Daccosim NG: co-simulation made simpler and faster

José Évora Gómez[1] José Juan Hernández Cabrera[2] Jean-Philippe Tavella[3] Stéphane Vialle[4]
Enrique Kremers[5] Loïc Frayssinet[6]

[1]Monentia SL, Spain, jose.evora@monentia.com
[2]SIANI, Spain, josejuanhernandez@siani.es
[3]EDF Lab Paris-Saclay, France, jean-philippe.tavella@edf.fr
[4]CentraleSupélec - University Paris-Saclay & LRI, France, {Stephane.Vialle}@centralesupelec.fr
[5]EIFER, Germany, enrique.kremers@eifer.org
[6]CETHIL - BHEE, France, loic.frayssinet@insa-lyon.fr

Abstract

This paper introduces the last evolution of Daccosim co-simulation environment, with Daccosim NG developed in 2018. Main features of Daccosim NG are described: enhanced Graphic User Interface and Command-Line Interface, algorithm and mechanism of co-simulation, co-execution shell, software architecture designed for both centralised and distributed architectures, aggregation of a co-simulation graph into a Matryoshka FMU, and declarative language to design large scale co-simulation graphs. A new industrial use case in simulation of energetic systems is also introduced, and first performances of Daccosim NG on multi-core architectures are analysed.

Keywords: co-simulation tool, multithreaded execution, master algorithm, FMI standard, energy system, runtime performance

1 Introduction

The study of Smart Grids, which are intelligent energy systems enhanced by additional communication means and modern IT features, requires a complex analysis of many components considering different aspects. These aspects are amongst others, the demand, production (including renewable), stability of the power grid and flexibility assessment. This is the case for Electricité de France (EDF) and the European Institute For Energy Research (EIFER), where Smart Grids and, more in general, Multi-Energy System analysis are performed through simulations representing the power grids considering multiple aspects. To this end, EDF and EIFER are working in the development of simulation models.

For instance, there are teams working in the modelling and simulation of customers by representing how devices consume energy at their homes: fridges, stoves, washing machines, etc. The analysis of the energy demand of these devices also requires to study thermal dynamics, since many of these devices produce heat or cold. Besides of thermal dynamics, the sociotechnical behaviour of the customers must also be represented as they are the ones who operate the devices. There are also teams developing models for representing thermal gains and loses for houses, buildings, districts, etc. Other teams are dedicated to optimise the grid operation with massive renewable energy and storage units.

Some examples of these kinds of business models are ThermoSysPro, BuildSysPro, PlantSysPro, TelSysPro and EPSL. ThermoSysPro (Hefni et al., 2011) is a library devoted to the modelling and simulation of power plants and energy systems. BuildSysPro (Plessis et al., 2014) is designed to be used in several contexts including building physics research, global performance evaluation, technology development and impact assessment. PlantSysPro is devoted to industrial processes like hot water system. TelSysPro is a new Modelica library able to model the impact of telecommunication networks on complex systems from failure/repair rate of components and stochastic latency.

These teams develop their models using the tool that is the most appropriate according to their work habit or affiliation. There are many tools or programming languages that can be used for developing these models: Anylogic (Borshchev, 2013), Dymola (Elmqvist et al., 1996), Matlab (Guide, 1998), Java (Gosling et al., 2014), Python (Rossum and al., 2007), etc. So, it happens very often that teams want to collaborate by making their models interoperable with others. This is challenging since models are developed in different tools. At this level, the interoperability challenge is double: syntactic and semantic (Hernandez et al., 2016).

The syntax challenge consists in being able to technically communicate models that are developed in different tools. For instance, this problem is equivalent to two people trying to speak when they do not have a common language. The semantic problem has several axis when talking about data exchange between two models: meaning of the words, units that are used, data types, etc. The most common semantic problem in models communication is to have different words to express the refer to the same concept.

The syntactic problem is addressed in FMI (Blochwitz et al., 2011). FMI, the Functional Mock-Up Interface, is a tool-independent standard that supports both model