Parametric Constraints Evaluator Tutorial

Rhapsody Version: 7.5.2
Last Updated: June 13, 2010

SysML Parametric Diagrams provide a way to integrate engineering analysis models described in mathematical equations and constraints, with design models describing the structural and behavioral aspects of systems.
The Parametric Constraint Evaluator (PCE) is a Rhapsody Add-on that allows the evaluation of parametric diagrams using a Computer Algebra System (CAS) such as MAXIMA or MATLAB Symbolic Math Toolbox (MSMT) to solve the mathematical expressions. PCE translates the SysML diagrams to the CAS specific language; then solves the expressions by calling the CAS. The results of this analysis can be fed back to the design specification by instructing PCE to update the model.
PCE is mainly intended to be used for doing engineering analysis directly from the SysML models including examining various design alternatives by comparison (Trade Studies)

Tutorial Pre-requisites

The Parametric Constraint Evaluator (PCE) Add-on is installed as part of the Systems Engineering Add-ons. It is specific to SysML models.

In addition it is required to install a Computer Algebra System (CAS): either MATLAB Symbolic Math Toolbox or MAXIMA. The order of installation is not important and one can install both CAS tools and switch between them via the *PCE Environment Settings* dialog (opened via the *Tools* menu in Rhapsody)

Lesson 1: Solving algebraic equations under constraints and doing linear optimization

Consider the following example problem:
A computer shop builds desktops and laptops. The profit for a laptop is $600 and the profit for a desktop is $400. The shop can make a total of 16 computers a day (laptops and desktops together). The shop can sell a total of up to 28 desktops and up to 14 laptops a day. The shop can ship a total of up to 25 desktops and up to 16 laptops per day.

We will answer the following questions:

1. What will be their daily profit if they produce 5 laptops and 6 desktops?
2. What if they produce 10 laptops and 10 desktops?
3. How many laptops and desktops should they produce to maximize the daily profit?
4. What happens if we change the price of the laptops and desktops (increase the profit)?

Step 1: Modeling the example problem

The complete model is available as a sample (the Rhapsody 7.5.2 installation it is under [Rhapsody][1]\Samples\SystemSamples\PCESamples\ComputerShop). It is possible to review the model instead of building it as described in this lesson.

1. Open Rhapsody and create a new SysML model, called "ComputerShopExample". Rename the Default package to ComputerShopPkg
2. Specify a block representing the computer shop with the attributes as shown in Figure 1. The values listed in the figure are the initial values of the ComputerShop attributes and will be used as input values for the parametric analysis.

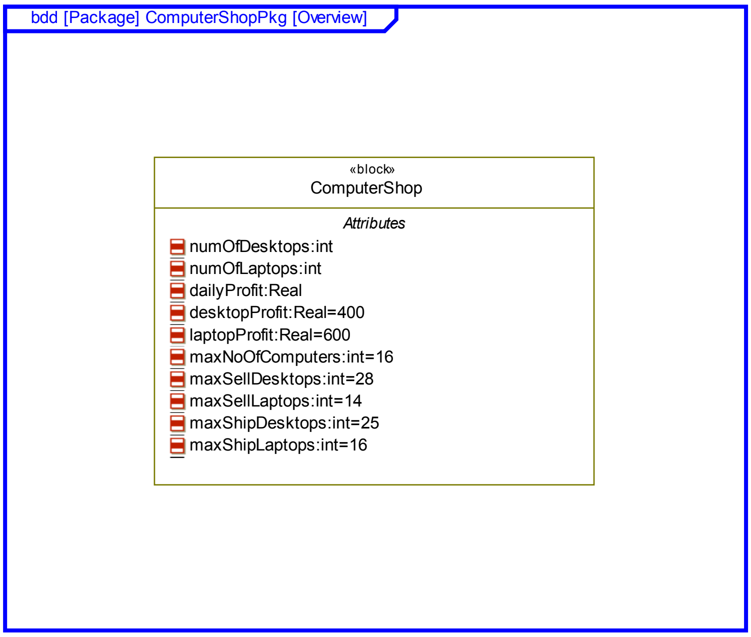

Figure 1: ComputerShop block and its attributes

3. Create a Parametric Diagram under ComputerShop and name it "Daily Profit".
4. Specify the parametric diagram as shown in Figure 2. DailyProfitCB is a ConstraintBlock that specifies all the equations and constraints. In this case we have one equation for the total daily profit and three inequalities showing the manufacturing, selling and shipping constraints. The language used for the constraint expressions is a subset of the Modelica language. Formally, in SysML, in order to use a Constraint Block one needs to specify a Constraint Property (a usage of a Constraint Block) and connect the Constraint Parameters with binding connectors (the lines shown on the diagram) to the relevant design attributes.

---

[1] [Rhapsody] = Rhapsody installation directory

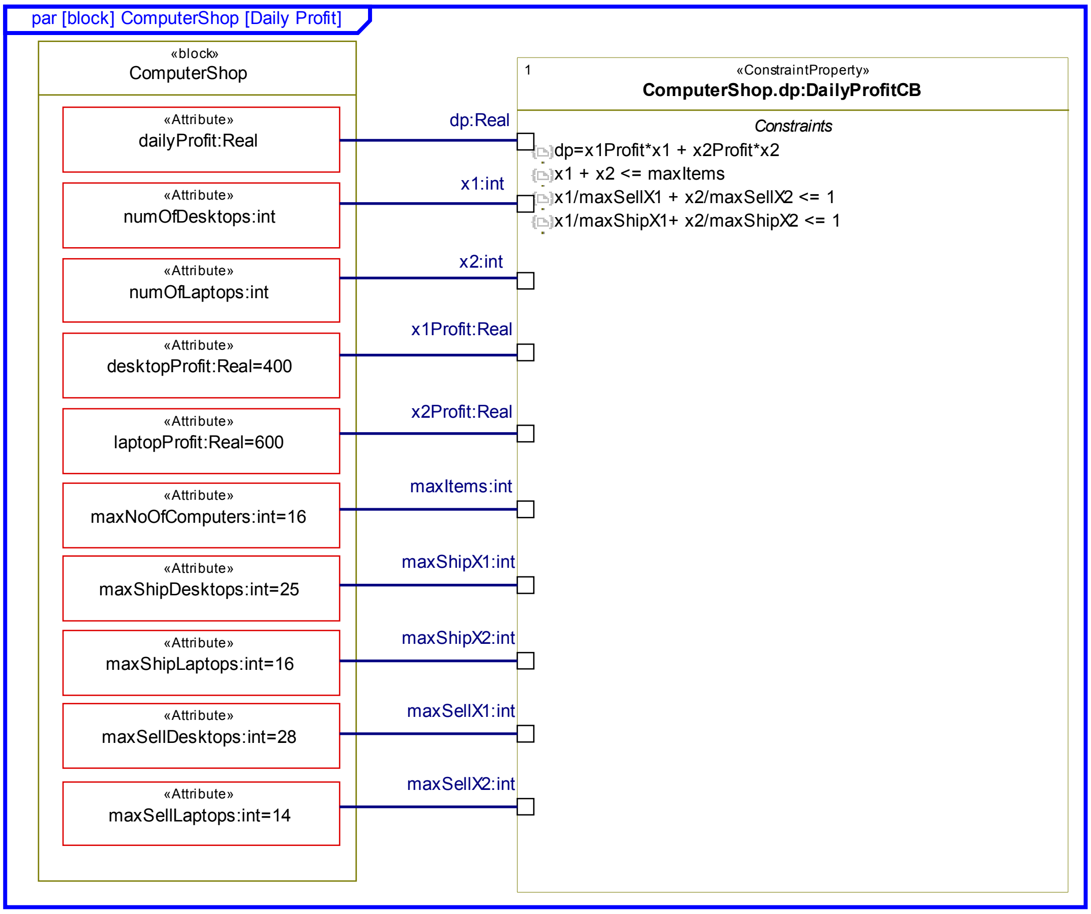

Figure 2: Parametric Diagram for Computer Shop Example

Step 2: Adding the PCE Profile and defining the Constraint View

In order to use PCE we need to add the PCE profile to the model. To do that, use the *Add Profile to Model...* menu item under the *File* menu. The PCE profile is located under the Profiles folder in a sub folder named PCE.

To perform the analysis we need to add a constraint view that references the ComputerShop parametric diagram:

1. Add the PCE Profile to the model
2. Create a Constraint View under ComputerShopPkg package and name it ComputerShopPCEView. A constraint view is the model element through which PCE performs the analysis as we demonstrate below.
3. Right click the constraint view (ComputerShopPCEView) and add a Reference to the Daily Profit PD parametric diagram. In general, a constraint view may reference several parametric diagrams.

Step 3: Doing the Analysis

In this step we will use the Constraint View created in the previous step to answer the questions listed in the beginning of the lesson

1. Right click on the ComputerShopPCEView, select *Open ConstraintsView…*
    a. If this is the first time you are running PCE, the dialog of the PCE environment settings will be displayed (Also available via the *Tools* menu of the main menu bar). This dialog allows one to switch and configure the CAS.

        Notes:

        1. For MATLAB Math Symbolic Toolbox (MSMT) make sure to point to MATLAB.exe (as opposed to matlab.exe which spawns MATLAB.exe). In MATLAB 2010a this is under a folder named win32.
        2. If you use MSMT you need to carry out some actions on the MATLAB side – make sure to follow the instruction after you OK the dialog
        3. If you use MAXIMA but also have a MATLAB installed (even without MSMT), configure the path to MATLAB.exe as well. This will allow you to write causal expressions in the M language using Constraint Blocks stereotyped *«ExpressionForMatlab»*.
2. "What will be their daily profit if they produce 5 laptops and 6 desktops?"
    a. In the value column fill in the number of laptops and desktops and click Evaluate.
    b. The dailyProfit value is displayed in the constraint view
3. "What if they produce 10 laptops and 10 desktops?"
    a. Fill in the value 10 for numOfDesktops and numOfLaptops and do Evaluate
    b. The violated constraints appear in red. Double clicking a constraint selects the constraint in the Rhapsody browser.
4. “How many laptops and desktops should they produce to maximize the daily profit”?
    a. Delete the values of numOfDesktops and numOfLaptops
    b. Set the Command cell for Daily Profit to Maximize and click *Evaluate*
    c. Answer: a max daily profit of $8800 will be achieved if 4 desktops and 12 laptops are produces daily
5. “What happens if we change the price of the laptops and desktops (increase the profit)?”
    a. In the *Value* column set the laptopProfit to 800 click *Evaluate* (while the command cell for daily profit is still set to Maximize)
    b. Now the maximum daily profit is $11200 for producing 14 laptops and zero desktops

Lesson 2: Time Dependent Ordinary Differential Equations

In this example we describe a point mass connected to a spring and solve a set of Ordinary Differential Equations (ODEs) to describe the position, velocity and acceleration of the mass (one dimensional movement)

The equation of motion is:

$$m\frac{d^2x}{dt^2} = -kx$$

Where m is the mass of the point, k is the spring constant, x is the relative position of the point mass and t denotes time

To solve the equations for a specific set of initial conditions we will use the following set of ODEs:

$$ma = -kx$$

$$v = \frac{dx}{dt}$$

$$a = \frac{dv}{dt}$$

$$x(t = 0) = 0$$

$$v(t = 0) = 5$$

Where v is the velocity, a is the acceleration.

For now we assume arbitrary units.

Step 1: Review the model

1. Open the HarmonicOscillation sample model (located under [Rhapsody]\Samples\SystemSamples\PCESamples\SpringAndMass)
2. Navigate to HarmonicOscillation package and open the Mass and Spring BDD and review their attributes
3. Open the Harmonic Oscillation parametric diagram
    a. Observe that in this case we defined only a Constraint Property, it's Constraint Block type is implicit
4. Review the equations and initial conditions:
    a. der means derivative over time
    b. observe how the initial conditions are set (x(0)=0 and v(0)=5)

Step 2: Specify the Constraints View and perform the analysis

1. Open the HarmonicOsc constraint view (it references the Harmonic Oscillation parametric diagram)
2. In the constraints view set the mass to 5 and the spring constant to 2
3. Click *Plot…*

4. In the dialog enter a name for the graph (e.g. m=5 k=2), press OK
5. Click on the graph name and press *Next*
6. Set the primary variable to time, set it's min to 0 and it's max to 20
7. Click *Next*
8. Check the acceleration, displacement and velocity. When selecting them from the list you can provide a more readable name using the *Label* field
9. Click *Finish* – you should get a plot similar to Figure 3
10. Change the constants and observe the behavior by creating additional plots
11. Exercise: Add a damping factor to the spring and observe the behavior. The Equation is

$$m\frac{d^2x}{dt^2} = -kx - \mu\frac{dx}{dt}$$

(Where μ represents the damping coefficient)

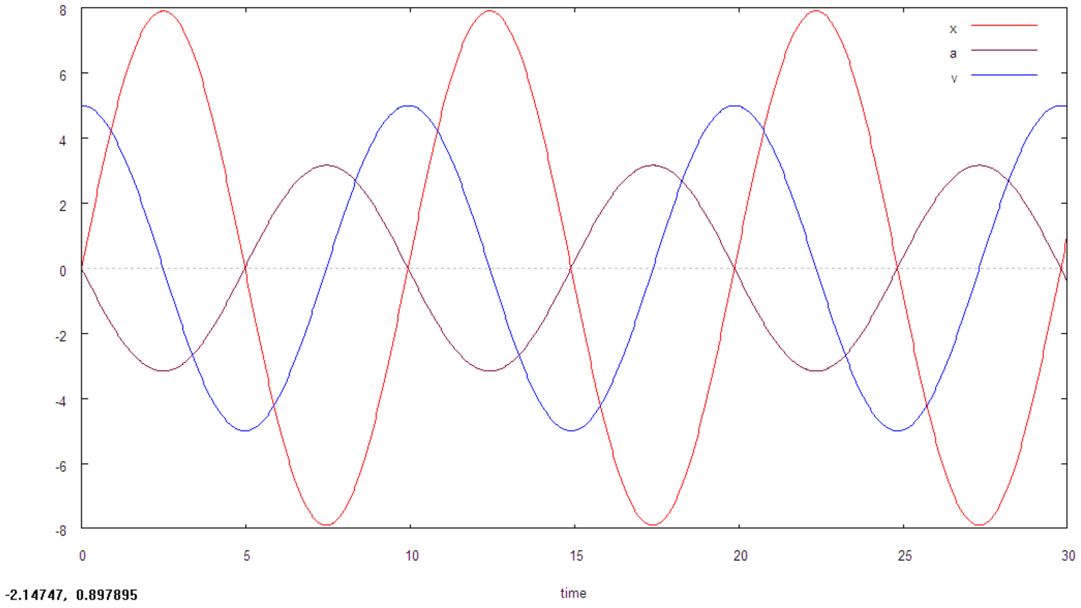

Figure 3: Plot of Harmonic Oscillation

Lesson 3: Analyzing two parametric diagrams and producing 3D plots

In this example we will analyze two parametric diagrams describing the fuel pressure and fuel demand of fuel injectors systems of a hybrid SUV. This example is based in part on the example provided in Annex B of the SysML 1.1 specification

The equations are:

$$\sin(\pi x^2)^2 + \cos(\pi y^2)^2 = d$$

$$r = p/(4*d)$$

Where:
d is the fuel injector demand
r is the fuel flow rate
p is the fuel pressure
x and y are parameters that we can use to control the pressure, demand and rate

Step 1: Review the model

1. Open the sample model PowerSubsystem ([Rhapsody]\Samples\SystemSamples\PCESamples\PowerSubsystem)
2. Navigate to the PowerSubsystemPkg package
3. Open the diagram “BDD for Power Subsystem” and review the equations and the specification of the PowerSubsystem block. Note that the Constraint Properties are owned by the PowerSubsystem block in this case
4. Review the two parametric diagrams: “Fuel Demand” and “Fuel Flow”

Step 2: Perform Analysis

1. Open the ConstraintsView PowerSunSystemCV
2. Observe that the ConstraintsView relates to both parametric diagrams, this was done by adding a reference to each of the diagrams (Using the right click popup menu on the ConstraintsView)
3. In the ConstraintsView click *Plot…*
4. For each of the two plots click Next and review the settings, until getting the plots. Observe that for the 3D plot you can rotate the graph by clicking on it and moving the mouse.

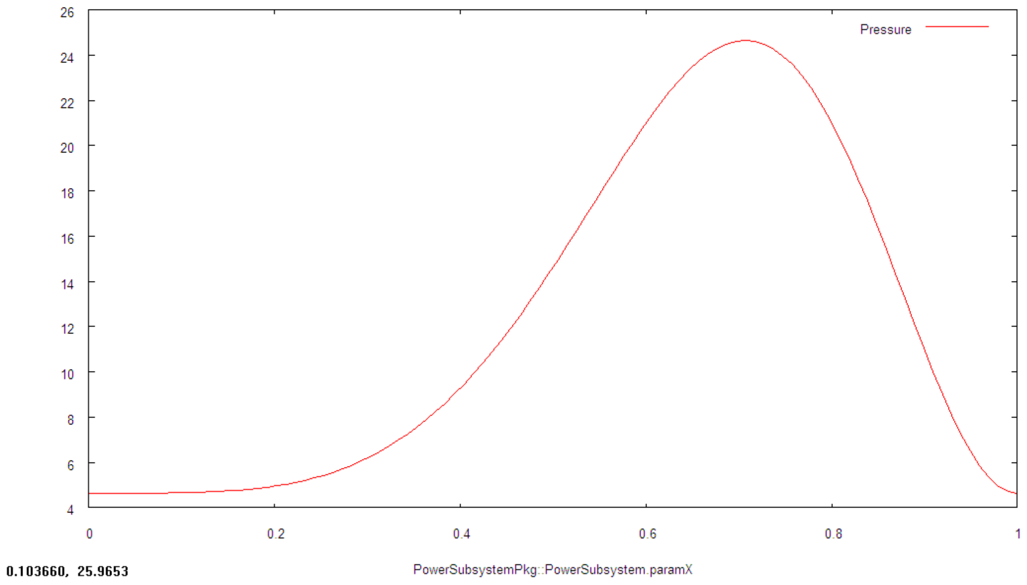

Figure 4: Fuel Pressure as a function of the x (for a constant y)

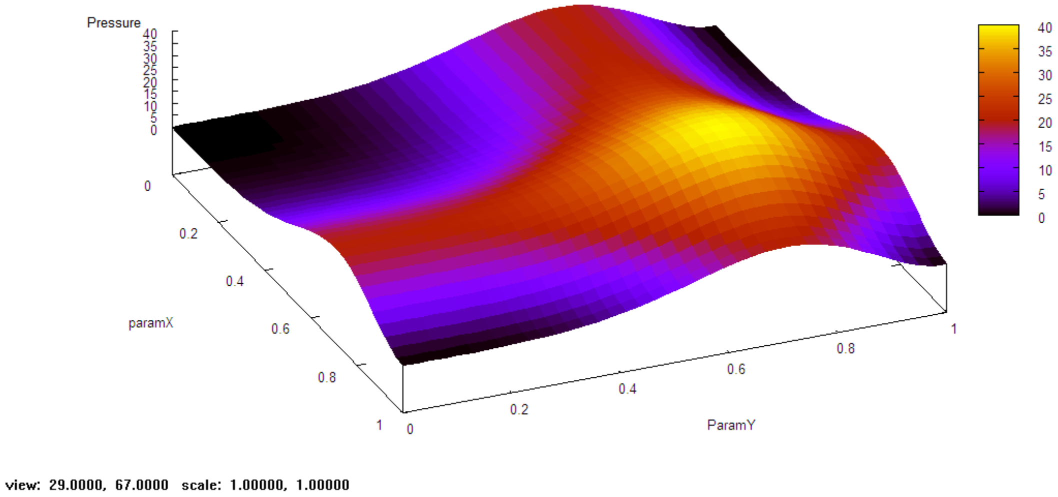

Figure 5: Fuel pressure as a function of x and y parameters

Lesson 4 (Case study): comparing design alternatives (classical trade study)

The model of this test case is attached to this tutorial: TestCaseModel

Problem Statement

We would like to design an optical X-Y Scanner to detect flaws on a 1 meter by 1 meter surface. A camera mounted on an XY translation table (Figure 6) scans a surface and transmits the images to a computer for image processing (to detect flaws).
The measures of effectiveness are cost, scan time (how long it would take to scan the surface) and the accuracy (how small a defect can the scanner detect). The scanner should detect defects of size 1 micron (0.001 mm) and smaller. The cost of the Motor and Camera should not exceed $700 and the scanning time should not exceed 60 minutes. We have to choose between three camera types and between three motor types for the X translation.

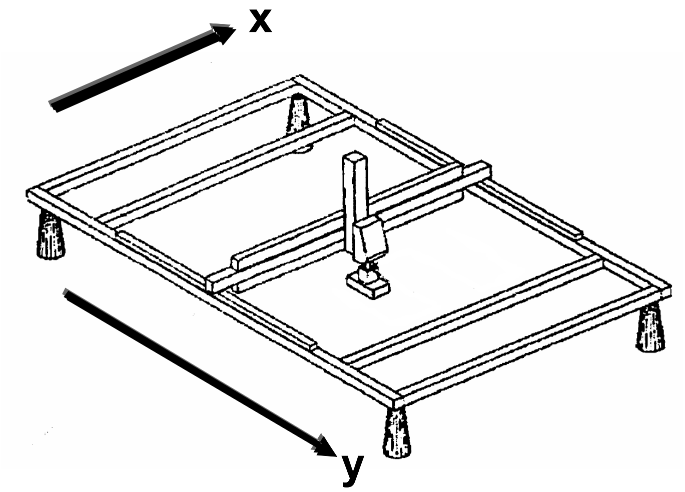

Figure 6:An illustration of the X-Y Scanner

Solution Steps

1. Creating a parametric diagram to compare design alternatives by weighted measures of effectiveness
2. Calculating spatial resolution/accuracy for each camera
3. Calculating scanning time for each combination of camera and x-motor
4. Calculating the cost of each combination of camera and x-motor
5. Concluding the trade study: identifying the best alternative

Creating a parametric diagram to compare design alternatives by weighted measures of effectiveness

Figure 7 shows the parts that make up the scanner. The white-diamond relations show that the Camera is assembled on the CameraMount, which is assembled on the XArm, which is assembled on the XArmMount, which is assembled on the YArm.
In the test case model this diagram is under the TestScan1800 package (ScannerParts).
Tip: The name of the package and the diagrams appear in the diagram frame.

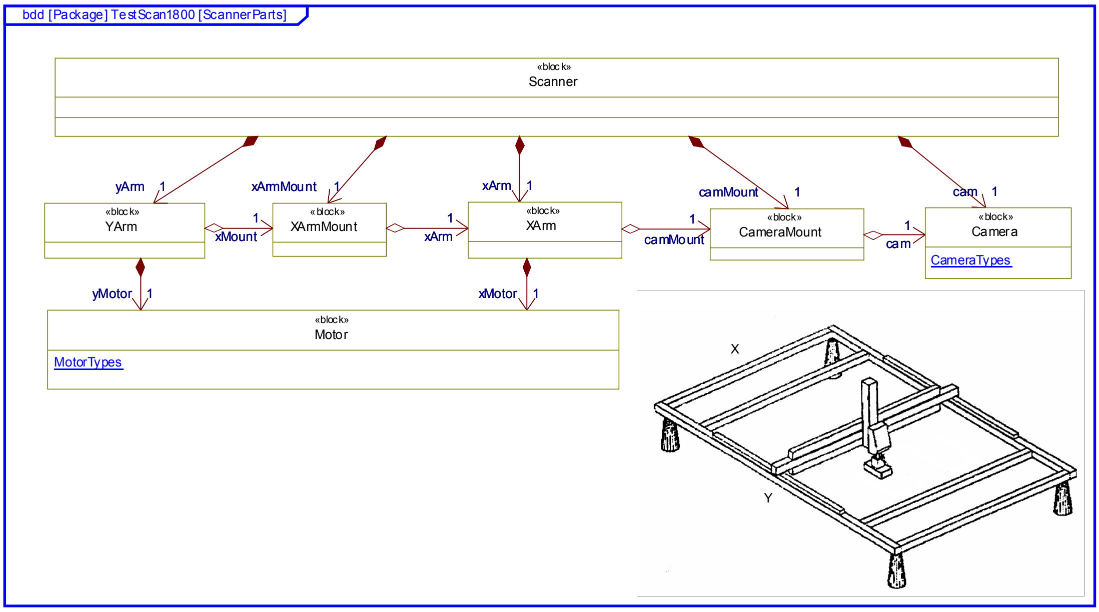

Figure 7: Bill of materials for the XY scanner

Figure 8 shows a parametric diagram to compute the normalized measure of effectiveness (MOE) and the "score" for every design alternative. As stated previously, the cost should not exceed $700, the scanning time should not exceed 60 minutes and the accuracy should be at least 1 micron (0.001 mm). The normalized values are between 0 and 10. The utility graphs for each MOE are shown in Figure 9. These plots can be produced using the UtilityView constraint view in the ParametricAnalysis package (sub package of TestScan1800 package)
In the ObjectiveFunction constraint property (typed by an implicit constraint block) we see that the accuracy MOE has a weight of 0.2 while the cost and scanning time MOEs have a weight of 0.4. The overall score of each of the alternatives is a number between 0 and 10 (higher number means better design alternative)

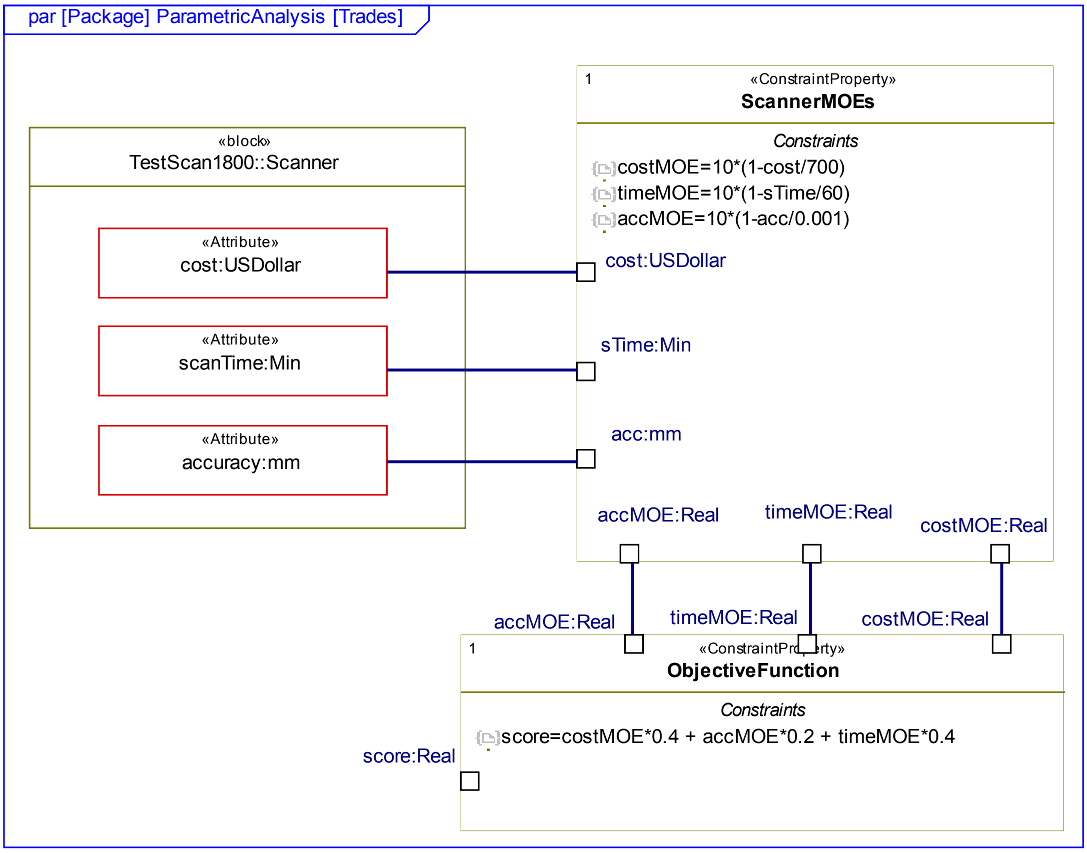

Figure 8: Trade study equations for the Scanner

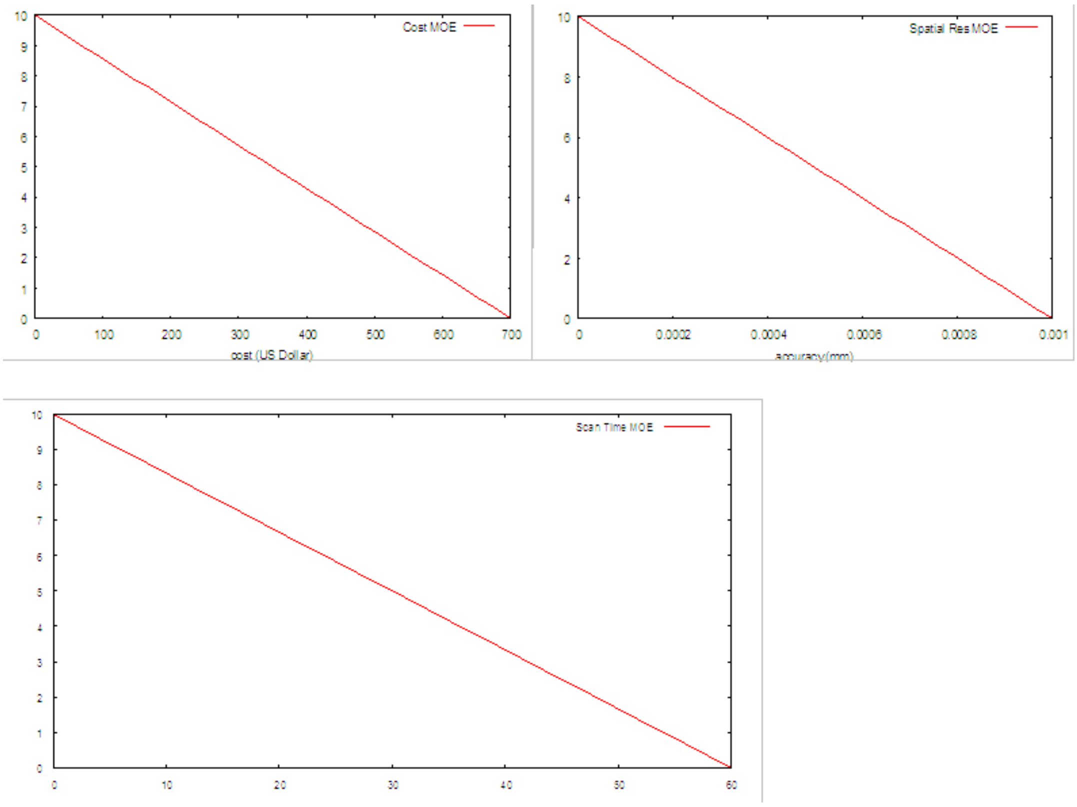

Figure 9: Utility Graphs of the normalized MOEs

Now we need to calculate the accuracy, scan time and cost that we will obtain using the cameras and motors we have as design options.

Calculating spatial resolution/accuracy for each camera

Let's start with the accuracy: the three types of cameras are modeled in Figure 10. All cameras have the same attributes; however the attributes have different values for every camera type. For each camera type we know the cost, mass, shotTime, imageRadius and the resolution. We need to calculate the spatial resolution of each camera type to see if it meets the requirement that the spatial resolution must be less than 1 micron. We assume that all camera types are mounted in a way that the lens is at the same distance from the surface and that the surface is in focus. We neglect other factors such as contrast and brightness.

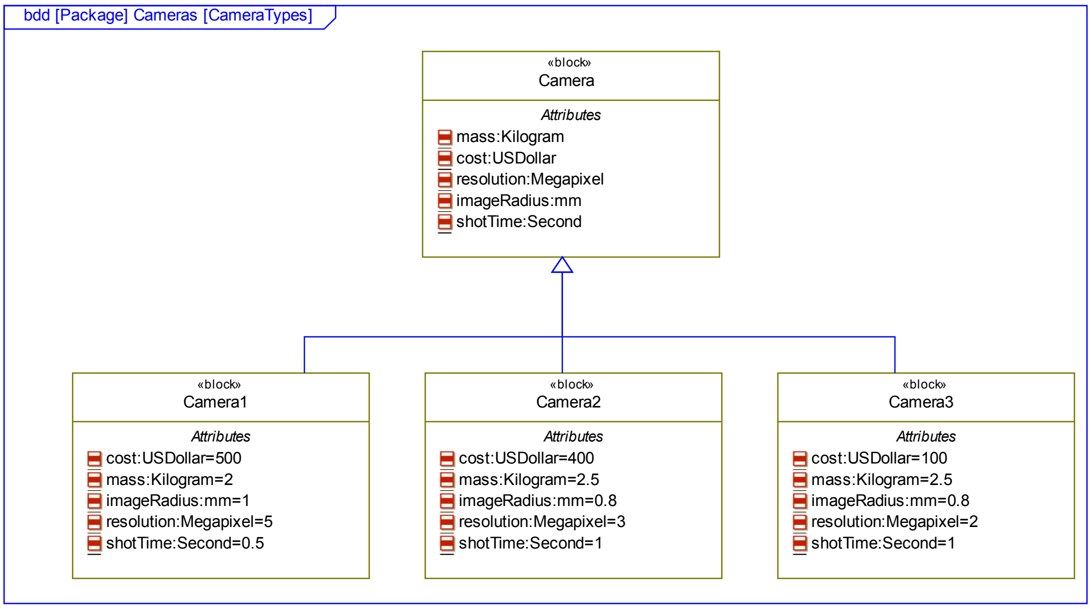

Figure 10: Three types of available cameras

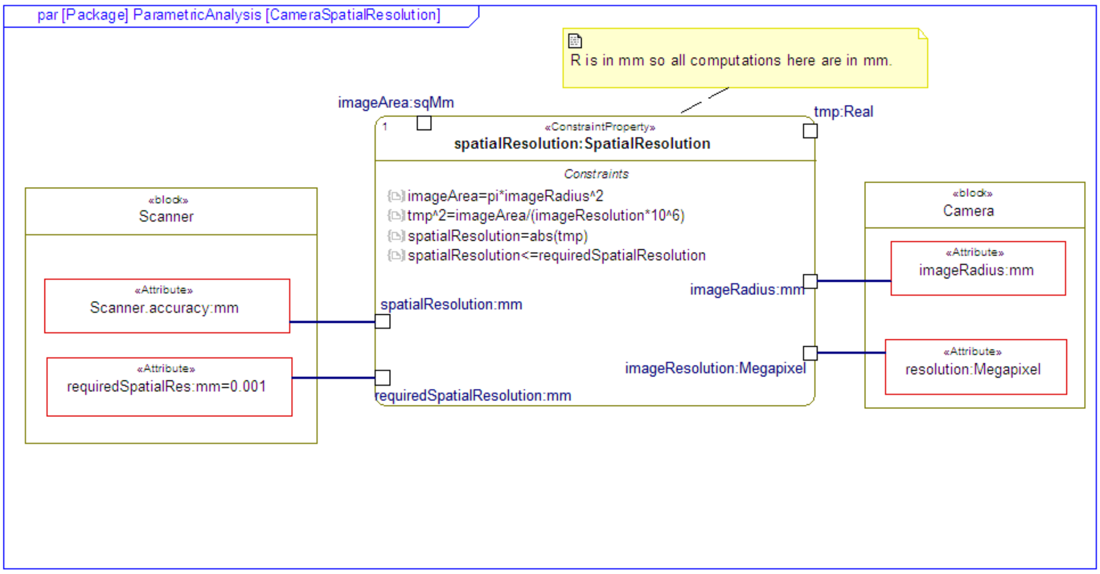
par [Package] ParametricAnalysis [CameraSpatialResolution]
R is in mm so all computations here are in mm.
imageArea:sqMm
tmp:Real
1
«ConstraintProperty»
spatialResolution:SpatialResolution
Constraints
imageArea=pi*imageRadius^2
tmp^2=imageArea/(imageResolution*10^6)
spatialResolution=abs(tmp)
spatialResolution<=requiredSpatialResolution
imageRadius:mm
spatialResolution:mm
imageResolution:Megapixel
requiredSpatialResolution:mm
«block»
Scanner
«Attribute»
Scanner.accuracy:mm
«Attribute»
requiredSpatialRes:mm=0.001
«block»
Camera
«Attribute»
imageRadius:mm
«Attribute»
resolution:Megapixel

Figure 11: Parametric diagram for calculating the spatial resolution of the camera and enforcing the spatial resolution constraint

Figure 11 shows how to calculate the spatial resolution using a parametric diagram with the attributes of the generic Camera block. We specify a parametric diagram as shown in Figure 12 for each of the camera types (the figure is referencing Camera1) in order to assign the specific values of the concrete cameras to the constraints specified in Figure 11. These diagrams are owned by the Camera blocks in the test case model. By creating constraint views for each camera type that reference the two parametric diagrams (e.g. Figure 11 and Figure 12) we can perform an evaluation for each of the cameras. See Camera1Check, Camera2Check and Camera3Check in the model. Evaluating each of these constraint views we see that Camera3 violates the resolution constraint while Camera1 and Camera2 satisfy the requirement.

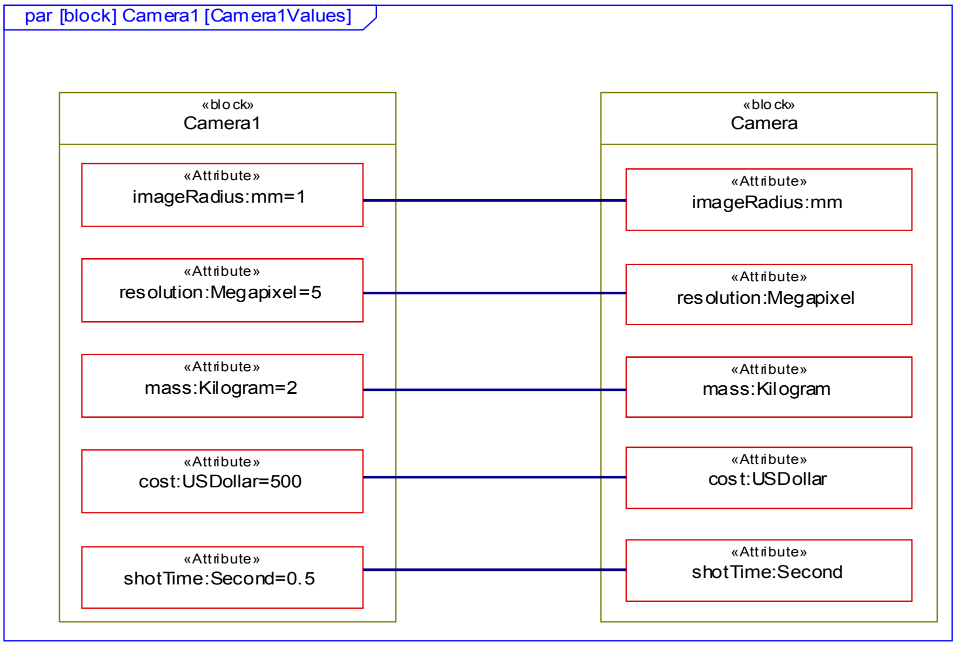

Figure 12: Parametric Diagram relating the general constraints with Camera1 values

Calculating scanning time for each combination of camera and x-motor

The scanning speed may depend both on the motor's angular speed (rounds per minute) and the time it takes for a camera to shoot the image along with the image size (the smaller the image, we need to scan more lines).

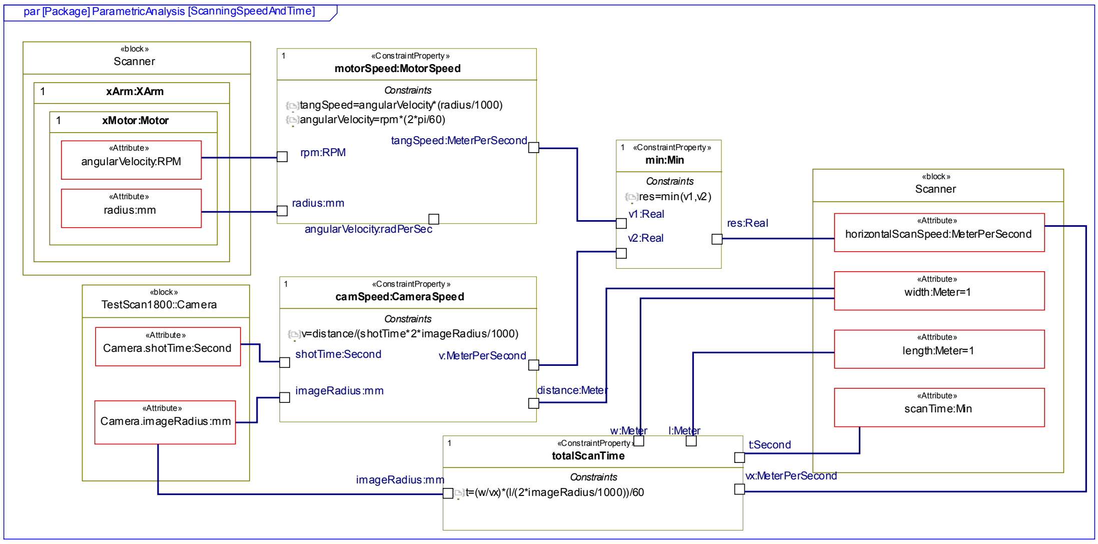

Figure 13: Parametric Diagram for solving the scanning speed and total scan time

Figure 13 shows a set of equations for calculating the total scan time. First the MotorSpeed constraint block is used for calculating the tangential speed produced by the motor. The other constraint in MotorSpeed converts the RPM angular velocity to Radians per second. The CameraSpeed constraint block calculates how the speed is limited by the camera by taking into account the shotTime (the time the camera requires to take a picture) and the image area. The lower of the two results (the motor speed and the camera speed) is the horizontal scan speed.

Note: We can use the SysML dot notation to reference the attributes of Motor in the context of the XArm of the scanner. Simply drag the attribute of Motor, right-click on it and do "bind to context..."

Once the speed is known we take into account the dimensions of the surface (1 meter by 1 meter) and the image size to calculate the total scan time (totalScanTime constraint property). Note that for this we use a constraint property with an implicit constraint block as its type (because it is a single usage)

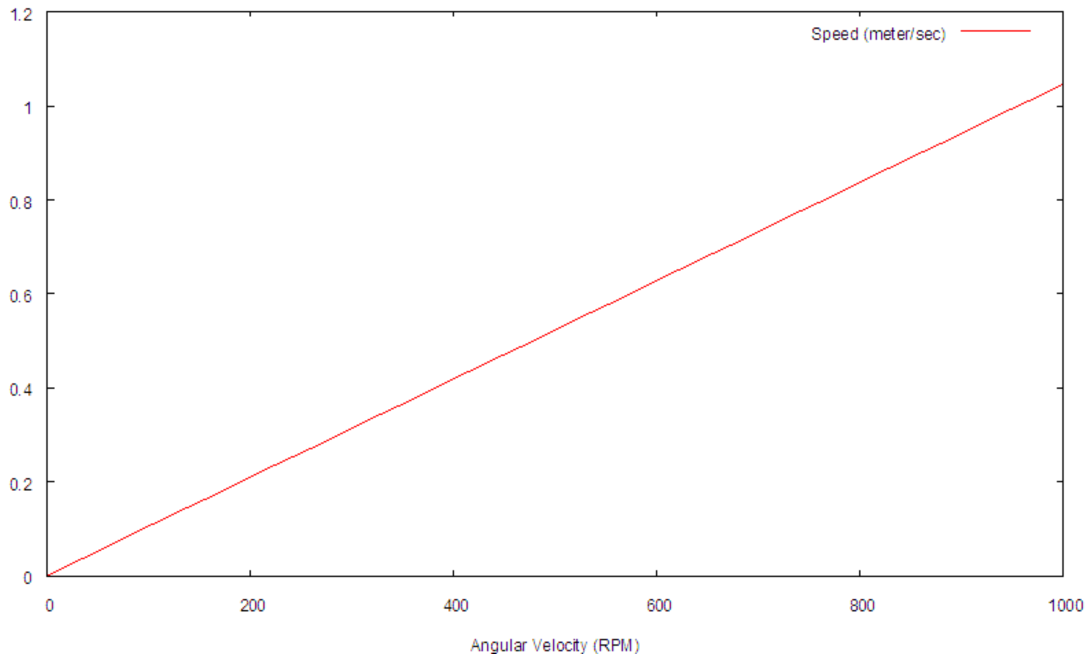

Figure 14: Horizontal Speed as a function of Motor Angular Velocity

We can also use the Plot functionality to see if the Camera Speed would actually limit the horizontal speed provided by the X Motor. We can plot the tangSpeed vs. angular velocity in RPM and in addition the speed as limited by the camera vs. the shotTime. Looking at Figure 14 and Figure 15 (where the base values are of Motor1 and Camera1) we see that the motor can bring us into speeds around 1 meter/sec while the shooting time limits us to 100 meter/sec so we have a difference of two orders of magnitudes. Hence, the camera speed is not actually limiting the horizontal speed of the scan.

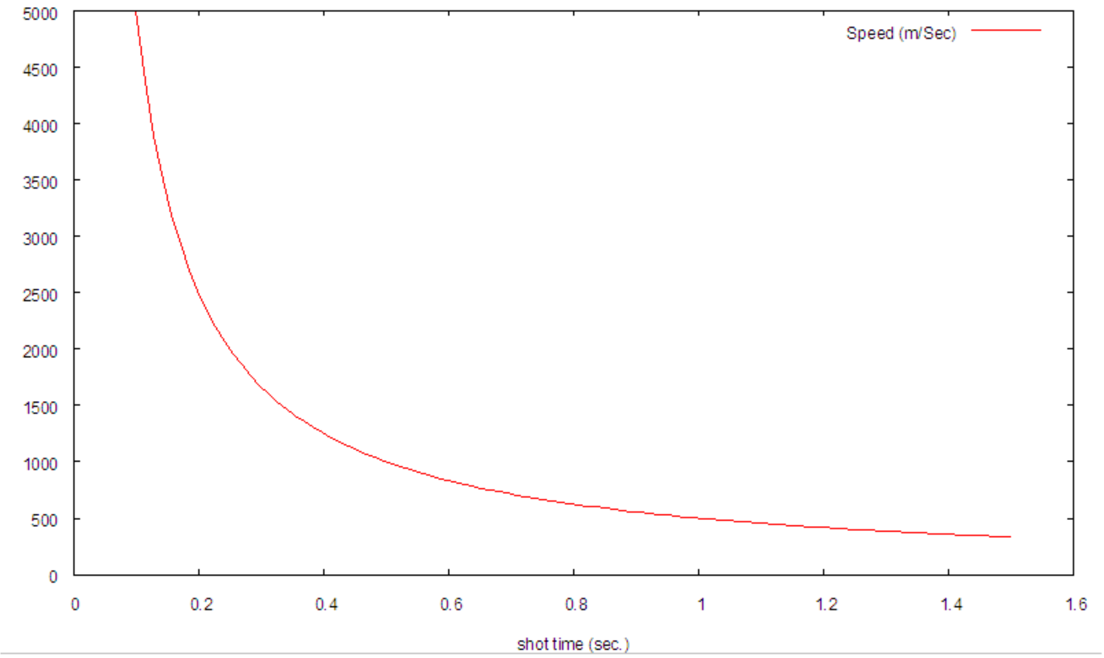

Figure 15: Horizonal Scan Speed as a function of shooting time

To use the values of the different motors we apply the same technique as we did for the cameras. However since we have two parts typed by Motor we bind the values to the xMotor part and not the Motor block to indicate that we are relating specifically to the motor mounted on the x arm of the scanner. This is shown in Figure 16.

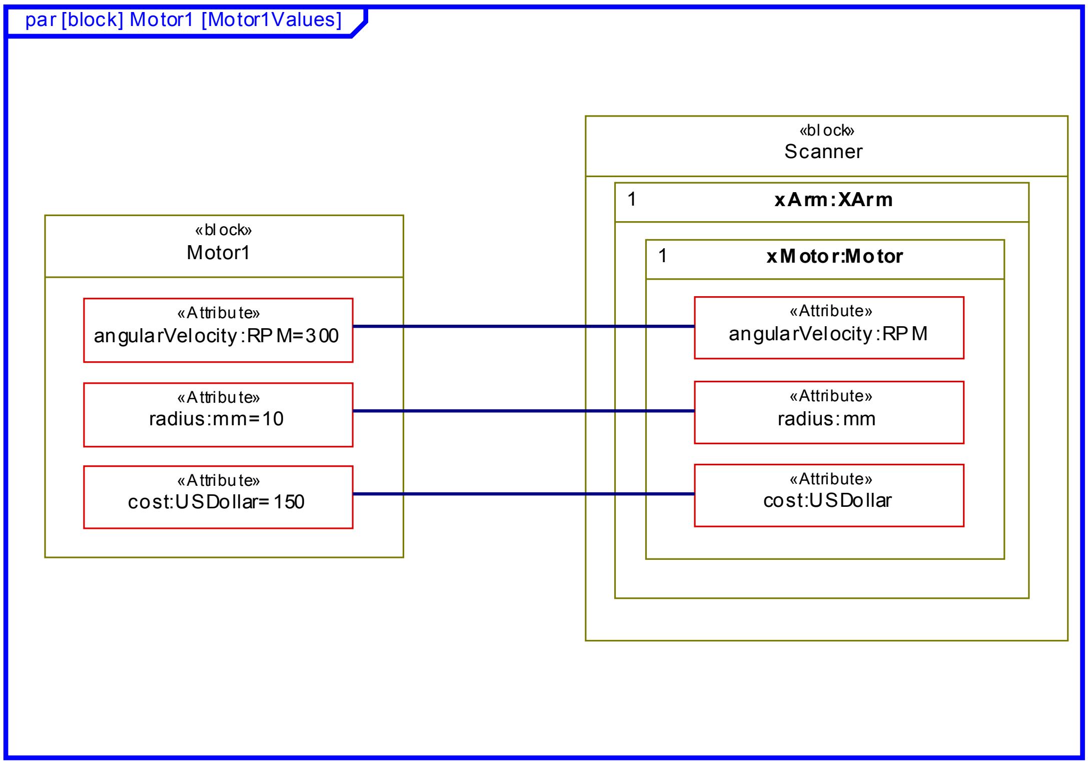

Figure 16: Parametric diagram to assign Motor1 values for the xMotor part of the Scanner

In the Test Model, under the ParametricAnalysis package, the constraint view ScanningTimeCam1Motor1 references three parametric diagrams and can be used to calculate the scanning time for a design alternative that uses Camera1 and Motor1 as the Camera and X-Motor of the Scanner.

Calculating the cost of each combination of camera and x-motor

To calculate the total cost of a given camera/motor combination we use the parametric diagram shown in Figure 17. We can include the same diagrams for camera values and motor values (the parametric diagrams owned by the camera and motor blocks) to get the cost of each design alternative

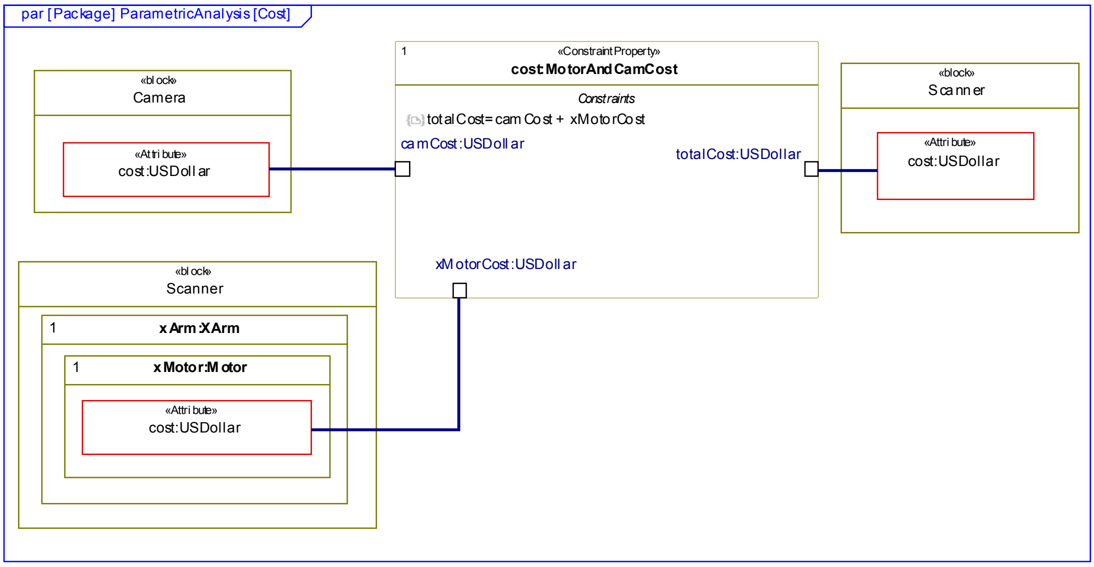

Figure 17: Cost of camera and motor

Concluding the trade study: identifying the best alternative

Now that we can calculate the cost, scanning time and accuracy we can go back to the parametric diagram for the trade study we previously specified (Figure 8).

To calculate the MOEs and the score for each design alternative we specify constraint views that reference the generic parametric diagrams (Figure 11, Figure 13, Figure 17) with the value parametric diagrams owned by the specialized Camera and Motor blocks. For example, in the test case model, Cam1Motor1 constraint view references Camera1Values and Motor1Values parametric diagrams as well the generic diagrams. All and all, we have six constraint views representing six alternatives.

Using each of the constraint views we can evaluate the cost, speed and accuracy and the weighted score of the trade study. After each evaluation we export the results to a .csv (comma separated file) file by doing *Export Data…*
Once we have all .csv files we can open all of them using an electronic spreadsheet and reference the results in a summary sheet.

The summary sheet is shown in Table 1. Looking at the table (in the form of a matrix) we see that Camera 2 with Motor 2 is the best design alternative according to our weight factors.

Score	**Motor1**	**Motor2**	**Motor3**
Cam1	2.931999245	3.701905592	3.587619877
Cam2	3.009339973	***4.000294334***	3.88600862
Cost (US Dollar)	**Motor1**	**Motor2**	**Motor3**
Cam1	650	670	690
Cam2	**550**	570	590
Scan Speed (Min.)	**Motor1**	**Motor2**	**Motor3**
Cam1	26.52576182	**13.26288091**	**13.26288091**
Cam2	33.15720227	16.57860114	16.57860114
Spatial Resolution (mm)	**Motor1**	**Motor2**	**Motor3**
Cam1	**7.93E-04**	**7.93E-04**	**7.93E-04**
Cam2	8.19E-04	8.19E-04	8.19E-04

Table 1: Summary of trade studies results

Note: The ResultsSummary.xls attached with the sample models contains links to the .csv files exported from Rhapsody PCE. To update the links you need to open all .csv files and then update the value in each cell (by typing in the cell = and then switching to the relevant .csv file and selecting the relevant cell there)

Tutorial Summary

PCE is an add-on that provides the capability to analyze/evaluate parametric diagrams. We showed how to solve algebraic and time dependent differential equations as well as validate constraints. We also showed how to use these capabilities to perform a classical trade study analysis within the context of the design model. There are additional capabilities, which were not covered here, such as using causal expressions (Constraint Blocks stereotyped *«ExpressionForMatlab»)* and exporting to the CAS language.

We believe that PCE allows systems engineers to perform various simple analysis tasks from within the SysML model; however we still expect that more complicated analysis would require special experts to perform analysis outside the context of the SysML design. We hope that PCE will serve as a basis to integrate with more advanced analytical tools and techniques in accordance with the INCOSE Model-Based Systems Engineering (MBSE) vision in which all the analysis is carried out in the context of design models.