University of Applied Sciences Wismar
University of Applied Sciences
Technology, Business and Design
Research Group Computational Engineering and Automation

Documentation

SESEcPy - SES Experiment Control Python

Experiment Control as Part of the Extended SES/MB Infrastructure

as of: August 15, 2019

by: Research Group CEA

Contents

1 Introduction and Description of SESEcPy

This is the Experiment Control (EC) SESEcPy as part of the System Entity Structure (SES)/Model Base (MB) infrastructure introduced in the documentation of the SES modeling tool SESToPy. It provides scripts for the procedure of automatic simulation experiment control. For information on the SES and its extensions to the extended SES/MB (eSES/MB) infrastructure please read the documentation of SESToPy and how to connect the SES to an MB please read the documentation of SESMoPy. This software was written in the Research Group Computational Engineering und Automation (CEA) at the University of Applied Sciences Wismar.
It is written in Python 3.4.1, but as of August 2018 it runs in current Python versions as well. Currently there are no more modules needed than given in the Python Standard Library. As simulation softwares Matlab R2018a (for Simulink), OpenModelica 1.12.0 and Dymola 2018 are used.
As shown in detail in the documentation of SESToPy the SES can be connected to an MB for generating models. It was extended to allow automatic model generation and execution of models. The process is depicted in Figure 1.1.

This software controls the specification of simulation experiments, automatic experiment execution, and the evaluation of simulation results. The part of the eSES/MB infrastructure as well as the interfaces needed are shown in Figure 1.2 for clarification.
A detailed explanation on the Python-based software structure supporting the eSES/MB infrastructure is given in the documentation of SESToPy.

SESEcPy has no graphical user interface.

SESEcPy has three scripts: a main script, a script to control the general process, and a script defining the experiment. In the main script functions of the two other scripts are called in a loop until the experiment goals are reached. Some further preparation needs to be done as well.
Experiment specific parts are defined in two functions in the experiment specific script. The function *initialSettings* sets the initial settings of the experiment like the SES file to be used and the path(s) of the MB(s). The second function of the script

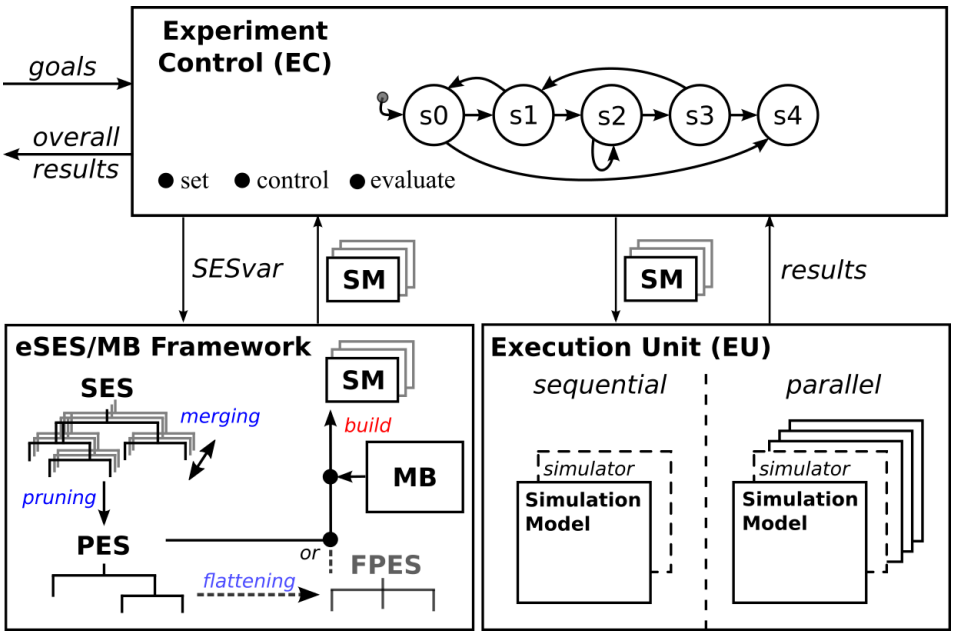

Figure 1.1: Extended SES/MB-based infrastructure.

defining the experiment *nextState* sets variables based on the simulation number and, if it is not the first run, based on the simulation results returned by SESEuPy. If the experiment goals are not met, it is defined which simulator and solver to use next, how to set the SES variable (SESvar)s next, which paths to use for the Pruned Entity Structure (PES) and Flattened Pruned Entity Structure (FPES) derived from the SES next, and the next simulator parameterization like the starttime and stoptime is set. Thus this allows a reactive control of an experiment. The script describing the general process then calls the methods pruning and flattening of SESToPy before the build method of SESMoPy is called and the model is executed with the tool SESEuPy which returns simulation results. An example makes that process clear.

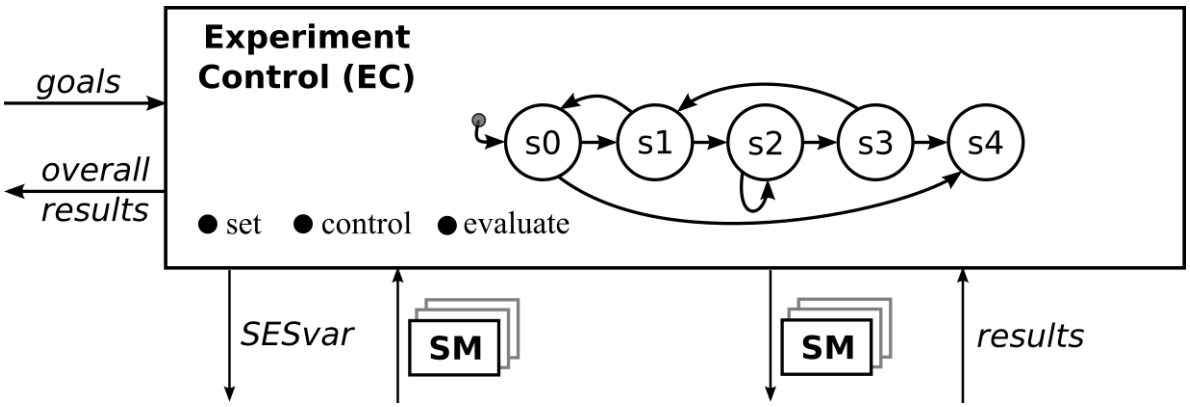

Figure 1.2: Part of the eSES/MB infrastructure supported by SESEcPy.

2 Example

This example can be found in the Example folder in the main directory of SESEcPy. It describes the optimization of a feedback control system with optional feedforward control and is introduced thoroughly in the documentation of SESToPy. This example shall clarify the usage of the eSES/MB infrastructure and present a template for other experiments.

This example is given using the native interface and using Functional Mock-up Interface (FMI) to support different simulators. For FMI it is given in two ways: Only using OpenModelica basic models on one hand and using Functional Mock-up Unit (FMU)s and OpenModelica basic models on the other hand. This is introduced in more detail in the documentation of SESMoPy.

In the SES the links to the basic models are given in the MB attribute (see the documentation of SESMoPy).
Using the native way the links are like “MB/blockname“, where “MB“ refers to the file “MB.slx“ for Simulink and to “MB.mo“ for OpenModelica and Dymola. For each simulator a dedicated simulator specific modelbase is needed. The files for the modelbase are in simulator specific subdirectories of the directory containing the FPES .jsonsestree file. These simulator specific directories are set in code in the *initialSettings* function.
Using FMI the links are like “MB.mo/blockname“ for OpenModelica basic models and “MB/blockname.fmu“ for FMUs. This refers to the file “MB.mo“ for OpenModelica basic models and the FMUs lie in a directory named “MB“. Only one modelbase (which can consist of OpenModelica basic models and FMUs) for each simulator is needed. The files for the modelbase are in the same directory as the directory containing the FPES .jsonsestree file. This directory is set in code in the *initialSettings* function.
Because of the different ways for the definition of MB attributes there are different SES for both ways.

The experiment to execute is defined in the experiment specific script of SESEcPy. The paths to the SES, the MB and the simulation behavior are set in the *initialSet-*

tings function. Remember to set the correct paths using the native interface or FMI. Furthermore in the *nextState* function the calculation of the state the simulation will enter next (either first run or based on already achieved experiment results) is defined. It calculates in this example whether control goals are achieved with the current simulation run and what to try next by setting SESvar values and simulation parameters. Different simulators and simulator configurations are tried and finally the feedforward is added by setting the SESvar *feedforward=1*. When the experiment goals are met, the overall results are returned.
This is given in the commented sourcecode of SESEcPy and shall not be discussed further. In the documentations of SESMoPy and SESEuPy the simulators and versions tested with this example are given.

The experimentation is started by starting SESEcPy's "main.py" file assuming that Python 3 is installed.

3 Related Work

Related work with contributions to the development of the SES/MB framework is given in the documentation of SESToPy. Some related work regarding model building in connection with the SES is given in the documentation of SESMoPy.

List of Figures

List of Tables

List of Listings

Acronyms

CEA Computational Engineering und Automation.

EC Experiment Control.

eSES/MB extended SES/MB.

FMI Functional Mock-up Interface.

FPES Flattened Pruned Entity Structure.

MB Model Base.

PES Pruned Entity Structure.

SES System Entity Structure.

SESvar SES variable.