2 * 3 * 4 * 5) will be computed and plotted. This is the same as we do an expansion with pen and paper: K = K1 * K2 = (x*y) * (p*q); then binded with the values (2 * 3) * (4 * 5), we get 120.

What's interesting here is that we intentionally define K2's equation to be "p = K2 / q" and this example still works.

We can easily solve "K2" to be "p * q" in this example, however, in some complex examples, it is extremely hard to solve a variable from an equation, however, the EA SysMLSim can still get it right.

In summary, this example shows you how to define a constraint block in a more flexible way by construct the constraint properties. Although we demonstrated only 1 layer down into the constraint block, this mechanism could work on complex models for arbitrary level of usage.

Flows in Physical Interactions

Default Value and Initial Values

Simulation Functions

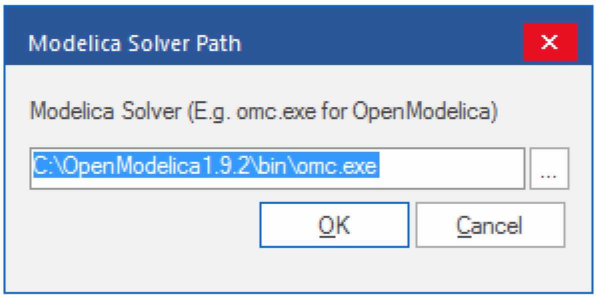
Value Allocation

SysML Simulation Configuration Window

Access

Ribbon	Simulate > SysMLSim > Manage > SysMLSim Configuration Manager
Element	Double click on an artifact with the SysMLSimConfiguration stereotype.

Toolbar

Toolbar Button	Action
Select Artfiact	Select and load an existing configuration from an artifact with the SysMLSimConfiguration.
Create Artifact	Create a new SysMLSimConfiguration or select and load an existing configuration artifact.
Select Package	Select a Package to scan for SysML elements to configure for simulation.
Reload	Reload configuration window with changes to the current package.
Configure Modelica Solver	Specify the path to the Modelica solver to use.  Modelica Solver Path Modelica Solver (E.g. omc.exe for OpenModelica) C:\OpenModelica1.9.2\bin\omc.exe
Save	Save the configuration to the current artifact.
Run	Generate, Compile, Run and display the results of the current configuration.
Export to CSV	
Generate Modelica Code	Generate the Modelica code without compiling or running.
Open Modelica Simulation Directory	Open the directory that Modelica code will be generated to.
Edit Modelica Templates	Open the Code Template Editor to allow customization of the code

	generated for Modelica.

Simulation Context

Field	Meaning
Artifact	Create a new SysMLSimConfiguration or select and load an existing configuration artifact.
Package	Select a Package to scan for SysML elements to configure for simulation.

Package Element List

Element Type	Behavior
ValueType	ValueType elements can be mapped to a corresponding simulation type.
Block	Block elements can be mapped to SysMLSimClass or SysMLSimModel. A SysMLSimModel is a possible top level element for a simulation, and allows creation of data sets against it.
Block Properties	Properties within a block can be configured to be either a SimConstant or SimVariable.
Constraint Block	

Simulation Panel

Field	Meaning
Model	Allows selection of the top level type for the simulation. This is pre-populated with the blocks marked as a SysMLSimModel.
Data Set	Allows selection of a dataset for the selected model.
Start	Specify the initial time where the simulation is started.
Stop	Specify the time where the simulation ends.
Format	
Parametric Plot	

Dependencies	Shows a list of the types that need to be generated to simulate this model.
Properties to Plot	Provides a list of properties that are involved with the simulation, and allows selection of which you want to plot.

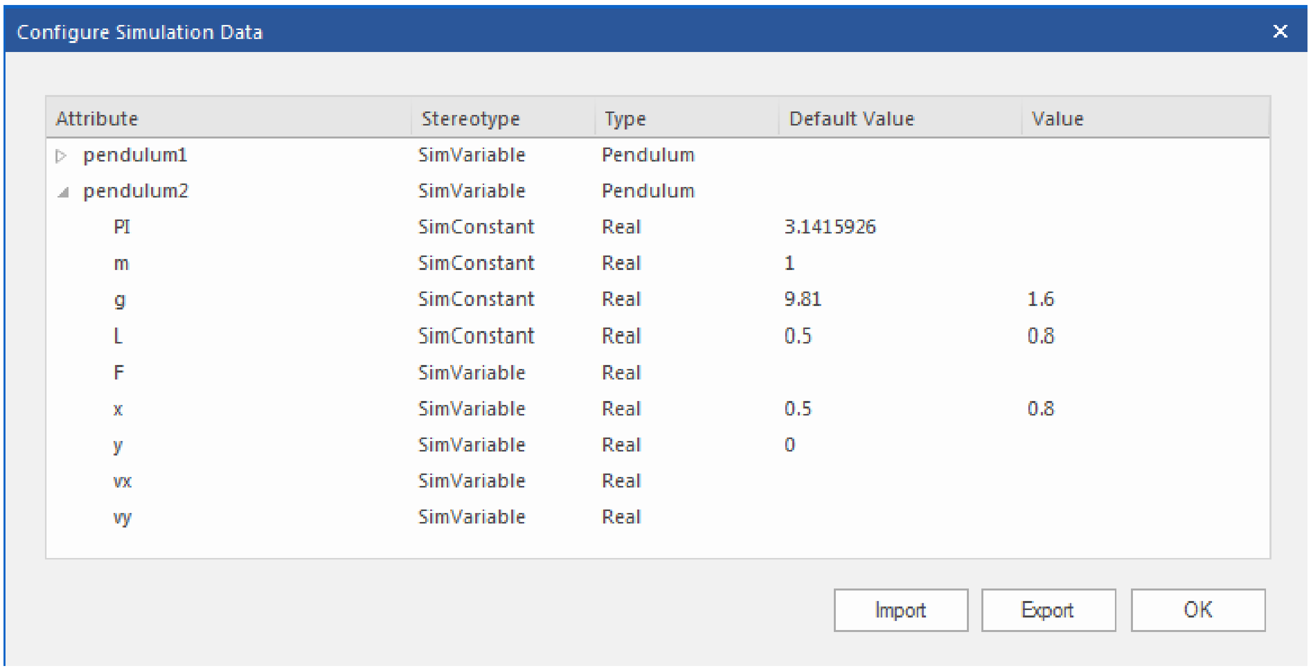
Model Analysis using Datasets

Every SysML block used in a Parametric model can have multiple datasets defined against them. This allows for repeatable simulation variations using the same SysML model. When running a simulation, a user is able to select from the datasets specified against the model before starting.

Dataset Management

Create	New Datasets can be created by right clicking on a block selecting 'Create Simulation Dataset'.
Duplicate	To duplicate an existing dataset as a base for creating a new dataset, right click on the dataset and select 'Duplicate'.
Delete	To remove a dataset that is no longer required, right click on the dataset and select 'Delete Dataset'.
Set Default	The default dataset used by a SysMLSimClass when used as a property type or inherited can be set by right clicking on the dataset and selecting 'Set as Default'. The properties used by a model will use this default configuration unless the model overrides them explicitly.

Configure Simulation Data



Attribute	Stereotype	Type	Default Value	Value
▷ pendulum1	SimVariable	Pendulum		
◢ pendulum2	SimVariable	Pendulum		
PI	SimConstant	Real	3.1415926	
m	SimConstant	Real	1	
g	SimConstant	Real	9.81	1.6
L	SimConstant	Real	0.5	0.8
F	SimVariable	Real		
x	SimVariable	Real	0.5	0.8
y	SimVariable	Real	0	
vx	SimVariable	Real		
vy	SimVariable	Real		

Attribute	The Attribute column provides a tree view of all the properties in the Block being edited.

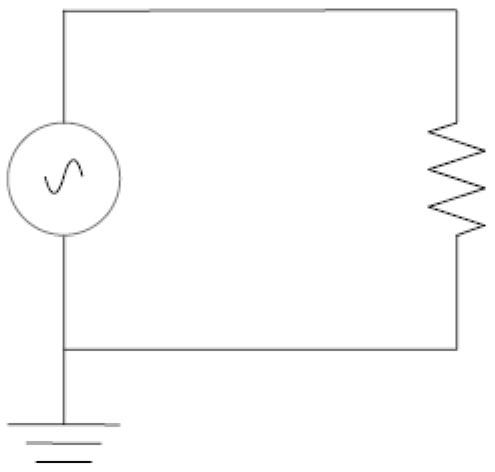
Stereotype	The stereotype column specifies for each property if it has been configured to be a constant for the duration of the simulation or variable so that the value is expected to change over time.
Type	The Type column describes the type used for simulation of this property. It can be either a primitive type (eg. Real) or a reference to a Block contained in the model. Properties referencing blocks will show the child properties specified by the referenced block below them.
Default Value	The default value column shows the value that will be used in the simulation if no override is provided. This can come from the Initial Value field in the SysML model or from the default dataset of the parent type.
Value	The value column allows the user to override the default value for each primitive value.
Export / Import	The export and import buttons allow the user to modify the values in the current dataset using an external application such as a spreadsheet before re-importing them.

SysML Simulation Example

In this section, we will walk through the creation of a SysML parametric model for a simple electrical circuit, and then use a parametric simulation to predict and chart the behavior of that circuit.

Circuit Diagram

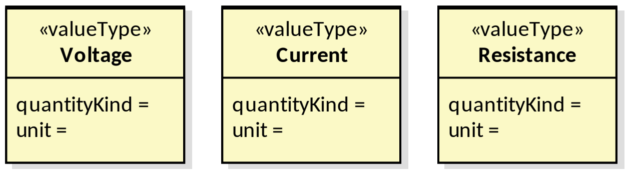
The electrical circuit we are going to model is shown below using a standard electrical circuit notation.

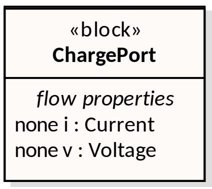
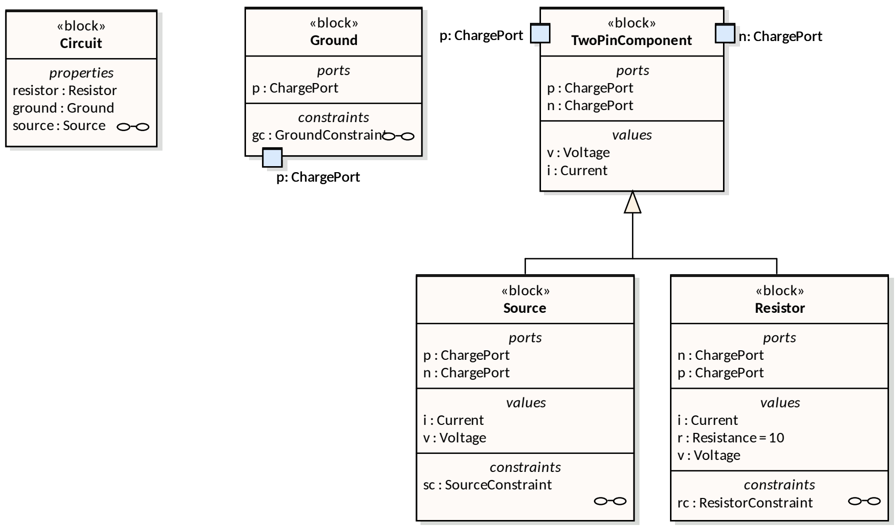


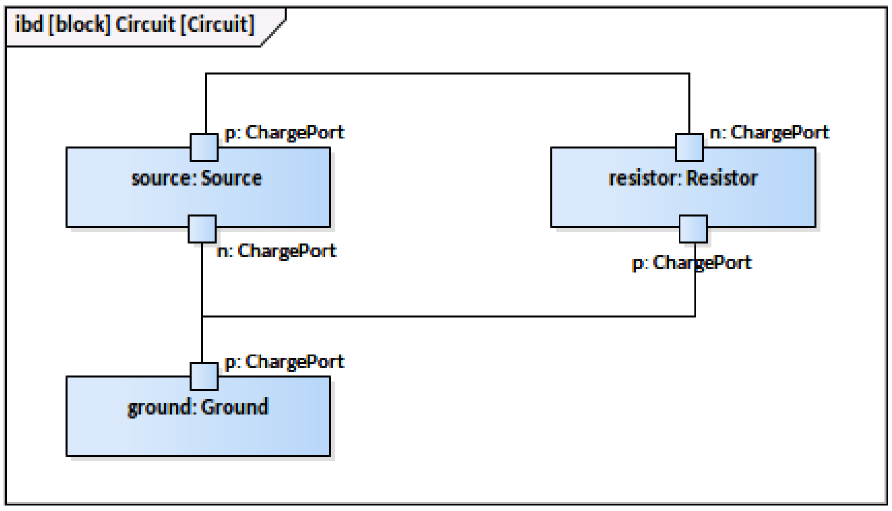
The circuit includes an AC power source, a ground and a resistor. Each of these being connected via wires.

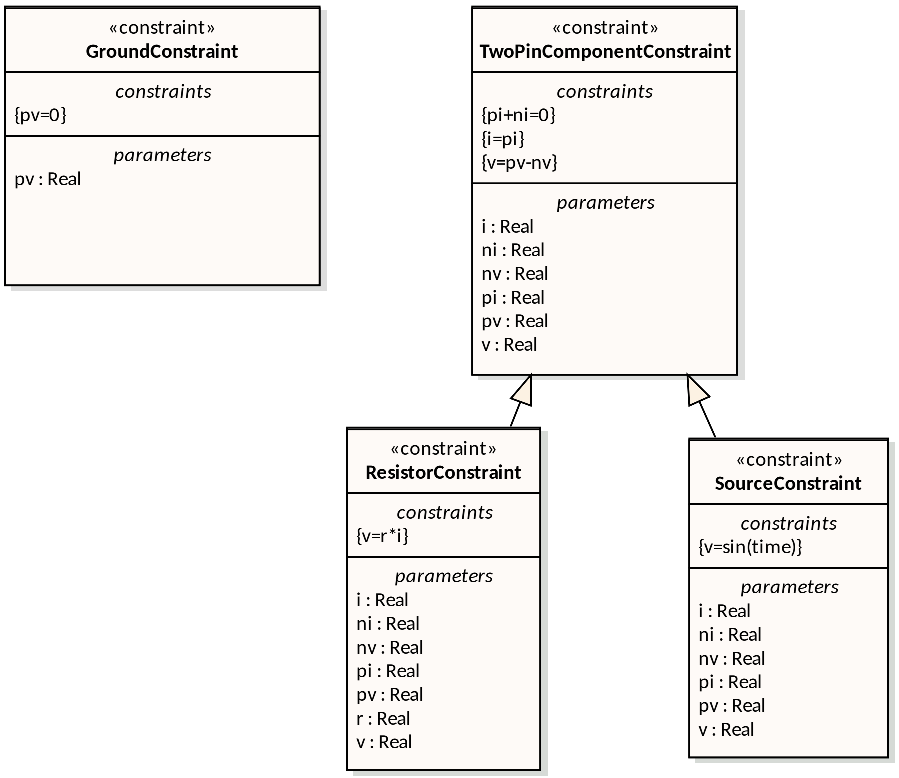
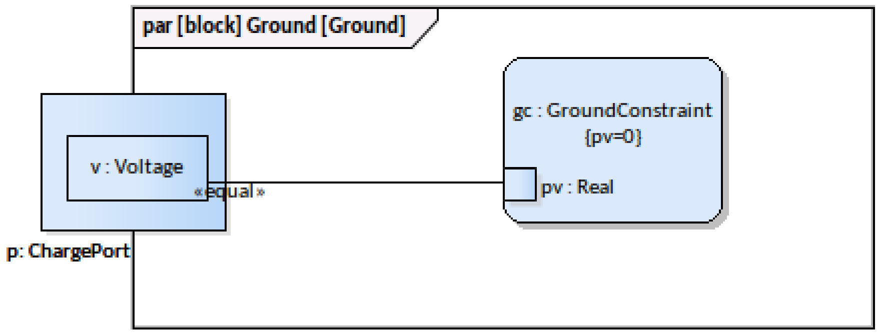
Create SysML Model

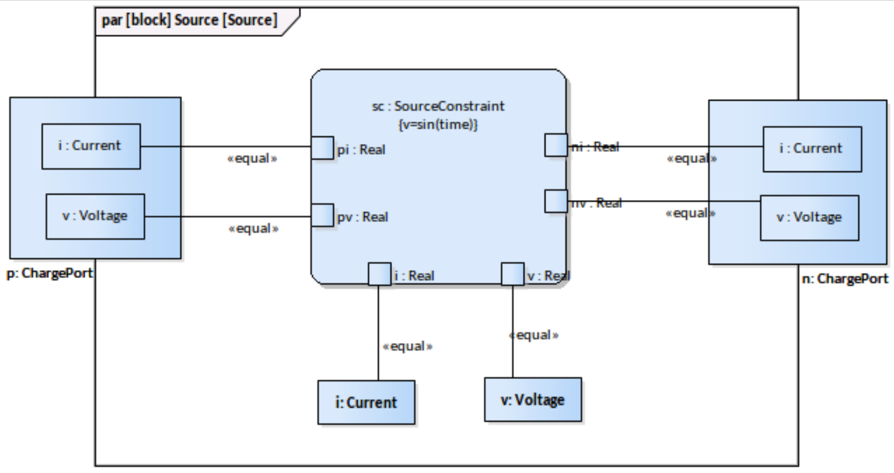
The following table shows how we can build up a complete SysML model to represent the circuit. Starting at the lowest level types and building up the model one step at a time.

Types	Define Value Types for each of Voltage, Current and Resistance. Unit and quantity kind are not important for the purpose of simulation but would be set if defining a complete SysML model. These types will be generalized from the primitive type 'Real'. In other models, you can choose to map a value type to a corresponding simulation type separate from the model.  «valueType» Voltage quantityKind = unit = «valueType» Current quantityKind = unit = «valueType» Resistance quantityKind = unit = Additionally, we define a composite type called ChargePort, which includes properties for both Current and Voltage. This type allows us to represent the electrical energy at the connectors between components.

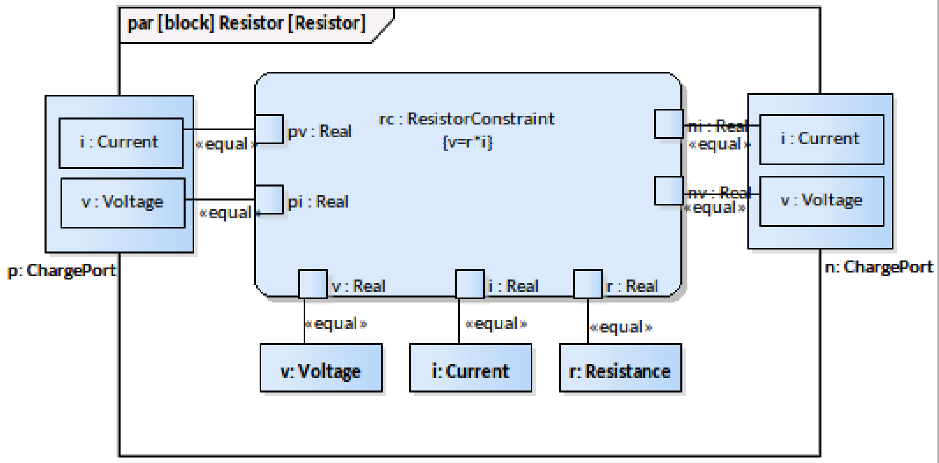
	 «block» ChargePort flow properties none i : Current none v : Voltage
Blocks	In SysML, the circuit and each of the components will be represented as Blocks. Create a Circuit block in a Block Definition Diagram (BDD). The circuit has three parts: a source, a ground, and a resistor. These parts are of different types, with different behaviors. Create a block for each of these part types. The three parts of the Circuit block are connected through ports, which represent electrical pins. The source and resistor have a positive and a negative pin. The ground has only one pin, which is positive. Electricity (electric charges) is transmitted through the pins. Create an abstract block TwoPinComponent with two ports (pins). The two ports are named p and n, and they are of type ChargePort. The following figure shows what the BDD should look like, with the blocks Circuit, Ground, TwoPinComponent, Source and Resistor.  «block» Circuit properties resistor : Resistor ground : Ground source : Source «block» Ground ports p : ChargePort constraints gc : GroundConstraint p: ChargePort p: ChargePort «block» TwoPinComponent n: ChargePort ports p : ChargePort n : ChargePort values v : Voltage i : Current «block» Source ports p : ChargePort n : ChargePort values i : Current v : Voltage constraints sc : SourceConstraint «block» Resistor ports n : ChargePort p : ChargePort values i : Current r : Resistance = 10 v : Voltage constraints rc : ResistorConstraint
Internal Structure	Create an Internal Block Diagram (IBD) for Circuit. Add properties for the source, resistor, and ground, typed by the corresponding blocks. Connect the ports with connectors. The positive pin of the source is connected to the negative pin of the resistor. The positive pin of the resistor is connected to the negative pin of the source. The ground is also connected to the negative pin of the source.

	 ibd [block] Circuit [Circuit] p: ChargePort source: Source n: ChargePort n: ChargePort resistor: Resistor p: ChargePort p: ChargePort ground: Ground Notice that this follows the same structure as the original circuit diagram, but the symbols for each component have been replaced with properties typed by the blocks defined above.
Constraints	Equations define mathematical relationships between numeric properties. In SysML, equations are represented as constraints in constraint blocks. Parameters of constraint blocks correspond to SimVariables and SimConstant of blocks (i, v, r in this example), as well as to SimVariables present in the type of the ports (pv, pi, nv, ni in this example). Create an constraint block TwoPinComponentConstraint to define parameters and equations common to sources and resistors. The equations should state that the voltage of the component is equal to the difference between the voltage at the positive and negative pin. The current of the component is equal to the current going through the positive pin. The sum of the current going through the two pins must add up to zero (one is the negative of the other). The ground constraint states that the voltage at the ground pin is zero. The source constraint defines the voltage as a sine wave with the current simulation time as parameter. The following figure shows what these constraints should look like in a BDD.

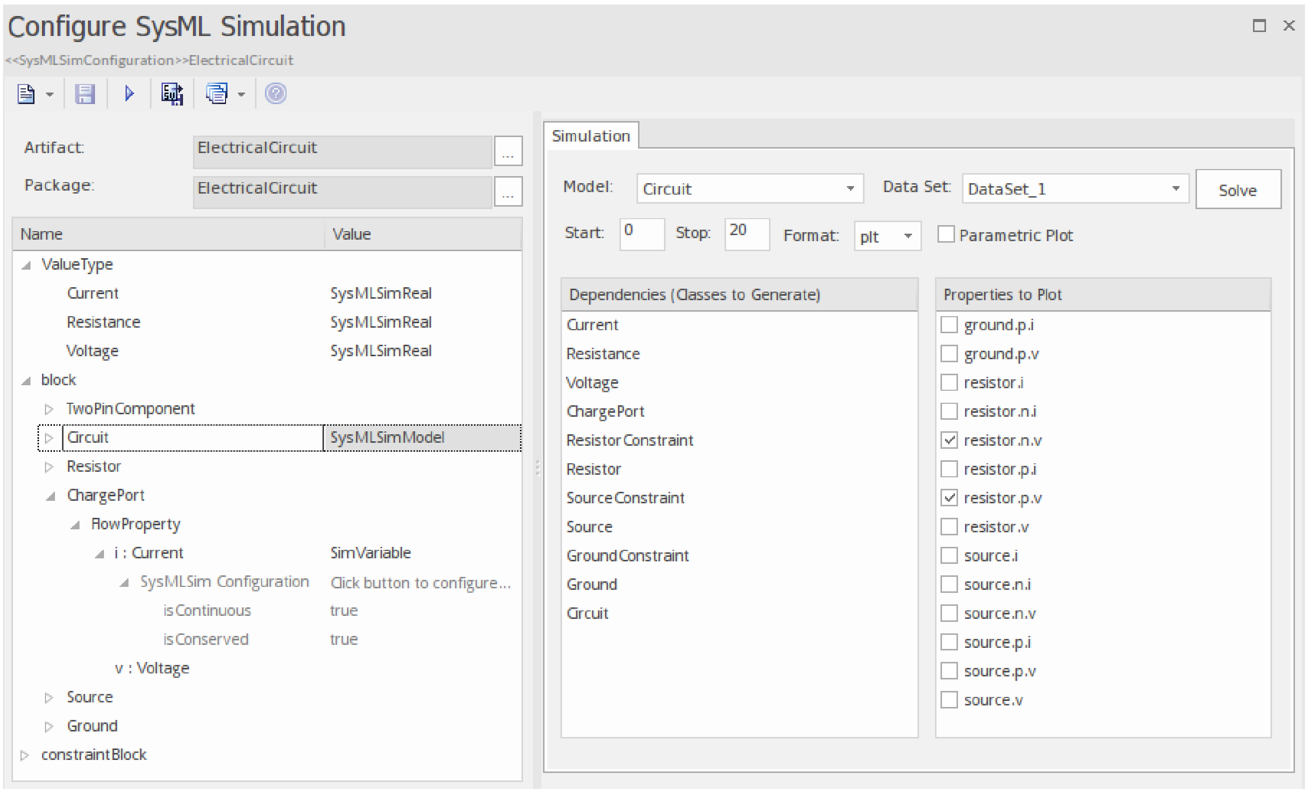
	 «constraint» GroundConstraint constraints {pv=0} parameters pv : Real «constraint» TwoPinComponentConstraint constraints {pi+ni=0} {i=pi} {v=pv-nv} parameters i : Real ni : Real nv : Real pi : Real pv : Real v : Real «constraint» ResistorConstraint constraints {v=r*i} parameters i : Real ni : Real nv : Real pi : Real pv : Real r : Real v : Real «constraint» SourceConstraint constraints {v=sin(time)} parameters i : Real ni : Real nv : Real pi : Real pv : Real v : Real
Bindings	The values of constraint parameters are equated to variable and constant values with binding connectors. Create constraint properties on each block (properties typed by constraint blocks) and bind the block variables and constants to the constraint parameters to apply the constraint to the block. block. The following figures show the bindings for the ground, the source, and the resistor respectively. For the ground constraint, bind gc.pv to p.v.  par [block] Ground [Ground] v : Voltage «equal» p: ChargePort gc : GroundConstraint {pv=0} pv : Real For the source constraint, bind sc.pi to p.i, sc.pv to p.v, sc.v to v, sc.i to i, sc.ni to n.i, and sc.nv to n.v .



For the resistor constraint, bind rc.pi to p.i, rc.pv to p.v, rc.v to v, rc.i to i, rc.ni to n.i, rc.nv to n.cflowsim.v, and rc.r to r.



Configure Simulation Behavior



Configure SysML Simulation

<<SysMLSimConfiguration>>ElectricalCircuit

Artifact: ElectricalCircuit

Package: ElectricalCircuit

Name	Value
ValueType	
Current	SysMLSimReal
Resistance	SysMLSimReal
Voltage	SysMLSimReal
block	
TwoPinComponent	
Circuit	SysMLSimModel
Resistor	
ChargePort	
FlowProperty	
i : Current	SimVariable
SysMLSim Configuration	Click button to configure...
isContinuous	true
isConserved	true
v : Voltage	
Source	
Ground	
constraintBlock	

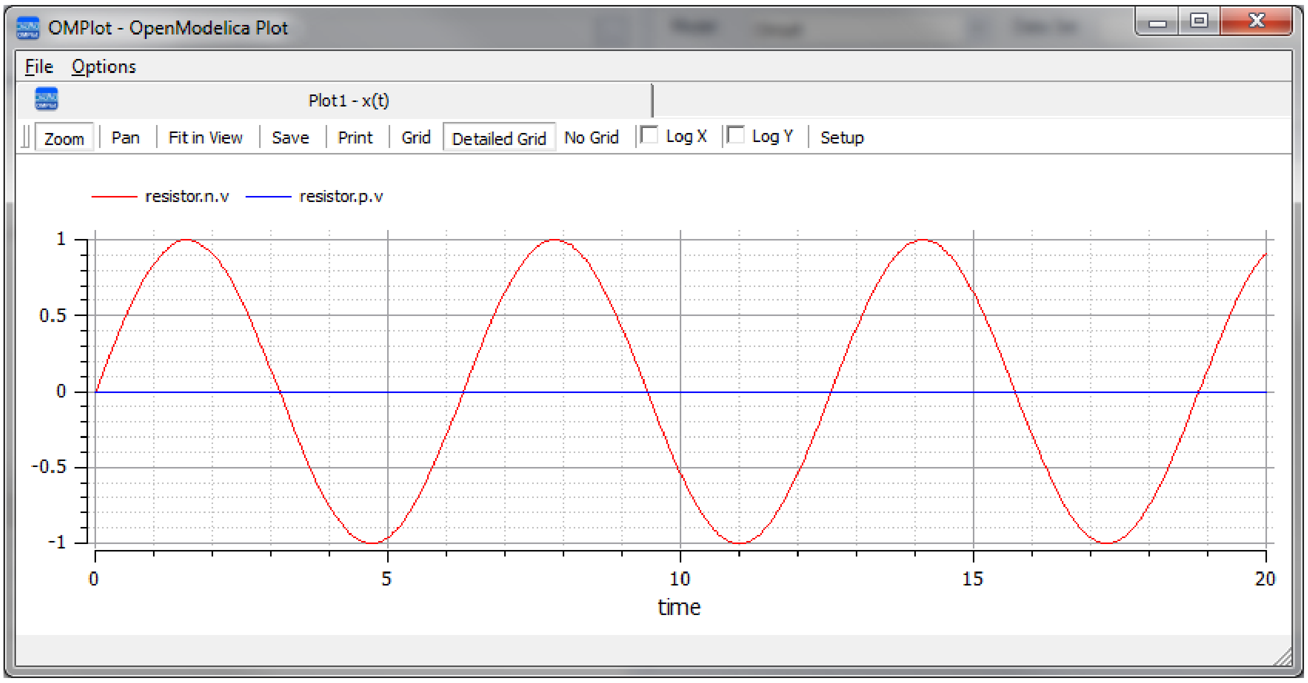
Simulation

Model: Circuit Data Set: DataSet_1 Solve

Start: 0 Stop: 20 Format: plt ☐ Parametric Plot

Dependencies (Classes to Generate)	Properties to Plot
Current	☐ ground.p.i
Resistance	☐ ground.p.v
Voltage	☐ resistor.i
ChargePort	☐ resistor.n.i
ResistorConstraint	☑ resistor.n.v
Resistor	☐ resistor.p.i
SourceConstraint	☑ resistor.p.v
Source	☐ resistor.v
GroundConstraint	☐ source.i
Ground	☐ source.n.i
Circuit	☐ source.n.v
	☐ source.p.i
	☐ source.p.v
	☐ source.v

Run Simulation



Troubleshooting OpenModelica Simulation

Common Simulation Issues

The following table describes some of the common issues that can prevent a model being simulated.

Check the output at "System Output | Build" window, the messages were dumped from OpenModelica compiler (omc.exe), it normally points you to the lines of the modelica source code. This will help you pick up most of the errors.
The number of equations are fewer than the number of variables: You might forgot to set some of the properties to be SimConstant, which means the value doesn't change during simulation. You might need to provide the SimConstant properties value before the simulation started. (Set the values through Simulation Data Set)
The blocks that are typing to ports may contain conserved properties. For example, a block ChargePort may contain two parts: "v : Voltage" and "i: Current", the property "i : Current" should be defined with SimVariable with attribute "isConserved = true".
SimConstants should be provided default values.
A SimVariable may need an initial value to start with.
The properties may typed by elements (blocks or value type) outside of the configured package; Use a package import connector to fix this.