LBNL-XXXX

ERNEST ORLANDO LAWRENCE BERKELEY NATIONAL LABORATORY

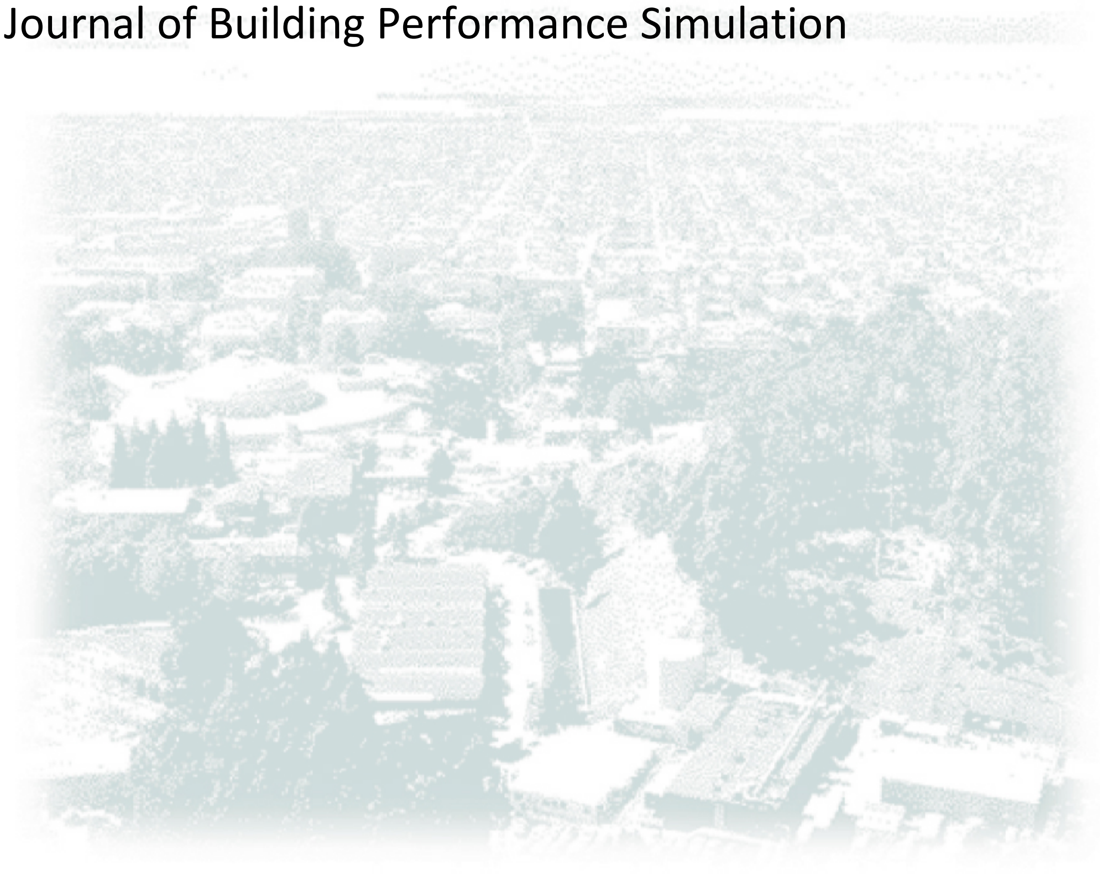
Functional Mock-up Unit for Co-Simulation Import in EnergyPlus

Thierry Stephane Nouidui, Michael Wetter, Wangda Zuo

Environmental Energy Technologies Division

November 2013

Submitted to
Journal of Building Performance Simulation



Functional Mock-up Unit for Co-Simulation Import in EnergyPlus

Thierry Stephane Nouidui[a]*, Michael Wetter[a], Wangda Zuo[a]

[a] *Lawrence Berkeley National Laboratory*
Environmental Energy Technologies Division
Building Technology and Urban Systems Department
Simulation Research Group
Berkeley, CA 94720, USA
[b] *University of Miami*
Department of Civil, Architectural, and Environmental Engineering
Coral Gables, FL 33124, USA

This article describes the development and implementation of the Functional Mock-up Unit (FMU) for co-simulation import interface in EnergyPlus. This new capability allows EnergyPlus to conduct co-simulation with various simulation programs that are packaged as FMUs. For example, one can model an innovative Heating, Ventilation, and Air Conditioning (HVAC) system and its controls in Modelica, export the HVAC system and the control algorithm as an FMU, and link it to a model of the building envelope in EnergyPlus for run-time data exchange.

The formal of FMUs is specified in the Functional Mock-up Interface (FMI) standard, an open standard designed to enable links between disparate simulation programs. An FMU may contain models, model description, source code, and executable programs for multiple platforms. A master simulator—in this case, EnergyPlus—imports and simulates the FMUs, controlling simulation time and coordinating the exchange of data between the different FMUs.

This article describes the mathematical basis of the FMI standard, discusses its application to EnergyPlus, and describes the architecture of the EnergyPlus implementation. It then presents a typical workflow, including pre-processing and co-simulation. The article concludes by presenting two use cases in which models of a ventilation system and a shading controller are imported in EnergyPlus as an FMU.

Keywords: Functional Mock-up Interface; Functional Mock-up Unit; co-simulation; building simulation;

1. Introduction

EnergyPlus (Crawley *et al.*, 2001) is a whole building energy simulation program that models heating, cooling, lighting, ventilation, and water use in buildings. EnergyPlus has been used for building design (Graça *et al.*, 2004; Wang *et al.*, 2009), fault detection and diagnostics (Jacob *et al.*, 2010; Pang *et al.*, 2012), and energy standard development (Borgeson and Brager, 2011; Thorton *et al.*, 2011). Since the focus of EnergyPlus is whole annual building energy simulation, it is not designed to study the controls performance of HVAC systems, to perform detailed daylighting analysis, or detailed temporal distribution of indoor airflow and contaminants that are better addressed by other simulation programs. To extend the capability of EnergyPlus for building simulation, EnergyPlus has been linked through co-simulation to various programs

*Corresponding author. Email: TSNouidui@lbl.gov from other domains, such as advanced lighting and daylighting simulation, Computational Fluid Dynamics (CFD), multizone airflow network , HVAC system and controls, for co-simulation.

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 that depend on state variables during run-time. We will now discuss three different ways to perform co-simulation with EnergyPlus.

The first approach is a one-to-one coupling (Figure 1). In this approach, specific interfaces need to be implemented in EnergyPlus and program B to enable the communication between the two programs. These interfaces can only be used for the co-simulation between EnergyPlus and program B. To link EnergyPlus with a program C, another set of interfaces have to be developed in both programs. The one-to-one approach has the benefit to enable a fast communication between the simulation programs. However, one has to modify the EnergyPlus code whenever EnergyPlus needs to be linked to another program. Furthermore, for many pratical co-simulations, the computing time is dominated by the simulation and not by the data exchange. Therefore, fast data exchange is not a high priority.

Using the one-to-one approach, Zhai and Chen (Zhai and Chen, 2005; Zhai *et al.*, 2002) coupled EnergyPlus with a CFD tool. The CFD tool was used in this context to provide more accurate convective heat transfer coefficients between the room air and the walls for the load calculation in EnergyPlus. Huang *et al.* (1999) coupled EnergyPlus with a Conjunction Of Multizone Infiltration Specialists (COMIS) multizone airflow network model. Using the boundary condition provided by EnergyPlus, COMIS calculated the airflow that was then used by EnergyPlus in subsequent heat balance calculations. Nghiem (2010) also used the one-to-one approach to develop a tool that allows coupling EnergyPlus with Matlab (MathWorks, 2012a) and Simulink (MathWorks, 2012b).

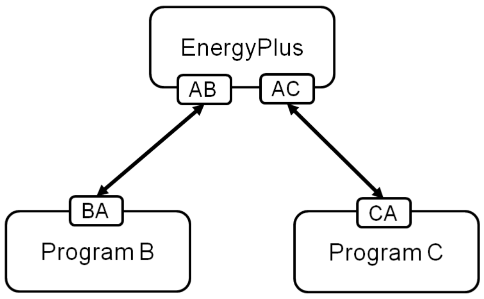


Figure 1. Co-simulation of EnergyPlus with programs B and C using a one-to-tone approach.

The second approach is a middleware coupling (Figure 2). The middleware acts as co-simulation master that orchestrates the simulation and manages the data exchange between the different simulation programs. In this approach, each simulation program needs to implement a specific interface to the middleware. As long as a program can communicate with the middleware, it can communicate with any programs linked to the middleware. Compared to the one-to-one approach, the middleware approach needs significantly less effort to enable co-simulation among various tools.

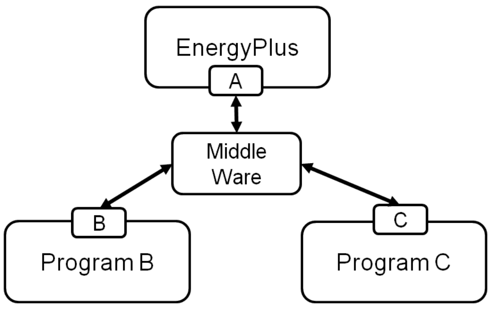


Figure 2. Co-simulation of EnergyPlus with programs B and C using a midleware approach.

Currently, the Building Controls Virtual Test Bed (BCVTB) (Wetter, 2011) is the only middleware which enables the co-simulation of EnergyPlus with other software and hardware. Through the BCVTB, EnergyPlus can do co-simulation with programs such as Matlab, Simulink, Dymola (Dassault Systemes AB, 2012), Radiance (Larson and Shakespeare, 2012), and ESP-r (Clarke, 2012). The BCVTB can also connect to hardware through the BACnet stack (ASHRAE, 2004) or through the analog/digital interface.

The third approach is to standardize the co-simulation interface (Figure 3). All simulation programs, including EnergyPlus, implement the same standard interface for co-simulation. Since the interface is standardized, it allows direct coupling of various simulation programs. Till date, standard interfaces have been used in different application domains to couple simulation programs (MODELISAR Consortium, 2012a; Accelera, 2012). Although this approach combines the advantages of both one-to-one and middleware approaches, it has not yet been implemented in EnergyPlus. Similar to the one-to-one approach, it enables the direct link between the simulation programs. By eliminating the transaction layer, the standard interface approach is more efficient and less complex than the middleware approach. This contribution describes the standard interface which has been implemented in EnergyPlus for co-simulation.

The standard interface we selected for EnergyPlus is the Functional Mock-up Interface (FMI) standard, which is an open standard for model exchange and co-simulation (MODELISAR Consortium, 2012a). Until 2012, the FMI has been supported by more than 35 (MODELISAR Consortium, 2012b) modeling and simulation environments. In the following, we will briefly introduce the FMI standard. We will then discuss the implementation of the standard in En-

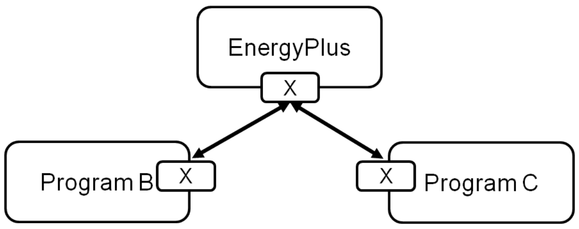


Figure 3. Co-simulation of EnergyPlus with programs B and C using a standard interface approach.

ergyPlus. After that we will show two examples which use the FMI to link EnergyPlus with another program for co-simulation.

2. Nomenclature

(1) We denote by $\mathbb{N} = \{0, 1, 2, ...\}$ the set of natural numbers, and by $\mathbb{R}$ the set of real numbers.

(2) Let $\mathbb{X}_k^n$ a set containing a sequence $\{x_i\}_{i=0}^k$ for some n, $k \in \mathbb{N}$ and $x \in \mathbb{R}^n$. We denote by $\underline{\mathbf{X}}_k^n$ the set of all $k+1$ element sequences in $\mathbb{X}_k^n$.

(3) $f(\cdot)$ denotes a function where $(\cdot)$ stands for the undesignated variables.

(4) $f(x)$ denotes the value of $f(\cdot)$ for the argument x.

3. FMI Standard

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

- 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. The exported model is independent of the target simulator because it does not use a simulator specific header file as in other approaches (MODELISAR Consortium, 2010b).
- The second part is FMI for co-simulation, an interface standard for coupling two or more simulation programs in a co-simulation environment. The data exchange between sub-systems is restricted to discrete communication points in time. In the time between two communication points, the sub-systems are solved independently from each other by their individual solver. A master algorithm controls the data exchange between sub-systems and the synchronization of all slave simulation programs (slaves). All information about the

slaves, which is relevant for the communication in the co-simulation environment, is provided in a slave specific XML-file (MODELISAR Consortium, 2010a).

A simulation model or program which implements the FMI standard is called FMU (Functional Mock-up Unit). An FMU comes along with a small set of easy to use C-functions (FMI functions) whose input and return arguments are defined by the FMI standard. These C-functions can be provided in source and/or binary form. The FMI functions are called by a simulator to create one or more instances of the FMU. The functions are also used to run the FMUs, typically together with other models. An FMU may either require the simulator to perform numerical integration (model-exchange) or be self-integrating (co-simulation). An FMU is distributed in the form of a zip-file that contains

- shared libraries, which contain the implementation of the FMI functions and/or source code of the FMI functions,
- an XML-file, also called the model description file, which contains the variable definitions as well as meta-information of the model,
- additional files such as tables, images or documentation that might be relevant for the model.

Figure 4 shows a schematic view of the structure of a zip-file of an FMU with the model description file, shared libraries for different target machines as well as additional resource information such as documentation, bitmaps and tables.

```
modelDescription.xml              // Description of model (required file)
model.png                         // Optional image file of model icon
documentation                     // Optional directory for the model documentation
        _main.html                // Entry point of the documentation
        <other documentation files>
sources                           // Optional directory containing all C-sources
binaries                          // Optional directory containing the binaries
        win32                     // Optional binaries for 32-bit Windows
                <modelIdentifier>.dll   // DLL of the model interface implementation
        win64                     // Optional binaries for 64-bit Windows
                ...
        linux32                   // Optional binaries for 32-bit Linux
                <modelIdentifier>.so    // DLL of the model interface implementation
        linux64                   // Optional binaries for 64-bit Linux
                ...
resources                         // Optional resources needed by the model
                < data in model specific files which will be read during initialization >
```

Figure 4. Structure of a zip-file of an FMU.

Figure 5 shows the dataflow of model exchange and co-simulation using the FMI standard. For model exchange, simulation program A will export an FMU through its FMU for model exchange export interface. The exported FMU is called ,,FMU for model exchange." It does not include any numerical solver. The FMU for model exchange is then imported into simulation program B through the FMU for model exchange import interface. The numerical solver of simulation program B is used to solve both sub-systems A and B.

For co-simulation, simulation program A will export an FMU through its FMU for co-simulation export interface. The exported FMU is called ,,FMU for co-simulation." It contains the model and numerical solver that can execute independently. The FMU for co-simulation is then imported into simulation program C through the FMU for co-simulation import interface. Simulation program C can then use its co-simulation master algorithm to link sub-systems A and C.

Since this paper is focusing on co-simulation, we use ,,FMU" and ,,FMU import" as abbreviations for ,,FMU for co-simulation" and ,,FMU for co-simulation import", respectively, below.

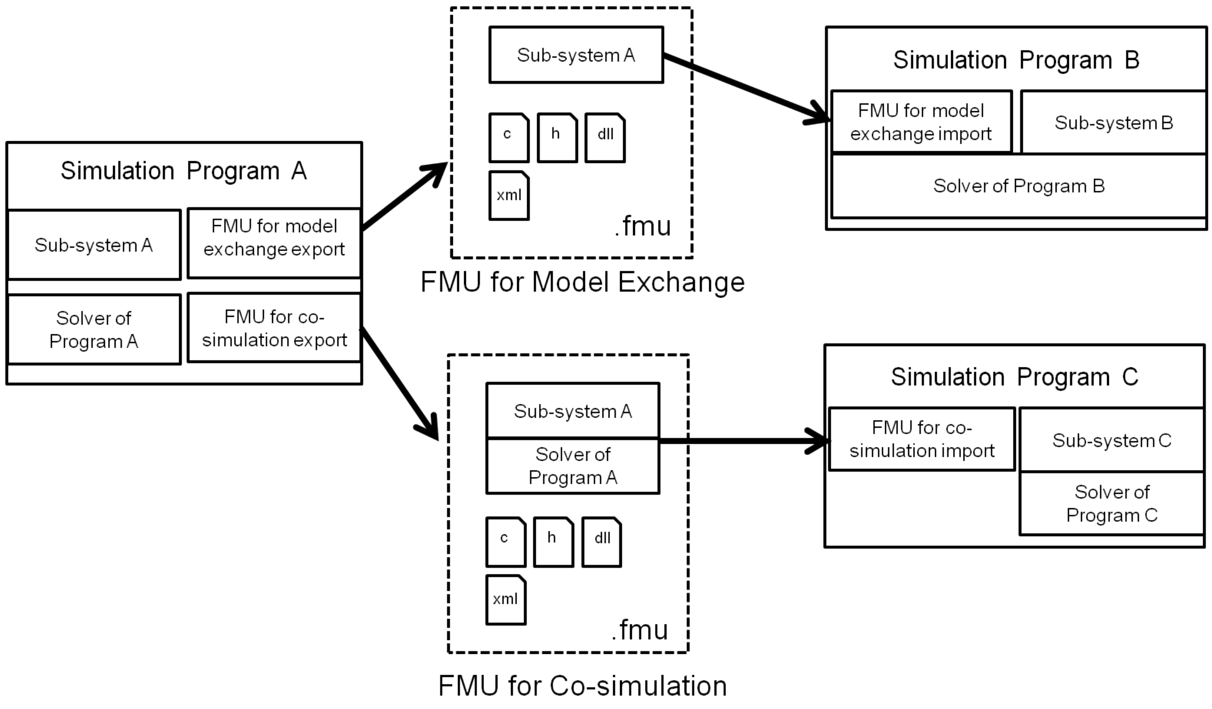


Figure 5. Dataflow of model-exchange and co-simulation using the FMI standard.

4. Implementation

EnergyPlus is a simulation program written in Fortran. It consists of many program modules that calculate the energy consumption of a building. In EnergyPlus all program modules are controlled by a Solution Manager (Department Of Energy (DOE), 2011a). The loops are divided into supply and demand sides, and the solution scheme generally relies on successive substitution iteration to reconcile supply and demand. To link EnergyPlus to one or multiple FMUs, the EnergyPlus module `ExternalInterface` (Department Of Energy (DOE), 2011b) has been modified so that data can be exchanged during run-time between EnergyPlus and FMUs. A shared library compiled from C has been developed for this. This library contains all functions needed to interface with FMUs. The primary functions in this shared library are called at runtime to unzip the FMUs, instantiate, initialize, set, and get values of defined variables and execute single time steps. In addition to the libraries, the EnergyPlus input data dictionary 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. In the current implementation, the data exchange is done at the beginning of each zone time step. There is no iteration between the FMUs and EnergyPlus, and the synchronization of the data exchange is controlled by EnergyPlus.

5. Architecture of the FMU import

Figure 6 shows the architecture of the connection between EnergyPlus and two FMUs. EnergyPlus imports the FMUs that connect to its external interface. These FMUs are generated by external simulation environments that implement the FMU for co-simulation export interface. See (MODELISAR Consortium, 2012a) for a list of simulation programs that export FMUs. In the external interface, the input/output signals that are exchanged between the FMUs and EnergyPlus are mapped to EnergyPlus objects.

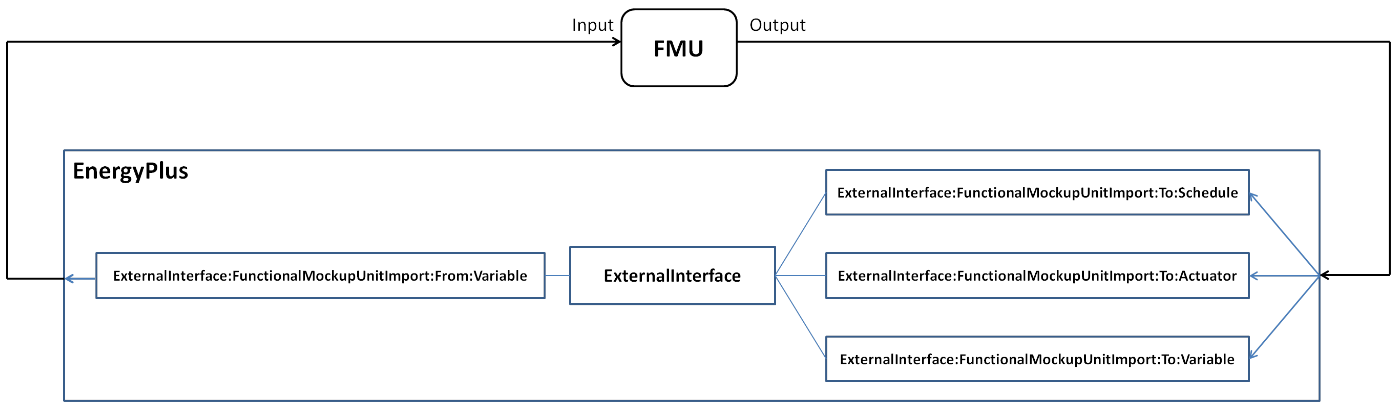

Input
FMU
Output
EnergyPlus
ExternalInterface:FunctionalMockupUnitImport:To:Schedule
ExternalInterface:FunctionalMockupUnitImport:From:Variable
ExternalInterface
ExternalInterface:FunctionalMockupUnitImport:To:Actuator
ExternalInterface:FunctionalMockupUnitImport:To:Variable

Figure 6. Architecture of the FMU import.

The external interface can map to four new EnergyPlus input objects called:

- `ExternalInterface:FunctionalMockupUnitImport:To:Schedule`,
- `ExternalInterface:FunctionalMockupUnitImport:To:Actuator`,
- `ExternalInterface:FunctionalMockupUnitImport:To:Variable`, and
- `ExternalInterface:FunctionalMockupUnitImport:From:Variable`.

The `ExternalInterface:FunctionalMockupUnitImport:To:Schedule` can be used to overwrite schedules, the `ExternalInterface:FunctionalMockupUnitImport:To:Actuator` and the `ExternalInterface:FunctionalMockupUnitImport:To:Variable` objects can be used in place of Energy Management System (EMS) actuators and EMS variables. These 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 the zone time step. The external interface also uses the `ExternalInterface:FunctionalMockupUnitImport:From:Variable` object which reads EnergyPlus objects `Output:Variable` and `EnergyManagementSystem:OutputVariable` to send data from EnergyPlus to FMUs.

6. Workflow of the FMU import

We identified two major steps when conducting co-simulation with FMUs in EnergyPlus: pre-processing and co-simulation. The pre-processing step generates a section of an EnergyPlus input file that can be used to configure the FMU import. The input file defines the input and output variables for both EnergyPlus and the FMUs. The co-simulation step performs co-simulation.

To support the user in the pre-processing step, we developed an FMU parser which unzips and extracts relevant information from the model description file of the FMU and writes them in a temporary EnergyPlus input file. The model description file is an XML-file which contains the definition of all variables in the model in a standardized way. The FMU parser is a code written in C that includes Expat (Clark *et al.*, 2011), which is an XML parser library written in C.

In the co-simulation step, the user launches the simulation by invoking EnergyPlus with the EnergyPlus input file. After EnergyPlus detects that the FMU import is activated, it will call the FMU parser and unzip the related FMU for executable files and description files. After that, EnergyPlus will conduct the co-simulation with FMUs.

7. Mathematics of data exchange for the FMU import

This section describes the data exchange between EnergyPlus and FMUs. Prior to describing the data exchange between EnergyPlus and FMUs, some definitions and terminologies used in the remainder of this document will be introduced.

Consider a model that is described by a system of differential algebraic equations. We say that a variable is a differential variable if its derivatives are present in the differential algebraic equations. Otherwise, we say that the variable is an algebraic variable (Fabian *et al.*, 2008).

Because in subsequent discussions, we will distinguish between algebraic and differential variables, we now introduce a new notation for different system of equations that involve algebraic and differential variables. Let n, m, $q \in \mathbb{N}$, and let $x^1 \in \mathbb{R}^n$, $x^2 \in \mathbb{R}^m$, and $u \in \mathbb{R}^q$.

- If x^1 and x^2 are differential variables, then the system is $F(\dot{x}^1, x^1, \dot{x}^2, x^2, u, t) = 0$, where $F : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^q \times \mathbb{R} \to \mathbb{R}^{n+m}$.

- If x^1 is a differential variable and x^2 is an algebraic variable, then the system is $G(\dot{x}^1, x^1, x^2, u, t) = 0$, where $G : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q \times \mathbb{R} \to \mathbb{R}^{n+m}$.

- If x^1 is an algebraic variable and x^2 is a differential variable, then the system is $H(x^1, \dot{x}^2, x^2, u, t) = 0$, where $H : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^q \times \mathbb{R} \to \mathbb{R}^{n+m}$.

- If x^1 is an algebraic variable and x^2 is an algebraic variable, then the system is $I(x^1, x^2, u, t) = 0$, where $I : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q \times \mathbb{R} \to \mathbb{R}^{n+m}$.

Throughout this section, let N denote the number of time steps and let t_k with $k \in \{0, ..., N\}$ denote the synchronization time instants. We will use the superscripts 1 and 2 to denote the variables and the functions that compute the next state variable of the simulator 1 and 2, respectively.

Figure 7 shows a case where an FMU is linked to an EnergyPlus model for co-simulation. The FMU and EnergyPlus could be linked through differential or algebraic variables.

Table 1 shows the different system configurations that are possible.

- In the first case, the variable x^1 and x^2 are differential variables in EnergyPlus and in the FMU.
- In the second case, the variable x^1 is a differential variable and the variable x^2 is an algebraic variable.
- In the third case, the variable x^1 is an algebraic variable and the variable x^2 is a differential

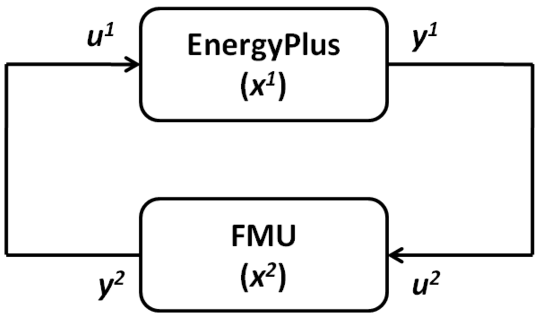


Figure 7. System with one FMU linked to EnergyPlus.

variable.

- In the fourth case, the variable x^1 is an algebraic variable and the variable x^2 is an algebraic variable.

Table 1. Use cases with different system configurations

Use Cases	EnergyPlus	FMU
1	Model1 (Differential variable)	Model2 (Differential variable)
2	Model1 (Differential variable)	Model2 (Algebraic variable)
3	Model1 (Algebraic variable)	Model2 (Differential variable)
4	Model1 (Algebraic variable)	Model2 (Algebraic variable)

The current implementation of the FMU import in EnergyPlus supports the first and second system configuration since the third and fourth case will require the FMU to be solved numerically in the iteration solver loop of EnergyPlus. This will necessitate a) the ability of the FMU to reject time steps (MODELISAR Consortium, 2010a), which is currently not supported by major tool vendors that export FMUs, and b) the ability of EnergyPlus to save its state variables and reset them when needed. This capability is hard to achieve in EnergyPlus, as it does not have a data structure for storing and resetting state variables. In the next sections, we describe and discuss the mathematics used for the data exchange in use cases 1 and 2.

7.1 Use Case: Linking EnergyPlus and FMU through differential variables

This use case could be for an application where a wall with a phase change material (PCM) is modeled in an FMU and linked to a room model in EnergyPlus. The room dry-bulb temperature is the differential variable in EnergyPlus and the temperature of the wall with PCM is the differential variable in the FMU. Each system solves a differential equation that is connected to the differential equation of the other system. For simplicity, we assume that $y^1(\cdot) = x^1(\cdot)$ and $y^2(\cdot) = x^2(\cdot)$. The systems are described by the ordinary differential equations

$$0 = f^1(\dot{x}^1, x^1, x^2), \text{ with } x^1(0) = x^{1,0}, \tag{1}$$

$$0 = f^2(\dot{x}^2, x^2, x^1), \text{ with } x^2(0) = x^{2,0}. \tag{2}$$

The first system computes, for $k \in \{0, ..., N\}$ and some $\tilde{F}^1 : \underline{\mathbf{X}}^n_k \times \underline{\mathbf{X}}^m_k \times \underline{\mathbb{R}}_{k+1} \to \mathbb{R}^n$, the sequence

$$x^1(t_{k+1}) = \tilde{F}^1(\underline{x}^1_k, \underline{x}^2_k, \underline{t}_{k+1}), \tag{3}$$

and, similarly, the simulator 2 computes for some $\tilde{F}^2 : \underline{\mathbf{X}}^m_k \times \underline{\mathbf{X}}^n_k \times \underline{\mathbb{R}}_{k+1} \to \mathbb{R}^m$,

$$x^2(t_{k+1}) = \tilde{F}^2(\underline{x}^2_k, \underline{x}^1_k, \underline{t}_{k+1}), \tag{4}$$

with initial conditions $x^1(0) = x^{1,0}$ and $x^2(0) = x^{2,0}$. The functions $\tilde{F}^1(\cdot,\cdot,\cdot,\cdot)$ and $\tilde{F}^2(\cdot,\cdot,\cdot,\cdot)$ are used to compute the values of the state variables at the new time instant.

To advance from time t_k to t_{k+1}, each system uses its own time integration algorithm. At the end of the time step, EnergyPlus sends the new state $x^1(t_{k+1})$ to the FMU and it receives the state $x^2(t_{k+1})$ from the FMU. The same procedure is done by the FMU.

7.2 Use Case: Linking EnergyPlus and FMU through algebraic and differential variables

This use case could arise in an application where a fan is modeled in an FMU and is linked to a room model in EnergyPlus. The room temperature is the differential variable in EnergyPlus and the pressure difference of the fan is the algebraic variable in the FMU. For simplicity, we assume that $y^1(\cdot) = x^1(\cdot)$ and $y^2(\cdot) = x^2(\cdot)$. In this application, the systems are described by the equations

$$0 = g^1(\dot{x}^1, x^1, x^2), \text{ with } x^1(0) = x^{1,0}, \tag{5}$$

$$0 = g^2(x^2, x^1). \tag{6}$$

Let N denote the number of time steps and let t_k with $k \in \{0, ..., N\}$ denote the time steps. We use the same superscripts 1 and 2 as for the first case to denote the variable and the function that computes the next variable of the simulator 1 and 2, respectively. The first system computes, for $k \in \{0, ..., N\}$ and some $\tilde{G}^1 : \underline{\mathbf{X}}^n_k \times \underline{\mathbf{X}}^m_k \times \underline{\mathbb{R}}_{k+1} \to \mathbb{R}^n$ the sequence

$$x^1(t_{k+1}) = \tilde{G}^1(\underline{x}^1_k, \underline{x}^2_k, \underline{t}_{k+1}), \tag{7}$$

and, similarly, the simulator 2 computes for $\tilde{G}^2 : \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$, the sequence

$$x^2(t_{k+1}) = \tilde{G}^2(x^2_k, x^1_k, t_{k+1}), \tag{8}$$

with initial conditions $x^1(0) = x_{1,0}$. The functions $\tilde{G}^1(\cdot, \cdot, \cdot, \cdot)$ and $\tilde{G}^2(\cdot, \cdot, \cdot)$ are used to compute the value of the state and algebraic variables at the new time step. To advance from time t_k to t_{k+1}, system 1 uses its own time integration algorithm. At the end of the time step, EnergyPlus sends the new state $x^1(t_{k+1})$ to the FMU and it receives the value $x^2(t_{k+1})$ from the FMU.

The coupling schemes described in Section 7.1 and 7.2 are based on loose coupling (Hensen, 1999), which is easier to implement, requires shorter synchronization time steps, is numerically more robust, and computed faster in the experiments reported by Trčka *et al.* (2007).

8. Applications

8.1 Coupling an HVAC system model, implemented in an FMU, with a room model in EnergyPlus

In this example, an HVAC system implemented in an FMU is linked to a room model in EnergyPlus. Figure 8 shows the linkage between the FMU and EnergyPlus. The HVAC system was implemented in Modelica and exported as an FMU. The room model was modeled in EnergyPlus. The HVAC system computes sensible and latent heat gain required to maintain a room set point temperature and humidity. The FMU needs as inputs the room dry-bulb temperature ($TRooMea$), the outdoor dry-bulb temperature ($TDryBul$) , the room air relative humidity ($rooRelHum$), and the outdoor air relative humidity ($outRelHum$). The outputs of the FMU are the sensible ($QSensible$) and latent ($QLatent$) heat transported across the thermodynamic boundary of air inlet and outlet of the thermal zone.

This is an application of the first use case where the differential variables are room dry-bulb temperature and room air relative humidity in EnergyPlus, and sensible and latent heat provided by the HVAC system in the FMU. We also sent the outdoor dry-bulb temperature and outdoor air relative humidity from EnergyPlus to the FMU. These are, however, functions of time that are not part of an algebraic loop and hence can be sent from one tool to another without changing the mathematical structure of the coupled system.

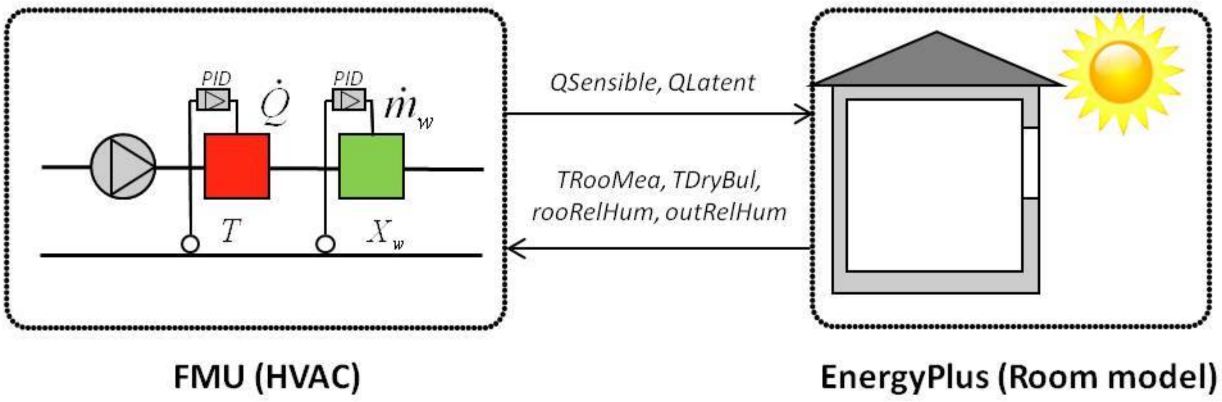


Figure 8. System with an HVAC system modeled in an FMU and a room model modeled in EnergyPlus ($\dot{Q}$ is the heat flow rate, T is the temperature, $\dot{m}_w$ is the water mass flow rate, and X_w is the water vapor mass fraction).

Figure 9 shows the Modelica implementation of the HVAC model with the four inputs that will be linked to outputs of EnergyPlus and the two outputs that will be connected to the inputs of EnergyPlus. The Modelica model represents an air-based heating system with an ideal heater and an ideal humidifier in the supply duct. The heater and humidifier are controlled with a feedback loop that tracks the room air temperature and room air humidity. The ideal heater adds heat to the thermal zone in the amount $\dot{Q} = y_h \cdot \dot{Q}_0$ and the humidifier adds water vapour in the amount $\dot{m}_w = y_w \cdot \dot{m}_{w,0}$ where y_h and y_w are the output signals of controllers that track the return air temperature and water vapour concentration, and $\dot{Q}_0$ and $\dot{m}_{w,0}$ are the nominal heating and humidification capacities.

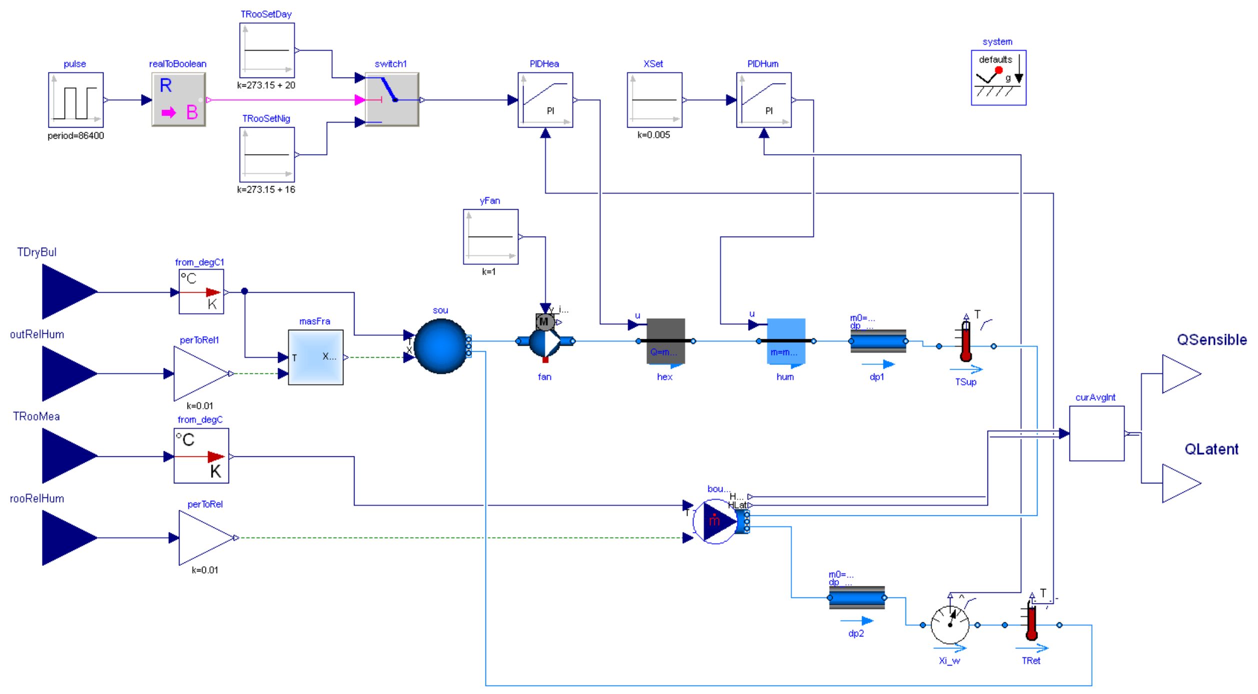


Figure 9. Modelica implementation of the HVAC model which is exported as an FMU.

To link the FMU with EnergyPlus, we send from the FMU to EnergyPlus two schedule values for the latent and sensible heat gain and from EnergyPlus to the FMU four output variables for the outdoor dry-bulb temperature, outdoor air relative humidity, room dry-bulb temperature and room air relative humidity at each zone time step. This can be accomplished by using two objects of type `ExternalInterface:FunctionalMockupUnitImport:To:Schedule` to write values in EnergyPlus and four objects of type `ExternalInterface:FunctionalMockupUnitImport:From:Variable` to read values from EnergyPlus. During the simulation, the two models are coupled by setting during the time interval $t \in [t_k,\ t_{k+1})$, the boundary conditions of the HVAC system model to the room dry-bulb temperature $TRooMea(t_k)$, to the room relative humidity $rooRelHum(t_k)$, to the outdoor dry-bulb temperature $TDryBul(t_k)$ and to the outdoor relative humidity $outRelHum(t_k)$. These values are sent from EnergyPlus to the FMU. The HVAC system model in the FMU then computes the sensible and latent heat $QSensible(t_k)$ and $QLatent(t_k)$ and sends these values to EnergyPlus. The simulation is performed at a one minute time step.

8.2 *Coupling a shading controller, implemented in an FMU, with a shading device in EnergyPlus*

In this example, a shading controller, represented by a finite state machine, was implemented in Modelica and exported as an FMU. It was then linked to a shading device which is part of an EnergyPlus room model. Figure 10 shows the coupling between the shading controller and EnergyPlus. The inputs of the 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 ($yShade$). The room model with the window and the shading device, which will be controlled by the FMU, has been implemented in EnergyPlus. The shading controller was implemented in Modelica and exported as an FMU.

This is an application of the second use case where the differential variable is the room dry-bulb temperature in EnergyPlus, and the algebraic variable is the shading actuation signal in the FMU. We also sent the solar irradiation from EnergyPlus to the FMU which is not part of an algebraic loop and thus can be sent from one tool to another without changing the mathematical structure of the coupled system.

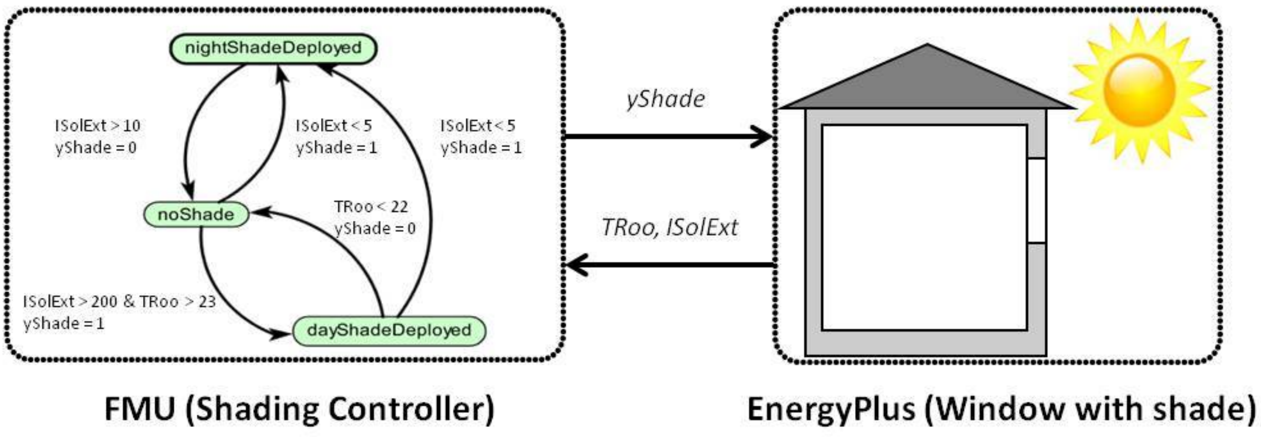


Figure 10. System with a shading controller in an FMU and a room model with a shading device modeled in EnergyPlus.

Figure 11 shows the Modelica implementation of the shading controller which has been modeled using a finite state machine of the Modelica Standard Library. The finite state machine switches between the states *nightShadeDeployed*, *noShade* and *dayShadeDeployed* depending on the guards defined in the transitions. Input of the EnergyPlus model is the shading actuation signal computed in the FMU. During the simulation, the two systems are coupled by setting during the time interval $t \in [t_k,\ t_{k+1})$ the boundary conditions of the shading controller to the room dry-bulb temperature $TRoo(t_k)$ and to the solar irradiation incident on the window $ISolExt(t_k)$. These values are sent from EnergyPlus to the FMU. The shading controller in the FMU then computes the actuation signal $yShade(t_k)$ and sends this value to EnergyPlus. The simulation and reporting are performed at a 10 minutes time step.

The coupling of the two systems in EnergyPlus can be achieved by using either one object of type `ExternalInterface:FunctionalMockupUnitImport:To:Actuator` or one object of type `ExternalInterface:FunctionalMockupUnitImport:To:Variable` to write values to EnergyPlus and two objects of type `ExternalInterface:FunctionalMockupUnitImport:From:Variable` to read values from EnergyPlus.

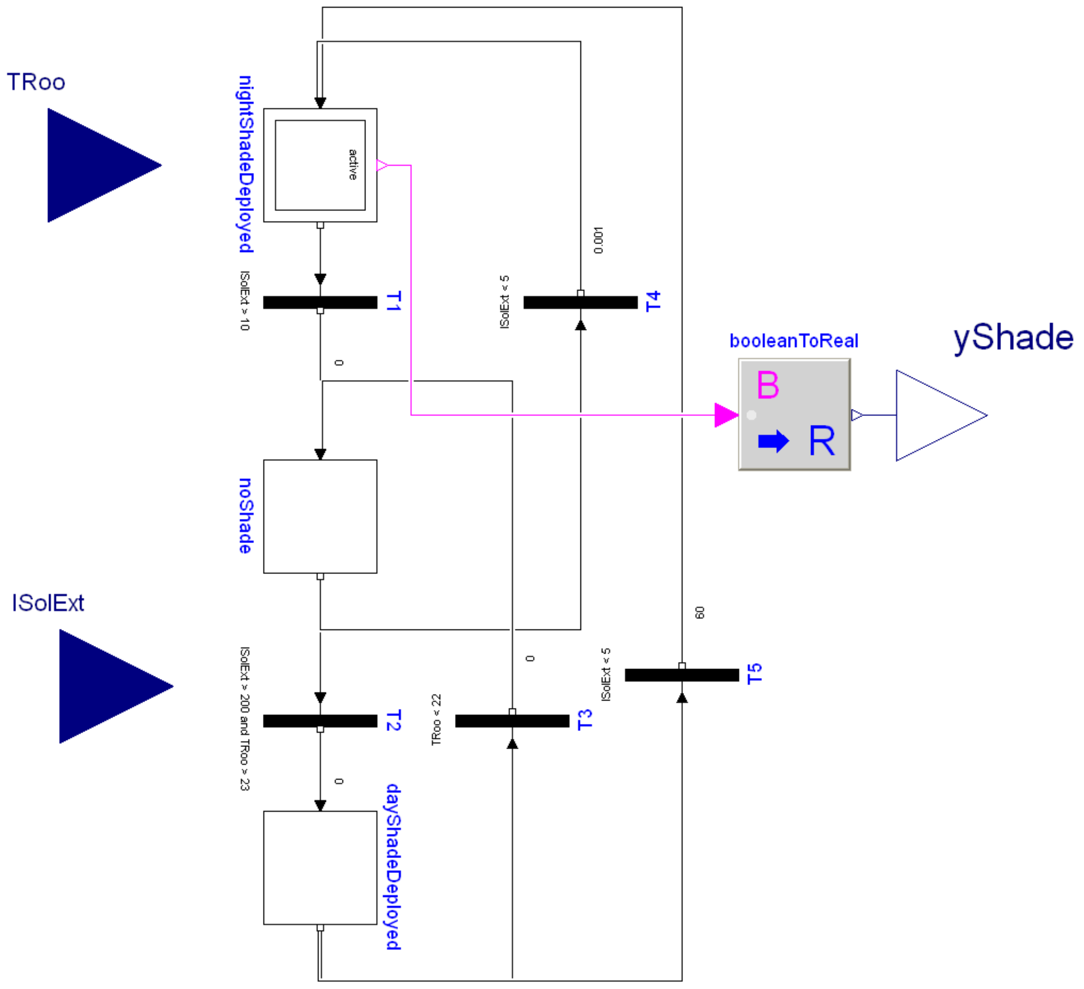


Figure 11. Modelica implementation of the shading controller model which is exported as an FMU.

Figure 12 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 model.

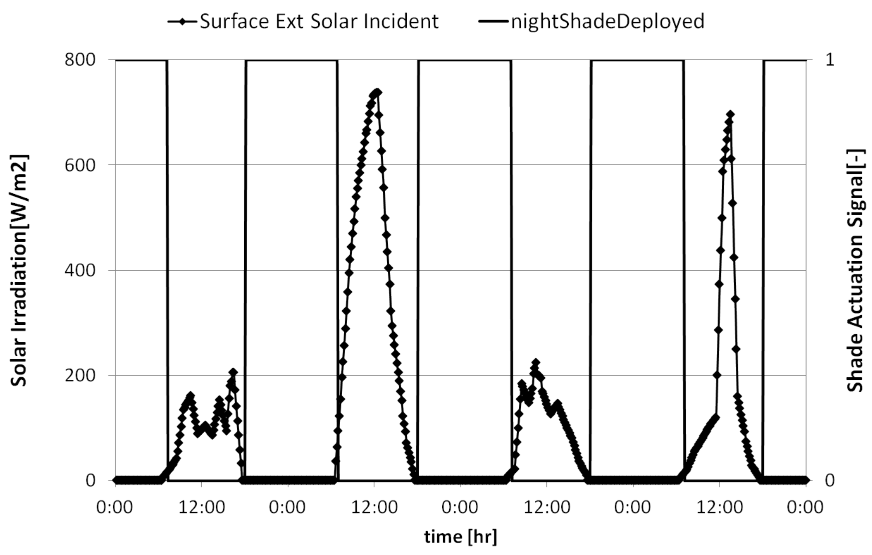


Figure 12. Simulation results showing the solar radiation incident on the shading device, and the shade actuation signal computed in the FMU (1 means that the shade is deployed).

9. Conclusion

Leveraging the FMI standard and creating an FMU import interface in EnergyPlus allowed EnergyPlus to be coupled with any simulation tool that is exported as an FMU for co-simulation. With the FMU import interface in EnergyPlus, it is possible prior to simulate a building to a) subdivide the building into different domains such as envelope, HVAC, and controls, b) model each domain in the simulation tool which is best suited for the problem at hand, e.g. Modelica for HVAC, Simulink for controls, EnergyPlus for the envelope, c) export the HVAC and the controls as FMUs, and d) link them through the FMU import interface in EnergyPlus. The FMI standard is a promising approach, developed by the co-simulation community and applied by the building simulation community. It is currently supported by a growing number of tools. We believe this approach to be robust and be able to contribute to solving challenges faced in integrated building simulations. 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.

10. Acknowledgment

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

References

Accelera, 2012. Accelera Systems Initiative. [online] `http://www.accellera.org/downloads/standards/systemc` [Last accessed 01 March 2013].

ASHRAE, 2004. ANSI/ASHRAE Standard 135-2004. BACnet - A Data Communication Protocol for Building Automation and Control Networks. [online] `http://www.bacnet.org/` [Last accessed 01 March 2013].

Borgeson, S. and Brager, G., 2011. Comfort standards and variations in exceedance for mixed-mode buildings. *Building Research and Information*, 39 (2), 118–133.

Clark, J., Cooper, C., and Drake, F., 2011. Expat XML Parser. [online] `http://sourceforge.net/projects/expat` [Last accessed 19 September 2012].

Clarke, J., 1974-2012. ESP-r. [online] `http://www.esru.strath.ac.uk/Programs/ESP-r.htm` [Last accessed 01 March 2013].

Crawley, D.B., *et al.*, 2001. EnergyPlus: Creating a New-Generation Building Energy Simulation

Program. *Energy and Buildings*, 33 (4), 319–331.

Dassault Systemes AB, 1991-2012. Dymola Dynamic Modeling Laboratory. [online] `http://www.3ds.com/products/catia/portfolio/dymola` [Last accessed 01 March 2013].

Department Of Energy (DOE), 2011a. EnergyPlus Engineering Reference. [online] `http://apps1.eere.energy.gov/buildings/energyplus/pdfs/engineeringreference.pdf` [Last accessed 01 March 2013a].

Department Of Energy (DOE), 2011b. External Interface Guide for EnergyPlus. [online] `http://apps1.eere.energy.gov/buildings/energyplus/pdfs/externalinterfaces_application_guide.pdf` [Last accessed 01 March 2013b].

Fabian, G., van Beek, D., and Rooda, J., 2008. Substitute equations for index reduction and discontinuity handling. *In*: *Proc. of the Third International Symposium on Mathematical Modeling*, Vienna, Austria.

Graça, G., Linden, P., and Haves, P., 2004. Use of simulation in the design of a large, naturally ventilated office building. *Building Services Eengineering Research and Technology*, 25 (3), 211–221.

Hensen, J.L.M., 1999. A comparison of coupled and de-coupled solutions for temperature and air flow in a building. *ASHRAE Transactions. American Society of Heating, Refrigerating, and Air-Conditioning Engineers*, 105 (2).

Huang, J., *et al.*, 1999. Linking the COMIS multizone airflow model with the EnergyPlus building simulation program. *In*: *Proc. of Building Simulation 1999*, 1065–1070.

Jacob, D., *et al.*, 2010. Black-box models for fault detection and performance monitoring of buildings. *Journal of Building Perfomance Simulation*, 3 (1), 53–62.

Larson, G.W. and Shakespeare, R.A., 1988-2012. Radiance. [online] `http://radsite.lbl.gov/radiance/` [Last accessed 01 March 2013].

MathWorks, 1994-2012a. MATLAB The Language of Technical Computing. [online] `http://www.mathworks.com/products/matlab/` [Last accessed 01 March 2013a].

MathWorks, 1994-2012b. MATLAB The Language of Technical Computing. [online] `http://www.mathworks.com/products/simulink/` [Last accessed 01 March 2013b].

MODELISAR Consortium, 2008-2012a. Functional Mock-up Interface. [online] `https://fmi-standard.org/` [Last accessed 01 March 2013a].

MODELISAR Consortium, 2008-2012b. Functional Mock-up Interface Support Tools. [online] `https://www.fmi-standard.org/tools` [Last accessed 01 March 2013b].

MODELISAR Consortium, 2010a. Functional Mock-up Interface for Co-Simulation. [online] `https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_CoSimulation_v1.0.pdf` [Last accessed 01 March 2013a].

MODELISAR Consortium, 2010b. Functional Mock-up Interface for Model-Exchange. [online] `https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_v1.0.pdf` [Last accessed 01 March 2013b].

Nghiem, T., 2010. MLE+: a Matlab-EnergyPlus Co-simulation Interface. [online] `http://www.seas.upenn.edu/~nghiem/software.html` [Last accessed 01 March 2013].

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

Thorton, B.A., *et al.*, 2011. Achieving the 30% Goal: Energy and Cost Savings Analysis of ASHRAE Standard 90.1-2010. [online] `http://www.pnl.gov/main/publications/external/technical_reports/PNNL-20405.pdf` [Last accessed 01 March 2013].

Trčka, M., Hensen, J.L.M., and Wetter, M., 2007. Comparison of co-simulation approaches for building and HVAC/R Simulation. *In*: *Proceedings of the 10th IBPSA Conference*, 1418–1425.

Wang, L.P., William, J., and Jones, P., 2009. Case study of zero energy house design in U.K.. *Energy and Buildings*, 41 (11), 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*, 3 (4).

Zhai, Z. and Chen, Q., 2005. Performance of coupled building energy and CFD simululations. *Energy and Buildings*, 37 (4), 333–344.

Zhai, Z., Chen, Q., Haves, P., and Klems, J., 2002. On approaches to couple energy simulation and computational fluid dynamics programs. *Building and Environment*, 37 (8-9), 857–864.