

## SysML Parametrics

Read and understand the Addition and Satellite examples from the Paramagic tutorial that has been posted on the class website. Reproducing parts of it by yourself using the MagicDraw tool will be of great benefit. You will also find the Paramagic User Guide online.

In this lab, you will be using MagicDraw to determine if a set of requirements can be achieved for a simple system. In this week's lab, you will be using Matlab as the solver and it will be up to you how to structure the requirements diagrams, the block diagrams, the parametric diagrams and the instance diagram using the tutorial as a guideline.

### Problem Statement

The customer wishes to use a pendulum clock to keep track of time. His requirement is that the clock loses no more than a second every hour.

**Requirement:** The clock loses no more than one second every hour.<sup>1</sup>

**Refer to the Modeling assignment on pendulums for a description of the formulas.**

### Matlab Pendulum Model

The pendulum will be modeled in Matlab and will have two input parameters, the length and the angle, and one output parameter, the period. Instructions on creating the model are given below.

This will be how the pendulum is modeled in Matlab. Create an mfile called "pendulum.m" with the following contents.

```
function xdot = pendulum(len, t, x)
xdot = [x(2); -9.81*sin(x(1))/len];
```

Here  $x$  is the state vector standing for  $\langle \theta, \omega \rangle'$ . Matlab will use this description to solve the system.

Create an additional mfile called "zeroevent.m" with the following contents.

```
function [lookfor stop direction] = zeroevent(t,x)
lookfor = x(1);
stop = 1;
direction = -1;
```

This file instructs Matlab's differential equation solver to look for  $\theta = 0$  and to stop when it occurs, with  $\theta$  having a negative derivative.

This next mfile should be called "computePeriod.m" and will call the ode solver to determine the period.

```
function period = computePeriod(len, angle)
x0 = [angle*(pi()/180), 0];
findEvent = odeset('Events', @zeroevent);
[t,x,te,xe,ie]=ode45(@(t,x)pendulum(len,t,x), [0,5], x0, findEvent);
period = 4*te;
```

Finally, to link the created model to MagicDraw UML, create a Matlab script called "periodscript.m".

```
insel = load('input.txt');
len = insel(1);
angle = insel(2);
period = computePeriod(len, angle);
save('output.txt', 'period', '-ASCII');
exit
```

<sup>1</sup>Hint: a requirement diagram can be used to capture this, but an additional constraint might be needed to model it.

**Linking the Matlab Model to a Constraint** When creating the Constraint Block that will capture the pendulum behavior, follow the instruction 4-6 that begin on page 48 of the Paramagic User Manual. In step 4.5, use the following settings.

**output Parameter:** period

**script\_or\_function:** scriptascii

**name\_of\_m-file:** periodscript.m

**input1:** length

**input2:** angle

This links the constraint to the Matlab model.

### Questions

1. Create a requirements diagram (showing how the requirements are verified), a block diagram showing the structure, as many parametric diagrams as necessary, and instance diagrams as needed to describe this problem.
2. Explore different values for the starting angle and see if you can determine a range over which the customer's demands are feasible.