Development of a simulation-based controls framework for implementation of control algorithms for complex fenestration systems

Thierry S. Nouidui, Andy McNeil, Eleanor S. Lee

Building Technologies Program, Environmental Energy Technologies Division, Lawrence Berkeley National Laboratory, Mailstop 90-3111, 1 Cyclotron Road, Berkeley, CA 94720 USA

Abstract

Effectively controlled dynamic windows can substantially reduce the energy consumption of buildings. Unfortunately, modular and extensible frameworks for testing and evaluating window system control algorithms that include the effects of thermal mass are missing in the research community.

This paper describes a modular and extensible simulation-based framework that uses different simulation tools such as EnergyPlus, Modelica, Radiance, and the Building Controls Virtual Test Bed (BCVTB) to develop and evaluate the performance of integrated façade control strategies. We present a simulation framework and a proof-of concept application using the framework to control a venetian blind in a physical test cell in order to reduce its zone thermal load based on solar inputs and internal gains measured at the test cell.

1 Introduction

Window systems, consisting of glazing and shading systems, are key factors of heat gains of buildings. They contribute substantially to overall energy performance of buildings. NREL states that windows contribute about 30 percent to building heating and cooling electrical loads nationwide (NREL, 2007).

Traditional window systems are static with limited interaction with envelope or HVAC systems. Emerging window systems are more dynamic (e.g., switchable glazing, automated venetian blinds) and the potential energy savings of use of such systems to reduce lighting, or HVAC energy consumption while ensuring visual comfort is becoming a key research activity in the buildings sector.

While the best solution to minimize the cooling energy in a room is to completely close the venetian blind or switch the windows to a low-transparent state, when daylight is considered, there is a tradeoff between energy savings from reducing cooling and electric lighting. Additionally, from an occupant's point of view, keeping the venetian blinds closed during all cooling hours is undesirable. Thus we need to find a balance between controlling for daylight and controlling for cooling loads. This can be achieved by using smart control algorithms, which can determine the optimal state of the dynamic facade system that will minimize energy consumption of both electric lighting and cooling while maintaining visual comfort in the space. The validity of the control algorithm strongly depends on the input received, thus it is important to have a framework that provides reliable inputs for the controls.

2 Objective

The objective of the current work is to develop a framework that leverages the capability of many simulation tools to create a simulation based platform which can be used to 1) accurately modeled whole buildings and 2) support the development, implementation and testing of control algorithms where different domains of buildings physics such as envelope, the HVAC are managed synergistically. The framework should support the modeling of building thermal mass, which is critical when developing thermal mass controls strategies, it should be easily extendable and

reconfigurable to support rapid prototyping of control algorithms. The framework should be able to use real-time sensors data for online simulations and interface with hardware remotely through the internet.

3 Challenges

This work was motivated by the need to solve challenges faced in the development and evaluation of control algorithms of dynamic façade systems integrated in buildings with thermal mass to minimize HVAC energy use.

Our simple example illustrating the challenges was to implement a control algorithm which modulates the slat angles of a venetian blind to find the position which minimizes the energy consumption of the test cell where the blind is installed. Solving the problem requires:

1) developing simulation model of the test cell which takes into account physical effects such as thermal mass, solar radiation for different blind positions,
2) calculating the HVAC energy use for all blind positions at every simulation time step;
3) saving state variables of the blind position that led to minimum energy use; and,
4) reinitializing the simulation model with these variables for the next simulation time step.

The main challenges that we faced were as follows:

a) There is no whole building simulation tool which can simultaneously and accurately model i) the heat transfer through envelope and within the room, ii) the impact of dynamic façade system such as a venetian blind on the solar radiation distribution within the room, and iii) HVAC systems for heating and cooling; and
b) The control algorithm relies on the capability of the simulation model to reinitialize its state variables during run-time which is not supported by legacy simulation tools such as such as EnergyPlus (EnergyPlus, 2012) or DOE-2 (eQuest, 1998). It is not possible to explicitly specify the values of state variables that are key in our control algorithm. Thus, we need to use simulation tools which support this capability.

4 Solution Approach

A framework for rapid prototyping of control algorithms that addresses the challenges specified in the previous section, will be capable of accurately modeling the building envelope, dynamic façade systems, and HVAC system and will enable testing and evaluation of control algorithms.

The proposed framework relies on the decomposition of the building into domains. Each domain uses a tool that best suits the simulation needs of the domain. A co-simulation environment links the domains to provide an integrated simulation.

The proof of concept implementation of the framework contains five domains:

1) Building envelope for the geometry and heat transfer through envelope
2) Solar radiation distribution within room and façade system
3) HVAC systems to calculate HVAC energy use
4) Control algorithm to interface façade system
5) Hardware (if the building possesses sensors that can be read or actuators that can be interfaced). The dynamic façade systems investigated in this current version of the framework was limited to venetian blinds.

4.1 Building envelope

We initially considered using EnergyPlus, a whole building energy simulation tool, to model the geometry and the heat transfer through the envelope. However, EnergyPlus is unable to reinitialize state variables during runtime. Reinitializing state variables is critical when considering thermal mass in control decisions. We decided to use the room model of the Modelica *Buildings* Library (Wetter et al., 2011).

The room model of the Modelica *Buildings* library simulates heat transport processes within rooms and through the building envelope. This model can model an unlimited number of opaque and transparent surfaces or entire buildings. The room model accommodates the following physical processes:

a) Transient or steady-state heat conduction through opaque surfaces, such as walls.

b) Heat transfer through glazing systems, frame, interior or exterior shading layer including solar radiation, infrared radiation from ambient environment, heat conduction and heat convection.

c) Convective heat transfer between the room (inside) air and room-facing surfaces using either a constant heat transfer coefficient or a temperature-dependent heat transfer coefficient.

d) Convective heat transfer between the outside air and outside-facing surfaces using either a constant heat transfer coefficient or a variable heat transfer coefficient as a function of wind-speed, wind-direction and temperature.

e) Solar and infrared heat transfer between the room enclosing surfaces.

f) Temperature, pressure and species balance equations inside the room volume.

Figure 1 shows a Modelica room model composed of six main parts. The parts are component models from the *Buildings* library.

- Part 1 labeled as heat sources in the Figure defines radiative, convective, and latent heat added to the room.
- Part 2, shading signal, interprets the signal for the shading layer. For example, if the shade signal is 0.2, then this model accounts for 80% of the glass that is not behind the shade.
- Part 3, material properties, defines the material properties used for the building enclosures.
- Part 4, weather file reader, reads *.epw weather files. It is also possible to specify user specified weather conditions which can be constant.
- Part 5, room air, defines the building envelope. This room air model is used to parameterize the room including geometry as well as building construction.
- Part 6, boundary conditions, defines conditions for surfaces exposed to boundary conditions.

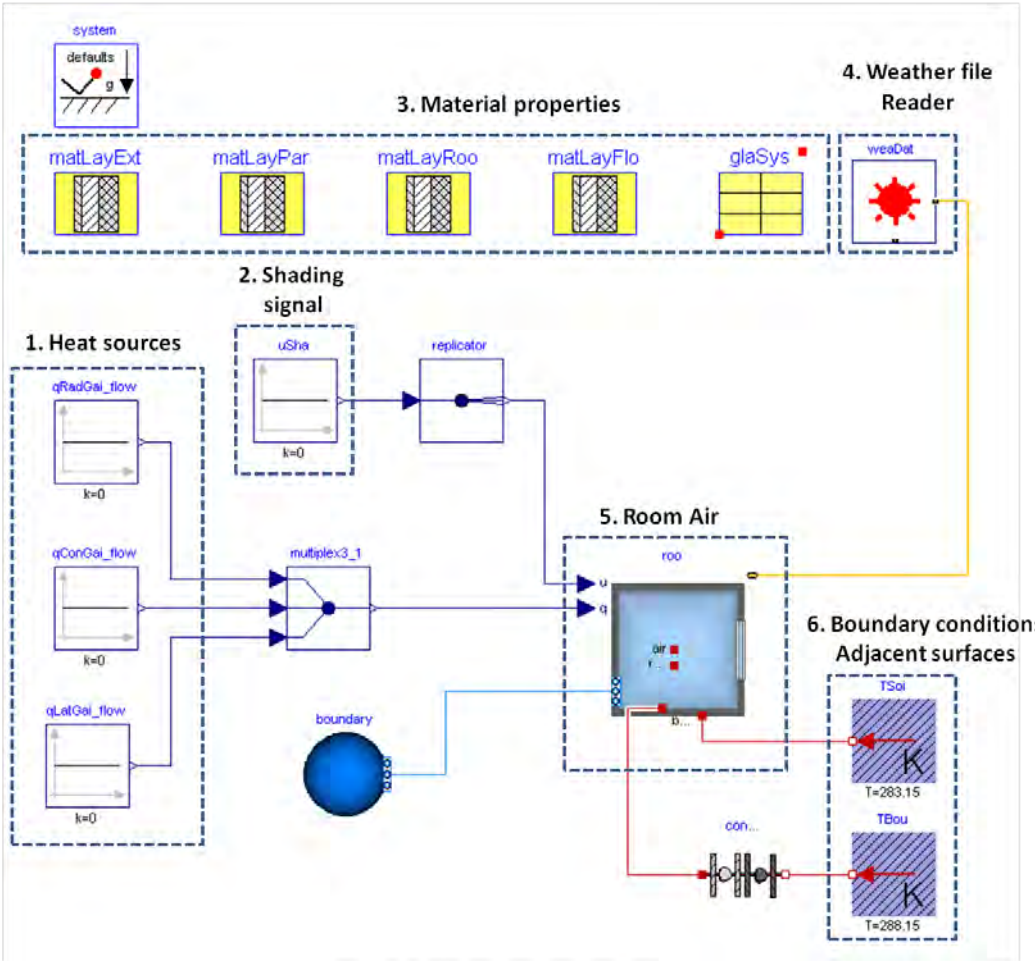


Figure 1: Example of a room model in Modelica *Buildings* Library

The *Buildings* library contains a window model, which was validated as part of this work (Nouidui et al., 2012).

The Modelica *Buildings* library room model does not contain a venetian blind. To emulate the effect in the heat and window balance calculations, we used Radiance (Ward, 1985) to compute the solar radiation distribution in layers of window and the shading system. These data were then passed to the room model, which used them in the window and heat balance calculation. A description of the solar radiation distribution method is given in the next section.

This simulation model is unable to accurately model convective heat transfer coefficient between the exterior window pane and the venetian blind for different blind positions. This paper does not explicitly address this limitation requiring further investigation.

4.2 Solar Radiation Distribution

Façade systems such as venetian blinds reflect solar radiation in a non-specular manner. In order to accurately consider their impact on the heat and window balance calculations of the room model of the Modelica *Buildings* library, we use the method proposed in (Klems, 1994) to compute with Radiance the absorbed solar radiation in shading layers as well as the incident solar radiation on interior opaque surfaces of building enclosures. This method requires the modeling of the test cell in Radiance for the computation of the solar radiation distribution data. These data are then use by the room model for the window and heat balance calculation within the room.

4.3 HVAC System

We leverage in our framework for the HVAC systems the component models of the Modelica *Buildings* library. This library more than 200 component models which can be used for rapid prototyping of building energy and controls systems, modeling of large HVAC systems with their controls, development of model based controls, modeling of controls sequences, fault detection and diagnostics.

By leveraging the component models of the *Buildings* library, we also i) leverage the key benefits of the Modelica language which separate the concerns between simulation and modeling allowing high degree of model reuse, graphical "plug and play" modeling, the integration of models from different domains, the coupling of models with different time dynamics, the use of state-of-the art numerical solvers and ii) leverage the models of the *Buildings* library allowing us to create real HVAC systems such as variable air volume flow system with terminal reheat, chiller plant, hydronic heating systems with real controls rather than idealized controls which are in simulation tools such as EnergyPlus or DOE-2.

4.4 Control algorithm

In current version of the framework, we used Python to implement control algorithms. Python was selected because it is a powerful open-source high level programming language with a large library with mathematical functions that can be used to implement control algorithms. In a simulation scenario, the control algorithm gets its inputs from external simulation blocks and returns a control signal which can then be sent to a virtual or a real hardware. In future work, we should extend the framework to use Modelica for controls since it supports advanced control algorithms which involves state machines and are suitable for model based control operations.

4.5 Hardware

In our framework, it is possible to get access to sensor data such as sky data or plug loads data through a wide area network. In this case, python scripts retrieve the data via http, parse them into numerical values and parameterize the simulation model. It is also possible to convert a control signal from our simulation into a data string, which can then be sent via http to hardware at a test cell to actuate a shading device.

4.6 Co-Simulation of physical domains

To link the domain specific tools which are used to different the physical domains, we use the Building Controls Virtual Test Bed (Wetter, 2011).The BCVTB is a powerful simulation tool which is based on Ptolemy II (Brooks et al., 2007) from UC Berkeley. The BCVTB allows experts users to link different simulation tools during run-time. It also allows to

link simulation tools with real hardware. The BCVTB orchestrates the start of the simulation, and manages the data exchange between the tools. The BCVTB can synchronize its time to real-time, which is needed for real-time model based operations.

In our co-simulation framework, the BCVTB uses the *SystemCommand* actor to call scripts that start the different simulation tools process the different simulation results and terminates the simulation tools.

A typical sequence of operations of the framework is to use this actor to call a script that

(1) obtains boundary conditions such as weather information required to perform a simulation

(2) pre-computes solar radiation distribution based on boundary conditions

(3) uses the results of (1) and (2) to parameterize a Modelica room model and run a simulation

(4) saves the state variables of the simulation run which can then be used if needed.

5 Application

The proof of concept implementation of our simulation-based controls framework controlled in real-time an window shading device (external venetian blind) of one room of the window test bed facility at LBNL (). The window test bed, an advanced window test facility with three identical test rooms, allows real world test of façade systems and evaluation of control strategies. Each of these test cells has a floor area of about *14* m^2, a room volume of about *47*m^3 and a south facing window. The window shading device of the left test cell is an exterior venetian blind which is controlled via our controls framework. The performance of the controlled test cell is compared to the middle test cell which has an exterior static venetian blind, and to the right test cell which has an interior static venetian blind.

Figure 2 The Advanced Window Test Facility at the Lawrence Berkeley National Laboratory

The room air temperature of the test cells is controlled to a fixed temperature using a constant volume dual-duct system which has a chiller for cooling and an electric heater for heating. There are several sensors in the test cells which measure room air temperatures, exterior glass surface temperatures at the upper and lower window surfaces, plug loads, lighting loads, fan loads as well as transmitted solar irradiation at the upper and lower window surface. There are also several sensors located outdoors to measure external environmental conditions, such as solar irradiances, outdoor temperature, and wind speed (see Figure 3).

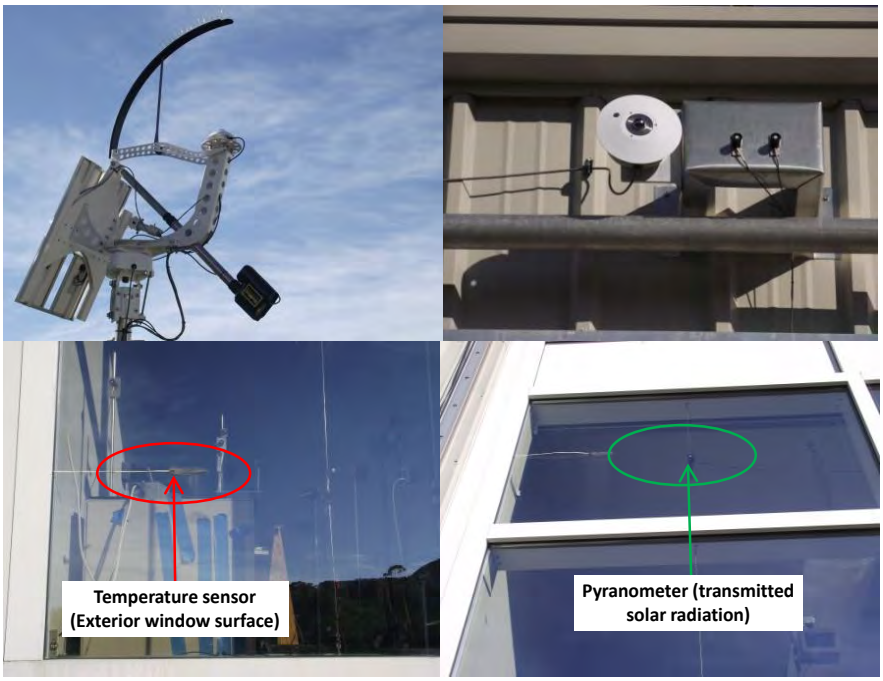


Figure 3 Instrumentation used at the Advanced Windows Test Facility

In this application, the room model of the *Buildings* library is used to model the left test cell. The window system installed in the test cell is a double pane window. The control framework remotely controls the exterior venetian blind by adjusting the slat tilt angle and retracting or deploying the blind.

The control algorithm implemented in the framework aims to reduce the zone thermal load of the test cell while keeping a constant indoor dry-bulb temperature of 24°C. The framework achieves the objectives by computing the HVAC energy required to maintain the set-point temperature for ten discrete blind configurations. The blind configuration requiring the least HVAC energy use is selected and converted into a control signal sent to the blind actuation hardware.

The evaluation of the performance of the control algorithm is achieved by comparing the zone thermal loads of three test cells. In our experiment, the interior and exterior static venetian blind are set to 35 degree blocking angle which corresponds to 10 degree tilt angle. This represents one common configuration for blinds generally set by users.

5.1 Available information

There are several sensors installed at the Advanced Windows Test Facility, our framework used just the following sensors:

- Global solar irradiation,
- Diffuse solar irradiation,
- Direct solar irradiation,
- Global vertical solar irradiation,
- Atmospheric infrared irradiation,
- Wind speed,
- Dry bulb outdoor temperature,

In addition to these data, following measurements specific to an individual test cell were recorded:

- Transmitted solar irradiation
- Lighting loads
- Plug loads
- Fan loads
- Shade height
- Slat angle

To make these data available through the remotely, there was an internet protocol defined at the Advanced Test Facility which enabled real-time access to the data and control.

5.2 Overview of the Controls Framework

At the beginning of the simulation, the BCVTB (see Figure 4) gets the start and end time of the simulation (a), the test cell number (b), the test cell password (c), and the time step that are pre-defined by the users (d). It then calls the *SystemCommand* actor (e) that invokes the time step calculation.

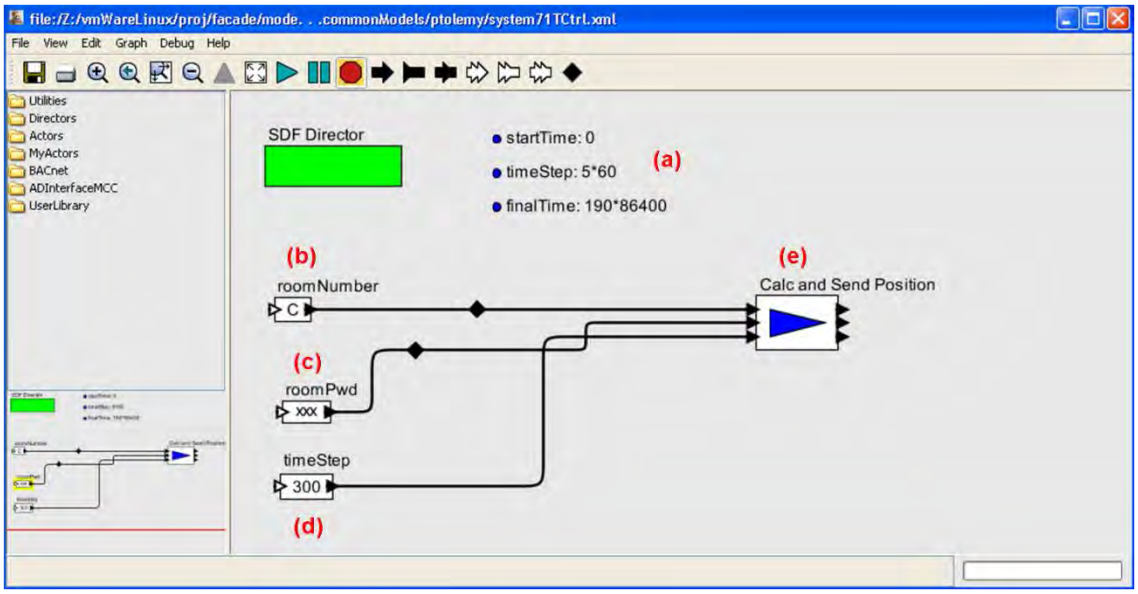


Figure 4 Configuration of the controls framework

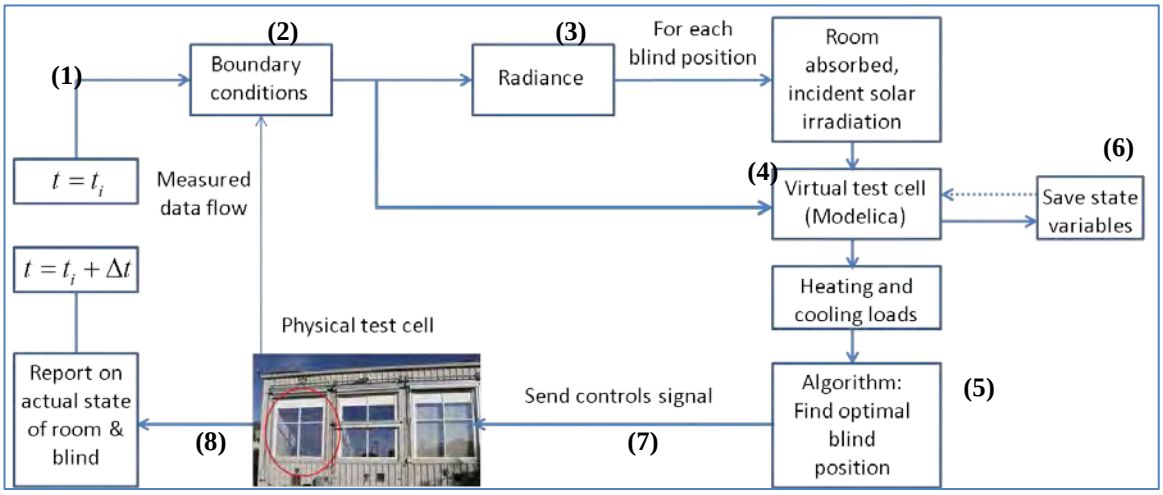


Figure 5 Simulation-based controls framework used to control one of the test cells of the user facility

Figure 5 shows the schematic of the time step workflow to determine the optimal blind position in a simulation time step. The workflow consists of 8 steps, which involve co-simulation using different simulation tools and communication between hardware and software. The entire process is managed by the Building Controls Virtual Test Bed. In our implementation, the simulation runs in real-time with a time step size of 5 minutes. Shorter or longer simulation time step can be set in the framework according to the application domain.

The eight steps of the time step calculation are as follows:

1. The BCVTB uses a Python (Van Rossum, 1991) script to send requests through the internet to get the current clock-time, as well as the data string which contains weather data, as well as plug, fan and lighting loads which are measured in the test cell.

2. The BCVTB writes a weather file and a load file. The weather file contains measured weather data including diffuse solar irradiation on the horizontal surface, direct solar irradiation, the atmospheric infrared solar irradiation, outdoor dry-bulb temperature, and wind speed. This file is written in the format used by the room model of the Modelica *Buildings* library. The load file contains the sum of plug, fan and lighting loads.

3. The BCVTB starts a Perl (Wall, 1987) script which invokes Radiance to calculate the incident solar radiation on interior wall surfaces of the test cell and solar irradiation absorbed in glass layers and shading layer of the window system for multiple possible blind positions. Radiance uses in its calculation measured sensors data such as global and direct solar irradiation to determine the solar radiation distribution. The results of the Radiance calculation are written in files read later by the Modelica room model for its window and heat balance calculations. This approach allows emulating the effect of the blind in the room model of the *Buildings* library, which does not support the modeling of venetian blinds yet. In our current simulation configuration, we consider 11 positions. The first position is with the blind fully retracted. The second to the 11^{th} position are with the blind set at to achieve blocking angles of 40, 35, 30, 25, 20, 15, 10, 5, 0, and -5 degrees. A blocking angle of -5 degrees is fully closed.

4. The BCVTB starts a script that simulates multiple instances of the Modelica room model using Dymola (Elmqvist, 1978). Each model simulates the blind set at one of the twelve positions. The model is initialized with the weather file, the load file, and the incoming and absorbed solar irradiation pre-calculated by Radiance. Figure 6 shows a screenshot of the Modelica implementation of the test cell. This model consists of 7 parts: part 1 defines the heat sources which are read from the load file, part 2 is the PI controller for heating, part 3 is the PI controller for cooling, part 4 models the building envelope, part 5 represents the material properties of the building envelope, part 6 provides the weather data, and part 7 computes the infiltration in the test cell. Each component model shown in the graph implements a specific equation. For instance a component model (Gain) which outputs the product of a gain value with the input signal will have following graphical and textual in Modelica.

5. The BCVTB calls a Python script that collects the Modelica simulation results for different blind positions and determines the optimal position, which has the lowest thermal load. This position is then written in a file named “chosenposition.txt”.

6. The BCVTB calls a script that saves the state variables of the room model with the optimal blind position. These state variables will be used as initial conditions in the next time step. The capability of Modelica to easily reinitialize state variables is one of the key reasons why Modelica is suitable for simulation-based controls operations.

7. The BCVTB calls a script that reads the optimal blind position from the “chosenposition.txt” file, converts it into a controls signal, and sends it through the internet to the actuator to set the position of the blind.

8. Finally, the hardware sends through internet the controls signal that has actually been set to the BCVTB which writes it in a log file. The BCVTB then pauses until the next time step is reached and restarts the process.

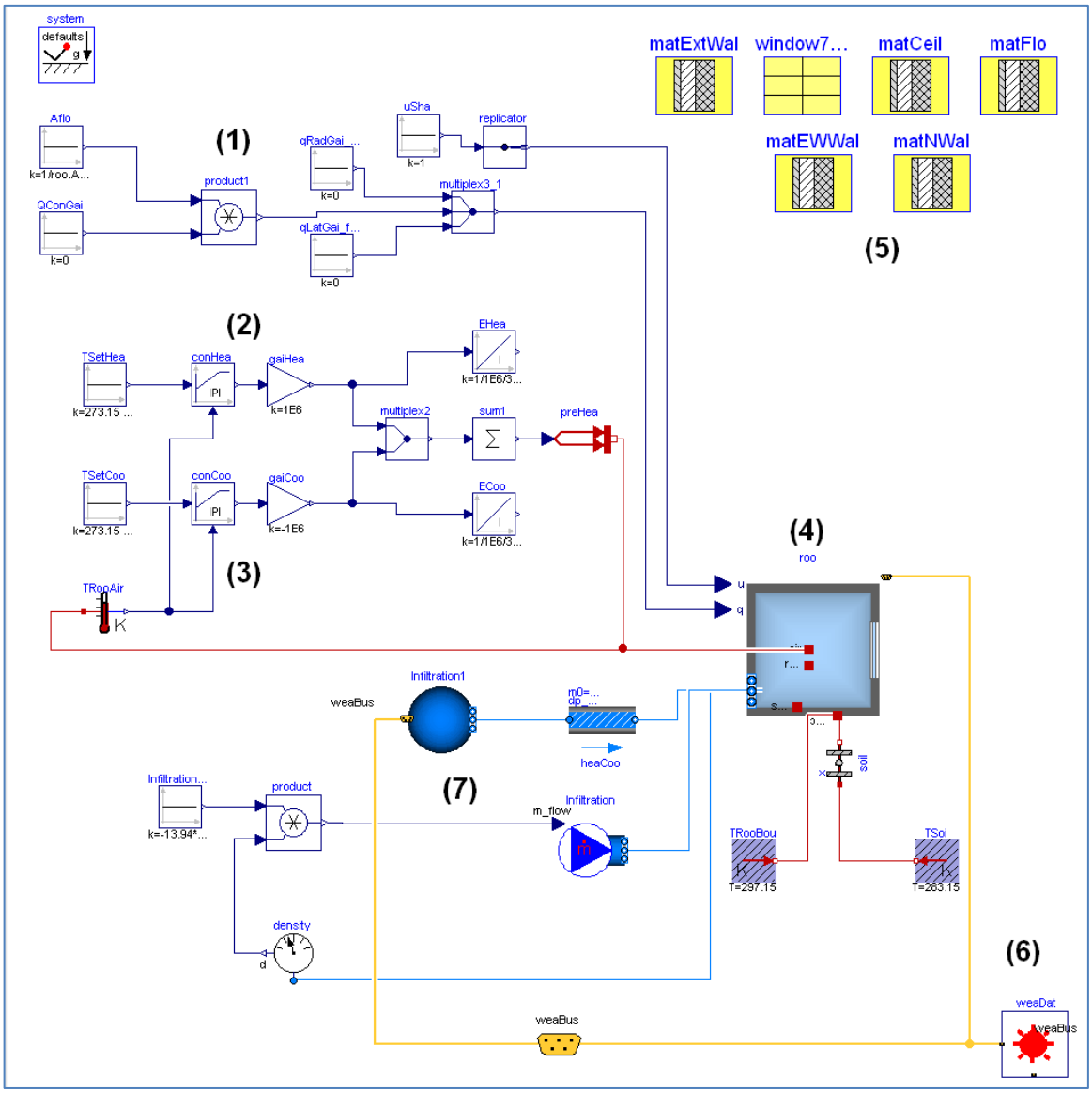
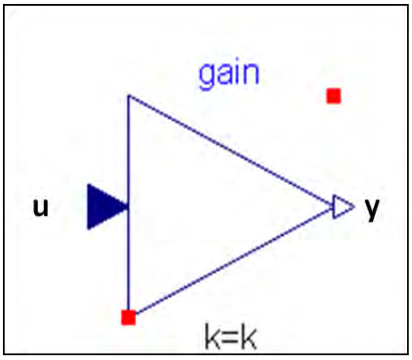


Figure 6 Modelica implementation of the physical test cell




```
block Gain "Output the product of a gain value with the input signal"
  parameter Real k(start=1, unit="1") "Gain value multiplied with input signal";
public
  Interfaces.RealInput u "Input signal connector"
    annotation (Placement(transformation(extent={{-140,-20},{-100,20}},
      rotation=0)));
  Interfaces.RealOutput y "Output signal connector"
    annotation (Placement(transformation(extent={{100,-10},{120,10}},
      rotation=0)));
equation
  y = k*u;
end Gain;
```

Figure 7 Graphical (left) and textual (right) representation of the Gain – model in Modelica

5.3 Simulation results

The test setup was cycled into the facility for shorter and longer testing periods depending on availability of the test cell. Both the control algorithm and the component models of the framework were continuously revised based on monitored results. After the control algorithm was operating satisfactorily, the data from 12 uninterrupted days from 07/16/2012 to 07/28/2012 were used to evaluate the performance of the controls framework.

The test facility was not designed to enable direct comparison of HVAC energy consumption thus to evaluate the performance of the controls framework, we compared for two exemplary days (07/20/2012-07/22/2012) the zone thermal loads derived from measurements obtained in the test cells. We determined the zone thermal load based on the method proposed by Klems (Lee E., 2006) and following equation:

$$ZoneThermalLoad = abs(CoolW - HeaW - FanW - LightW - PlugW) \quad (1)$$

With:

$$CoolW = \dot{m}_{flow} * c_w * (T_{out} - T_{in}) \quad (2)$$

T_{out}, T_{in} being the water outlet and water inlet temperature measured at the test cells, c_w being the heat capacity of water and $\dot{m}_{flow}$ being the mass flow rate which is derived from the volume flow rate measured in the test cells. $HeaW$ was the power of the electric heater measured in the test cells, and *FanW*, *LightW*, *PlugW* were the fan, lighting and plug loads measured in the test cells.

Figure 7 and Figure 8 show for the two exemplary days the outdoor dry bulb temperature as well as the global solar irradiation measured at the test facility during this experiment.

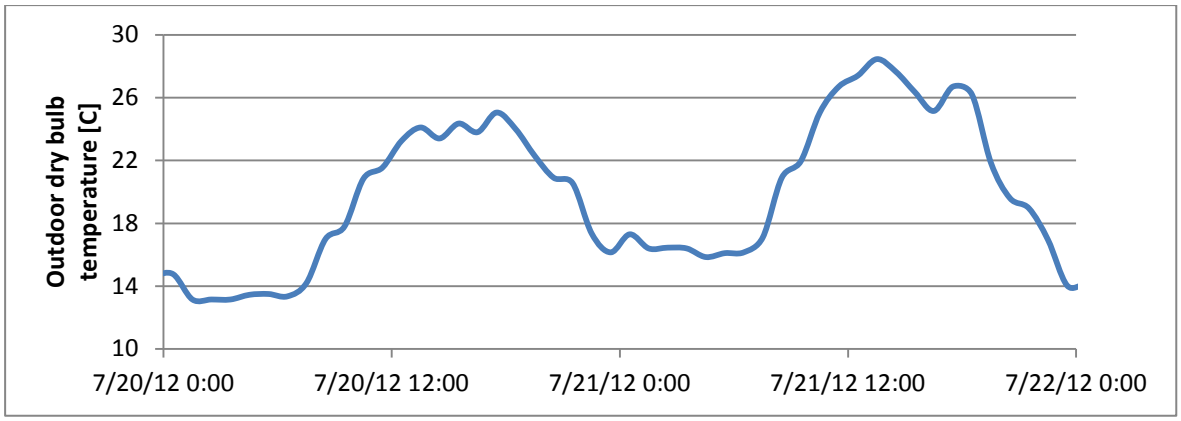


Figure 7: Outdoor dry bulb temperature measured at the Advanced Windows Test Facility

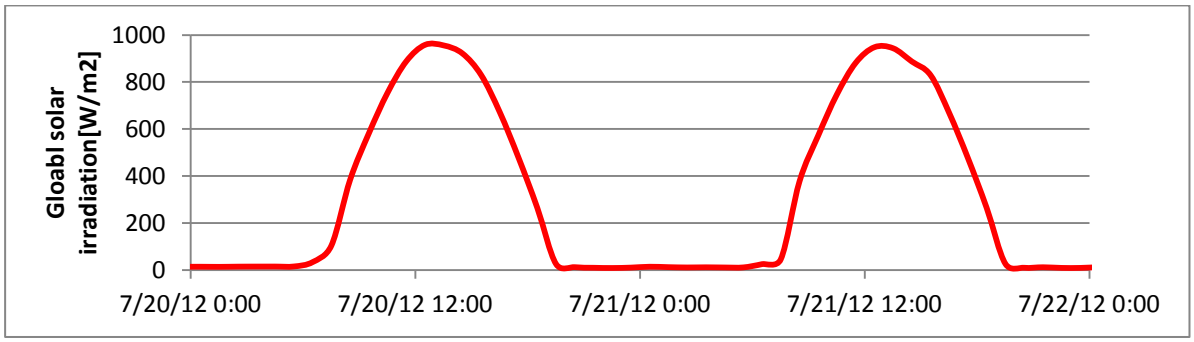


Figure 8: Global solar irradiation measured at the Advanced Windows Test Facility

Figure 9 shows the value of the zone thermal load (*ZoneThermalLoadTotal*) derived from measurements obtained in the three test cells. This figure shows that the controlled test cell (Room C) performed much better than the test cells with static exterior (Room B) and interior blinds (Room A). The use of the controlled test cell can reduce within day time up to twice and triple zone thermal load in the peak compared to the test cell with the exterior and interior static blind respectively. However at night time, the response of the controlled test cell did not lead to optimal performance. We believe the suboptimal performance was caused by 1) uncertainty in the model particularly in the limitation of the simulation model to perfectly model convective heat transfer coefficients between glass and blind for different blind positions which effect is important during night and 2) uncertainty in the measurements which were not allowing direct comparisons of HVAC energy use. Nonetheless, the simulation results demonstrate that the implemented framework work, and potential energy savings due to the controls can be achieved.

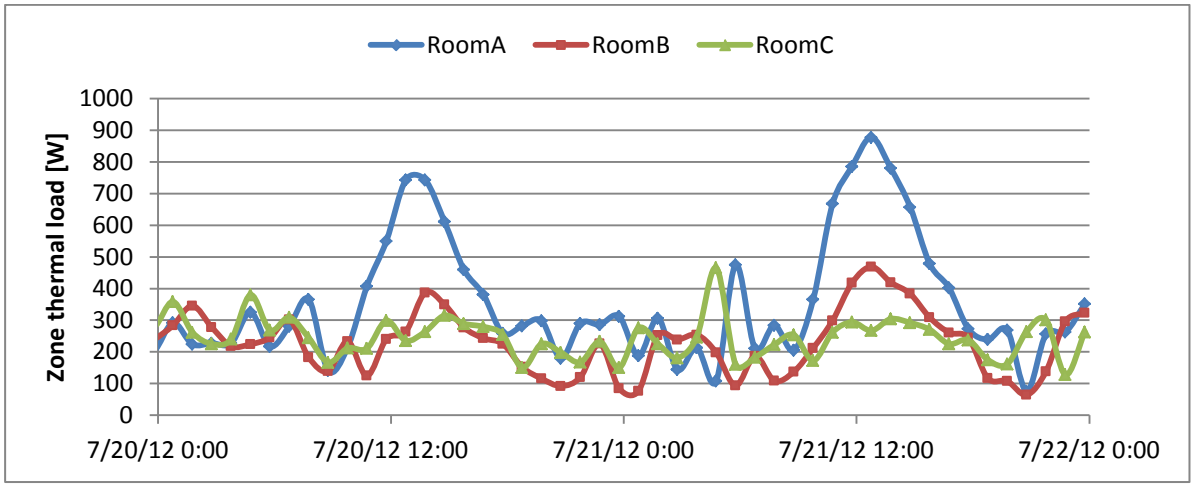


Figure 9: Zone thermal load measured at the advanced windows test facility

In the experiments conducted, the control algorithm set the blind to be constantly closed as shown on Figure 10 because of the high solar intensity. This was leading to a reduction in zone thermal load compared to the two other test cells where the tilt angles of the blinds were set to 10 degrees. Figure 10 also shows for the 07/21/2012 how the requested blind position (tilt_angle_predicted) was not set (tilt_angle_measured). This indicates some of the challenges faced when transferring and converting the simulation results in the real physical world.

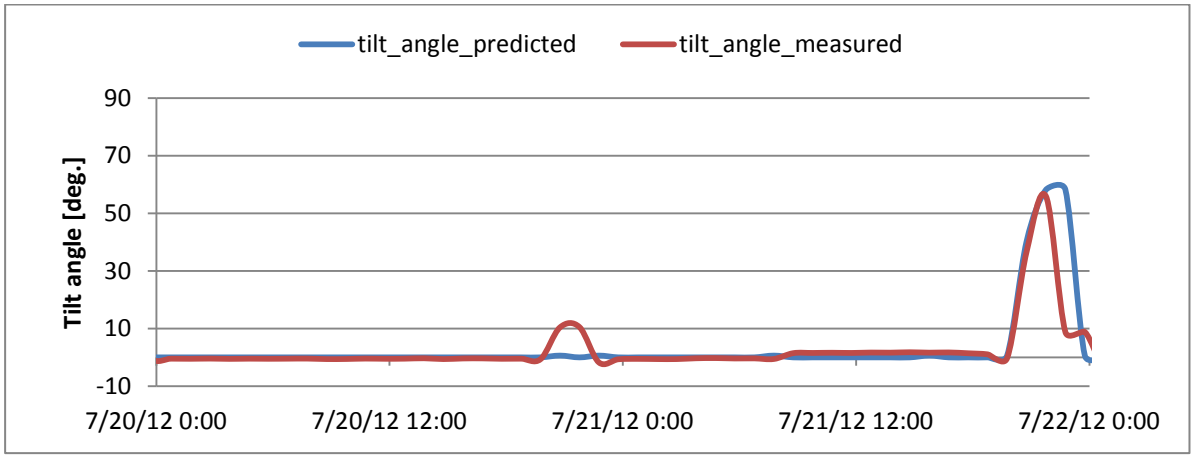


Figure 10: Tilt angle predicted and measured in the controlled test cell

6 Lessons learned

6.1 Reinitialization of state variables

Reinitialization of state variables during run-time can be a serious issue when using thermal model as part of control algorithms that test multiple scenarios. Reinitilization requires the model to expose its state variables to the simulation environment where it is used and also requires the simulation environment to be able to access and change these variables during runtime. Legacy simulation tools such as EnergyPlus or DOE-2, which were conceived mainly for annual simulations, were not designed for this application and should not be used in this context. Simulation tools or languages such as Modelica or MATLAB (MathWorks, 1994-2012) are more appropriate in the context of this application.

6.2 Network

During our framework testing, we experienced several challenges with the network reliability. Since the framework obtains real time data at every time step, loss of connectivity causes serious problems. After losing network connectivity the control algorithm has to be reinitialized with historical data stored at the test cell. A solution would be to enable the framework to access the historical data for automatic reinitialization.

6.3 Hardware-Latency

Another issue identified while testing the framework was the latency in the actuator used to set the blind to a position requested. The blind was not always actuated to the position selected by the framework because of a mechanism implemented to prevent mechanical damage from short cycling. Thus it was important to keep track and consider the position which was actually set by the hardware when analyzing experimental results. Implementing the anti-short cycling mechanism in the controls framework and disabling the mechanism in the blind controller could resolve this issue.

6.4 Complexity

Although, current framework is very powerful since it relies on the effective use of many domain-specific tools different physical domains. The framework is also needlessly complex, requiring the user to have knowledge of different simulation tools such as the Buildings Control Virtual Test Bed, Radiance, Modelica, programming languages Python and Perl. Simplification efforts would improve the framework's practicality. One solution path could be to have preprocessing tools that generate models for the different physical domains. These models could be packaged as a Functional Mockup Unit (MODELISAR, 2008-2011) and linked through the BCVTB. Functional Mock-up Units are self-contained models or programs implemented in Functional Mock-up Interface (FMI). The FMI defines an open interface for model or simulation exchange to be implemented by an FMU. The FMI functions are used by a simulator to create one or more instances of the FMU, called models, and to run these models, typically together with other models. An FMU is distributed in form of a zip-file. This zip file contains:

- an XML file containing among other things the definition of the variables used by the FMU
- all the equations used by the model (defined as a set of C functions)
- optional other data, such as parameter tables, user interface, documentation which may be needed by the model.

7 General Recommendation

Figure 11 shows a use case of the simulation-based controls framework. In this use case a controls engineer would like to test a new control algorithm for façade systems in physical test cell. The controls engineer could 1) use the simulation-based controls framework to implement the model of the physical test cell if it doesn't exist yet, 2) implement the control algorithm, 3) test verify and optimize the control algorithm in the simulation environment using historical and reference data, 4) do the full scale testing in the physical test cell, 5) evaluate and eventually deploy the algorithm that can be extrapolated to other climates.

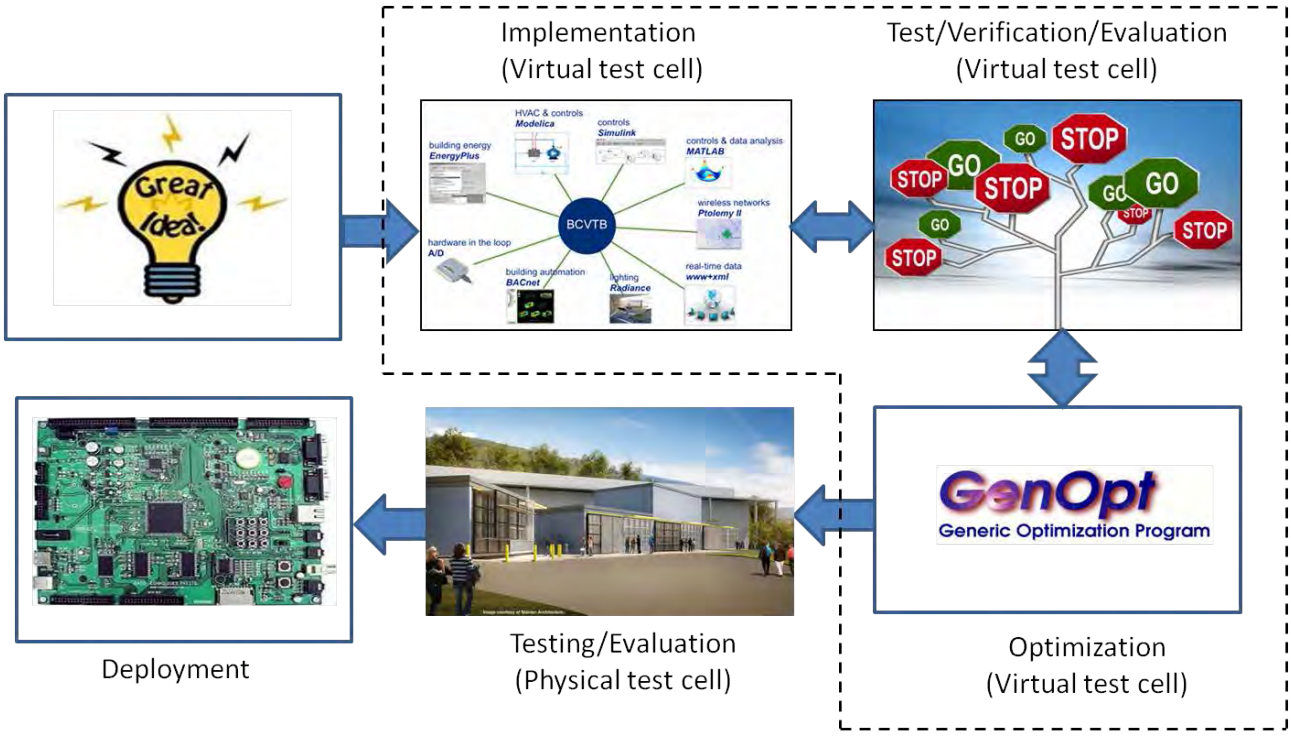


Figure 11: Use case of the simulation-based controls framework

8 Conclusions

We have developed a flexible and expandable framework for testing and evaluating of control algorithms for façade systems. This framework addresses the reinitialization of state variables, a key issue arising from the use of a thermal simulation model in the controls loop. This issue is resolved by using the Modelica room model of the *Buildings* library to represent the building envelope. Another key component of the framework is the accurate modeling of solar radiation distribution of complex fenestration systems such as a venetian blind, by using the daylight simulation program *Radiance*. This extends the capability of the framework to develop control algorithms for complex fenestration systems described using bidirectional scattering distribution function and thus study the potential energy impacts of CFS which cannot routinely be done in current whole building simulations. We discussed the software architecture of the framework and explained the component models. We've illustrated the use of the framework by implementing a control algorithm to control the venetian blind of a physical test cell in real time. Results show that using the framework to control the venetian blind of the test cell reduces the zone thermal load in the cell compared to rooms with a static blind.

Future research activities should address the limitations of the simulation-based framework particularly its complexity to make it more usable to others. Also the framework should be extended with new control algorithms that can be used in buildings to reduce energy consumption while keeping thermal and visual comfort. The use of Modelica in the framework could significantly simplify the integration of such new control algorithms.

9 Acknowledgments

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technology, State and Community Programs, Office of Building Research and Standards of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 and by the California Energy Commission through its Public Interest Energy Research (PIER) Program on behalf of the citizens of California.

References

Brooks, C. et al., 2007. *Ptolemy II – Heterogeneous Concurrent Modeling and Design in Java*. Berkeley.

Elmqvist, H., 1978. *Dymola*. [Online] Available at: http://www.3ds.com.

EnergyPlus, 2012. *http://apps1.eere.energy.gov/buildings/energyplus/*. [Online].

eQuest, 1998. *http://doe2.com/equest/*. [Online].

Klems, J.H., 1994. A new method for predicting the solar heat gain of complex fenestration systems: II. Detailed description of the matrix layer calculation. *ASHRAE Transactions 100 (1)*, pp.1073-86.

Lee E., D.D..K.J..Y.M..a.S.S., 2006. Monitored energy performance of electrochromic. *ASHRAE Transactions*, 112, pp.122-41.

MathWorks, 1994-2012. *MATLAB*. [Online] Available at: http://www.mathworks.com/ [Accessed 24 September 2012].

MODELISAR, 2008-2011. *http://www.modelisar.com/*. [Online] [Accessed 21 September 2012].

Nouidui, T.S., Wetter, M. & Zuo, W., 2012. Validation of the window model of the Modelica Buildings library. In *Proceedings of SimBuild2012*. Madison, 2012. Accepted for publication.

NREL, 2007. *http://archrecord.construction.com/tech/techFeatures/0712feature-1.asp*. [Online] [Accessed 21 September 2012].

Van Rossum, G., 1991. *Python*. [Online] Available at: http://www.python.org/.

Wall, L., 1987. *Perl*. [Online] Available at: http://www.perl.org/.

Ward, G., 1985. *Radiance*. [Online] Available at: http://radsite.lbl.gov/radiance/index.html.

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), Available at: http://simulationresearch.lbl.gov/bcvtb/releases/1.1.0/doc/manual/bcvtb-manual.pdf [Accessed 11 May 2012].

Wetter, M., Zuo, W. & Nouidui, T.S., 2011. Recent developments of the Modelica buildings library for building energy and control systems. In *Proceedings of the 8th International Modelica Conference. Dresden, Germany, March 2011*., 2011.

Winkelmann, F., 2001. Modeling Windows in EnergyPlus. In *Proceedings of IBPSA, Buildig Simulation 2001. LBNL-47972.* Rio de Janeiro, 2001.