Enterprise Architect version 13 Beta ~ Simulation using SysML 1.4 and OpenModelica

Sparx Systems released the first Beta of Enterprise Architect version 13 to registered users on June 3rd 2016. Since then a further build of the Beta was released on July 20th 2016.

Numerous changes have been made in version 13 (a full summary can be found at http://www.sparxsystems.com/products/ea/13/). I have not tried them all, but one of the more complex features is explored in this tutorial, namely, Simulation using SysML 1.4 and OpenModelica.

In the help for Enterprise Architect version 13 is a good section, including examples, on simulation using SysML 1.4 and OpenModelica. These examples are also included in the example repository EA Example.eap which is part in the installation of Enterprise Architect version 13 and accessible via the Help Ribbon. However I found that whilst these are good, they are lacking in detail, hence the reason why I wrote this tutorial.

Rather than just expand upon the examples provided by Sparx Systems, I created my own example (used within this tutorial). I certainly learnt a lot more creating this, than I did by simply replicating one of the Sparx Systems Examples.

Once you have completed this tutorial, revisit these examples, and they should make more sense!

Overview

Simulation using SysML 1.4 and OpenModelica is the ability to create a Systems Engineering Model using SysML 1.4, model mathematical behaviour and perform time simulation using OpenModelica.

The subject of time simulation and in particular OpenModelica is complex, so this tutorial will illustrate a fairly basic example to get you started.

The simulation provided by Enterprise Architect Version 13 is a graphical plot for a number of variables (which have the same dimension) on the y-axis against time on the x-axis.

The Example and its Mathematical Theory

For this tutorial I have chosen to model a simple Weapon that is capable of launching a projectile, simply known as Ballistics.

The Weapon will have:

an initial launch velocity for its projectile.
an angle of launch for the projectile.
The simulation will plot **two variables** which have a **dimension of distance**:

- the **parabolic trajectory** of the projectile (the time the projectile is in the air).
- the **range** of the projectile.

**The Math!**

To conduct a simulation we have to bite the bullet and apply some math. If you are not a Systems Engineer then most equations for simulations like **motion theory**, **pendulums**, and indeed **ballistics** can be found on the **World Wide Web**.

I have included the math here, since it is fundamental to our tutorial.

First let us assume that we are on earth where the force of gravity is 9.81 m/s

Also let us assume that we are not considering resistance due to air etc in our calculations.

Give the following definitions:

- _initial projectile velocity is defined as u_
- _initial launch angle is defined as θ_
- _horizontal position at any point in time is defined as x_
- _vertical position at any point in time is defined as y_
- _time is defined as t_
- _force of gravity is defined as g_

Then we have:

\[
x = u \times t \times \cos\theta
\]

\[
y = u \times \sin\theta - \frac{1}{2} \times g \times t
\]

Combining these equations using Pythagoras' Theorem we obtain:

\[
\text{parabolic trajectory at any point in time : } y = x \times \tan\theta - \frac{g \times x^2}{2 \times u^2 \times \cos^2\theta}
\]
For the range of the projectile we use:

\[ \text{range} = \frac{u^2 + \sin 2\theta}{g} \]

In addition, OpenModelica trigonometrical functions expect their parameter angle to be specified in radians whereas in the “real world” we usually use degrees. A simple conversion formula is:

\[ \text{radians} = \theta \times \frac{\pi}{180} \quad \text{where } \theta \text{ is the angle in degrees} \]

Now we have the math behind us we can build our model ready for simulation.

Creating the Model

Before creating the model ensure that you have installed OpenModelica, this can be downloaded free of charge from

https://www.openmodelica.org/

Plan of Attack

I will be using linear approach to building the model, this approach is re-usable and applicable to any such Systems Modelling.

- Create a new repository.
- Model any valueTypes (this is optional).
- Model any Blocks that will type Ports (in this tutorial none of our Blocks require Ports).
- Model the equations as ConstraintBlocks.
- Model the structural design using Blocks including Value Properties and Constraint Properties.
- If the Blocks have Ports, model the interconnectivity between these Blocks.
- Model a Parametric for the Block to be simulated.
- Model variations for the Block to be simulated.
- Configure the Simulation.
- Run the Simulation(s),
Create a New Repository

Create a new model repository in a location of your choice using a name of your choice.

**NOTES**

When using the Model Wizard for SysML 1.4, you will observe that the ability to add a structure for the System Model is not present (it is for SysML 1.3). We do not need such a structure for this tutorial, but if you do require this structure for your own projects, then activate both SysML 1.3 and SysML 1.4 MDGs. Create you structure, then deactivate SysML 1.3.

As code will be generated during the simulation process, avoid using spaces or punctuation when naming any element including Packages in your model.

In this tutorial I added the PrimitiveValueTypes.

Add a Package named Ballistics to your Model Root and add further Packages to Ballistics so as to create the following structure:

```
Model Browser

<table>
<thead>
<tr>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
</tr>
<tr>
<td>Ballistics</td>
</tr>
<tr>
<td>Constraints</td>
</tr>
<tr>
<td>Structure</td>
</tr>
<tr>
<td>Comparison</td>
</tr>
<tr>
<td>PrimitiveValueTypes</td>
</tr>
<tr>
<td>ModelLibrary</td>
</tr>
</tbody>
</table>
```

Model the Equations using ConstraintBlocks

In the Package named Constraints add a SysML 1.4 Block Definition Diagram named Constraints.

Using the Toolbox add a ConstraintBlock named Trajectory to this diagram.

Whilst the Properties dialog is still open, enter the following (make sure that your constraint is syntactically correct!) as the name of a new constraint:

\[ y = (x \cdot \tan(\theta)) - \left( \frac{g \cdot x^2}{2 \cdot u \cdot u \cdot \cos(\theta) \cdot \cos(\theta)} \right) \]

Save the Constraint.

Using the Toolbox add Properties named theta
To the **ConstraintBlock** named **Trajectory**. There is no need to type these properties, since they will default to **Real** when we come to simulation.

Press **Delete** key for each of these **Properties** to remove them from the diagram and you should see the following:

If you do not see the **constraints** compartment, use the **Diagram / Element** properties to make the **Constraints** compartment visible.

Repeat the above process to create **ConstraintBlocks** for

**HorizontalPosition**
NOTE
We do not have to create a parameter for time since this is pre-defined already for simulation.

DegToRad

Range
Model the Structural Design

For this tutorial, this is really straightforward as our structural design just consists of a single block for the Weapon.

In the Package named Structure add a SysML 1.4 Block Definition Diagram named Structure.

Using the Toolbox add a Block named Weapon to this diagram.

Add the following Properties to the Block (there is no need to type them as they will all be Real and this is the assumed default for simulation).

- g (set the default value to 9.81)
- range
- theta
- u
- y

Add the following ConstraintProperties typed to their corresponding ConstraintBlock to this Block.

<table>
<thead>
<tr>
<th>ConstraintProperty Name</th>
<th>Typed to ConstraintBlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>d2r</td>
<td>DegToRad</td>
</tr>
<tr>
<td>hp</td>
<td>HorizontalPosition</td>
</tr>
<tr>
<td>mg</td>
<td>Range</td>
</tr>
<tr>
<td>traj</td>
<td>Trajectory</td>
</tr>
</tbody>
</table>
NOTE
You can add a **ConstraintProperty** to a Block by:

Adding a Property from the Toolbox.

Setting its **Stereotype** to **SysML1.4::constraintProperty** (use the … navigate button, select **SysML 1.4** from the Profile dropdown, then select **constraintProperty** from the list of stereotypes).

Your **Block** should be as shown below:

```
«block»
Weapon

properties
  g = 9.81
  range
  theta
  u
  y

constraints
  d2r : DegToRad
  hp : HorizontalPosition
  rng : Range
  traj : Trajectory
```

NOTE
To aid locating the **Package** containing the **ConstraintBlocks** consider setting the **Package** as a **Namespace Root**, by right-clicking the **package** and selecting **Code Engineering | Set as Namespace Root**.

Model the Parametric for the Structural Design
Right-click the **Block** named **Weapon** and select **New Child Diagram | Parametric Diagram**, accepting the default diagram name of **Weapon**.

Right-click the **Frame** and select **Synchronize Structural Elements**. All the **Properties** and **Constraint Properties** will now be added to the **Frame**.

Arrange these as shown below:

Right-click the **ConstraintProperty** named **d2r** and select **Structural Elements…**

Click **Show Owned/Inherited**.

Select all the **Properties**.

Repeat the above for the remaining **ConstraintProperties**.

After some rearrangement, your diagram shown be as shown below:
Using **Binding Connectors** connect corresponding **Parameters** so your diagram is as shown below:
Modelling the Variations

We now model variations of the Block named Weapon so that we can use the simulation to compare different values of Initial Velocity and Angle of Elevation.

In the Package named Comparisons add a SysML 1.4 Block Definition Diagram named Comparisons.

Using the Toolbox add a Block named WeaponCompare to this diagram.

To this Block add three Properties named weapon1, weapon2 and weapon3 respectively. These Properties should all be typed to the Block named Weapon.
Once completed, your diagram should be as shown below:

![Diagram of WeaponComparison block with weapon1, weapon2, and weapon3]

**Configure the Simulation**

Select the **Ribbon** named **Simulate** then select

![Ribbon with Manage and SysMLSim]

Select **SysMLSim Configuration Manager**.

This opens the **SysMLSim** workspace as shown below:

![SysMLSim workspace]

If this is the first time you have run a simulation, then select

![Folder icon from the top toolbar]

Select **Configure Modelica Solver**

Navigate to the installation folder for **OpenModelica** (as shown below)
Click **OK**.

The first step for new simulation is to create a **SysMLSimConfiguration Artifact** (subsequently you can simply select a **SysMLSimConfiguration Artifact Artifact**).

Select ![image](image.png) from the top toolbar

Select **Create Artifact**

Select **Ballistics** in the Select Package Dialog and click **Add New** to create a new **SysMLSimConfiguration Artifact** named **Ballistics**.

We now have to set the **Package** using the … navigation adjacent to **Package**. It is important to select a **Package** that contains **all elements** in the model to be simulated, in this tutorial this is the **Package** named **Ballistics**.

The model structure is then added to the **SysMLSim Configuration Manager** as shown below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>block</td>
<td></td>
</tr>
<tr>
<td>Weapon</td>
<td></td>
</tr>
<tr>
<td>WeaponCompare</td>
<td></td>
</tr>
<tr>
<td>constraintBlock</td>
<td></td>
</tr>
</tbody>
</table>

Expand the node named **Weapon** and then expand **Part**:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>block</td>
<td></td>
</tr>
<tr>
<td>Weapon</td>
<td></td>
</tr>
<tr>
<td>Part</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>Real</td>
</tr>
<tr>
<td>range</td>
<td>Real</td>
</tr>
<tr>
<td>theta</td>
<td>Real</td>
</tr>
<tr>
<td>u</td>
<td>Real</td>
</tr>
<tr>
<td>y</td>
<td>Real</td>
</tr>
</tbody>
</table>
First we need to inform the simulator that **Weapon** is a **Class**.
Select **SysMLSimClass** from its drop down.

Next we need to configure whether these **Parts** are **Variables** or **Constants**.

For this tutorial we have **two variables**, `y` and `range`, the remaining **Parts** are **Constants**.

For each part, click on its drop down and select either **SimVariable** or **SimConstant** as shown below:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weapon</td>
<td>SysMLSimClass</td>
</tr>
<tr>
<td>g : Real</td>
<td>SimConstant</td>
</tr>
<tr>
<td>range : Real</td>
<td>SimVariable</td>
</tr>
<tr>
<td>theta : Real</td>
<td>SimConstant</td>
</tr>
<tr>
<td>u : Real</td>
<td>SimConstant</td>
</tr>
<tr>
<td>y : Real</td>
<td>SimVariable</td>
</tr>
<tr>
<td>constraintProperty</td>
<td></td>
</tr>
<tr>
<td>BindingConnector</td>
<td></td>
</tr>
<tr>
<td>WeaponCompare</td>
<td></td>
</tr>
<tr>
<td>constraintBlock</td>
<td></td>
</tr>
</tbody>
</table>

We are going to simulate the **Block** named **WeaponCompare** (so as to compare different weapon settings), this is known as the **SysMLSimModel**.

This is selected by using the drop down for **WeaponCompare** and selecting **SysMLSimModel**.

This populates the **Simulator** section of the **SysMLSim Configuration Manager** as shown below:
So that we can run several simulations each using different settings for the **Weapon Parameters**, we create one or more **DataSets** for the **Simulation Model**.

Right-click the entry named **WeaponCompare** and select **Create Simulation DataSet**.

Do this twice more, and then expand **WeaponCompare**.

You can rename each **DataSet** by clicking on its default name **DataSet_1** etc. and then typing a new name for example rename:

**DataSet_1** as **Same Velocity Different Angle**
DataSet_2 as Same Angle Different Velocity
DataSet_3 as Different Velocity Different Angle

We now configure each DataSet by clicking **Click button to configure...** followed by clicking the **...** button. This opens the **Configure Simulation Data** dialog (the one shown is for the DataSet named Same Velocity Different Angle):

Values are entered in **Value** column for their corresponding attribute as shown below:
Click **OK** when done.

Repeat the above using the following values for **theta** and **u**.

### DataSet : Same Angle Different Velocity

<table>
<thead>
<tr>
<th>Weapon</th>
<th>theta</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>weapon1</td>
<td>45</td>
<td>200</td>
</tr>
<tr>
<td>weapon2</td>
<td>45</td>
<td>400</td>
</tr>
<tr>
<td>weapon3</td>
<td>45</td>
<td>600</td>
</tr>
</tbody>
</table>

### DataSet : Different Angle Different Velocity

<table>
<thead>
<tr>
<th>Weapon</th>
<th>theta</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>weapon1</td>
<td>60</td>
<td>200</td>
</tr>
<tr>
<td>weapon2</td>
<td>45</td>
<td>600</td>
</tr>
</tbody>
</table>
Save the settings by clicking the on in the toolbar.

**Run the Simulation**

To run a simulation:

Select a **Model** to simulate.

Select a **DataSet** from this **Model**.

Select a **Start** and **Stop** time (initially set to 0 and 20).

Select the **Properties to Plot**.

Click **Solve**

If all has gone well, you should see a simulation graph (if not refer to section named **Troubleshooting** in this tutorial).

For this tutorial and for the first simulation:

Select **WeaponCompare** as the **Model**.

Select **Same Velocity Different Angle** as the **DataSet**

Enter 85 as the **Stop** time (I found this value by running the simulation several times, until I had an optimum value)

Select all Properties to Plot.

Click **Solve**.
If you do not see this graph refer to the section named **Troubleshooting**.

The trajectory for **weapon3** is difficult to see since it is coloured **yellow**, this can be changed by clicking **Setup** and selecting a different colour (for example **Black**).

I believe, it is not possible to save these colour settings for future simulations.
You can also set titles and a legend.
Some Observations

You may notice that the parabolic curve for the trajectory, does not look exactly the same as some examples you may have found elsewhere. This is due to the fact that most ballistic graphs plot distance on the x-axis and distance on the y-axis.

In Enterprise Architect Version 13 SysML Simulation always plots time on the x-axis, hence the parabolas above represent the time that the projectile is in the air.

Also notice that the plot continues for negative values of y. (There may be a method of controlling this using Modelica Functions created as an operation stereotyped as SimFunction and using Modelica code as the operation’s behaviour, but I have not tried this).

Also notice that we appear to have only one plot for the Range (which is a straight line since its value does not vary with time). This is due to the fact that complimentary angles (such as 30 and 60) always produce the same range value. In addition in this example, the Range values for the angles 30 / 60 and 45 are very close, and cannot be distinguished in the scale used for the y-axis.

Repeat the simulations for the other two datasets and you should see this following graphs.

Same Angle Different Velocity
Different Velocity Different Angle
Troubleshooting

When trying SysML Simulation for myself, I encountered numerous errors which prevented the graph from being displayed.

Error messages are displayed in the System Output window, the most common I encountered where:

**Too many equations over determined system** – the model has $N$ equations and $M$ variables

**Too few equations under determined system** – the model has $M$ equations and $N$ variables

**Class name missing**

Where $N$ is greater than $M$ and name is the name of the Block you are simulating.

The cause of the first two errors is simply that the Parts have the wrong type (you either have too many SimVariables or too few SimVariables. The most common cause of this I found is forgetting to set Parts to SimConstant (it appears that no setting is similar to a SimVariable).

The last error message is more puzzling, since the model class is plainly present. What the error actually means, is that code generated for the class has failed to parse (or compile), the most usual cause is a syntax error in the constraint text (miss-matched brackets etc).

There may be many other errors yet to be encountered, but hopefully the above will solve most of your problems.

Conclusion

In this article I have provided a tutorial to illustrate the new Simulation using SysML 1.4 and OpenModelica functionality provided in Enterprise Architecture version 13 Beta.

I hope you found this article useful and informative and please keep a lookout for further mini tutorials on the new and exciting features of EA version 13.

Phil Chudley
Principal Consultant
Dunstan Thomas Consulting
@SparxEAGuru