

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

Michael Wetter¹, Jianjun Hu¹, Milica Grahovac¹, Brent Eubanks² and Philip Haves¹

¹Lawrence Berkeley National Laboratory, Berkeley, CA

²Carbon 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).

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 building 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 Consortium 2014). Therefore, the language needs to be declarative (in order for it to be translated to other representations), simple (to lower the barrier to develop translators) 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 deterministic 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).

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

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 implementation 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 control 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 subset 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 implementations 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 generation. 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 language, and only a subset is needed for controls. The sub-set of Modelica that was chosen for inclusion in CDL allows the graphical composition of block diagrams, forming new composite blocks, for example to encapsulate sequences as their own objects, and instantiating arrays of controllers, for example to model hundreds of VAV terminal 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 parameters, 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 systems, 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 support), redeclare and hence also constrained by to simplify 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.

```

1 block AddParameter
2   "Add a parameter to the input signal"
3   parameter Real p "Value to be added";
4   parameter Real k "Gain of input";
5   Interfaces.RealInput u
6   "Connector of Real input signal";
7   Interfaces.RealOutput y
8   "Connector of Real output signal";
9 equation
10  y = k*u + p;
11  annotation(Documentation(info(
12    "<html> ... [omitted] </html>"));
13 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 explicitly 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 example, if one would like to implement a custom proportional controller with output limiter that computes the

output signal $y = \min(ke, y_{\max})$, where k is a parameter, and e and y_{\max} are control inputs, one can implement 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 Modelica, and hence any Modelica modeling environment can be used to compose, translate and simulate CDL conformant 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

```

1 block CustomPWithLimiter
2   "P controller with variable output limiter"
3   parameter Real k "Constant gain";
4   CDL.Interfaces.RealInput yMax "Maximum value of output signal" annotation(...);
5   CDL.Interfaces.RealInput e "Control error" annotation(...);
6   CDL.Interfaces.RealOutput y "Control signal" annotation(...);
7   CDL.Continuous.Gain gain(final k=k) "Constant gain" annotation(...);
8   CDL.Continuous.Min minValue "Outputs the minimum of its inputs" annotation(...);
9   equation
10    connect(yMax, minValue.u1) annotation(...);
11    connect(e, gain.u) annotation(...);
12    connect(gain.y, minValue.u2) annotation(...);
13    connect(minValue.y, y) annotation(...);
14    annotation (Documentation(info="<html><p>
15    Block that outputs <code>y = min(yMax, k * e)</code>, where ...</p></html>"));
16    //comment omitted
17 end CustomPWithLimiter;

```

Figure 2: Implementation of a composite control block that outputs $y = \min(ke, y_{\max})$ where k is a parameter. (Graphical annotations are omitted.)

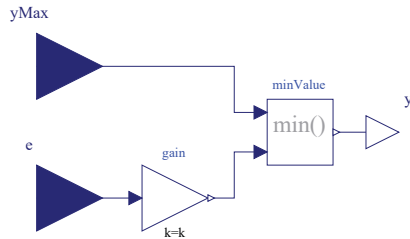


Figure 3: Graphical rendering of a the composite control block shown inImpleme

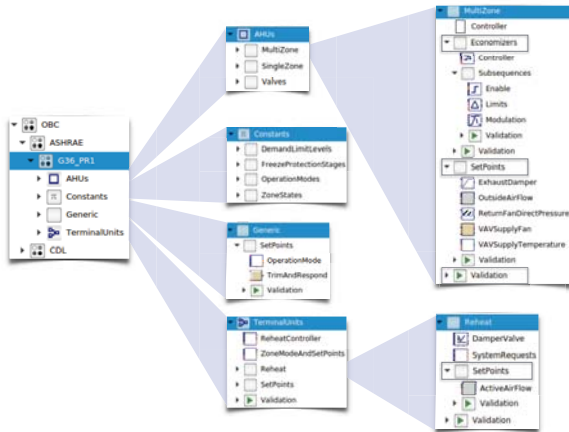


Figure 4: Overview of the ASHRAE Guideline 36 package implemented in the Modelica Buildings library 5.0.0

from the VAV terminal unit controllers. The reset is dynamic using the trim and respond logic.

- The base case resets the supply fan static pressure set- point based on the VAV damper position, which is only modulated during cooling, whereas the Guideline 36

sequence resets the fan static pressure setpoint based on zone pressure requests from the VAV terminal con-trollers. The reset is dynamic using the trim and re-spond logic.

- The base case controls the economizer to track a mixed air temperature setpoint, whereas Guideline 36 controls the economizer based on the supply air temperature control loop signal.
- The base case controls the VAV dampers based on the zone's cooling temperature setpoint, whereas Guideline 36 uses the heating and cooling loop signal to control the VAV dampers.

Methodology

All models are implemented in Modelica, using models from the Buildings library 5.0.0 (Wetter et al. 2014). As a test case, we used a simulation model that consists of five thermal zones that are representative of one floor of the new construction medium office building for Chicago, IL, as described in the set of DOE Commercial Building Benchmarks (Deru et al. 2011). There are four perimeter zones and one core zone. The envelope thermal properties meet ASHRAE Standard 90.1-2004. The system model consist of an HVAC system, a building envelope model and a model for air flow through building leakage and through open doors based on wind pressure and flow imbalance of the HVAC system.

The HVAC system is a variable air volume (VAV) flow system with economizer and a heating and cooling coil in the air handler unit. There is also a reheat coil and an air damper in each of the five zone inlet branches. In the VAV model, all air flows are computed based on the duct static pressure distribution and the performance curves of the fans, which take the speed control signal as an input. Air exchange within the building depends on wind pres-

sure, HVAC static pressure, and buoyancy forces among the thermal zones that induce air flow among the thermal zones, HVAC system and outside.

For the base case, we implemented the control sequence VAV 2A2-21232 of the Sequences of Operation for Common HVAC Systems. In this control sequence, the supply fan speed is regulated based on the duct static pressure. The duct static pressure is adjusted so that at least one VAV damper is 90% open. The economizer dampers are modulated to track the setpoint for the mixed air dry bulb temperature. The supply air temperature setpoints for heating and cooling are constant during occupied hours, which may not comply with some energy codes. Priority is given to maintaining a minimum outside air volume flow rate. In each zone, the VAV damper is adjusted to meet the room temperature setpoint for cooling, or fully opened during heating. The room temperature setpoint for heating is tracked by varying the water flow rate through the reheat coil. Local loop control is implemented using proportional and proportional-integral controllers, while the supervisory control is implemented using a finite state machine that transitions the mode of operation of the HVAC system between different occupancy modes. For the detailed implementation of the control logic, see the model Buildings.Examples.VAVReheat.ASHRAE2006.

Our implementation differs from VAV 2A2-21232 as follows:

- We removed the return air fan as the return air path has a small flow resistance that does not require a fan. With a return fan, building static pressure was too low.
- In order to have the identical mechanical system as for Guideline 36, we do not have a minimum outdoor air damper, but rather control the outdoor air damper to allow sufficient outdoor air if the mixed air temperature control loop would yield too little outdoor air.

For the Guideline 36 case, we implemented the multi-zone VAV control sequence based on ASHRAE Guideline 36. Fig. 6 shows the sequence diagram, and the implementation is available in the model Buildings.Examples.VAVReheat.Guideline36.

In the Guideline 36 sequence, the duct static pressure is reset using trim and respond logic based on zone pressure reset requests, which are issued from the terminal box controller based on whether the measured flow rate tracks the set point. The economizer dampers are modulated based on a control signal for the supply air temperature set point, which is also used to control the heating and cooling coil valve in the air handler unit. Priority is given to maintaining a minimum outside air volume flow rate. The supply air temperature setpoints for heating and cooling at the air handler unit are reset based on outdoor air temperature, zone temperature reset requests from the terminal boxes and operation mode.

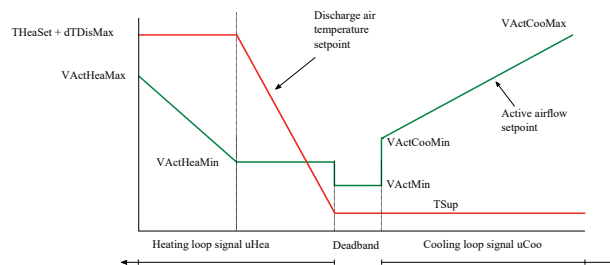


Figure 5: Control sequence for VAV terminal unit. Table 1: Model and simulation statistics.

Quantity	Base case	Guideline 36
Number of components	2826	4400
Number of variables (prior to translation)	33,700	40,400
Number of continuous states	178	190
Number of time-varying variables	3400	4800
Time for annual simulation in minutes	70	100

In each zone, the VAV damper and the reheat coil is controlled using the sequence shown in Fig. 5, where THeaSet is the set point temperature for heating, dTDisMax is the maximum temperature difference for the discharge temperature above THeaSet, TSup is the supply air temperature, VAct* are the active airflow rates for heating (Hea) and cooling (Coo), with their minimum and maximum values denoted by Min and Max.

All simulations were done with Dymola 2018 FD01 beta3 using Ubuntu 16.04 64 bit. We used the Radau solver with a tolerance of 10^{-6} . This solver adaptively changes the time step to control the integration error. Also, the time step is adapted to properly simulate time events and state events.

Tab. 1 shows an overview of the model and simulation statistics. The differences in the number of variables and in the number of time varying variables shows that the Guideline 36 control is significantly more detailed than what may otherwise be used for simulation of what the authors believe represents a realistic implementation of a feedback control sequence.

Results

As shown in Fig. 7, the Guideline 36 control saves around 30% site HVAC energy under comparable thermal comfort. These are significant savings that can be achieved through software only, without the need for additional hardware or equipment. In Fig. 7, for the cases labeled $\pm 50\%$, the internal gains have been decreased by 50% in the North zones, and increased by 50% in the South

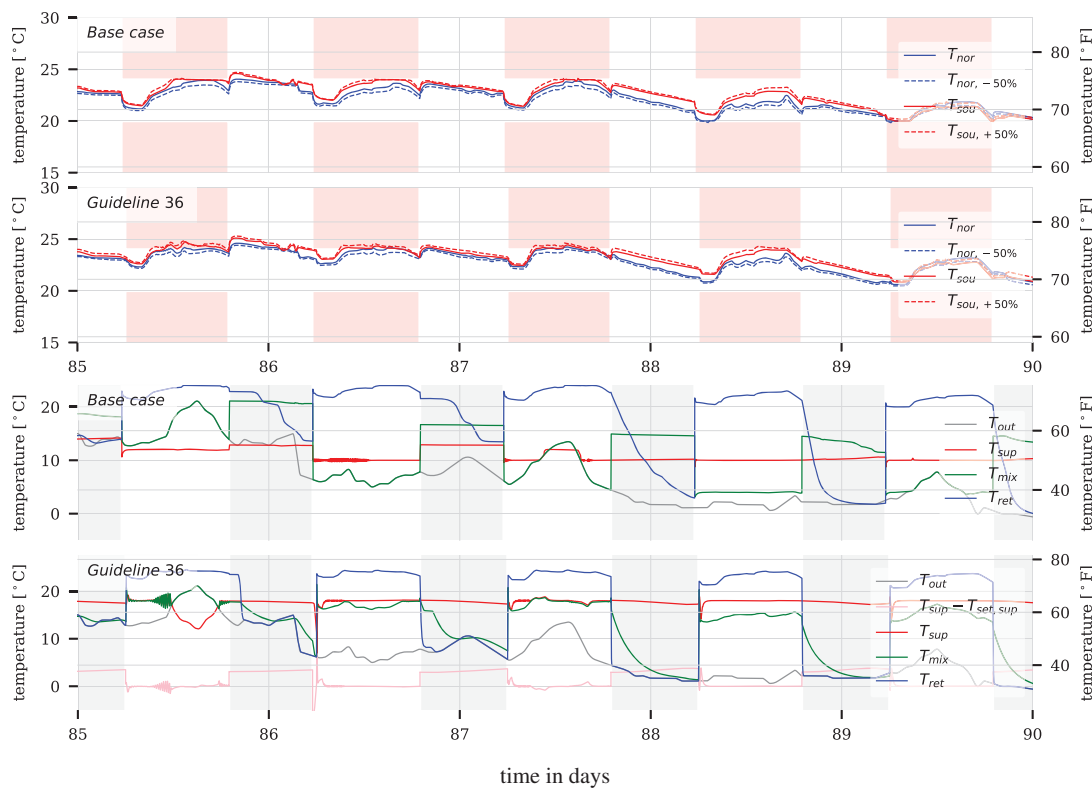


Figure 8: Temperatures for five days in spring for the base case and the Guideline 36 control. In the top two charts, regions outside the dual-setpoint for the room temperatures are displayed in red; in the bottom two charts, night-time operation is displayed in grey.

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, executable specification against which vendor-specific implementations 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 California 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 systems, 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 Information 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 Association, Munich, Germany, 477–484.
- Broman, David, Lev Greenberg, Edward A. Lee, Michael Massin, Stavros Tripakis, and Michael Wetter. 2015, April. “Requirements for Hybrid Cosimulation 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 Yazdani, Joe Huang, and Drury Crawley. 2011, February. “U.S. Department of Energy Commercial Reference Building Models of the National Building Stock.” Technical Report NREL/TP- 5500-46861, National Renewables Energy Laboratory, 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. Modelica Conf.* Modelica Association, Prague, Czech Republic, 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 Building Automation and Control Systems.” *2010 31st IEEE Real-Time Systems Symposium*. 105–116.