FMU software component orchestration strategies for co-simulation of building energy systems

Abbass Raad, Vincent Reinbold, Benoit Delinchant and Frédéric Wurtz
Univ. Grenoble Alpes, CNRS, Grenoble INP, G2Elab, 38000 Grenoble, France
Email: firstname.name@g2elab.grenoble-inp.fr

Abstract:

This paper is about system simulation and co-simulation issues such as interoperability. The Functional Mock-up Interface (FMI[1]) which is proposed as an interoperability standard has been studied in the field of the building energy systems to make an effective co-simulation of different models with heterogeneous time constants.
A chaining algorithm based on scalar value exchanges is presented. Our experimental building (PREDIS Smart Building) has been modelled in different software environments for the thermal behaviour of the skin and for the heating and ventilation system. These models are loosely coupled with a same or a different temporal discretization.

The results are about coupling mechanisms and their implementation using FMI standard. Moreover, for advanced coupling strategies, encountered difficulties and FMI specifications lacks are highlights.

I. Introduction

Recently, complex systems have become an important field of study, which attracts the interest of the researchers from different approaches. Energy systems associated to their control strategies, are an example of such a complex system which can benefit from holistic approaches in order to take emerging effects into account. Emerging effect is the case where the global behaviour is greater than the sum of the behaviour of their agents, and it strongly depends on the interactions between these agents.

Simulation is an interesting approach for this study because it allows us to study the dynamic aspects of the system, while highlighting these behaviours and interactions.

Nowadays, it is common to use different models to simulate the behaviour of complex systems and products. These models are rarely interoperable with each other, and do not operate in an aggregate environment that would allow multi-physics simulation of the entire system.

The standard FMI (Functional Mock-up Interface) allows modelling tools (Dassault Dymola, LMS AMESim, ANSYS Simplorer, etc.) to generate C or binary code which represents a dynamic system model that can be fully integrated with other environmental modelling and simulations [1][2]. This is a standard "black box" representation of dynamic models, regardless of the tools or languages permitting their description.

Firstly, a general presentation of simulators issues in the building field will be presented. Then, the paper focuses on the FMI standards as a solution to the interoperability problem. In the fourth part, we will present tools and the chaining algorithm in order to perform a simulation and a co-simulation using FMUs through a python interface. Finally, a simulation of a building, including ventilation and heating system will be presented as a use-case.

II. Issues related to simulators

• The multiplicity of simulation tools:

In the area of Intelligent Building, simulation is becoming increasingly diverse and heterogeneous involving multiple components from different engineering fields. Hundreds of simulation tools have been developed to help architects, office designers of studies and researchers across the world and follow in the different phases of the project of a building. The US Department of Energy has identified[2] more than 400 simulation tools related to the evaluation of energy efficiency, renewable energy and sustainable development in the building sector.

[1] https://www.fmi-standard.org/

[2] http://apps1.eere.energy.gov/buildings

A proliferation of tools implies a broader coverage of diversity necessary models, but does not necessarily mean greater modelling ease. On the contrary, this variety represents a barrier at the time of selection of tools. Indecision problems in the selection can be discussed.

• The non-suitability of simulation tools to new needs:

The building is a constantly evolving area which may imply that tools widely used at a given time, may be discharged later because of their inability to follow developments in the sector.

• The specialization of simulation tools:

Many simulation tools have been developed in the building sector. However, there is no tool that is able to do everything itself. Each tool is generally dedicated to specific applications.

• Needs interoperability between building simulation tools:

We can summarize the issues related to building simulators as:

- A sector in constant evolution imposes to dynamically meet the requirements and needs derived therefrom.

- Various capacities and at the same time limited to specialties and data project phases.

To overcome the limitations of specialization of existing simulation tools to meet the constantly changing requirements of users, it becomes necessary to use interoperability solutions to improve the modularity of tools (for import and export features) and / or to provide a collaborative work covering the simulation needs of the building system.

In the following, the paper focuses on the standard Functional Mock-up Interface as an interoperability solution.

III. The standard Functional Mock-up Interface

Functional Mock-up Interface (FMI) is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of xml-files and compiled C-code. The first version, FMI 1.0, was published in 2010, followed by FMI 2.0 in July 2014 [2].

From November 2011, several simulation tools have supported the FMI component is in import, export or both. An updated list of these tools and details can be found in the web pages of FMI[3].

To make a system simulation, FMI creates the virtual system assembled from a set of models, each representing a sub part of the system and each possessing its own behavioural laws.

FMU (Functional Mock-up Unit), a component that implements the interface FMI, is a compressed file (.fmu) that contains the XML description (variable names, orientations, types, etc.) of the model and its implementation in binary code.

Note that the FMI standard is still in development and this can lead to simulation bugs depending on the software implementation of the norm.

1. *FMU Model Exchange (FMU ME)*

This is a dynamic system model described by algebraic differential equations, discrete time equations, or state machines. These models can be very complex and require significant computing resources, but can also be used in real time in embedded control systems for micro-processors.
The FMU ME uses the simulator solver to run the simulation (c.f. Figure 1). It must be coupled with this solver to integrate in time the model.

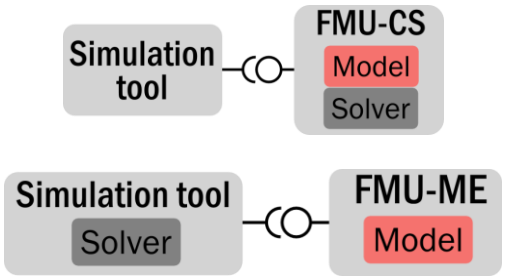

Figure 1 FMU-ME and FMU-CS

2. *FMU Co-Simulation (FMU CS)*

The FMU CS contains its own solver that will be built when generating the tool (c.f. Figure 1). The advantage of this model is to combine two or more simulation tools in a co-simulation environment. The exchange of data between the subsystems is limited to "Communication Points". Between two “Communication Points”, the

[3] https://www.fmi-standard.org/tools

subsystems are solved independently from each other by their individual solver.
In a master-slave concept, slaves simulate sub-problems while the master is responsible for the coordination of the overall simulation and data transfer [3][5][6]. To unify the interface between the master and slave, the FMU CS was developed. The master algorithm controls the exchange of data between subsystems and the synchronization of all the slaves of simulation solvers [6].

IV. Python library and chaining algorithm

1. Python library : PyFMI[4]

PyFMI is a python package dedicated to the FMI. PyFMI is used to load and introspect components of the FMU (ME and CS). A wrapper for the JModelica.org simulation software allows a simple simulation of these FMUs.

2. Implementation in Python

Two commands from the PyFMI package can make the simulation of an FMU :

- The command "Model.simulate()" : This method allows the simulation of the model, with the default solver Assimulo, but can also be connected to other simulation algorithms. The simulation method for MODEL-EXCHANGE models and CO-SIMULATION (FMUModelME / FMUModelCS) has configuration parameters: start_time, final_time, input, algorithm = 'AssimuloFMIAlg' and other options.

- The command "Model.do_step()" : This method is specific for the co-simulation of several FMUs. It has configuration parameters: current_t, step_size, new_step. The "Model.set() " allows to introduce the values of the inputs.

Main python commands for simulation:

```
#FMU library
from pyfmi import load_fmu #FMU command to load fmu file into FMU Object Model

# FMU loading
model = load_fmu(fmu_name)

# Define input variable
u_traj = N.transpose(N.vstack((t_input,u_output)))
input_object = ('u', u_traj)

#Simulation of a ‘model’
res = model.simulate(final_time=t_end, input=input_object)

# Get ‘x1’ and ‘x2’ outputs of the model.
X1 = res['x1']
X2 = res['x2']
t = res['time']
```

Main python commands for co-simulation:

```
#FMU library
from pyfmi.fmi import FMUModelCS1   #FMU CS Object Model
 # FMU loading
model_A = FMUModelCS1('./models_CS/model_A_CS.fmu')
model_A.initialize()

# Set input variable: FMU_CS allow only constant input (no temporal)
model_A.set("x2", X2)

#Simulation of a ‘model’ (FMUModelCS1 class) for a synchronisation step.
res = model_A.do_step(current_t=t,  step_size=hStep, new_step=True)

# Get ‘x1’ output of the model.
X1_A = model_A.get('x1')
```

Note that the *model.simulate()* method is only used in a simulation context. Indeed, in a co-simulation context, it can lead to an increase of the computation time. As a result, for a co-simulation, the *model.doStep()* method is preferred .

We have made available a complete tutorial on Dimocode platform[5]. Those methods are used in the following.

[4] http://www.jmodelica.org/page/4924

[5] http://www.v3.dimocode.fr > Composants MUSE > FMU Components > "TUTORIAL for using FMU-CS with Python"

3. *Chaining algorithm for co-simulation (loosely coupled)*

Chaining algorithm ensures low coupling between two subsystems, the entry of a model at a given time is the output of another model at the previous time [3][4]. The consistency of the coupling is then never checked but convergence is generally obtained with small enough steps. Figure 2 shows the steps of a weak coupling chaining two subsystems (SS).

At the first time step, the initial solution of the sub-system 2 is exchanged to the sub-system 1 in order to simulate the first sub-system (step 2). The solution of the sub-system 1 is then exchanged to the sub-system 2 (step 3) for the simulation of the sub-system 2 (step 4). In this case, it simulates the subsystem 2 for 4 time-steps.

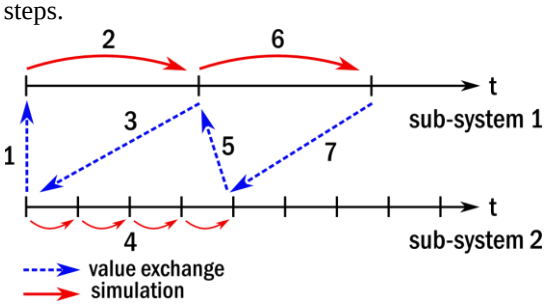

Figure 2 : Chaining Algorithm

In the following, the chaining algorithm is used for a co-simulation of building models.

V. Application to the building simulation

1. *Description of the PREDIS platform*[6]

The platform PREDIS / MHI is dedicated to teaching, research and industrial innovation in the field of intelligent energy management of buildings. Established on 2500 m^2 of premises INPG school ENSE3 France (School of energy, water and environment), the platform fits within PREDIS technological and scientific platform. Its vocation is to make available to all players in the energy of a training and research tool based on technology demonstrators developed through a strategy of alliances and partnerships with industry and local authorities. This platform aims at supporting work on the subject of energy efficiency at the scale of a building or territory, efficiency and safety of power distribution networks, taking into account the diversity of sources and the ability of users to sell their electricity production.

2. *Sample of Models used to export FMUs:*

Several tools[7] are compatible with the FMI interface at Export/Import for both components FMU, Model-Exchange (ME) and co-simulation (CS).
Example: JModelica, Dymola, LMS AMESim, EnergyPlus, CATIA, NI LabVIEW, Ptolemy II, etc.

Here is some model PREDIS exported as FMU:

a. *PREDIS Building (LMS AMESim)*

It models the thermal and aeraulics parts of the PREDIS envelope:
Inputs: External temperature (T_{ext}), solar contributions (P_{sun}), and internal gains ($P_{heating}$).
Outputs: internal temperature ($T_{internal}$).
Note that the internal gain is only represented by the heating power for simplicity.

b. *HVAC (LMS AMESim)*

It models the heating ventilation and air conditioning (HVAC) of the platform (cf. Figure 3).
Inputs: Temperature of injected fresh air & return air, air flows, heat exchanger rotation speed.
Outputs: temperature of fresh air blown & stale air.

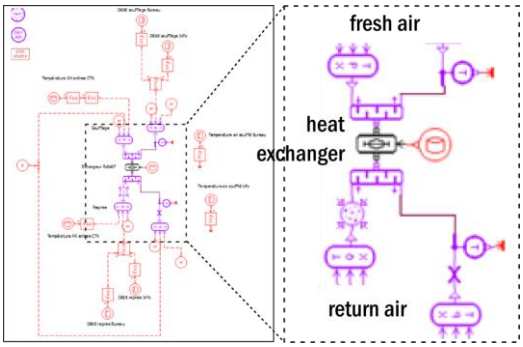

Figure 3 : HVAC modelling (LMS AMESim)

c. *Heating System (Dymola)*

This model represents a type of hysteresis heating to regulate the temperature around 20°C.
Inputs: internal temperature ($T_{internal}$).
Outputs: Heating power ($P_{heating}$).

[6]http://predis.grenoble-inp.fr/smartbuilding

[7] https://www.fmi-standard.org/tools

3. Simulation of the HVAC

The HVAC model was simulated in LMS AMESim to obtain the reference values.
The FMU generated from this model was simulated using the package PyFMI.

The figures below show the validation of this method of simulation. Indeed, the deviations of the results from the simulation under AMESim are almost zero (see Figure 4 and Figure 5).

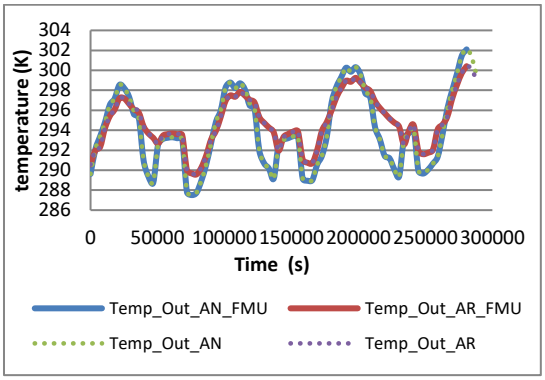

Figure 4 : Comparison of Python _ PyFMI simulation results with AMESim

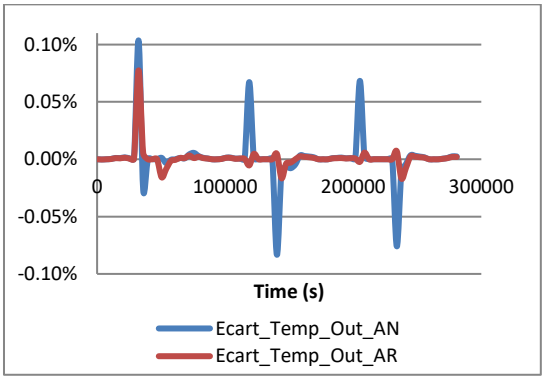

Figure 5 : Deviation of simulation results in %

4. Co-Simulation of the building and the heating system

An application of the co-simulation algorithm (chaining) is applied on the FMUs models "PREDIS Building" and "Heating" generated from AMESim and Dymola. This will highlight the interest of the co-simulation to couple models from different tools (see Figure 6).

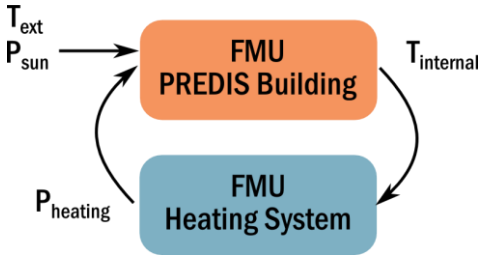

Figure 6 : Co-simulation of FMUs PREDIS Building and Heating

The Figure 7 shows the results of this co-simulation which are coherent and meet expectations. The temperatures are well controlled around 20 ° C.

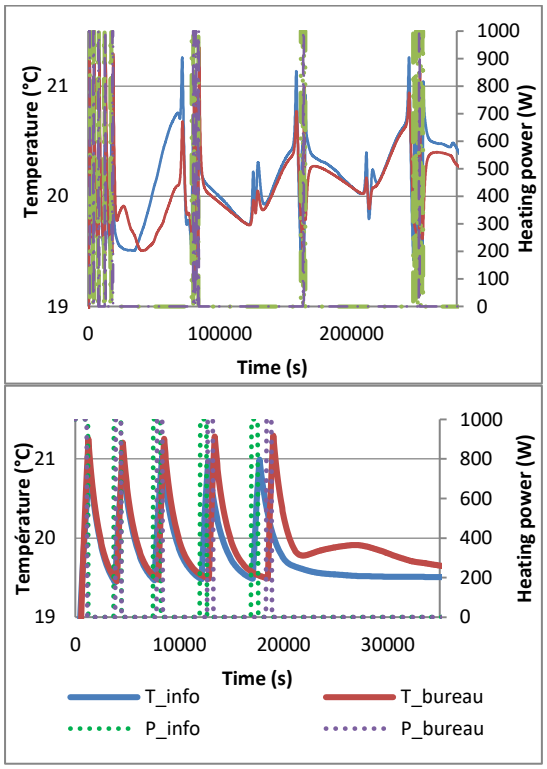

Figure 7 : Results of co-simulation PREDIS Building and Heating

The choice of the time step is the key point for the FMU co-simulation in terms of time consumption and accuracy.

VI. Conclusion and Perspectives

1. General conclusion

The standard interface FMI allows, using these FMUs components, to perform co-simulations between different environments and tools of different kind (thermal, electrical, hydraulic ...), which offers the possibility to benefit from the expertise of every tool in its field and to couple them together.

The variety of tools and simulation environments that integrate these components makes FMUs more useful.

2. Perspectives for co-simulation algorithm

Co-simulation developed in the previous algorithm is simple, smarter co-simulations will be our goals in the following:

- Advanced orchestration strategies [7]:

In this approach, the concept of the event will be taken into account, which refines the co-simulation. The FMU specifications are currently limited to simple strategies such as chaining. Event-based strategies are confronted with the available capacity in the components (via the standard it may be insufficient, but mostly because of the tools that generate FMU with the bare minimum).

- Waveform Relaxation Method – WRM [8]:

The idea is to combine several heterogeneous systems (different dynamics), the coupling being performed by an iterative method on the waveforms (see Figure 8). Each system is solved in time throughout the time domain considered, and its solution, the entire waveform source used for other systems.

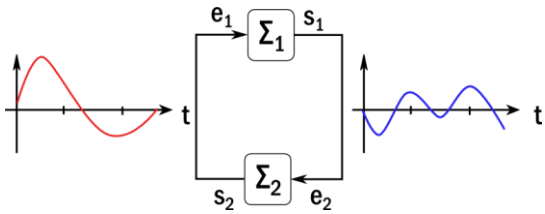

Figure 8 : Wave Form Relaxation Method

VII. Acknowledgment

Alexandre LEBLOND, Engineer from LMS Imagine for his help in modelling using AMESim software. And the French National Agency : ANR Precision. « VILLES ET BATIMENTS DURABLES »

VIII. Bibliography

[1] Blochwitz T., Otter M., Arnold M., Bausch C., Clauß C., Elmqvist H., Wolf S. The functional mockup interface for tool independent exchange of simulation models. In 8th International Modelica Conference, 2011, Dresden (pp. 20-22).

[2] Blochwitz T., Otter M., Åkesson J., Arnold M., Clauss C., Elmqvist H., Viel A. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In 9th International Modelica Conference, 2012.

[3] Wetter, M. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. Journal of Building Performance Simulation, Taylor & Francis, 2011, 4, 185-203.

[4] Sicklinger S., Belsky V., Engelmann B., Elmqvist H., Olsson H., Wüchner R., & Bletzinger K. U. Interface Jacobian-based Co-Simulation. International Journal for Numerical Methods in Engineering, 98(6), 418-444, 2014.

[5] Broman D., Brooks C., Greenberg L., Edward A. L., Masin M., Tripakis S., Wetter M. Determinate Composition of FMUs for Co-Simulation, In Proceedings of the 13th International Conference on Embedded Software (EMSOFT), Montreal, Canada, September 29 - October 4, 2013.

[6] Bastian J., Clauß C., Wolf S., Schneider P. Master for co-simulation using FMI. In 8th International Modelica Conference, Dresden, 2011.

[7] Gaaloul S., Le X. H. B., Delinchant B., Wurtz F., Ploix S. Architecture à composants de co-simulation appliquée au couplage de la thermique du bâtiment au comportement de l'usager. In Journées AUGC et IBPSA 2012.

[8] Lelarasmee E., Ruehli A. E., Sangiovanni-Vincentelli A. L. The waveform relaxation method for time-domain analysis of large scale integrated circuits. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 1(3), 131-145, 1982.