2018 Building Performance Modeling Conference and
SimBuild co-organized by ASHRAE and IBPSA-USA
Chicago, IL
September 26-28, 2018

OpenBuildingControl: Modeling Feedback Control as a Step Towards Formal Design, Specification, Deployment and Verification of Building Control Sequences

Michael Wetter[1], Jianjun Hu[1], Milica Grahovac[1], Brent Eubanks[2] and Philip Haves[1]
[1]Lawrence Berkeley National Laboratory, Berkeley, CA
[2]Carbon Lighthouse, San Francisco, CA*

ABSTRACT

This paper presents ongoing work to develop tools and a process that will allow building designers to instantiate control sequences, configure them for their project, assess their performance in closed loop building energy simulation, and then export these sequences (i) for the control provider to bid on the project and to implement the sequences through machine-to-machine translation, and (ii) for the commissioning agent to verify their correct implementation.

The paper reports on the following: (i) The specification of a Control Description Language, (ii) its use to implement a subset of the ASHRAE Guideline 36 sequences, released as part of the Modelica Buildings library, (iii) its use in annual closed-loop simulations of a variable air-volume flow system, and (iv) lessons learned regarding simulation of closed loop control.

In our case study, the Guideline 36 sequences yield 30% lower annual site HVAC energy use, under comparable comfort, than sequences published earlier by ASHRAE. The 30% differences in annual HVAC energy consumption due to changes in the control sequences raises the question of whether the idealization of control sequences that is common practice in today's building energy simulation leads to trustworthy energy use predictions.

INTRODUCTION

Built-up HVAC systems which require custom-control solutions are common in large buildings. The current state is that, at best, the mechanical designer specifies the building control sequences in an English language specification. However, such a specification cannot be tested formally for correctness. It is also ambiguous, leaving room for different implementations, including variants that were not intended by the designer or may not work correctly. The implementation of the sequences is often done by a controls contractor who either attempts to implement the sequence as specified, or use a sequence of a similar project that appears to have the same control intent. During commissioning, the lack of an executable specification of the control sequence against which the implementation can be tested makes commissioning of the control sequences expensive and limited in terms of code coverage (Gnerre and Fuller 2017). Not surprisingly, pro-gramming errors are the dominating issue among control-related problems that impact energy use in buildings (Bar-wig et al. 2002).

*The research was performed while the author was with Integral Group, Oakland, CA.

To change this status quo, we are developing tools, and borrowing processes from other industries, that allow formal specification of control sequences, testing of these sequences in simulation, export of the sequences in a control-vendor independent format, and conducting verification tests during commissioning and/or continuous operation. This project, called OpenBuildingControl, is further described at http://obc.lbl.gov/. This paper describes a Control Description Language (CDL) that we developed to support this process, the implementation of a subset of ASHRAE Guideline 36 (ASHRAE 2016) using CDL, and to demonstrate its use for comparing a variable air volume flow (VAV) control sequence as published in Guideline 36 with a sequence earlier published by ASHRAE (2006). If applied to an identical HVAC and building model, the two sequences result in a 30% difference in annual site HVAC energy consumption under comparable comfort. This raises the question of whether today's common simulation practice of idealizing control leads to trustworthy energy use predictions if different implementations of a control sequence lead to 1/3 differences in energy use.

The intention of CDL is that it can be used within annual simulations, and that it can be executed on various build-ing automation systems, either through code generation that translates CDL into a representation needed by the particular control system, or through translation to C code, which may be encapsulated as a Functional Mockup Unit in order to use an open standard (MODELISAR Consor-tium 2014). Therefore, the language needs to be declar-ative (in order for it to be translated to other representations), simple (to lower the barrier to develop transla-tors) yet expressive enough to be able to express control sequences and to be used as part of an annual building energy simulation. Moreover, as we intend to formally verify that the sequences installed in a building control system meet the design specification, we require control sequences that conform to CDL to allow a determinis-tic computation. I.e., the control signals, computed by implementations of the sequence in two different control systems or simulators, need to be identical (see Broman et al. 2015).

Related work includes the following: Husaunndee et al. (1997) developed a MATLAB/Simulink-based toolbox of models of HVAC components and plants for the design and test of control systems called SIMBAD. SIMBAD has been used for testing and emulation of building control sequences, and is commercially distributed by CSTB France. Bonvini and Leva (2012) developed an industrial control library in Modelica. Yang et al. (2010) developed a tool chain that maps Simulink and Modelica models into an intermediate format, and then refined it for implemen-tation in distributed controllers. Our approach borrows from their methodology. Schneider, Peßler, and Steiger (2017) implemented a Modelica library with standardized control functions for building automation. They use con-trol functions from VDI 3813-2:2011 and state graph representations from VDI 3814-6:2009. Our approach differs from their work as they document the control using Unified Modeling Language (UML) class and activity diagrams. Also, they used semantic control connectors, but subsequently removed them for version 1.0.0. Our approach differs from Bonvini and Leva (2012), Yang et al. (2010) and Schneider, Peßler, and Steiger (2017) in that we use elementary control blocks that form a basic library of control functions, and simple composition rules that we believe suffice for composing building control sequences. Below we describe both CDL and an ASHRAE Guideline 36 control sequence library. CDL is a proper subsets of Modelica and have been tested for compatibility with the Modelica modeling and simulation environments Dymola, OpenModelica and JModelica. For CDL, we use a subset of Modelica to keep it simple to use and understand for HVAC designer and control providers, to respect the limitations of current control systems with respect to expressing certain modeling constructs (for example, we currently do not allow finite state machines in CDL) and to reduce the barrier for industry to develop translators that process our representation to generate implementa-tions for their control system.

METHODOLOGY

This section summarizes the methodology that we used to design the language, implement a library of basic building blocks, and implement a subset of the control sequences of ASHRAE Guideline 36.

Language

To be able to conduct efficient simulations, we based the CDL on Modelica, to allow the following: (i) its use with the Modelica Buildings library, (ii) the efficient simulation of control sequences with continuous time, discrete time and discrete event dynamics, coupled to HVAC and building models, and (iii) the use of CDL with various tools for modeling, simulation and code gen-eration. The use of Modelica for CDL also fits with a current project that redesigns EnergyPlus to allow more modular modeling of HVAC and controls (see https:// lbl-srg.github.io/soep/). However, Modelica is a large lan-guage, and only a subset is needed for controls. The sub-set of Modelica that was chosen for inclusion in CDL al-lows the graphical composition of block diagrams, form-ing new composite blocks, for example to encapsulate se-quences as their own objects, and instantiating arrays of controllers, for example to model hundreds of VAV termi-nal box controllers.

The CDL consists of the following elements:

- A list of elementary control blocks, such as a block that adds two signals and outputs the sum, or a block that represents a PID controller.
- Connectors through which these blocks receive values and output values.
- Permissible data types. •

Syntax to specify how to

  - instantiate these blocks and assign values of pa-rameters, such as a proportional gain,
  - connect inputs of blocks to outputs of other blocks,
  - document blocks,
  - add annotations such as for graphical rendering of blocks and their connections, and
  - specify composite blocks.

- A model of computation that describes when blocks are executed and when outputs are assigned to inputs.

To simplify the support of CDL for tools and control sys-tems, the following Modelica keywords are not supported in CDL: extends as we don't allow inheritance (however, single inheritance could be added as this is easy to sup-port), redeclare and hence also constrainedby to sim-plify the use and support of the language, and inner and outer to keep blocks self-contained.

Also, the following Modelica language features are not supported in CDL:

- Clocks (which are used in Modelica for hybrid system modeling).
- algorithm sections (because the elementary building blocks are black-box models as far as CDL is concerned and thus CDL compliant tools need not parse the algorithm section).
- initial equation and initial algorithm sections.
- package-level declaration of constant data types. As the model of computation, CDL uses the synchronous data flow principle and the single assignment rule, which are defined below. (The definition is adopted from and consistent with the Modelica 3.3 Specification, Sec. 8.4.) • All variables keep their actual values until these values

  are explicitly changed. Variable values can be accessed at any time instant.

```
block AddParameter
  "Add a parameter to the input signal"
  parameter Real p "Value to be added";
  parameter Real k "Gain of input";
  Interfaces.RealInput u
    "Connector of Real input signal";
  Interfaces.RealOutput y
    "Connector of Real output signal";
equation
  y = k*u + p;
  annotation(Documentation(info(
  "<html> ... [omitted] </html>"));
end AddParameter;
```

Figure 1: Implementation of an elementary CDL block.(Graphical annotations are omitted.)

- Computation and communication at an event instant does not take time. (If computation or communication time has to be simulated, this property has to be explic-itly modeled.)
- Every input connector shall be connected to exactly one output connector.

In addition, the dependency graph from inputs to outputs that directly depend on inputs shall be directed and acyclic. I.e., connections that form an algebraic loop are not allowed.

CDL will also support the use of tags to add additional features, such as to connect it semantically to efforts like Brick (http://brickschema.org/) and Haystack (http://project-haystack.org/). The design of tag-ging is ongoing and not reported in this paper.

A more detailed specification of the CDL language and a description for how to translate CDL-conformant control sequences to control product lines can be found at http: //obc.lbl.gov.

CLD Library

The CDL library, implemented in the Modelica package Buildings.Controls.OBC.CDL, contains elementary building blocks for composing control sequences. The CDL library structures 134 elementary building blocks into 11 packages that contain blocks for real-valued continuous-time signals, for boolean signals, for time-sampled systems etc. Each package contains examples for each elementary building block. As different control vendors use different programming languages, CDL prescribes the functionality of the elementary building blocks, but not their implementation. For example, the block AddParameter is implemented in CDL as shown in Figure 1.

CDL also defines syntax for connecting input to outputs, for propagating parameter values, and for encapsulating composite blocks and storing them in a library. For ex-ample, if one would like to implement a custom pro-portional controller with output limiter that computes the output signal $y = \min(k\,e, y_{max})$, where k is a parameter, and e and y_{max} are control inputs, one can imple-ment the code shown in Figure 2 and store it in a file CustomPWithLimiter.mo.

A visual editor may render the block as shown in Figure 3. One may recognize that the syntax is a subset of Model-ica, and hence any Modelica modeling environment can be used to compose, translate and simulate CDL confor-mant control sequences.

ASHRAE Guideline 36 Library

Using the composition rules defined in the CDL language specification, library developers and end-users such as mechanical engineers can compose control sequences for a given system, and optionally store them in a library for use in other projects or to share them with others.

Our vision is that the simulation community will create packages of ready-to-use control sequences. Similarly, if a design firm uses their own, possibly proprietary sequences, they can build a library and share them within their company. Hence, the library will become a means to share expertise in building control sequences, and also to continually improve the sequences from one project to another.

In the Modelica Buildings library, we used CDL to implement a subset of the sequences published in ASHRAE Guideline 36 (ASHRAE 2016). Figure 4 shows an overview of the implemented sequences. The implementation is structured hierarchically into packages for air handler units, into constants that indicate operation modes, into generic sequences such as for a trim and respond logic, and into sequences for terminal units. For every sequence, there is a validation package that illustrates the use of the sequence.

CASE STUDY

In this section, we compare the performance of two different control sequences, applied to a floor of a prototypical office building. The objectives are to demonstrate the setup for closed loop performance assessment, to demonstrate how to compare the control performance, and to assess the difference in annual energy consumption. For the basecase, we implemented a control sequence published in ASHRAE's Sequences of Operation for Common HVAC Systems (ASHRAE 2006). For the other case, we implemented the control sequence published in the first public review draft of ASHRAE Guideline 36 (ASHRAE 2016). The main conceptual differences between the two control sequences are as follows: •
The base case uses constant supply air temperature set-

points for heating and cooling during occupied hours, whereas the Guideline 36 sequence uses one supply air temperature setpoint, which is reset based on outdoor air temperature and zone cooling requests, as obtained

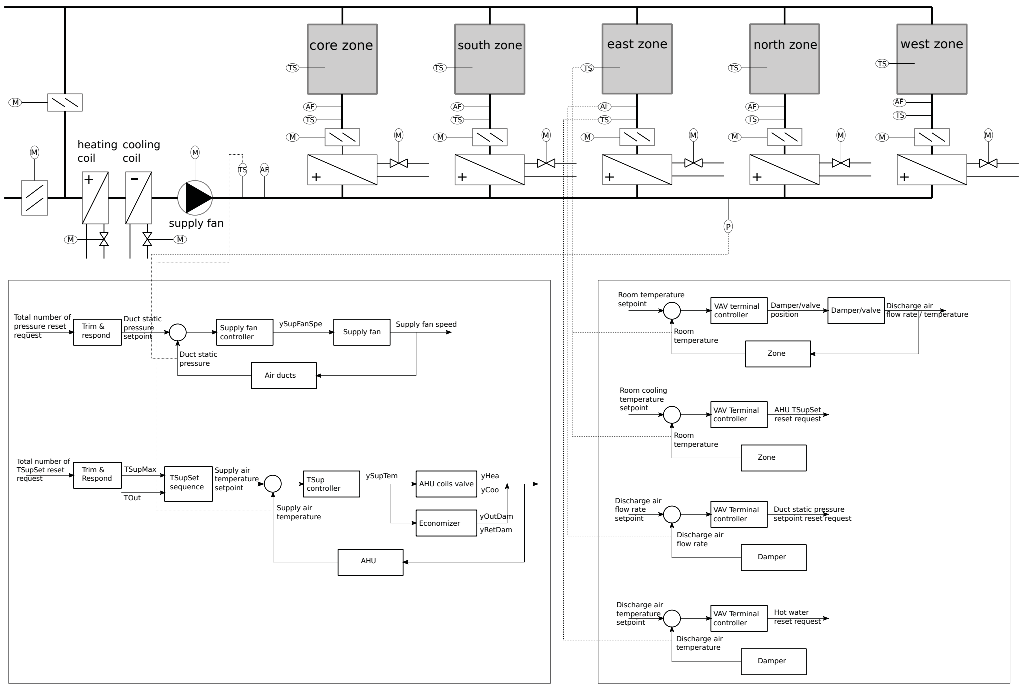

Figure 6: Control schematics of Guideline 36 case.

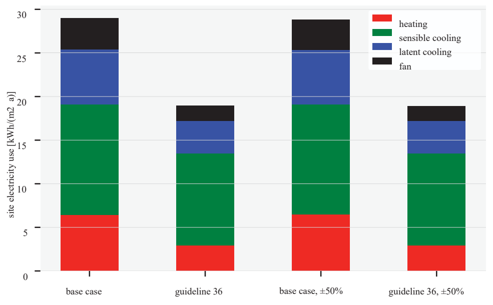

Figure 7: Comparison of energy use.

zones. For both control sequences, this does not noticeably affect energy use. Fig. 8 shows for five spring days that both control sequences successfully compensate for this load imbalance. The room air temperature trajecto-ries are practically on top of each other. Both control se-quences are comparable in terms of compensating for this diversity. The figure also shows for the same five days the outdoor air temperature T_{out}, supply air temperature T_{sup}, mixed air temperature T_{mix} and return air tempera-ture T_{ret}, as well as the control error for the supply air temperature for the Guideline 36 case. For this period, Guideline 36 operates with higher mixed air and supply air temperatures. On day 85, some oscillations can be observed for T_{mix}. These could be removed by retuning the control gains.

Lessons learned regarding the simulations

This is probably the most detailed building control simulation that has been performed using the Modelica Buildings library. A few lessons have been learned and are reported here. Note that related best-practices are also available in the users' guide for the Buildings library.

- *Fan with prescribed mass flow rate:* In earlier implementations, we converted the control signal for the fan to a mass flow rate, and used a fan model whose mass flow rate is equal to its control input, regardless of the pressure head. During start of the system, this caused a unrealistic large fan head of 4000Pa (16 inch of wa-ter) because VAV dampers opened slowly. The large pressure drop also led to large power consumption and hence unrealistic temperature increase across the fan in the order of 10 Kelvin.
- *Fan with prescribed pressure head:* We also tried to use a fan with prescribed pressure head. Similarly as above, the fan maintains the pressure head as obtained from its control signal, regardless of the volume flow rate. This caused unrealistic large flow rates in the re-turn duct which has very small pressure drops. (Subse-quently, we removed the return fan as it is not needed for this system.)
- *Time sampling certain physical calculations:* Dymola 2018FD01 uses the same time step for all continuous-

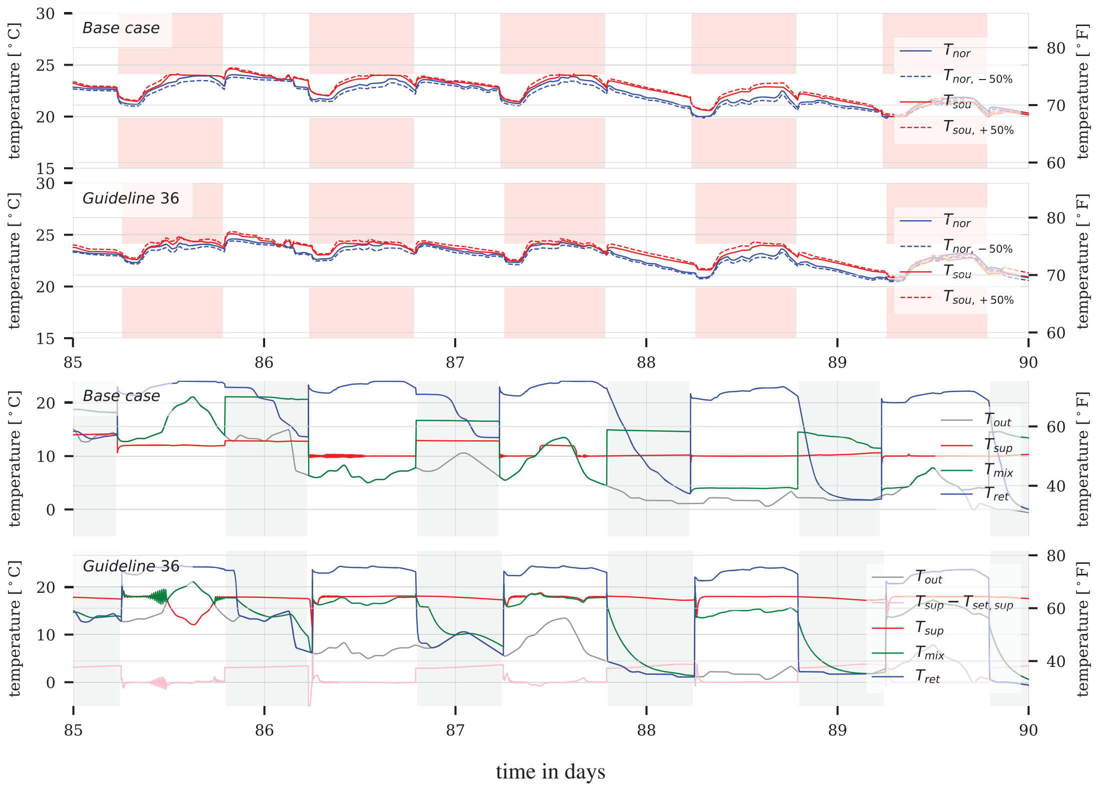

Figure 8: Temperatures for five days in spring for the base case and the Guideline 36 control. In the top two charts, regions outside the dual-setpoint for the room temperatures are displayed in red; in the bottom two charts, night-time operation is displayed in grey.

time equations. Depending on the dynamics, this can be smaller than a second. Since some of the control samples every 2 minutes, it has shown to be favorable to also time sample the long-wave radiation network, whose solution is time consuming.

- *Non-convergence:* In earlier simulations, sometimes the solver failed to converge. This was due to errors in the control implementation that caused event iterations for discrete equations that seemed to have no solution. In another case, division by zero in the control implementation caused a problem. The solver found a way to work around this division by zero (using heuristics) but then failed to converge. Since we corrected these issues, the simulations are stable.
- *Properly handling hard switches:* For continuous time control, all switches that lead to discontinuities need hysteresis or timers, and the selected numerical solver need to be able to handle time and state events.
- *Too fast dynamics of coil:* The cooling coil is implemented using a finite volume model. Heat conduction along the water and air flow paths used to be neglected as the mode of heat transfer is dominated by forced convection if the fan and pump are operating. However, during night when the system is off, the small infiltration due to wind pressure caused in earlier simulations the water in the coil to freeze. Adding diffusion along the flow path circumvented this problem, and the coil model in the library includes now by default a small diffusive term.

This indicates that certain simplifications that are common in many building simulation programs, such as prescribed mass flow rates, can give unrealistic simulations once control and flow friction is no longer idealized.

DISCUSSION AND CONCLUSIONS

Differences in properly designed conventional control sequences can have a substantial impact on energy consumption; in our example, site HVAC electricity use is reduced by 30%. Such a sensitivity of energy consumption with respect to the control sequence raises the fundamental question of whether today's simplified control implementations, as found in the major building energy simulation programs, are adequate for predicting annual energy consumption.

Conducting the case study indicated that correct implementation of sequences such as published in Guideline 36 is difficult and time-consuming. Verification of the correct implementation by mere inspection is not possible due to the various timers, switches and interlocks. In fact, we detected certain implementation errors only when we conducted closed loop simulations in different seasons. Fortunately, by providing ready-to-use libraries of such sequences that can be configured to a particular building, as we demonstrated with our implementation, this complexity of implementation can be hidden from the end-user. Therefore, a library of carefully designed and implemented control sequences has the potential to substantially reduce energy consumption.

We demonstrated that using a certain subset of Modelica, which suffices for block-diagram modeling and which lowers the effort to develop code generators, enables the performance comparison of different control strategies within annual building energy simulations.

In future work, we will expand a library of control sequences and further develop a tool chain to

1. enable mechanical designers to create specifications based on the simulation model,
2. allow control providers to translate CDL-conformant control sequences to their product line, and
3. allow the commissioning provider to verify correct implementation by executing the control model with inputs and parameters obtained from the real implementation, and comparing the simulated with the real actuator signals.

We also recommend implementing Guideline 36 using the CDL language. This would allow a non-ambiguous, exe-cutable specification against which vendor-specific imple-mentations could be tested and certified using free, open-source tools.

In addition, R&D is ongoing in a project called "Spawn of EnergyPlus" to redesign EnergyPlus so that it supports this process (see https://lbl-srg.github.io/soep/).

ACKNOWLEDGMENT

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231, and the Cali-fornia Energy Commission's Electric Program Investment Charge (EPIC) Program.

REFERENCES

ASHRAE. 2006. *Sequences of Operation for Common HVAC Systems*. Atlanta, GA: ASHRAE.

ASHRAE. 2016, June. *ASHRAE Guideline 36P, High Performance Sequences of Operation for HVAC sys-tems, First Public Review Draft.* ASHRAE.

Barwig, Floyd E., John M. House, Curtis J. Klaassen, Morteza M. Ardehali, and Theodore F. Smith. 2002, August. "The National Building Controls Informa-tion Program." Summer Study on Energy Efficiency in Buildings. ACEEE, Pacific Grove, CA.

Bonvini, Marco, and Alberto Leva. 2012, September. "A Modelica Library for Industrial Control Systems." *Proc. of the 9-th Int. Modelica Conf.* Modelica As-sociation, Munich, Germany, 477–484.

Broman, David, Lev Greenberg, Edward A. Lee, Michael Massin, Stavros Tripakis, and Michael Wet-ter. 2015, April. "Requirements for Hybrid Cosimu-lation Standards." *18th Int. Conf. on Hybrid Systems: Computation and Control*. ACM Press.

Deru, Michael, Kristin Field, Daniel Studer, Kyle Benne, Brent Griffith, Paul Torcellini, Bing Liu, Mark Halverson, Dave Winiarski, Michael Rosenberg, Mehry Yazdanian, Joe Huang, and Drury Craw-ley. 2011, February. "U.S. Department of Energy Commercial Reference Building Models of the Na-tional Building Stock." Technical Report NREL/TP- 5500-46861, National Renewables Energy Labora-tory, Golden, CO.

Gnerre, Bill, and Kevin Fuller. 2017, December. "When Building Controls Veer Off Course." In *Facility Executive*, Volume 462707, 32. Tinton Falls, NY: Group C Media, Inc.

Husaunndee, A., R. Lahrech, H. Vaezi-Nejad, and J.C. Visier. 1997. "SIMBAD: A Simulation Toolbox for the Design and Test of HVAC Control Systems." Edited by Jean Jacques Roux and Monika Woloszyn, *Proc. of the 5-th IBPSA Conf.* 269–276.

MODELISAR Consortium. 2014. *Functional Mock-up Interface for Model-Exchange and Co-Simulation version 2.0.* https://www.fmi-standard.org.

Schneider, Georg Ferdinand, Georg Ambrosius Peßler, and Simone Steiger. 2017, may. "Modelling and Simulation of Standardised Control Functions from Building Automation." *Proc. of the 12-th Int. Model-ica Conf.* Modelica Association, Prague, Czech Re-public, 209–218.

Wetter, Michael, Wangda Zuo, Thierry S. Nouidui, and Xiufeng Pang. 2014. "Modelica Buildings library." *Journal of Building Performance Simulation* 7 (4): 253–270.

Yang, Y., A. Pinto, A. Sangiovanni-Vincentelli, and Q. Zhu. 2010, November. "A Design Flow for Build-ing Automation and Control Systems." *2010 31st IEEE Real-Time Systems Symposium*. 105–116.