Tool coupling for the design and operation of building energy and control systems based on the Functional Mock-up Interface standard

Thierry Stephane Nouidui, Michael Wetter
Lawrence Berkeley National Laboratory
One Cyclotron Road, Berkeley, CA
TSNouidui@lbl.gov

Abstract

This paper describes software tools developed at the Lawrence Berkeley National Laboratory (LBNL) that can be coupled through the Functional Mock-up Interface standard in support of the design and operation of building energy and control systems. These tools have been developed to address the gaps and limitations encountered in legacy simulation tools. These tools were originally designed for the analysis of individual domains of buildings, and have been difficult to integrate with other tools for runtime data exchange. The coupling has been realized by use of the Functional Mock-up Interface for co-simulation, which standardizes an application programming interface for simulator interoperability that has been adopted in a variety of industrial domains.

As a variety of coupling scenarios are possible, this paper provides users with guidance on what coupling may be best suited for their application. Furthermore, the paper illustrates how tools can be integrated into a building management system to support the operation of buildings. These tools may be a design model that is used for real-time performance monitoring, a fault detection and diagnostics algorithm, or a control sequence, each of which may be exported as a Functional Mock-up Unit and made available in a building management system as an input/output block. We anticipate that this capability can contribute to bridging the observed performance gap between design and operational energy use of buildings.

Keywords: Co-simulation; Functional Mock-up Interface; Building Management System; Niagara^{AX}

1 Introduction

Building thermal systems, ventilation systems, electrical systems and control systems are becoming more and more integrated to increase the energy efficiency and to improve the interoperability with the electrical grid. This leads to a higher level of complexity for the design, installation, commissioning and operation of these systems. Modeling and simulation of such systems is challenging in today's simulation tools because it requires the tool to support multiple physical domains, multi-time scales, and also different formalisms for how systems evolve in time, in particular if they involve supervisory control with state transitions.

At present, the simulation of controls, rapid prototyping of new building energy and control systems and the use of simulation for building operations and building retrofits are constrained by current simulation tools. Most legacy whole building energy simulation tools such as EnergyPlus [1] or TRNSYS [2] perform well for annual energy analysis, but their model representation and numerical methods do not allow simulating systems with fast dynamics nor do they allow the proper representation of controls. For example, in EnergyPlus, the smallest time step is one minute, TRNSYS has a fixed time step, and neither can handle state events. To nevertheless use these tools with other programs that better address controls or systems with fast transients, but may lack comprehensive libraries of building components, they can be coupled with each other through co-simulation. By co-simulation, we mean a technique that allows individual component models described by differential algebraic or discrete equations to be simulated by different simulation tools running simultaneously and exchanging data during runtime.

Co-simulation somewhat remedies the limitations of individual tools. While this allows addressing many practical questions, see [3-5], one has to keep in mind that hybrid systems formed through this tool coupling still have some deficiencies. We refer to [6] for properties that would need to be satisfied by the individual tools to allow a proper treatment of hybrid systems.

This paper is structured as follows: Section 2 introduces the Functional Mock-up Interface (FMI) standard which is the open standardized interface used in this paper for co-simulation. Section 3 describes a) FMIs added to the Building Controls Virtual Test Bed (BCVTB), and EnergyPlus to support their co-simulation with various tools, and b) an FMI added to a building management system to support error-free development and deployment of control algorithms. Section 4 presents our conclusions.

2 Functional Mock-up Interface for Co-Simulation

The FMI standard has originally been developed in the Information Technology for European Advancement (ITEA2) project MODELISAR.

The FMI standard supports both model exchange and co-simulation of dynamic models using a combination of XML[1]-file, C-code and/or shared libraries.

The FMI standard version 1.0, which has been used in this contribution, consists of three parts:

- FMI for model exchange, which standardizes an interface for coupling simulation tools that are integrated in time by an external solver [7].
- FMI for co-simulation, which standardizes an interface for coupling simulation tools that contain their own solver for time integration [8].
- FMI for Product Lifecycle Management, which provides a standardized way to handle FMI related data [9].

A system model or simulation tool which implements the FMI standard is called a Functional Mock-up Unit (FMU). An FMU comes in the form of a zip-file, which contains the FMI model description file, which is an XML-file with information needed by an import tool, C-code and/or shared libraries required to interface with the model or simulation tool, and resource files such as tables, or documentation.

This contribution uses the FMI for co-simulation application programming interface (API). This API provides the means for two implementations namely *CoSimulation_Tool*, and *CoSimulation_StandAlone.* In the *CoSimulation_Tool* implementation, the FMU contains a wrapper for shared libraries that interact with the slave tool so that a master tool which imports the FMU can interface with the slave tool in a standardized way. In the *CoSimulation_StandAlone* implementation, the FMU contains the model and its solver.

[1]XML stands for Extensible Markup Language.

3 Co-simulation using the FMI Standard

3.1 FMU for Co-Simulation Import Interface in the BCVTB

The BCVTB [10] is a free, open-source middleware based on Ptolemy II [11]. It allows users to couple different simulation tools such as EnergyPlus, TRNSYS, MATLAB/Simulink [12], Modelica [13], or ESP-r [14] at runtime for co-simulation. The BCVTB also allows calling system commands, for example to run a shell script, which may start a Radiance-based [15] daylighting simulation. Figure 1 gives an overview of tools coupled to the BCVTB. The BCVTB also allows simulation tools to be coupled with hardware through its BACnet interface or its analog/digital interface [16].

The BCVTB is essentially a special configuration of Ptolemy II, with the addition of actors[2] and examples that are of interest to the buildings community.

The BCVTB has been used in several applications such as agent-based simulation [5], real-time simulation [3], controls of networked sensors and actuators [17], and performance prediction of HVAC systems [18].

As the BCVTB has been developed at the same time as the first version of the FMI standard, it contains its own API for co-simulation. This API is however much more limited than FMI and is not supported by all tools that export FMUs. Therefore, an FMU for co-simulation import interface has been added to the BCVTB. This interface allows the BCVTB to import simulation tools which have been exported as FMUs (Figure 2). This interface complies with the FMI for co-simulation API.

This new capability has several benefits:

- It allows users to couple the simulation tools shown in Figure 1, which are not all available as an FMU, to any other simulator that can be exported as an FMU for co-simulation.

[2]Actors are software components that execute concurrently and share data with each other by sending messages via ports.

DOI 10.3384/ECP14096311

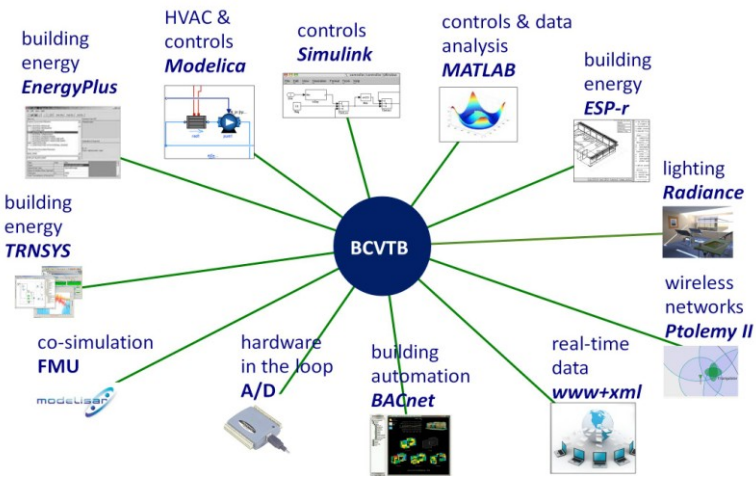

Figure 1: Simulation tools and hardware that can be coupled to the BCVTB.

- The BCVTB can be used as a master algorithm for co-simulation of FMUs using the Synchronous Data Flow domain of Ptolemy II.
- FMUs can be linked to hardware through the BCVTB.
- The BCVTB provides a graphical user interface for linking and simulating FMUs for co-simulation.
- The BCVTB allows synchronizing the simulation of FMUs to real-time.

The use of the BCVTB has the drawback that it introduces an additional transaction layer between the different simulators. As computing time is for most applications dominated by the simulation code inside the FMUs, the BCVTB middleware generally has no noticeable effect on the computing time. However, users need to have some familiarity with the use of the BCVTB. This increases the learning curve and could be a barrier for users who are familiar with one simulation tool and do not have resources to learn how to use this middleware. In addition, in some use cases, it may be more expedient to work directly in a domain-specific modeling environment. For example, when analyzing different façade systems, one may want to use a graphical user interface of a building simulation program that imports the model of the façade controller as an input/output block. Conversely, when developing a controller, one may want to take advantage of the visual editor and plotting capabilities of a Modelica modeling and simulation environment, while using an input/output block for a building model that takes as input the control action and outputs a sensor signal. For the first use case, we developed an FMU for co-simulation import interface in EnergyPlus, for the second use case, we developed a facility to export EnergyPlus as an FMU for co-simulation. The next two sections describe these technologies.

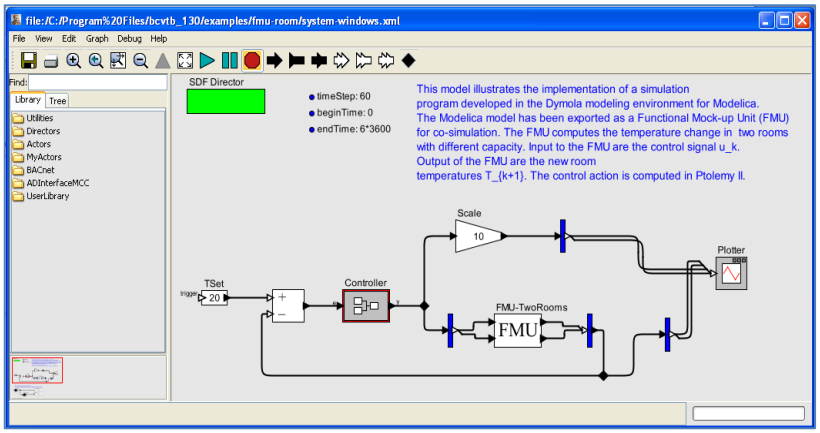

Figure 2: FMU for co-simulation import interface in the BCVTB.

3.2 FMU for Co-Simulation Import Interface in EnergyPlus

EnergyPlus is a whole building energy simulation tool. It is used by engineers, architects, and researchers for the modeling and simulation of energy use in buildings. EnergyPlus was not intended to be used for detailed modeling of airflow, dynamic response of heating, ventilation and air-conditioning equipment, and modeling of control systems other than scheduling of setpoints by simple supervisory control algorithms. To overcome these limitations, EnergyPlus has been coupled to various simulation tools such as COMIS [19], Computational Fluid Dynamics [20], MATLAB [21], or Modelica [22].

LBNL added to EnergyPlus 7.2 and higher an FMU for co-simulation import interface to allow the import of any simulation tool that is available as an FMU for co-simulation (Figure 3). This interface complies with the FMI for co-simulation API.

To facilitate the import of FMUs in EnergyPlus, we developed a utility called FMUParser. This utility is distributed with EnergyPlus and can be found in its *PreProcess* folder. When invoked, it unzips the FMU, extracts relevant information from the model description file of the FMU, and writes this information to a temporary EnergyPlus input file. A user can then complete this temporary input file to create the EnergyPlus input file. This parser has been developed so that users do not need to read the model description file, which can easily contain more than thousand lines of xml syntax.

To support the import of FMUs, we extended the data structure of EnergyPlus with four new objects [23]. These objects are used to map the inputs and outputs of the FMU to internal EnergyPlus variables once the FMU has been imported in EnergyPlus. We also implemented a set of C-functions which are distributed with EnergyPlus as a shared library. EnergyPlus uses these functions to call the FMI functions of the imported FMU.

Figure 4 shows how the FMU for co-simulation import interface was used to couple an HVAC system, implemented in Modelica and exported as an FMU, to a room modeled in EnergyPlus. This example is described in detail in [24]. The HVAC system computed sensible and latent heat exchange with the room, using the air inlet and outlet as the thermodynamic boundary. The room model computed the temporal evolution of the room air temperature and humidity, using the sensible and latent heat exchange as inputs to its energy balance. The FMU uses 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) to compute the sensible and latent heat exchange (QSensible, QLatent) which are sent to EnergyPlus through its outputs. EnergyPlus uses these values to compute the new room air temperature and humidity.

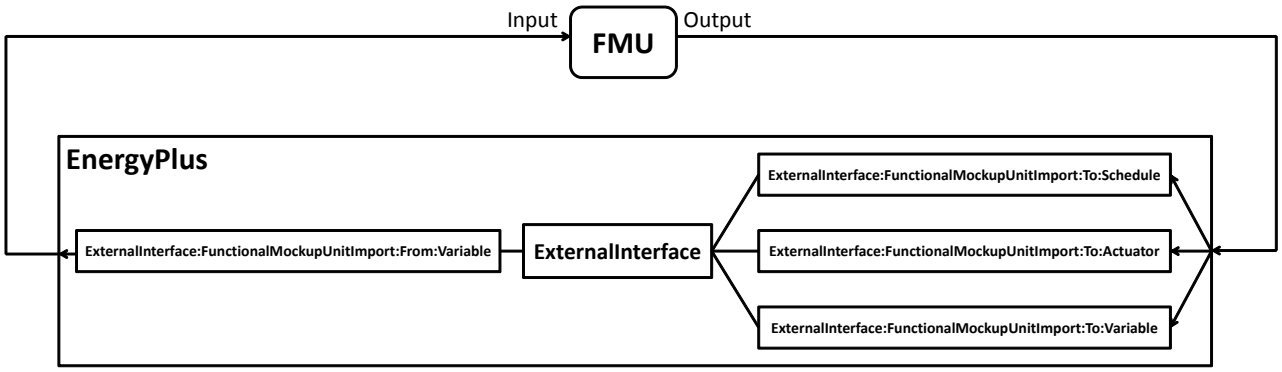

Figure 3: Importing an FMU for co-simulation in EnergyPlus through its ExternalInterface [25].

DOI 10.3384/ECP14096311

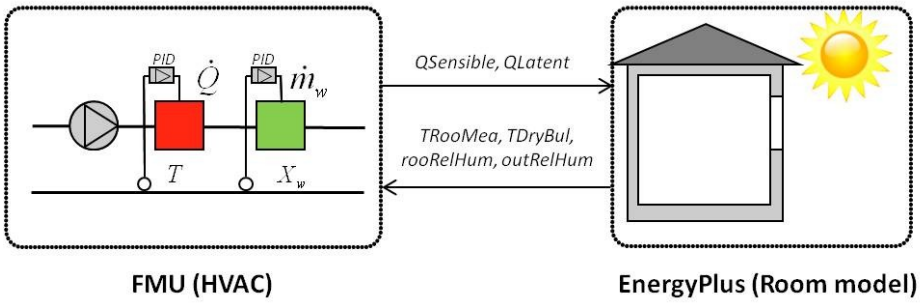

Figure 4: Linking an HVAC model developed in Modelica to an EnergyPlus room model using the FMU for co-simulation import interface.

3.3 FMU for Co-Simulation Export Interface of EnergyPlus

Although interfaces and middleware exist that facilitate the coupling of EnergyPlus with various software, they might not be widely used in the building simulation community since they still require users to be familiar with EnergyPlus so they can set it up and link it with other simulation tools. We thus exported EnergyPlus as an input/output block using the FMI standard. This allows importing EnergyPlus into any simulation tool that allows importing FMUs for co-simulation.

To export EnergyPlus 8.0 and higher as an FMU for co-simulation, we developed and released a software module called *EnergyPlusToFMU* [26], which exports EnergyPlus as an FMU for co-simulation. EnergyPlus implements in this configuration the FMI for co-simulation in the *CoSimulation_Tool* method.

Figure 5 shows how EnergyPlus is imported in Dymola as an input/output block which can be connected to other Modelica blocks.

To support the export of EnergyPlus as an FMU for co-simulation, we extended the data structure of EnergyPlus with four new objects [23]. These objects map the inputs and outputs of the FMU to variables that are internal to EnergyPlus.

Exporting EnergyPlus as an FMU for co-simulation can support various applications. For example, as described above, EnergyPlus may be used as an input/output block when designing a controller in a Modelica modeling and simulation environment or in MATLAB/Simulink. EnergyPlus building models may be linked to electrical grid and control models to design an electrical demand response controller for a campus that controls building electrical loads as a function of tariff, power quality and state of charge of batteries.

Table 1 shows a comparison between the different import and export capabilities. The table also lists their strengths and weaknesses. Although not exhaustive, this table should guide users in the selection of an import or export facility that is adequate for their specific application.

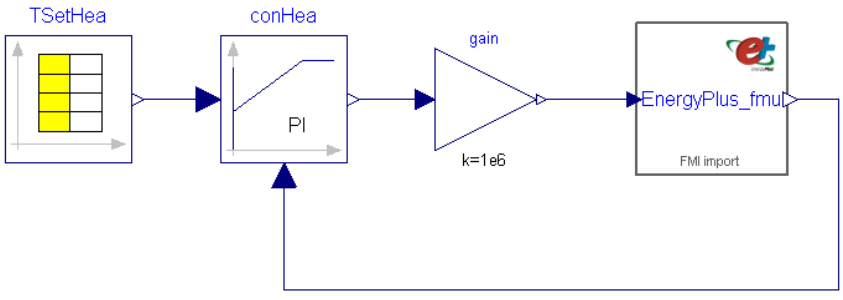

Figure 5: Coupling of an EnergyPlus model exported as an FMU with a PI-controller using Dymola.

Table 1: Comparisons between different technologies to couple simulators[3].

	Graphical user interface	Minimum knowledge required	Pros	Cons
BCVTB	Yes	BCVTB, and tools to be coupled.	Links to various tools and hardware. Reuse functionality of Ptolemy II.	Learning curve, transaction layer.
BCVTB (FMU Import)	Yes	BCVTB and FMU.	Links to various tools, FMU for co-simulation and hardware. Reuse functionality of Ptolemy II.	Learning curve, transaction layer.
EnergyPlus (FMU import)	No	EnergyPlus and FMU.	No need to learn new tool. May be able to use graphical user interface of EnergyPlus[4].	Tool to couple needs to be available as an FMU for co-simulation.
EnergyPlus (FMU export)	No	EnergyPlus and FMU.	EnergyPlus can be imported in other tools. Can use EnergyPlus as an input/output block inside a block diagram editor.	Import tool needs to support FMI for co-simulation.

[3]Simulator refers to simulation tool, system model, or tool exported as an FMU for co-simulation.
[4]Not all graphical user interfaces of EnergyPlus may support all EnergyPlus features and thereby support the FMU import interface.

DOI 10.3384/ECP14096311

The previous sections described the coupling of multiple simulators. In the next section, we describe an import interface that we implemented in an open framework for building controls that allows linking FMUs for co-simulation with different building management systems.

3.4 FMU for Co-Simulation Import Interface in Niagara^{AX}

Niagara^{AX} is a Java-based framework and development environment for creating internet-enabled products, device-to-enterprise applications and distributed internet-enabled automation systems. It is a commercial product from Tridium that is often overlaid to other building management systems to facilitate their interoperability (Figure 6). Niagara^{AX} uses a unified component model (Common Object Model) to transform the data from diverse external systems into uniform software components. These components form the foundation for building applications to manage and control the devices.

LBNL added to Niagara^{AX} an FMU for co-simulation import interface. This interface complies with the FMI for co-simulation API.

We selected the Niagara^{AX} framework because of its open-source architecture, which is based on Baja (Building Automation Java Architecture) [27] and its wide use in the buildings industry.

To implement the FMI interface in the Niagara^{AX} framework, we used JFMI [28], a Java wrapper for FMI, and created two new classes `BFMUService` and `BFMUComponent`. These classes are used in the framework to interface with imported FMUs for co-simulation.

The `BFMUService` is used by the Niagara^{AX} framework to process FMUs and to make their relevant information available to the framework. The `BFMUComponent` class represents an FMU instance. When instantiated, it appears in the Niagara^{AX} framework as an input/output block, which can be connected to other components of the Niagara^{AX} framework.

Figure 7 shows an FMU for co-simulation which has been imported in Niagara^{AX} as an input/output block. This block can then be linked to any other block available in the Niagara^{AX} framework.

Adding an FMU for co-simulation import interface enables various applications. For example:

- An HVAC designer may create a simulation model during the design of a building. She/he then exports the model as an FMU for co-simulation and imports it to Niagara^{AX}. In Niagara^{AX}, she/he links the model input to measured data. The design model can then be used to compute expected energy consumption, which in turn can be used to compare measured with expected performance. See [3] for such a use case.
- A researcher or product developer may develop a fault detection and diagnostic algorithm, test it on a simulation model, and then export the algorithm as an FMU for co-simulation. This FMU can then be linked through Niagara^{AX} with an actual building energy system.
- A researcher, product developer or advanced HVAC designer may develop and test an advanced control sequence in simulation, export it as an FMU for co-simulation, and import it to Niagara^{AX} to link it to an actual building.

We anticipate that the use of FMI in building management systems supports a robust and low cost implementation, and an error-free deployment of controls or FDD algorithms.

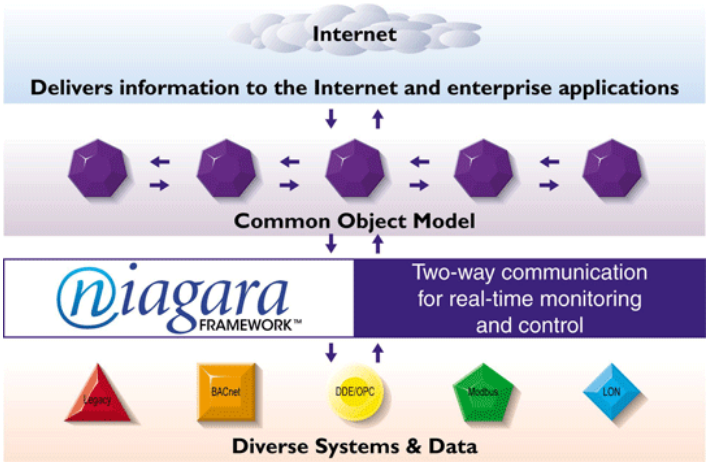

Figure 6: Niagara^{AX} framework (Courtesy: Tridium).

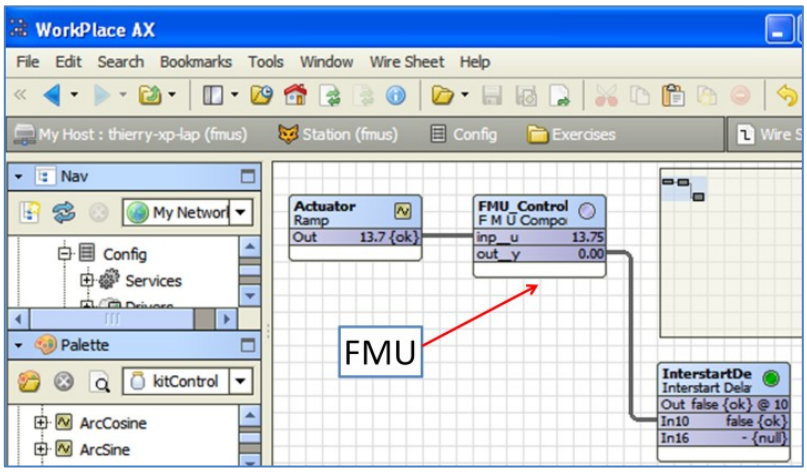

Figure 7: FMU for co-simulation import interface in Niagara^{AX}.

4 Conclusions

We anticipate the integration of FMU for co-simulation interfaces in the BCVTB and EnergyPlus to support a better simulation-based design and operation of buildings.

We believe FMI to be well positioned to become a de-facto standard for implementing, and deploying control sequences. We thus see the integration of an FMU for co-simulation import interface in building management system as a promising approach and a natural extension of its application to date.

Acknowledgments

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.

We thank Pacific Controls for sponsoring the development of a Functional Mock-up Unit for co-simulation import interface in Niagara^{AX}.

We thank Edward A. Lee, Christopher Brooks, David Broman, and Stavros Tripakis from the University of California at Berkeley for their support in developing JFMI and implementing the Functional Mock-up Unit for co-simulation import interface in the BCVTB.

References

[1] D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, C. O. Pedersen, *et al.*, "EnergyPlus: creating a new-generation building energy simulation program," *Energy and Buildings,* vol. 33, pp. 319-331, Apr 2001.

[2] A. Klein, J. A. Duffie, and W. A. Bechman, "TRNSYS - A transient simulation program.," *ASHRAE Transactions,* vol. 82, pp. 623-683, 1976.

[3] X. F. Pang, M. Wetter, P. Bhattacharya, and P. Haves, "A framework for simulation-based real-time whole building performance assessment," *Building and Environment,* vol. 54, pp. 100-108, Aug 2012.

[4] D. Gyalistras, C. Sagerschnig, and M. Gwerder, "A Multi-stage Approach For Building And HVAC Model Validation And Its Application To A Swiss Office Building," in *13th International Conference of the International Building Performance Simulation Association*, Chambery, France, 2013.

[5] D.-W. Kim, J.-H. Kim, S.-L. Park, K.-C. Kim, and C.-S. Park, "Traditional Vs. Cognitive Agent Simulation," in *13th International Conference of the International Building Performance Simulation Association*, Chambery, France, 2013.

[6] D. Broman, C. Brooks, L. Greenberg, E. A. Lee, M. Masin, S. Tripakis, *et al.*, "Determinate Composition of FMUs for Co-Simulation," in *Proc. of the International Conference on Embedded Software (EMSOFT 2013)*, Montreal, Canada, 2013.

[7] MODELISAR-Consortium. (2010). *Functional Mock-up Interface for Model-Exchange*. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_ModelExchange_v1.0.pdf

[8] MODELISAR-Consortium. (2010). *Functional Mock-up Interface for Co-Simulation*. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_CoSimulation_v1.0.pdf

[9] MODELISAR-Consortium. (2011). *Functional Mock-up Interface for Product Lifecycle Management*. Available: https://svn.modelica.org/fmi/branches/public/specifications/FMI_for_PLM_v1.0.pdf

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

[11] E. A. Lee, S. Neuendorffer, and G. Zhou, *System Design, Modeling, and Simulation Using Ptolemy II*, Claudius Ptolemaeus ed.: Ptolemy.org, 2014.

[12] Mathworks. (2013). *MATLAB & Simulink*. Available: www.mathworks.com/

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

[14] J. Clarke, "Moisture flow modelling within the ESP-r integrated building performance simulation system," *Journal of Building Performance Simulation,* vol. 6, pp. 385-399, Sep 1 2013.

[15] A. Grynberg, "Validation of Radiance," Lawrence Berkeley Laboratory, Berkeley LBID 1575, 1989.

[16] T. S. Nouidui, M. Wetter, Z. Li, X. Pang, P. Bhattacharya, and P. Haves, "BACnet and Analog/Digital Interfaces of the Building Controls Virtual Test Bed," in *12th International Building Simulation Association Conference*, Sydney, Australia, 2011.

[17] Y.-J. Wen, D. DiBartolomeo, and F. Rubinstein, "Co-simulation Based Building Controls Implementation with Networked Sensors and Actuators," in *BuildSys'11*, Seattle, WA, 2011.

[18] M. Trcka, M. Wetter, and J. Hensen, "An implementation of co-simulation for performance prediction of innovative integrated HVAC systems in buildings," in *11th International Conference of the International Building Performance Simulation Association*, Glasgow, Scotland, 2009.

[19] J. Huang, F. Winkelmann, F. Buhl, C. Pedersen, D. Fisher, R. Liesen, *et al.*, "Linking the COMIS multizone airflow model with the EnergyPlus building simulation program," in *Building Simulation 1999*, Kyoto, Japan, 1999, pp. 1065-1070.

[20] Z. Zhai and Q. Chen, "Performance of coupled building energy and CFD

simulations," *Energy and Buildings,* vol. 33, pp. 319-331, 2005.

[21] W. Bernal, T. Nghiem, M. Behl, and R. Mangharam, "MLE+: A Tool for Integrated Design and Deployment of Energy Efficient Building Controls," in *4th ACM Workshop On Embedded Sensing Systems For Energy-Efficiency In Buildings*, Toronto, Canada, 2012.

[22] C. Nytsch-Geusen, J. Huber, and Y. Nie, "Simulation-based design of PV cooling systems for residential buildings in hot and dry climates," in *13th International Conference of the International Building Performance Simulation Association*, Chambery, France, 2013.

[23] EnergyPlus. (2013). *Input Output Reference - The Encyclopedic Reference to EnergyPlus Input and Output*. Available: http://apps1.eere.energy.gov/buildings/energyplus/pdfs/inputoutputreference.pdf

[24] T. S. Nouidui, M. Wetter, and W. Zuo, "Functional Mock-up Unit for Co-Simulation Import in EnergyPlus," *Journal of Building Performance Simulation,* vol. 7, pp. 192-202, 2013.

[25] EnergyPlus, "External Interface(s) Application Guide, Guide for using EnergyPlus with External Interface," 2013.

[26] T. S. Nouidui, D. M. Lorenzetti, and M. Wetter. (2013). *EnergyPlusToFMU*. Available: http://simulationresearch.lbl.gov/fmu/EnergyPlus/export/index.html

[27] Tridium and Sun-Microsystems. (2000). *Baja: A Java - based Architecture Standard for the Building Automation Industry*. Available: http://www.automatedbuildings.com/wsim/Baja_White_Paper.pdf

[28] C. Brooks, E. A. Lee, M. Wetter, T. S. Nouidui, D. Broman, and S. Tripakis. (2012). *JFMI - A Java Wrapper for the Functional Mock-up Interface*. Available: http://ptolemy.eecs.berkeley.edu/java/jfmi/

 DOI 10.3384/ECP14096311