Enterprise Architect

User Guide Series

Parametric Simulation using OpenModelica

Author: Sparx Systems
Date: 3/06/2016
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.

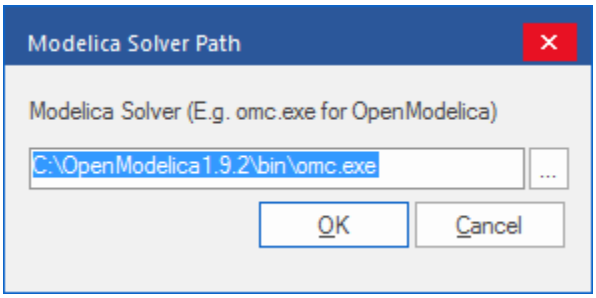
SysML Simulation Example

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. 
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	

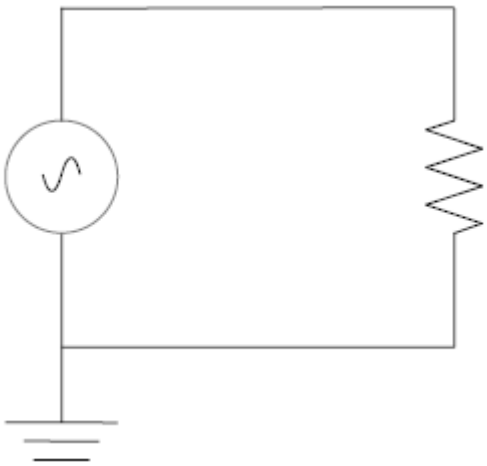
Dependencies	Shows a list of the types that need to be generated to simulate this model.
Properties to Plot	Provides a list of properties that are involved with the simulation, and allows selection of which you want to plot.

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

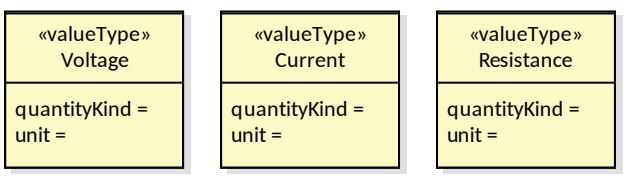
The electrical circuit we are going to model is shown below using a standard electrical circuit notation.

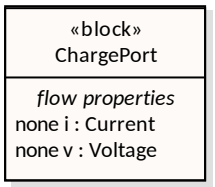


The circuit includes an AC power source, a ground and a resistor. Each of these being connected via wires.

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.

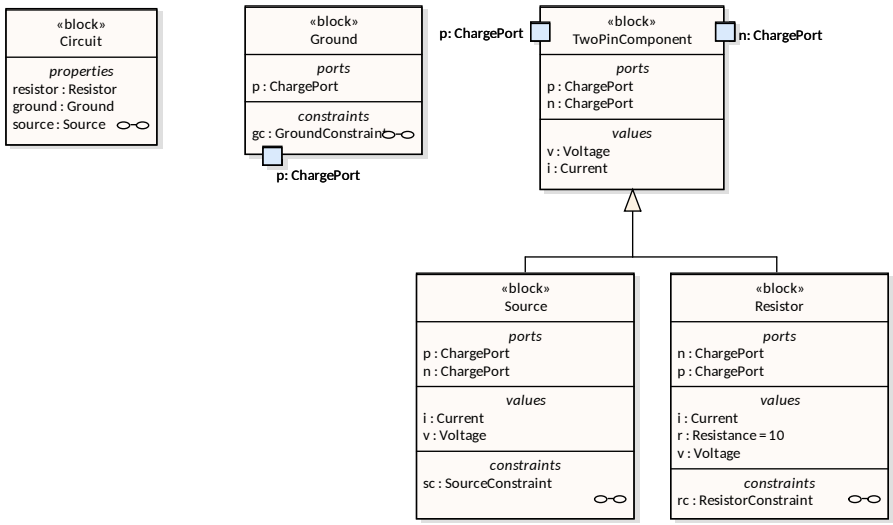
Types	Define Value Types for each of Voltage, Current and Resistance. Unit and quantity kind are not important for the purpose of simulation but would be set if defining a complete SysML model. These types will be generalized from the primitive type 'Real'. In other models, you can choose to map a value type to a corresponding simulation type separate from the model.  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.



Blocks

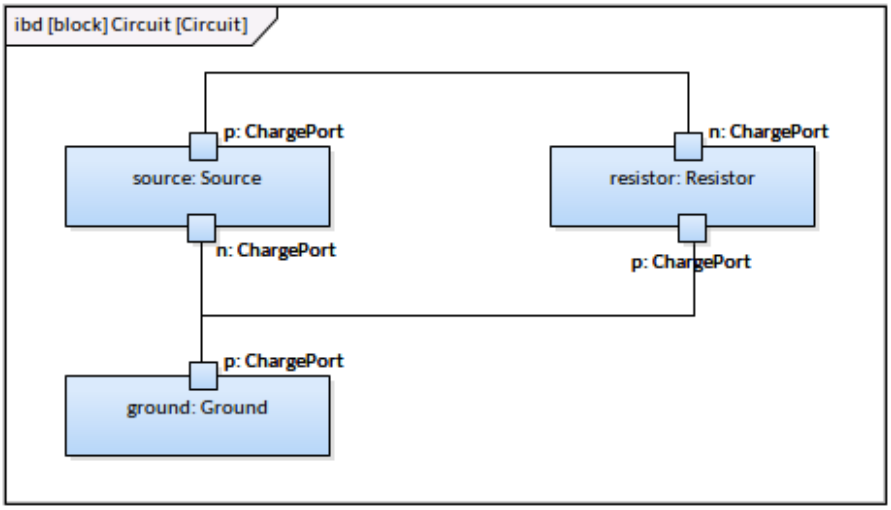
In SysML, the circuit and each of the components will be represented as Blocks. Create a Circuit block in a **Block Definition Diagram** (BDD). The circuit has three parts: a source, a ground, and a resistor. These parts are of different types, with different behaviors. Create a block for each of these 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.



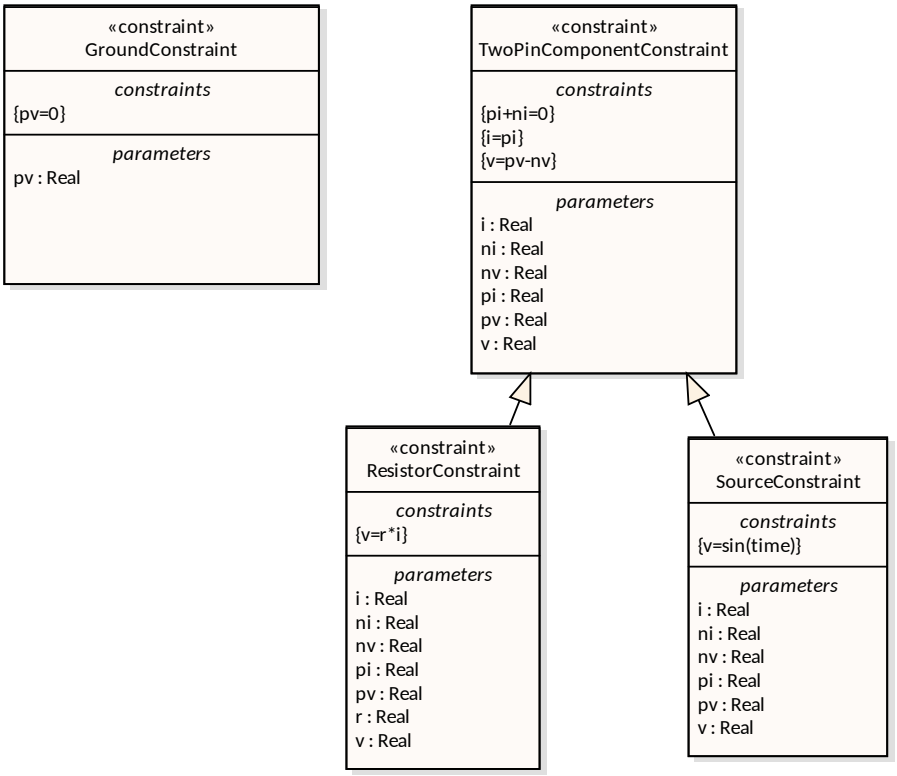
Internal Structure

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.



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.

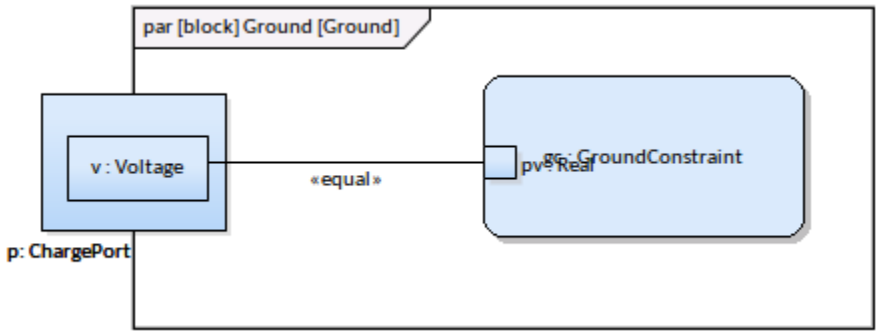
Constraints	Equations define mathematical relationships between numeric properties. In SysML, equations are represented as constraints in constraint blocks. Parameters of constraint blocks correspond to SimVariables and SimConstant of blocks (i, v, r in this example), as well as to SimVariables present 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.



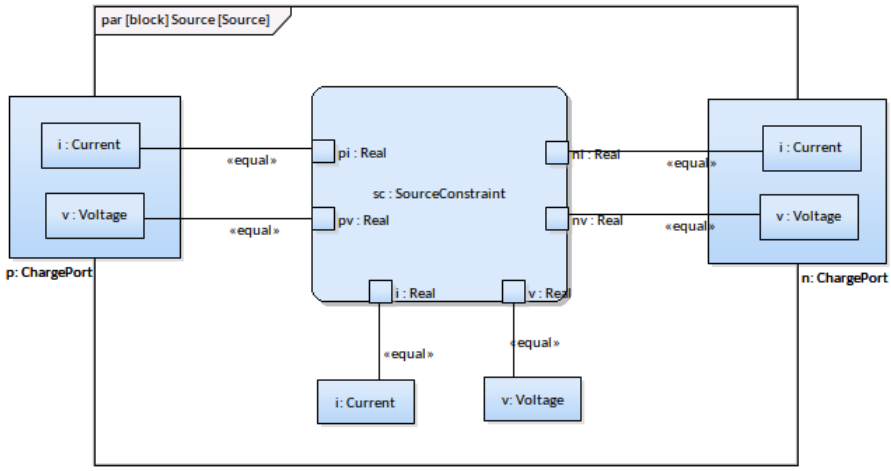
Bindings

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.

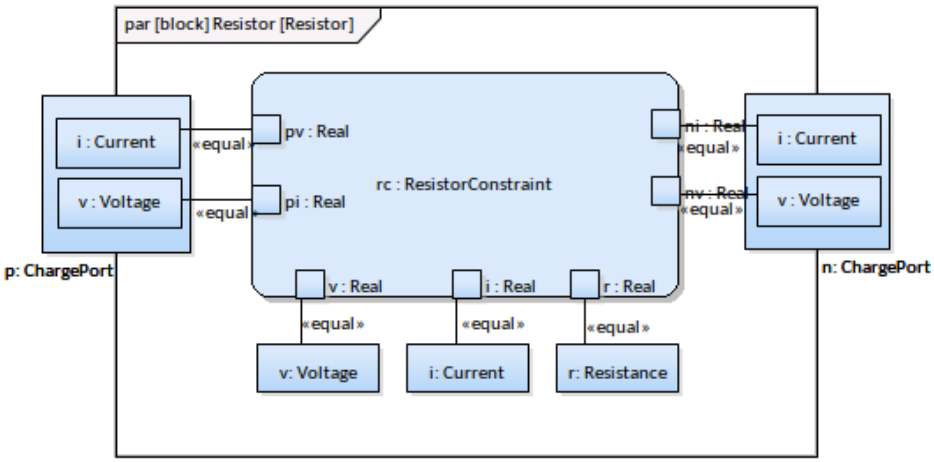
For the ground constraint, bind gc.pv to p.cflowsim.v.



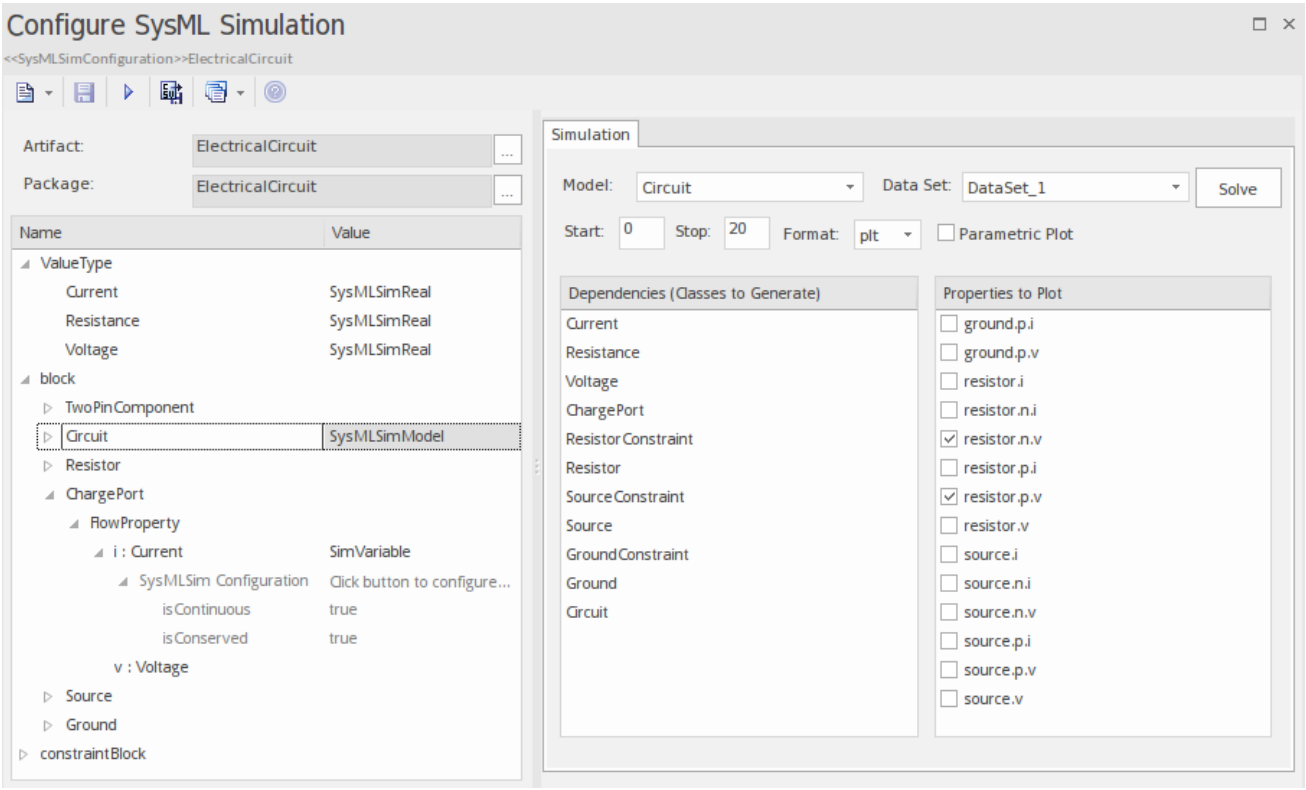
For the source constraint, bind sc.pi to p.cflowsim.i, sc.pv to p.cflowsim.v, sc.v to v, sc.i to i, sc.ni to n.cflowsim.i, and sc.nv to n.cflowsim.v .



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



Configure Simulation Behavior



Run Simulation