Software Engineering
Module: Model Based Software Engineering

Project requirements
AY 2020/21

Enrico Tronci
Computer Science Department, Sapienza University of Rome
Via Salaria 113 - 00198 Roma - Italy

tronci@di.uniroma1.it

Version: 2020-12-07

1 Exam

The exam for this part (MBSE in the following) of the *Software Engineering* course is worth 3 credits. Since *Software Engineering* is a 6 credit course, the MBSE exam is worth 50% of the total grade for the *Software Engineering* course.

The MBSE exam consists of two parts: a written test and a project. The written test is worth 50% of the total grade for the MBSE exam. The project contributes for the remaining 50% of the MBSE exam.

The written test for the MBSE exam and the project for the MBSE exam can be completed in any order. When they are both completed the final grade for the MBSE exam will be computed.

In the following we describe the project requirements.

2 Project structure

A project will consist of software along with a short technical report as outlined below.

2.1 Software to be delivered

You should deliver the following software.

1. All `.mo` files for your Modelica models.
2. An OpenModelica script `run.mos` to run your Modelica model.
3. A Python script `verify.py` verifying that the system requirements are meet for all operational scenarios. You can use Montecarlo sampling to carry out the above verification. The important thing is that any operational scenario has a non-zero probability of being selected (and thus tested).

4. A Python script `synth.py` computing values (e.g., using Montecarlo) for some of the system parameters so that for all operational scenarios all system requirements are meet and some Key Performance Indicator (KPI) is optimised.

2.1.1 Software structure

All software will be inside a directory named `Prj`.

The directory `Prj` will in turn consists of two sub-directories: `Models` and `MLibrary`.

The directory `MLibrary` will contain all packages you will use as a library in your `.mo` files.

The directory `Models` will contain all your models as detailed below.

2.1.2 Files in the directory `Models`

The directory `Models` will contain the following elements.

1. All `.mo` files for your Modelica model.
2. The OpenModelica script `run.mos` to run your model.
3. The Python script `verify.py`.
4. The Python script `synth.py`.

All software will be tested on a Linux machine using Python 2.7 with OpenModelica and OMPython installed. So, test it in the same environment before delivering.

Be careful with filenames and filepaths. For example, in Unix filenames are case sensitive (thus `Models` is different from `models`) and filepaths will be formed with `/` while in Windows with \.

Keep your code portable across directories (you do not know where your code will run). So, for example, to point to the directory `MLibrary` from the `run.mos` script or from `verify.py` or `synth.py`, you should use `../MLibrary` in order to be robust with respect to the directory where your project (namely, your `Prj` folder) will be deployed.

2.1.3 Models

Your Modelica code will model the following software design elements.

1. **Operational scenarios (*use cases*)**. This will consists of one or more `.mo` files. The model for the operational scenarios will have to be reasonably *complete* (*i.e.*, all plausible scenarios should be generated) and *sound* (only plausible scenarios should be generated). Completeness avoids declaring correct a system that is not. Soundness avoids *over-engineering*, that is, designing a system for situations that will never occur.

2. **System Architecture**. This will consists of one or more `.mo` files modelling the system architecture as well as the external behaviour of the main system components.

3. **Monitors**. Functional as well as non-functional requirements are modelled using monitors. You should model at least a functional requirement and at least a non-functional requirement. Thus monitors will be modelled through two (one functional and one non-functional) or more `.mo` files.

2.2 Technical report

Your technical report will consists of the following sections.

1. **System description**. A short (max one page) description of the overall system.

2. **Operational scenarios**. A short (max one page) description of:

   (a) the operational scenarios for your system;

   (b) how you modelled them with Modelica;

   (c) why we may consider that your Modelica model for the operational scenarios is reasonably complete;

   (d) why we may consider that your Modelica model the the operational scenarios is reasoably sound.

3. **System Architecture**. A short (max one page) description of the system architecture and of how you modelled it within your Modelica model.

4. **System requirements**. A short (max one page) description of the system functional and non-functional requirements you considered and of how you modelled them within your Modelica model.

5. **Experimental results**. A short description and discussion of the experimental results. You should run at lest the following experiments:

   (a) Run of the system using the script `run.mos`. For this case show: compile time and simulation time as returned by OpenModelica along with a picture of the plot showing monitors output and any other output you consider relevant.

   (b) Run the verification script `verify.py` using 100, 1000, 10000 samples. Show the verification results along with the CPU time needed to complete the verification activity in each case.

   (c) Run the synthesis script `synth.py` using 100, 1000 samples. Show the synthesis results along with the CPU time needed to complete the synthesis activity in each case.