LBNL-6413E

ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

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

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 Nouidui[1], Michael Wetter[1], Wangda Zuo[2*]
[1]Lawrence Berkeley National Laboratory, Berkeley, U.S.A.
[2]University 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 work was performed while the author was at the Lawrence Berkeley National Laboratory

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).

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.
The exported simulator or model is called an FMU. An FMU is distributed in form of a zip file. This file may contain

- The FMI model description file.
- The C sources of the FMU, including the needed run-time libraries used in the model, and/or binaries for one or several target machines
- Additional FMU data (such as tables, diagram) in FMU specific file formats.

There are currently more than 34 modeling and simulation environments which support or plan to support FMI (MODELISAR-Consortium, 2008-2012a).

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.

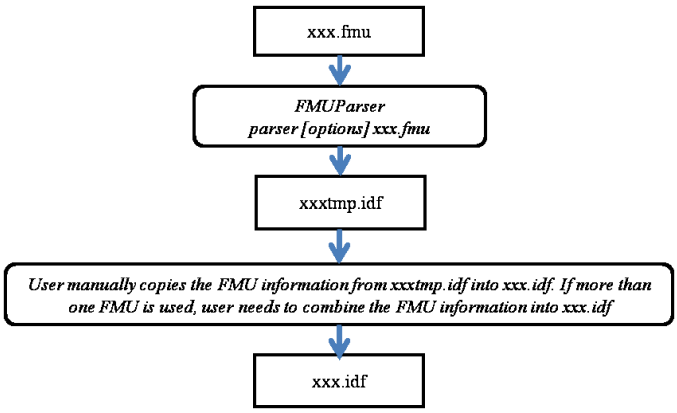

Figure 1 Workflow for the pre-processing.

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.

Otherwise, the parser will stop with an error message. The FMUParser is distributed with EnergyPlus and can be found in the PreProcess folder (FMUParser) of the EnergyPlus installation.

Co-simulation

For the co-simulation with EnergyPlus, we developed a shared library written in C, which contains all functions needed to interface with FMUs. The primary functions in the shared library are called at runtime to instantiate, initialize, set, and get values of defined variables and execute single time steps.

This shared library is accessed by EnergyPlus through the *ExternalInterface* module (EnergyPlus, 2012a). The *ExternalInterface* is a module in EnergyPlus and was originally developed to support the co-simulation of EnergyPlus with the BCVTB. To support the interface with FMUs, EnergyPlus has been extended with four new objects. These objects are used to map the input/output signals that are exchanged between the FMUs and EnergyPlus.

The *ExternalInterface* can map to three EnergyPlus input objects called

- `ExternalInterface:FunctionalMockupUnitImport:To:Schedule`
- `ExternalInterface:FunctionalMockupUnitImport:To:Actuator`
- `ExternalInterface:FunctionalMockupUnitImport:To:Variable`.

The `ExternalInterface:FunctionalMockupUnitImport:To:Schedule` can be used to overwrite schedules, and the other two objects can be used in place of Energy Management System (EMS) actuators and EMS variables. The objects have similar functionality as the objects `Schedule:Compact`, `EnergyManagementSystem:Actuator` and `EnergyManagementSystem:GlobalVariable`, except that their numerical value is obtained from the external interface at the beginning of each zone time step, and will remain constant during this zone time step.
The external interface also uses the `ExternalInterface:FunctionalMockupUnitImport:From:Variable` object which maps to EnergyPlus objects `Output:Variable` and `EnergyManagementSystem:OutputVariable` to send data from EnergyPlus to FMUs at each zone time step.

Mathematics of data-exchange

This section describes the algorithm for exchanging data between EnergyPlus and a simulation program packaged as an FMU.

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 $k \in \{0,...,N\}$ 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 $k \in \{0,...,N-1\}$ the sequence

$$x_1(k+1) = f_1(x_1(k), x_2(k)), \quad (1)$$

$$x_2(k+1) = f_2(x_2(k), x_1(k)), \quad (2)$$

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

$$dx/dt = h(x), \quad (3)$$

$$x(0) = x_0, \quad (4)$$

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.

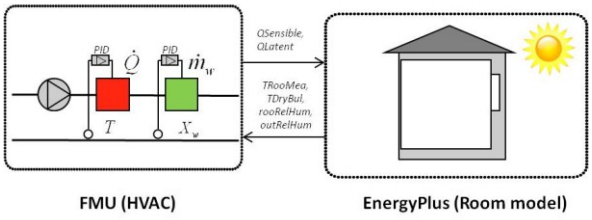

Figure 2 System with an HVAC system modeled in an FMU and a room modeled in EnergyPlus.

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.

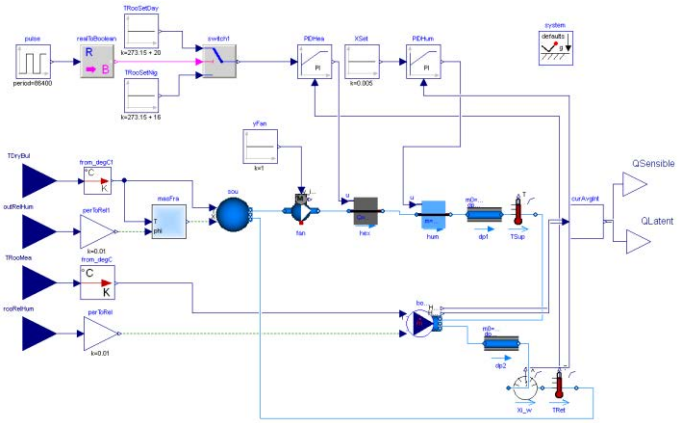

Figure 3 Modelica implementation of the HVAC system model.

The Modelica model of the HVAC system is exported as an FMU with the name of *MoistAir.fmu*. The FMU is an input/output block (Figure 4) which contains the simulation model and exposes inputs, outputs and events indicators of the model which can be interfaced using the FMI API.

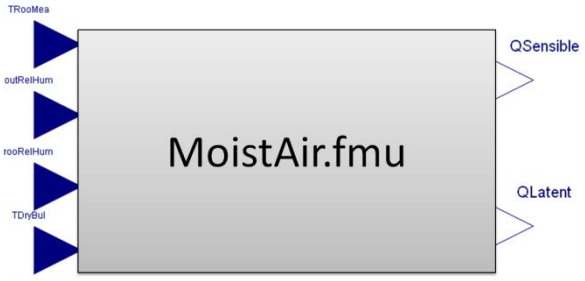

Figure 4 Modelica model of the HVAC system exported as an FMU for co-simulation.

This FMU is then imported into Energylus using the FMU import interface. The FMU needs as input the outdoor dry-bulb (`TDryBul`) temperature, outdoor air relative humidity (`outRelHum`), the room dry-bulb temperature (`TRooMea`) and the room air relative humidity (`rooRelHum`). The outputs of the FMU are the latent (`QLatent`) and sensible (`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*.

```
<?xml version="1.0" encoding="UTF-8"?>
<fmiModelDescription fmiVersion="1.0"
modelName="Buildings.Utilities.IO.BCVTB.Examples.MoistAir"
modelIdentifier="Buildings_Utilities_IO_BCVTB_Examples_Mois
tAir"guid="{814b15aa-dcbb-4adf-b55b-f5f841fafad1"
description="MoistAir" version="1.1" generationTool="Dymola
Version 2012 FD01 (32-bit), 2011-11-22"
  generationDateAndTime="2012-02-16T22:57:53Z"
  variableNamingConvention="structured"
  numberOfContinuousStates="13"
  numberOfEventIndicators="68">
  <UnitDefinitions>
    <BaseUnit
      unit=""/>
    <BaseUnit
      unit="K">
      <DisplayUnitDefinition
        displayUnit="degC"
        offset="-273.15"/>
    </BaseUnit>
  …
  <ScalarVariable
      name="TDryBul"
      valueReference="352321536"
      causality="input">
      <Real
        declaredType="Modelica.Blocks.Interfaces.RealInput"
        min="-273.15"
        start="0.0"/>
    </ScalarVariable>
    …
    <ScalarVariable
      name="QSensible"
      valueReference="335544320"
      causality="output">
      <Real
     declaredType="Modelica.Blocks.Interfaces.RealOutput"/>
    </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.

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 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.

More details on the input fields of the EnergyPlus objects can be found in the Input/Output Reference of EnergyPlus (EnergyPlus, 2012b).

```
ExternalInterface,
  FunctionalMockupUnitImport; !- Name of External Interface

ExternalInterface:FunctionalMockupUnitImport,
  MoistAir.fmu,     !- FMU Filename
   ,                !- FMU Timeout in milli-seconds
   ;                !- FMU LoggingOn Value

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
   ,                !- EnergyPlus Key Value
   ,                !- EnergyPlus Variable Name
   MoistAir.fmu,    !- FMU Filename
   ,                !- FMU Instance Name
   TDryBul;         !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
   ,                !- EnergyPlus Key Value
   ,                !- EnergyPlus Variable Name
   MoistAir.fmu,    !- FMU Filename
   ,                !- FMU Instance Name
   TRooMea;         !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
   ,                !- EnergyPlus Key Value
   ,                !- Energyplus Variable Name
   MoistAir.fmu,    !- FMU Filename
   ,                !- FMU Instance Name
   outRelHum;       !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
   ,                !- EnergyPlus Key Value
   ,                !- Energyplus Variable Name
   MoistAir.fmu,    !- FMU Filename
   ,                !- FMU Instance Name
   rooRelHum;       !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:To:,
   ,                !- EnergyPlus Variable Name
   MoistAir.fmu,    !- FMU Filename
   ,                !- FMU Instance Name
   QSensible,       !- FMU Variable Name
   ;                !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:To:,
   ,                !- EnergyPlus Variable Name
   MoistAir.fmu,    !- FMU Filename
   ,                !- FMU Instance Name
   QLatent,         !- FMU Variable Name
   ;                !- Initial Value
```

Figure 6 Temporary idf input file generated by the FMUParser.

The next step in the pre-processing consists of

- copying the temporary idf information 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

```
ExternalInterface,
  FunctionalMockupUnitImport; !- Name of external interface
```

To define the FMU that will be linked to EnergyPlus, we use

```
ExternalInterface:FunctionalMockupUnitImport,
  MoistAir.fmu,          !- FMU Filename
  15,                    !- FMU Timeout in milli-seconds
  0;                     !- FMU LoggingOn Value
```

To enter output variables from which the external interface reads data from and sends data to FMUs, we use

```
ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    Environment,              !- EnergyPlus Key Value
    Outdoor Dry Bulb,         !- EnergyPlus Variable Name
    MoistAir.fmu,             !- FMU Filename
    Model1,                   !- FMU Instance Name
    TDryBul;                  !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    ZONE ONE,                   !- EnergyPlus Key Value
    Zone Mean Air Temperature,  !- EnergyPlus Variable Name
    MoistAir.fmu,               !- FMU Filename
```

```
    Model1,                     !- FMU Instance Name
    TRooMea;                    !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    Environment,                !- EnergyPlus Key Value
    Outdoor Relative Humidity,  !- EnergyPlus Variable Name
    MoistAir.fmu,               !- FMU Filename
    Model1,                     !- FMU Instance Name
    outRelHum;                  !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    ZONE ONE,                   !- EnergyPlus Key Value
    Zone Air Relative Humidity, !- EnergyPlus Variable Name
    MoistAir.fmu,               !- FMU Filename
    Model1,                     !- FMU Instance Name
    rooRelHum;                  !- FMU Variable Name
```

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,                !- Key Value
    Outdoor Dry Bulb,           !- Variable Name
    TimeStep;                   !- Reporting Frequency

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

Output:Variable,
    Environment,                !- Key Value
    Outdoor Relative Humidity,  !- Variable Name
    TimeStep;                   !- Reporting Frequency

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

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

```
ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
    FMU_OthEquSen_ZoneOne,   !- EnergyPlus Variable Name
    Any Number,              !- Schedule Type Limits Names
    MoistAir.fmu,            !- FMU Filename
    Model1,                  !- FMU Instance Name
    QSensible,               !- FMU Variable Name
    0;                       !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
    FMU_OthEquLat_ZoneOne,   !- EnergyPlus Variable Name
    Any Number,              !- Schedule Type Limits Names
    MoistAir.fmu,            !- FMU Filename
    Model1,                  !- FMU Instance Name
    QLatent,                 !- FMU Variable Name
    0;                       !- Initial Value
```

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 called 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.

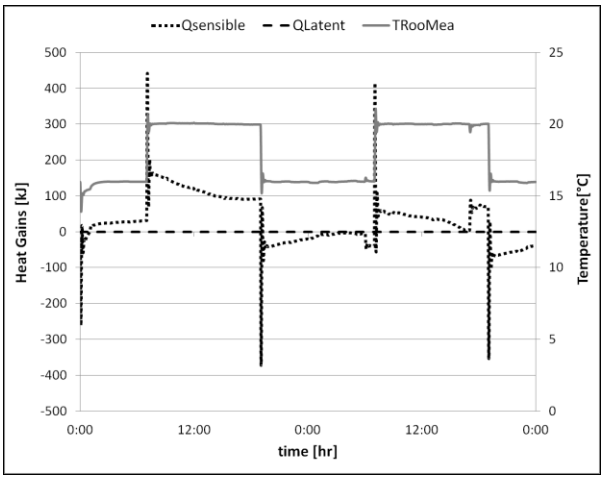

Figure 7 Simulation results showing the sensible heat gain, the latent heat gain, and the room dry-bulb temperature reported by EnergyPlus.

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 (`TRoo`) and the solar irradiation (`ISolExt`) that is incident on the window. The output of the FMU is the shading actuation signal (`yShade`).

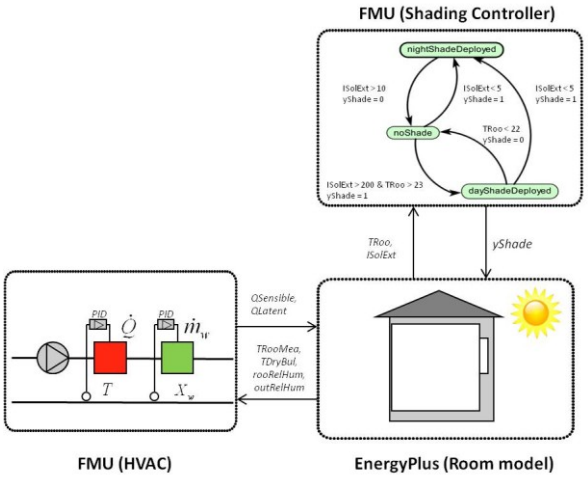

Figure 8 System with a HVAC system and a shading controller in FMUs and a room model with a shading device modeled in EneryPlus.

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.

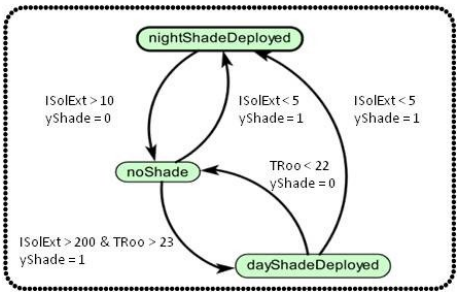

Figure 9 Finite state machine of the shading controller.

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

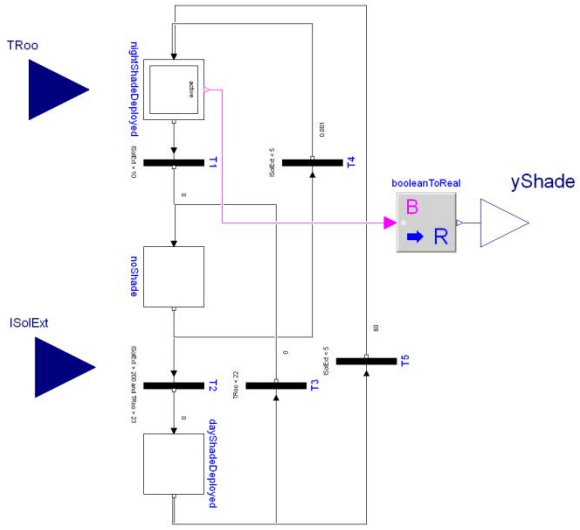

Figure 10 Modelica implementation of the shading controller system model.

To simulate the current system model in EnergyPlus, we extend the idf input file by defining a new FMU object for our shading controller.

```
ExternalInterface:FunctionalMockupUnitImport,
  ShadingController.fmu, !- FMU Filename
  15,                    !- FMU Timeout in milli-seconds
  0;                     !- FMU LoggingOn Value
```

We enter output variables from which the external interface reads data from and sends data to the FMU's shading controller.

```
ExternalInterface:FunctionalMockupUnitImport:From:Variable,
  ZONE SUBSURFACE 1 EAST WINDOW,!- EnergyPlus Key Value
   Surface Ext Solar Incident,  !- EnergyPlus Variable Name
   ShadingController.fmu,       !- FMU Filename
   Model1,                      !- FMU Instance Name
   ISolExt;                     !- FMU Variable Name


ExternalInterface:FunctionalMockupUnitImport:From:Variable,
    ZONE ONE,                   !- EnergyPlus Key Value
    Zone Mean Air Temperature,  !- EnergyPlus Variable Name
    ShadingController.fmu,      !- FMU Filename
    Model1,                     !- FMU Instance Name
    TRoo;                       !- FMU Variable Name
```

We specify these output variables in the idf file.

```
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:

```
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:

```
! EMS program. The first assignments sets the shading
! status and converts it into the
!              EnergyPlus signal (i.e., replace 1 by 6).
!              The second assignment sets yShade to
!              an EnergyManagementSystem:OutputVariable
!              which will be read by the external
!              interface.
EnergyManagementSystem:Program,
 Set_Shade_Control_State,          !- Name
 Set Shade_Signal = 6*yShade,      !- Program Line 1

! Declare an actuator to which the
EnergyManagementSystem:Program ! will write
EnergyManagementSystem:Actuator,
Shade_Signal,           !- Name
ZONE SUBSURFACE 1 EAST WINDOW,!- Actuated Component Unique
!Name
Window Shading Control, !- Actuated Component Type
Control Status;         !- Actuated Component Control Type

! Declare a global variable to which the
! EnergyManagementSystem:Program will write
```

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/m^2, which is defined in the transition of the shading controller's model.

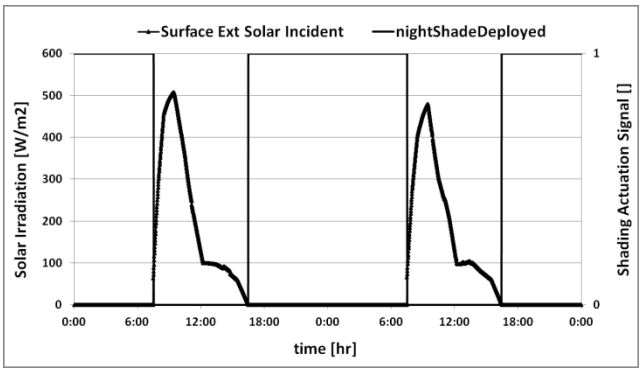

Figure 11 Simulation results showing the solar radiation incident on the shading device, and the shade actuation signal computed in the FMU (1 means that the shade is deployed).

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

Brooks, C., Lee, E. A., Liu, X., Neuendorffer, S., Zhao, Y. & Zheng, H. 2007. Heterogeneous Concurrent Modeling and Design in Java.

Clark, J., Cooper, C. & Drake, F. 2011. *Expat XML Parser* [Online]. Available: http://sourceforge.net/projects/expat [Accessed January 14 2013].

Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J. & Glazer, J. 2001. EnergyPlus: creating a new-generation building energy simulation program. *Energy and Buildings,* 33**,** 319-331.

Deringer, J. J., Nahman, J. E., Heming, K., Wetter, M., Pang, X. F. & Konstantoglou, M. 2012. LearnHPB and eLAD – Two Related Online eLearning Platforms for High Performance Buildings. *2012 ACEEE Summer Study on Energy Efficiency in Buildings.* Pacific Grove, CA.

Energyplus 2012a. External Interface(s) Application Guide, Guide for using EnergyPlus with External Interface.

Energyplus 2012b. Input Output Reference - The Encyclopedic Reference to EnergyPlus Input and Output.

Huang, J., Winkelmann, F., Buhl, F., Pedersen, C., Fisher, D., Liesen, R., Taylor, R., Strand, R. & Lawrie, L. Linking the COMIS multizone airflow model with the EnergyPlus building simulation program. Building Simulation 1999, 1999 Kyoto, Japan. 1065-1070.

Mattsson, S. E. & Elmqvist, H. 1997. An international effort to design the next generation modeling language. *7th IFAC Symposium on Computer Aided Control Systems Design.* Gent, Belgium.

Modelisar-Consortium. 2008-2012a. *Functional Mock-up Interface* [Online]. Available: https://fmi-standard.org/ [Accessed January 14 2013].

Modelisar-Consortium. 2008-2012b. *Functional Mock-up Interface for Co-Simulation* [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_CoSimulation_v1.0.pdf [Accessed January 14 2013].

Modelisar-Consortium. 2008-2012c. *Functional Mock-up Interface for Model-Exchange* [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_v1.0.pdf [Accessed January 14 2013].

Modelisar-Consortium. 2008-2012d. *Functional Mock-up Interface for Product Lifecycle Management* [Online]. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_PLM_v1.0.pdf [Accessed January 14 2013].

Pang, X. F., Wetter, M., Bhattacharya, P. & Haves, P. 2012. A framework for simulation-based real-time whole building performance assessment. *Building and Environment,* 54**,** 100-108.

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.

Trcka, M., Hensen, J. L. M. & Wetter, M. 2009. Co-simulation of innovative integrated HVAC systems in buildings. *Journal of Building Performance Simulation,* 2**,** 209-230.

Wang, L. P., William, J. & Jones, P. 2009. Case study of zero energy house design in UK. *Energy and Buildings,* 41**,** 1215-1222.

Wetter, M. 2011. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation,* 4**,** 185-203.

Wetter, M., Zuo, W., Nouidui, T. S. & Pang, X. 2013. Modelica Buildings library. To appear: *Journal of Building Performance Simulation*.

Zhai, Z. & Chen, Q. 2005. Performance of coupled building energy and CFD simululations. *Energy and Buildings,* 33**,** 319-331.