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

Documentation

SESEuPy - SES Execution Unit Python

Execution Unit for Models Built with SESEuPy

as of: March 19, 2019

by: Research Group CEA

Contents

1 Introduction

This is the execution unit SESEuPy, executing executable models built with the software SESMoPy. For information on the System Entity Structure (SES) please read the documentation of SESToPy and how to connect the SES to an Model Base (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. There are scripts for building an executable in Windows using cx_Freeze and a runnable in Linux using PyInstaller. Currently there are no more modules needed than given in the Python Standard Library.

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, depicted in Figure 1.1.

This software provides the execution of models created with the modelbuilder soft-
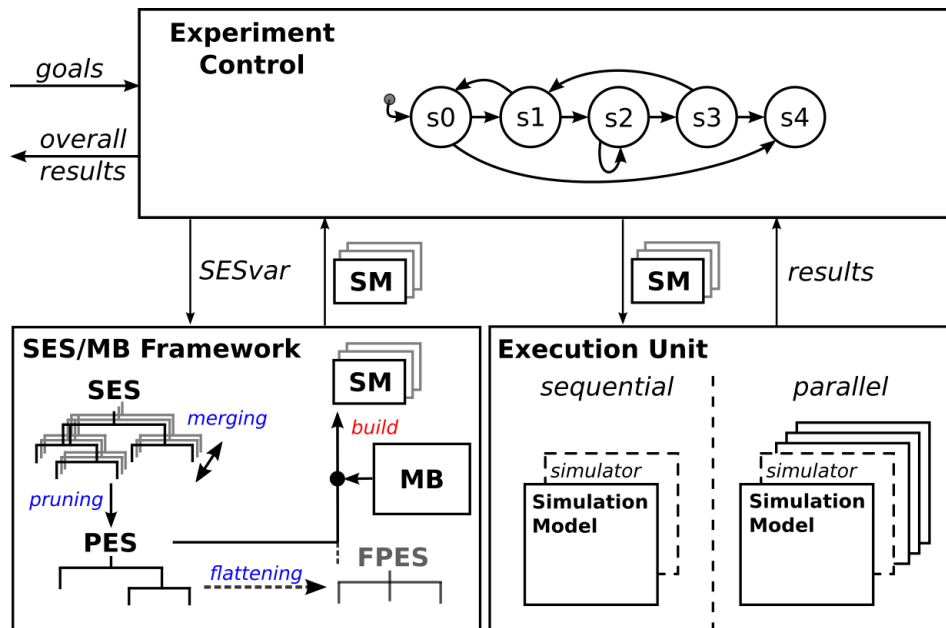

Figure 1.1: Extended SES/MB-based infrastructure.

3

ware SESMoPy sequential or in parallel. This is shown in Figure 1.2 for clarification. *A detailed explanation on the Python-based software structure supporting the SES/MB infrastructure is given in the documentation of SESToPy.*

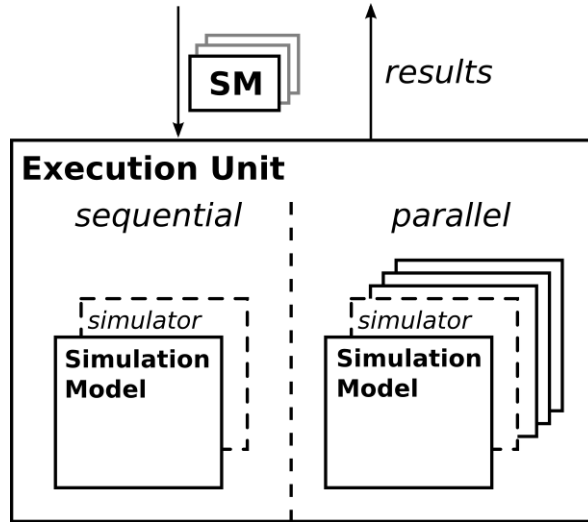SESEuPy has no graphical user interface. The simulators as described in the

Figure 1.2: SES/MB supported by SESEuPy.

documentation of the modelbuilder software SESMoPy are supported.

2 Working with SESEuPy

SESEuPy is used for automatic execution of simulation model (SM)s in the context of the infrastructure given in Figure 1.1. In the following sections of this chapter information on the function of SESEuPy shall be given.

2.1 Prerequisites

As seen in Figures 1.1 and 1.2, SESEuPy receives a number of SMs to execute. Then the SMs can be executed sequential or in parallel. The simulation results are collected and returned.

Since this software is part of the infrastructure, it receives all for the execution necessary data including the simulator settings in the configuration file originally created by the modelbuilder software SESMoPy, which is extended in the Experiment Control. There is no need for a graphical user interface.

The configuration file needs to have the entries according to Table 2.1. There can

Table 2.1: Entries in the configuration file.

variable	Description
MODEL	Complete path to a modelfile (*.m or *.mo)
SIMULATOR	"Simulink" or "OpenModelica" or "Dymola"
MODELBASE	Complete path to a modelbase file
STARTTIME	The simulation will start at this time.
SOLVER	The solver to use for the simulation, e.g. "ode45" for Simulink or "dassl" for OpenModelica and Dymola.
STOPTIME	The simulation will stop, when the stoptime is reached.
MAXSTEP	Maximum stepwidth for the simulation.
EXECTYPE	Type of the execution: "sequential" or "parallel".
NSIGANA	Names of the signals to analyze in the simulation output: ["Block1.Output", "Block2.Output", ...] Needed for OpenModelica and Dymola to filter the simulation results of interest. Notice the quotes!

be several entries of the type MODEL. Every entry refers to one model description

file including the absolute path, e.g. ("/Path/to/Model.m"). Furthermore there can be several entries of the type MODELBASE. Every entry refers to a file containing basic models. There can be more entries in the configuration file, which are not used by SESEuPy and therefore ignored by SESEuPy.

2.2 Performing a Simulation

When performing a simulation run, the configuration file in the folder containing the models to simulate is analysed. Depending on the simulator to use, there are different steps taken.

Simulink   If the simulator is selected as Simulink (in the configuration file coming from the modelbuilder software SESMoPy), the .m script describing the model (generated by SESMoPy) is extended. The simulation information solver, stoptime and maxstep as in Table 2.1 are inserted. Furthermore, there are instructions for generating a CSV file of the simulation results.
Finally, the simulation is executed using a Matlab command calling the extended .m script.

OpenModelica   If the simulator is selected as OpenModelica (in the configuration file coming from the modelbuilder software SESMoPy), a file with the ending .mos (MOdelica Script) is created. This file holds instructions on loading the MB and the model in OpenModelica. Furthermore, simulation information e.g. solver, number of steps (calculated of stoptime and maxstep) and stoptime as in the configuration file in Table 2.1 are defined. A command to save the simulation results as CSV is applied. The simulation then is executed by calling the created .mos script with the OpenModelica simulator program "omc". By default, all simulation results are printed in the result CSV file. The configuration file has the entry NSIGANA, in which the signals of interest in the simulation results are specified. These are extracted and saved in a CSV file with the name of the model.
Please be aware, that for simulation with OpenModelica the name of the model may not start with a number.

Dymola   If the simulator is selected as Dymola (in the configuration file coming from the modelbuilder software SESMoPy), a file with the ending .mos (MOdelica Script) is created. This file holds instructions on loading the MB and the model in

Dymola. Furthermore, simulation information e.g. solver, number of steps (calculated of stoptime and maxstep) and stoptime as in the configuration file in Table 2.1 are defined. The simulation then is executed by calling the created .mos script with a Dymola command. By default, Dymola saves the simulation results as .mat file. The tool "alist" lying in the same folder as the Dymola executable is shipped with Dymola, allowing to convert the .mat to CSV files. The signals to convert need to be defined. The configuration file has the entry NSIGANA, in which the signals of interest in the simulation results are specified. These are converted using "alist" and saved in a CSV file with the name of the model.

Generally   It is important, that the simulation programs are on the user's path, they need to be startable from the command window / shell.
For Windows, the bin folder of Matlab / OpenModelica needs to be on the path of the user / system. For Dymola this is the bin or bin64 folder.
For Linux, Matlab / omc of OpenModelica / Dymola must be able to be started with the command "matlab" / "omc" / "dymola" from the shell. Otherwise a symbolic link to the Matlab / omc / Dymola executable with the name "matlab" / "omc" / "dymola" in the /usr/bin folder needs to be placed.

The models can be called sequentially or in parallel to be simulated in their respective simulation software. Therefore the configuration file in Table 2.1 has the entry EXECTYPE, which can have the values "sequential" or "parallel".
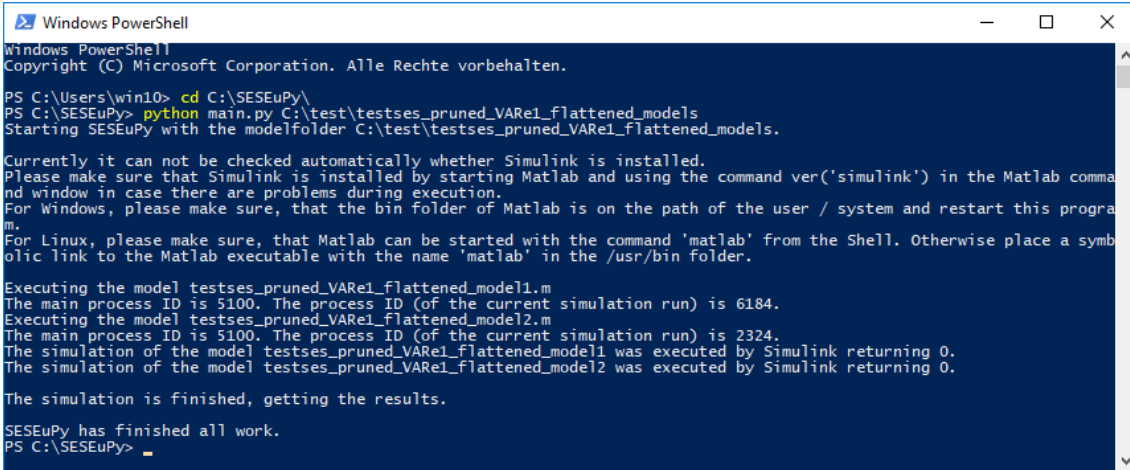
Experimental Frame

In the configuration file is defined how to summarize the simulation results.

2.3 Calling SESEuPy / Execution Unit Interface

The execution unit can be started from the command line as seen in Figure 2.1. Remember, that for execution the file containing the MB has to be in the same directory as the generated models by the modelbuilder software SESMoPy. Usually, it should already be placed there by the modelbuilder software SESMoPy when generating the models.
 Figure 2.1 shows the usage of the execution unit interface when performing a simulation run with Simulink from the command line. The folder in which the models,

Figure 2.1: Interface of the execution unit SESEuPy.

the modelbase and the configuration file reside, is passed as first argument to the call of SESEuPy. A second argument, which is optional and not shown in Figure 2.1, specifies whether the modelfolder, which is created for a simulation run, shall be deleted after modeling. It can be "True" or "False". If it is "True", the modelfolder is deleted after simulation. It defaults to "False". There are two models simulated in parallel processes (see different process IDs). The execution run returns the value 0 for both processes, so the simulation was successful. Finally, the results of the simulation runs are fetched and a file with the results is created in the directory above the modelfolder(s). If the modelfolders are not deleted, the results can be found in CSV files in the modelfolders as well.

2.4 Building an Executable / Runnable of SESEuPy in Windows and Linux

For Windows and Linux there is the possibility to build an executable / runnable of the program. There is no installation of Python3 / PyQt5 needed on machines for executing these compiled files.

Executable in Windows   In the main program directory of SESEuPy is the file *CreateRunnableWindows.cmd*. This file is a Windows-Command script to execute in order to compile an .exe file of this program. Please note, that the executable is dependent of the processor architecture and the operating system. It is not platform

independent anymore. A folder containing the .exe file and all for the execution necessary libraries is created. Compilation is done with cx_Freeze 4.3.4. The program cx_Freeze needs a configuration file. This configuration is stored in the file *setup-windows.py* the same directory as the file *CreateRunnableWindows.cmd* is located.

Runnable in Linux   In the main program directory of SESEuPy is the file *CreateRunnableLinux.sh*. This file is a Linux Shellscript to execute in order to compile a runnable file of this program. Please note, that the runnable is dependent of the processor architecture and the operating system. It is not platform independent anymore. A .tar.gz archive containing the runnable file and all for the execution necessary libraries is created. Compilation is done with PyInstaller.

3 Examples

Examples can be found in the documentation of the modelbuilder software SES-MoPy. The models built of these can be executed calling the execution unit. Example #1 of the modelbuilder software SESMoPy is discussed here using the different supported simulators.

Simulator Simulink

Simulator OpenModelica

Simulator Dymola

4 Related Work

Some related work regarding model building and executing 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.

MB Model Base.

SES System Entity Structure.

SM simulation model.