Title
Functional Mock-Up Unit Import in EnergyPlus For Co-Simulation

Permalink
https://escholarship.org/uc/item/02d8r3d0

Author
Nouidui, Thierry Stephane

Publication Date
2014-04-21
Functional Mock-Up Unit Import in EnergyPlus For Co-Simulation

Authors, Thierry Stephane Nouidui, Michael Wetter, Wangda Zuo

Environmental Energy and Technologies Division

August 2013
Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association, Chambéry, France, August 26-28
DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or The Regents of the University of California.
Acknowledgements

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Program, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.
FUNCTIONAL MOCK-UP UNIT IMPORT IN ENERGYPLUS FOR CO-SIMULATION

Thierry Stephane Nouidui1, Michael Wetter1, Wangda Zuo2*
1Lawrence Berkeley National Laboratory, Berkeley, U.S.A.
2University of Miami, Miami, U.S.A.

ABSTRACT

This paper describes how to use the recently implemented Functional Mock-up Unit (FMU) for co-simulation import interface in EnergyPlus to link EnergyPlus with simulation tools packaged as FMUs. The interface complies with the Functional Mock-up Interface (FMI) for co-simulation standard version 1.0, which is an open standard designed to enable links between different simulation tools that are packaged as FMUs. This article starts with an introduction of the FMI and FMU concepts. We then discuss the implementation of the FMU import interface in EnergyPlus. After that, we present two use cases. The first use case is to model a HVAC system in Modelica, export it as an FMU, and link it to a room model in EnergyPlus. The second use case is an extension of the first case where a shading controller is modeled in Modelica, exported as an FMU, and used in the EnergyPlus room model to control the shading device of one of its windows. In both cases, the FMUs are imported into EnergyPlus which models the building envelope and manages the data-exchange between the envelope and the systems in the FMUs during run-time.

INTRODUCTION

EnergyPlus (Crawley et al., 2001) is a well established whole building energy simulation program which has been used for various applications such as building design (Wang et al., 2009) and fault detection and diagnostics (Pang et al., 2012). EnergyPlus was primarily developed for annual building energy simulations as opposed to investigating controls performance of HVAC systems or detailed daylighting simulations.

To extend the capability of EnergyPlus for building simulation, EnergyPlus has been linked for co-simulation with various programs from other domains, such as multizone airflow network (Huang et al., 1999), Computational Fluid Dynamics (CFD) (Zhai and Chen, 2005), and HVAC system and controls (Wetter, 2011). The coupling of EnergyPlus with these external programs were realized either by creating specific interfaces for the external programs in the EnergyPlus source code (Huang et al., 1999, Zhai and Chen, 2005) or using the Buildings Controls Virtual Test Bed (BCVTB) middleware (Wetter, 2011). The major limitation of the former is the lack of reusability since the interface is only for a specific program. For the latter, having an additional transaction layer into the communication increases the complexity of the co-simulation. A more promising approach to link EnergyPlus with external programs would be to communicate through a standardized interface where all simulation tools can communicate directly using the same standard. As a result, the direct link and the elimination of the transaction layer will facilitate and enhance the co-simulation of EnergyPlus with external simulation tools. This contribution describes the standard interface which has been implemented in EnergyPlus. This interface is based on the Functional Mockup Interface (FMI) version 1.0, an open standard designed to link simulation programs during runtime (MODELISAR-Consortium, 2008-2012a).

FMI Related Work

Recently, the Energy Conservation in Buildings and Community Systems program of the International Energy Agency has approved a 5-year project (Annex 60) to develop the next generation computational tools based on the Modelica (Mattsson and Elmqvist, 1997) and the FMI standards. An FMI for co-simulation has been implemented in WUFI Plus (Pazold et al., 2012), a whole building simulation program, to extend its capabilities to be linked to external Modelica based simulation programs. The Institute for the Sustainable Performance of Buildings has been developing a web-based eLearning tool in which a Web interface communicates with a Functional Mock-up Unit (FMU) for co-simulation that simulates heat transfer through building envelope, HVAC systems, control systems and equipment faults of a building, and visualizes this response at an interactive web browser through WebGL (Deringer et al., 2012). Work is in progress at UC Berkeley and Lawrence Berkeley National Laboratory to integrate an FMU import interface in Ptolemy II (Brooks et al., 2007).

*The work was performed while the author was at the Lawrence Berkeley National Laboratory.
Work is also in progress to export EnergyPlus as an FMU for co-simulation.

**OVERVIEW OF FMI AND FMU**

The FMI is the result of the Information Technology for European Advancement (ITEA2) project MODELISAR. It is a tool independent and non-proprietary standard to support both model exchange and co-simulation of dynamic models using a combination of XML-file, C-header files, and C-code in source or binary form. The FMI standard version 1.0 consists of three parts:

1. The first part is FMI for model exchange. This part of the standard specifies how a modeling environment can generate C-code of a dynamic system model that can be utilized by other modeling and simulation environments (MODELISAR-Consortium, 2008-2012c).

2. The second part is FMI for co-simulation which provides an interface standard for coupling two or more simulation tools in a co-simulation environment (MODELISAR-Consortium, 2008-2012b). Co-simulation refers in this context to a technique that allows individual component models described by differential algebraic or discrete equations to be simulated by different simulation programs running simultaneously and exchanging data during run-time.

3. The third part is FMI for Product Lifecycle Management (PLM) which provides a generic way to handle FMI related data needed in a simulation of systems in a PLM system (MODELISAR-Consortium, 2008-2012d).

The FMI standard defines a set of C-functions that are needed to perform co-simulation with other simulation programs. The FMI also defines an XML-file (model description file) which contains all information required by the importing tool to inquire information about the model and its interface variables. Tools that support FMI can import and/or export a simulator for co-simulation.

Since the interface described in this contribution leverages the FMI for co-simulation Application Programming Interface (API), FMI and FMU will refer from now on to the second part of the FMI standard. For the co-simulation, the FMU must contain the model and its solver.

**FMU IMPORT INTERFACE IN ENERGYPLUS**

**Implementation**

**Pre-processing**

To support the co-simulation of EnergyPlus with FMU, we developed a utility called FMUParser. This utility is a code written in C. It includes Expat (Clark et al., 2011) which is an XML parser library written in C. The FMUParser can facilitate the set-up of the EnergyPlus input file by extracting relevant information from the FMU and writing it in a temporary EnergyPlus input file.

Figure 1 shows the workflow for pre-processing. First, the FMUParser parses an FMU file (i.e. xxx.fmu) and generates a temporary EnergyPlus input file (i.e. xxxtmp.idf). The temporary EnergyPlus input file is not complete as it just contains information related to the FMU, such as the name of the FMU and properties of the FMU variable including their names and input/output types. The user then needs to manually copy the FMU information from xxxtmp.idf into the EnergyPlus input file xxx.idf. The user finally needs to modify the xxx.idf file to link the FMU variables with EnergyPlus variables.

![Figure 1 Workflow for the pre-processing.](image-url)

In the pre-processing step, the FMUParser will be called with the command option --printidf. This will request the parser to unzip the FMU, parse the XML-file with the model description of the FMU and write the FMU information in a format of the EnergyPlus input file (*.idf). The parser will check if all the required fields from the FMU (see next section for details) in the *.idf file are correctly specified. If the check succeeds, the parser will successfully close.
Suppose we have a system with two simulation programs. Simulation program 1 is the slave simulation program, which is packaged as an FMU for co-simulation; and simulation program 2 is EnergyPlus, which is the master simulation program and imports the FMU for co-simulation. Each program solves an initial-value ordinary differential equation that is coupled to the differential equations of the other program.

Let $N \in \mathbb{N}$ denote the number of time steps and let $k \in \{0, \ldots, N-1\}$ denote the time steps. We will use the subscripts 1 and 2 to denote the simulation program 1 and 2, respectively.

Then programs 1 and 2 compute, for $k \in \{0, \ldots, N-1\}$ the sequence

$$\begin{align*}
x_1(k+1) &= f_1(x_1(k), x_2(k)), \\
x_2(k+1) &= f_2(x_2(k), x_1(k))
\end{align*}$$

with initial conditions $x_1(0) = x_{1,0}$ and $x_2(0) = x_{2,0}$.

To advance from time $k$ to $k+1$, each program uses its own integration algorithm. At the end of the time step, program 1 sends its new state $x_1(k+1)$ to program 2, and receives the updated state $x_2(k+1)$ from program 2. The same procedure is done with the program 2. Program 2, which is the master simulation program, manages the data-exchange between the two programs.

In comparison to numerical methods of differential equations, this coupling scheme resembles an explicit Euler integration that solves an ordinary differential equation with specified initial values

$$\begin{align*}
\frac{dx}{dt} &= h(x), \\
x(0) &= x_0
\end{align*}$$

on the time interval $t \in [0, 1]$. The integration sequence is as follows:

**Step 0:** Initialize counter $k=0$ and number of steps $N \in \mathbb{N}$.

Set initial state $x(k) = x_0$ and set time step $\Delta t = 1/N$.

**Step 1:** Compute new state $x(k+1) = x(k) + h(x(k)) \Delta t$.

Replace $k$ by $k+1$.

**Step 2:** If $k=N$ stop, else go to Step 1.

The above scheme does not require each simulation tool to use explicit Euler for its internal time-stepping; the analogy to explicit Euler applies only to the data exchange between programs. In the situation where the differential equation is solved using co-simulation, the above algorithm becomes
Step 0: Initialize counter $k=0$ and number of communication steps $N \in \mathbb{N}$.

Set initial state $x_1(k) = x_{1,0}$ and $x_2(k) = x_{2,0}$.

Set the communication time step $\Delta t = 1/N$.

Step 1: Compute new states

$x_1(k+1) = x_1(k) + f_1(x_1(k), x_2(k)) \Delta t$, and

$x_2(k+1) = x_2(k) + f_2(x_2(k), x_1(k)) \Delta t$.

Replace $k$ by $k+1$.

Step 2: If $k=N$ stop, else go to Step 1.

In this algorithm, there is no iteration between the two simulation programs within one time step.

It is worth mentioning that the current implementation of the FMU import interface assumes that there are no direct dependencies between input and output of any FMU. Moreover, the coupling scheme used in the implementation is based on loose coupling which, compared to strong coupling, is easier to implement, requires shorter synchronization time steps, is numerically more robust, and computed faster in the experiments reported in (Trcka et al., 2009).

COUPLING AN HVAC SYSTEM MODEL, IMPLEMENTED IN AN FMU, WITH A ROOM MODEL IN ENERGYPLUS

In this example, a room with its HVAC system are simulated in EnergyPlus version 7.2. The building envelope of the room is modeled in EnergyPlus whereas the HVAC system is implemented in Modelica. Figure 2 shows the system configuration with the HVAC system modeled in an FMU and the room model modeled in EnergyPlus.

The HVAC system model has been developed using component models of the Modelica Buildings library (Wetter et al., 2013). This model computes sensible and latent heat gain required to maintain a room set point temperature and humidity. Figure 3 shows the Modelica implementation of the HVAC system.

This FMU is then imported into EnergyPlus using the FMU import interface. The FMU needs as input the outdoor dry-bulb ($T_{DryBul}$) temperature, outdoor air relative humidity ($\text{outRelHum}$), the room dry-bulb temperature ($T_{Room}$) and the room air relative humidity ($\text{rooRelHum}$). The outputs of the FMU are the latent ($\text{QLatent}$) and sensible ($\text{QSensible}$) heat transported across the thermodynamic boundary of air inlet and outlet of the thermal zone.

In this example, we use the ExternalInterface:FunctionalMockupUnitImport:To:Schedule to send the latent and sensible heat gain from the FMU to EnergyPlus. We also use the ExternalInterface:FunctionalMockupUnitImport:From:Variable object to send outdoor dry-bulb temperature, outdoor air relative humidity, room dry-bulb temperature and room air relative humidity from EnergyPlus to the FMU. The data exchange between the FMU and EnergyPlus occurs at the zone time step of EnergyPlus.
The following section gives a step-by-step instruction on how to set-up and run the simulation in EnergyPlus.

**Pre-processing – Creating the EnergyPlus input file**

An FMU comes along with a model description file, which contains among other information the input and output variables of the FMU. Figure 5 shows a snippet of a section of the model description file of MoistAir.fmu.

```
<ScalarVariable>
  name="TDryBul"
  valueReference="352321536"
  causality="input"
</ScalarVariable>
```

```
<ScalarVariable>
  name="QSensible"
  valueReference="355544320"
  causality="output"
</ScalarVariable>
```

Figure 5 modelDescription.xml of FMU (MoistAir.fmu).

The model description file can contain more than thousand lines of information depending on the complexity of the model, but we are just interested in the input and output variables that must be mapped to the EnergyPlus variables here. We use the FMUParser to extract the relevant information from the FMU by calling from a DOS or Unix/Linux shell the command:

```
parser --printidf MoistAir.fmu
```

This calls the parser to process the FMU and generate a temporary idf file as shown in Figure 6.

```
ExternalInterface,
  FunctionalMockupUnitImport;  !- Name of External Interface
ExternalInterface:FunctionalMockupUnitImport,
  MoistAir.fmu,  !- FMU Filename
    ;  !- FMU Timeout in milli-seconds
    ;  !- FMU LoggingOn Value
```

The first object in the temporary input file instructs EnergyPlus that the FMU import interface should be activated. The second object specifies the FMU that will be imported in EnergyPlus. The next four objects are used by the `ExternalInterface` to read data from EnergyPlus and send data to the inputs of the FMU. The last two objects are used by the `ExternalInterface` to get data from the FMU output variables and write them to EnergyPlus.


Figure 6 Temporary idf input file generated by the FMUParser.

The next step in the pre-processing consists of:
- copying the temporary idf file into the full idf input file, and
- modifying the full idf file to link the FMU variables with EnergyPlus variables.

The idf excerpts below shows how the objects look like in the complete EnergyPlus input file.

To activate the external interface, we use the `ExternalInterface`, `FunctionalMockupUnitImport`, and `Modelica.Blocks.Interfaces.RealInput/RealOutput` objects.

To define the FMU that will be linked to EnergyPlus, we use the `ExternalInterface:FunctionalMockupUnitImport` object.

To enter output variables from which the external interface reads data from and sends data to FMUs, we use the `Modelica.Blocks.Interfaces.RealInput/RealOutput` objects.

The output variables that will be mapped to the input of the FMU also need to be specified in the idf file:

```
Output:Variable, Environment, Outdoor Dry Bulb, TimeStep; Output:Variable, ZONE ONE, Zone Mean Air Temperature, TimeStep; Output:Variable, Environment, Outdoor Relative Humidity, TimeStep; Output:Variable, ZONE ONE, Zone Air Relative Humidity, TimeStep;
```

To enter schedules to which the external interface writes, we use

```
ExternalInterface:FunctionalMockupUnitImport:To:Schedule, FMU_CheEqSen_ZoneOne, EnergyPlus Variable Name Any Name, Schedule Type Limits Names MoistAir.fmu, FMU Filename Model1, FMU Instance Name Qsensible, FMU Variable Name 0; Output:Variable, FMU_CheEqLat_ZoneOne, EnergyPlus Variable Name Any Name, Schedule Type Limits Names MoistAir.fmu, FMU Filename Model1, FMU Instance Name QLatent, FMU Variable Name 0;
```

This completes the configuration that is required to simulate EnergyPlus with the FMU.

Co-simulation

In the co-simulation process, EnergyPlus which is the co-simulation master calls the methods implemented and stored in the shared library. The main steps involved in the co-simulation processes are

- unpacking the FMU,
- creating an instance of the FMU,
- initializing the FMU,
- setting the input variables of the FMU,
- getting the output variables of the FMU,
- conducting the time integration,
- terminating and freeing the memory of the FMU.

Figure 7 shows how the room dry-bulb temperature in the EnergyPlus model changes with the sensible and latent heat gains, which are computed in the HVAC system model packaged as an FMU.

COUPLING A HVAC SYSTEM AND A SHADING CONTROLLER, IMPLEMENTED IN FMUS WITH A ROOM MODEL IN ENERGYPLUS

In this example, a shading device is added to one of the window of the room model discussed before. The shading device is controlled by a finite state machine. The shading controller is developed in Modelica and exported as an FMU.

Figure 8 shows the new system configuration which consists of EnergyPlus which is linked to two FMUs. The inputs of the shading controller’s FMU are the room dry-bulb temperature (TRoom) and the solar irradiation (ISolExt) that is incident on the window. The output of the FMU is the shading actuation signal (yShade).

Figure 8 System with a HVAC system and a shading controller in FMUs and a room model with a shading device modeled in EnergyPlus.
Figure 9 shows the finite state machine which switches between the states nightShadeDeployed, noShade and dayShadeDeployed if guards defined in the transitions evaluate to true.

![Finite state machine](image)

Figure 9 Finite state machine of the shading controller.

Figure 10 shows the Modelica implementation of the finite state machine.

![Modelica implementation](image)

Figure 10 Modelica implementation of the shading controller system model.

We specify these output variables in the idf file.

```idf
Output:Variable,
  ZONE SUBSURFACE 1 EAST WINDOW,  !- Key Value
  Surface Ext Solar Incident,       !- Variable Name
  TimeStep;                         !- Reporting Frequency

Output:Variable,
  ZONE ONE,                        !- Key Value
  Zone Mean Air Temperature,       !- Variable Name
  TimeStep;                        !- Reporting Frequency
```

To write data from the external interface to an EnergyPlus EMS variable, we use the following item in idf file:

```idf
ExternalInterface:FunctionalMockupUnitImport:To:Variable,
  Shade_Signal,                !- EnergyPlus Variable Name
  ShadingController.fmu,       !- FMU Filename
  Model1,                     !- FMU Instance Name
  yShade,                     !- FMU Variable Name
  1;                          !- Initial Value
```

which declares a variable with name yShade that can be used in an Energy Runtime Language (Erl) program to actuate the shading control of the window ZONE SUBSURFACE 1 EAST WINDOW as follows:

```erl
EnergyManagementSystem:Program,          !- Name
  Set_Shade_Control_Signal = 6*yShade,   !- Program Line

EnergyManagementSystem:Actuator,        !- Name
  Shade_Signal, ZONE SUBSURFACE 1 EAST WINDOW, !- Actuated Component Unique Name
  Window Shading Control, Control Status; !- Actuated Component Type

This completes the configuration that is required to simulate EnergyPlus with the two FMUs.

Figure 11 shows how the shading controller sets the night shade to be active during time when the incident solar radiation on the windows is smaller than the threshold of 5 W/m², which is defined in the transition of the shading controller’s model.
CONCLUSION & DISCUSSION
The FMU import interface developed in EnergyPlus extends the capability of EnergyPlus to import any simulation program, which is exported as an FMU for co-simulation. The FMI approach is very promising since it standardizes the co-simulation and model exchange between simulators. Future work should include the evaluation of the performance of the co-simulation approach versus mono-simulation where the entire simulation is done in a single environment.

ACKNOWLEDGEMENT
This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

REFERENCES


Pazold, M., Burhenne, S., Radon, J., Herkel, S. & Antretter, F. 2012. Integration of Modelica models into an existing simulation software using FMI for Co-Simulation. 9th International Modelica Conference, September 03 2012 Munich, Germany.


