Verification of Control Sequences within OpenBuildingControl

Michael Wetter, Antoine Gautier, Milica Grahovac, Jianjun Hu
Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract

The OpenBuildingControl project develops tools and processes for the performance evaluation, specification and verification of building control sequences. This paper describes the tools developed to verify that building control sequences are implemented as specified in a vendor-neutral, executable specification. The verification is done by testing whether trended time series, compared to the simulated control response, are within user-specified tolerances for time and for the trended variable. Morever, sequence diagrams are used for inspection of the control response.

The paper gives an overview of the OpenBuilding-Control process and describes tools for the control verification. It presents and example in which we successfully verified that an actual implemented control sequence conforms to its vendor-independent specification. It closes with discussion of experiences collected during this verification and with alternate approaches that can improve the operational performance of buildings.

Introduction

A significant fraction of inefficiency in buildings is caused by sub-optimal design, implementation and commissioning of control sequences (Fernandez et al., 2017). Barwig et al. (2002) report that the most frequent control related failures are due to software faults, accounting for 32% of all reported failures. Gnerre and Fuller (2017) reports that the lack of a non-ambiguous, executable control sequence specification prolongs the commissioning and decreases the quality of the control sequence verification, thereby increasing the fault probability over the lifetime of the system. Moreover, as building control sequences become increasingly complicated, a manual verification as is done today during commissioning becomes difficult due to the various timers, interlocks and mode switches that are present in advanced sequences such as the ones published in ASHRAE Guideline 36 (ASHRAE, 2018).

To contribute to the increased use of advanced control sequences and their error-free implementation, we are executing a project called OpenBuildingControl. The OpenBuildingControl project develops tools and processes for the performance evaluation, specification and verification of building control sequences. It aims to provide methodologies and tools to close the gap between energy modeling tools, controls specification and verification of correct implementation of control sequences. The project specified a workflow, described below, that starts with instantiating and possible adapting a control sequence from a library, optionally test its performance using energy simulation, exporting the sequence in a vendor independent format for cost estimation and subsequent implementation on control product lines by control providers, and formal verification of the correct implementation of the sequence by commissioning providers.

The focus of this paper is to describe the process and software for the verification of this control implementation. Given a set of control input signals, such as measured room and outside air temperatures, the verification tests formally whether measured control outputs, such as setpoints or actuator signals from the building automation system, agree with these signals as computed using the original specification. This process is to be done as part of the building commissioning. Related work from the project can be found as follows: The overall project is documented at `https://obc.lbl.gov`. Wetter et al. (2018a) describe a vendor-independent language called Control Description Language (CDL), which is a subset of the Modelica language (Mattsson and Elmqvist, 1997) that we use to express control sequences and simulate their behavior. Wetter et al. (2018b) describe the overall workflow and discuss simulation of control sequences with an energy model in the loop. They also show that changing the control sequence for a variable air volume flow system between a sequence published by ASHRAE in 2006 and a sequence published in ASHRAE Guideline 36, annual HVAC site electricity use is reduced by 30%, thereby indicating the importance of control sequences to reduce energy use. Moreover, the large difference of 1/3 of HVAC site electricity use questions the validity of today's common practice of idealizing control sequences in building energy simulation. To respond to the need for better representing controls and integrate energy modeling with the control delivery process, we are currently redesigning EnergyPlus through a project called Spawn of EnergyPlus (`https://www.energy.gov/eere/buildings/downloads/spawn-energyplus-spawn`).

This paper is structured as follows: We first give an overview of the control design and deployment process. Next, we present tools developed to conduct verification of the correct implementation of the control sequences. Then, we present an example and we close with discussions and conclusions.

Verification

To put the verification of the control sequence into the context of the overall workflow, we now describe the process that we follow in OpenBuildingControl for selecting, deploying and verifying a control sequence. Given regulations and efficiency targets, labeled as (1) in figure 1, a design engineer selects, configures, tests and evaluates the performance of a control sequence using building energy simulation (2), starting from a control sequence library that contains ASHRAE Guideline 36 sequences, as well as any user-added sequences (3), linked to a model of the mechanical system and the building (4). If the sequences meet closed-loop performance requirements, the designer exports a control specification, including the sequences and functional verification tests expressed in the Controls Description Language CDL (5). Optionally, for reuse in similar projects, the sequences can be added to a user-library (6). This specification is used by the control vendor to bid on the project (7) and to implement the sequence (8) in product-specific code. Prior to operation, a commissioning provider verifies the correct functionality of these implemented sequences by running functional tests against the electronic, executable specification in a commissioning and functional verification tool (9). If the verification tests fail, the implementation needs to be corrected and the tests repeated until the tests pass.

For closed-loop performance assessment, step (2) in the figure, Modelica models of the HVAC systems and controls (Wetter et al., 2014) can be linked to a Modelica envelope model (Wetter et al., 2011) or to an EnergyPlus envelope model. This can currently be done through the External Interface (`http://simulationresearch.lbl.gov/fmu/EnergyPlus/export/index.html`). A more direct coupling is in development through the Spawn of EnergyPlus project. Library of control sequences, step (3), have been released with the Modelica Buildings Library 5.0.0 and more sequences are currently added to this library. To export control sequences in a vendor-neutral format, step (5), a translator from CDL to a json intermediate format is being developed at `https://github.com/lbl-srg/modelica-json`. The json intermediate format is intended to be used as input for cost estimation tools and translators to vendor-specific product lines. This translators also outputs an English language description of the control sequence and its block diagram representation.

Methodology

We will now describe how to verify the correct implementation of control sequences. Note that this step only verifies that the control logic is implemented correctly. Hence, it should be conducted in addition to other functional tests, such as tests that verify that sensor and actuators are connected to the correct inputs and outputs, that sensors are installed properly and that the installed mechanical system meets the specification.

For clarity, we remind that this verification tests whether the implementation of the control sequence conforms with its specification. In contrast, validation would test whether the control sequence, together with the building system, is such that it meets the building owner's need, which is done in step (2) in figure 1.

The process is as follows: A commissioning agent exports trended control inputs and outputs and stores them in a CSV file. The commissioning agent then executes the CDL specification for the trended inputs, and compares the following:

1. Whether the trended outputs and the outputs computed by the CDL specification are close to each other.

2. Whether the trended inputs and outputs lead to the expected sequence diagrams, for example, whether an air handler's economizer outdoor air damper is fully open when the system is in free cooling mode.

Technically, step 2 is not needed if step 1 succeeds. However, feedback from mechanical designers indicate the desire to confirm during commissioning that the sequence diagrams are indeed correct (and hence the original control specification is correct for the given system).

Figure 2 shows the verification flow diagram. Rather than using real-time data through BACnet or other protocols, set points, inputs and outputs of the actual controller are stored in an archive, here a CSV file. This allows to reproduce the verification tests, and it does not require the verification tool to have access to the actual building control system. During the verification, the archived data are read into a Modelica model that conducts the verification. The verification will use three blocks. The block labeled *input file reader* reads the archived data, which may typically be in CSV format. As this data may be directly written by a building automation system, its units will differ from the units used in CDL. Therefore, the block called *unit conversion* converts the data to the units used in the CDL control specification.

Next, the block labeled *control specification* is the control sequence specification in CDL format. This is the specification that was exported during design and sent to the control provider. Given the set points and measurement signals, it outputs the control signals according to the specification. The block labeled *time series verification* compares this output with trended control signals, and indicates where the signals differ by more than a prescribed tolerance in time and in signal value. The block labeled *sequence chart* creates

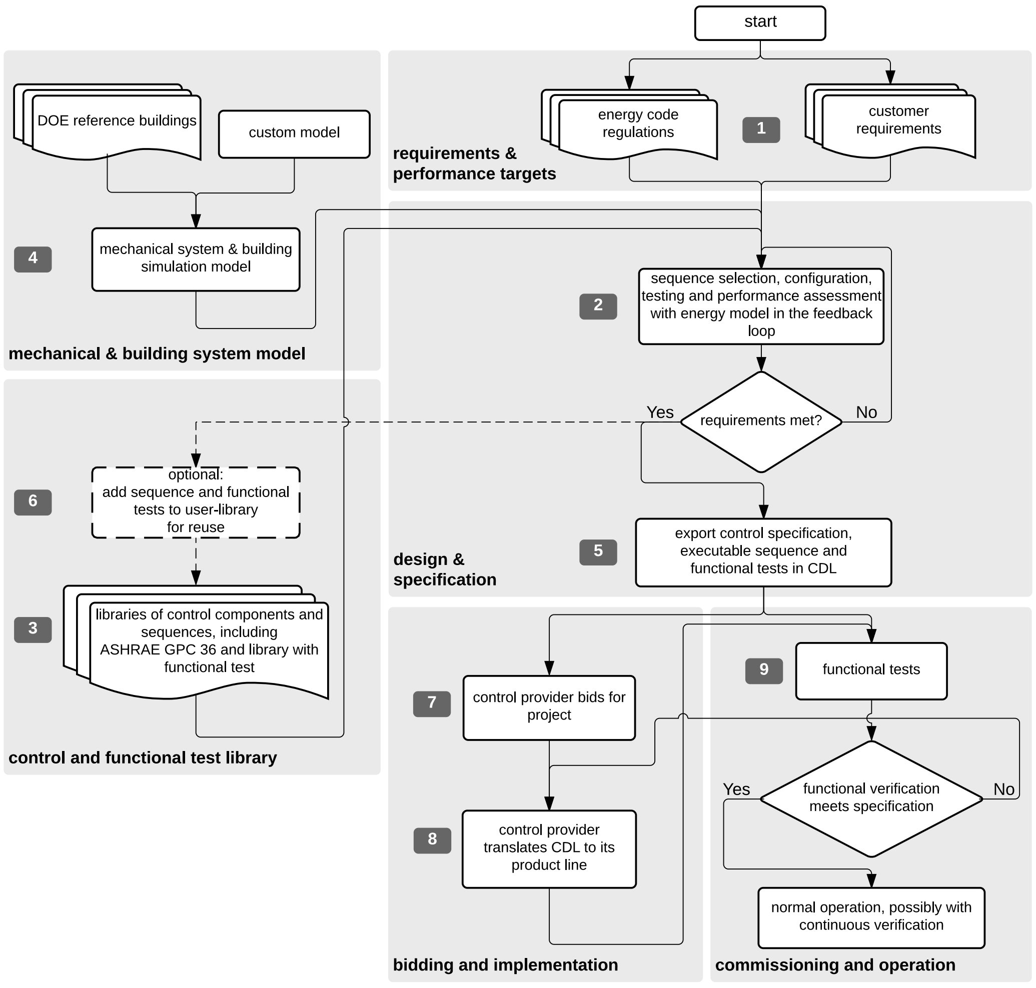

start
DOE reference buildings
custom model
energy code regulations
1
customer requirements
requirements & performance targets
4
mechanical system & building simulation model
mechanical & building system model
2
sequence selection, configuration, testing and performance assessment with energy model in the feedback loop
Yes
requirements met?
No
6
optional: add sequence and functional tests to user-library for reuse
5
export control specification, executable sequence and functional tests in CDL
design & specification
3
libraries of control components and sequences, including ASHRAE GPC 36 and library with functional test
control and functional test library
9
functional tests
7
control provider bids for project
Yes
functional verification meets specification
No
8
control provider translates CDL to its product line
normal operation, possibly with continuous verification
bidding and implementation
commissioning and operation

Figure 1: Overview of the OpenBuildingControl control design, deployment and verification process.

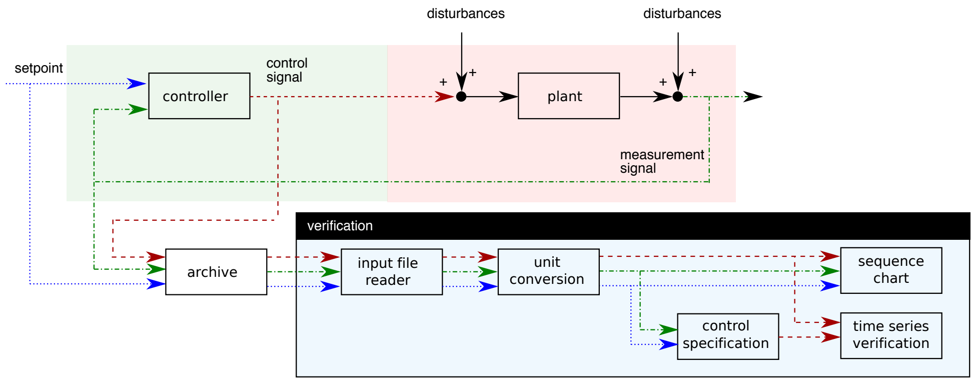


Figure 2: Overview of the verification that tests whether the installed control sequence meets the specification.

x-y or scatter plots. These can be used to verify for example that an economizer outdoor air damper has the expected position as a function of the outside air temperature.

Below we further describe the blocks in the box labeled *verification*.

Modules of the verification test

To conduct the verification, the following models and tools are used.

Input file reader

Input files are read with `Modelica.Blocks.Sources.CombiTimeTable` from the Modelica Standard Library.

Unit conversion

Building automation systems store physical quantities in various units. To convert them to the units used by Modelica and hence also by CDL, we developed the Modelica package `Buildings.Controls.OBC.UnitConversions` that provides blocks to convert between SI units and units that are commonly used in the HVAC industry.

Sequence chart

To create sequence charts, we developed the Modelica package `Modelica.Utilities.Plotters`. This package consists of models that generate time series plots or scatter plots, and save them in one or multiple html files. The plotters allow for example to plot control sequences such as the one shown in figure 5. The blocks that accumulate the data to be plotted can be activated and deactivated, for example to plot data only when the HVAC system is operating for at least 5 minutes.

Time series verification

We developed a cross-platform, C-based software called *funnel* (`https://github.com/lbl-srg/funnel`) to conduct time series comparison. The software reads two CSV files, one containing the reference data set and the other the test data set. Both CSV files contain time series that need to be compared against each other. The comparison is performed by computing a funnel around the reference curve. For this funnel, users can specify the tolerances with respect to time and with respect to the trended quantity. The L^1-norm is used to build the tolerance domain. The algorithm then checks whether the time series of the test data set is within this domain. For points outside the domain, it computes the corresponding error.

Examples

We will now present examples for the sequence diagram verification and the time series verification.

Sequence diagram verification

This is an example of the verification that tests whether an implemented control sequence leads to the expected relationship between the values of the control signal and the controlled variable, as visualized in a sequence diagram.

We verified that an implementation of the Guideline 36 supply air temperature heating and cooling setpoint sequence `Buildings.Utilities.Plotters.Examples.SingleZoneVAVSupply_u` leads to the expected sequence diagram as illustrated in figure 3. While in this example we used the control output of the CDL implementation, during commissioning one would use the control signal from the building automation system.

Figure 4 shows the verification model that we created using Modelica. On the left are the blocks that generate the control input. In a real verification, these would be replaced with a file reader that reads data that have been archived by the building automation system. In the center is the control sequence implementation. Some of its output is converted to degree Celsius, and then fed to the plotters on the right that generate a scatter plot for the temperatures and a scatter plot for the fan control signal. The block la-

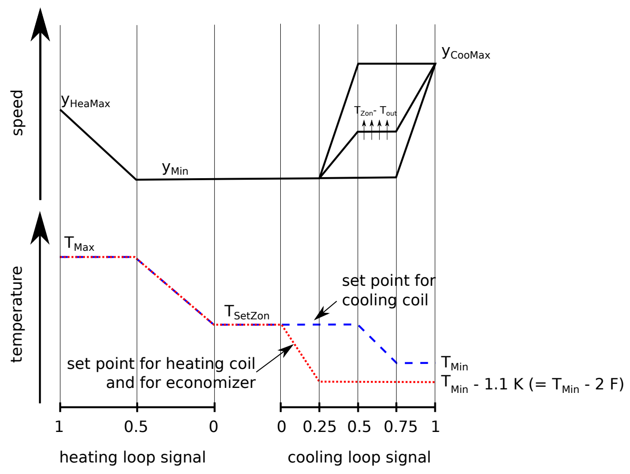


Figure 3: Control sequence diagram for the variable air volume flow single zone control sequence from ASHRAE Guideline 36.

beled `plotConfiguration` configures the file name for the plots and the sampling interval.

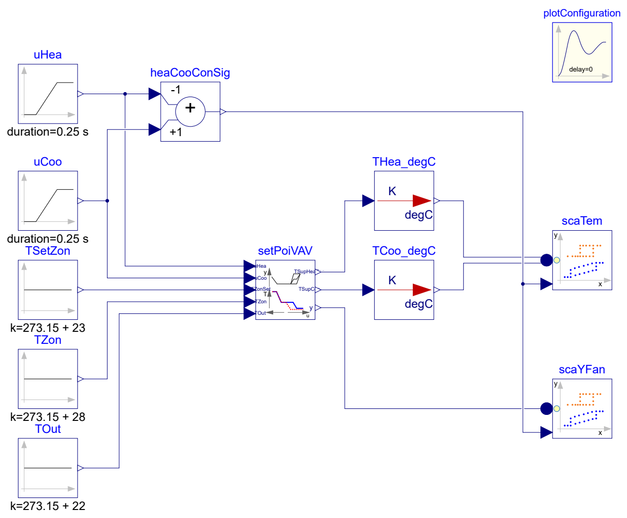


Figure 4: Modelica model that verifies the sequence diagram.

Simulating the model shown in figure 4 generates an html file that contains the scatter plots shown in figure 5. The plots complies with the specification provided in figure 3.

Sequence output time series verification

The process for the time series verification is as follows: Given a CDL specification of the implemented control sequence, a commissioning agent trends the sequence inputs and the outputs, reads the trended inputs into the CDL specification, executes the CDL specification, and verifies CDL computed outputs against the trended control outputs. This will be done for the top-level sequences as well as for lower level sequences.

In this example we validated a trended output of a control sequence that defines the cooling coil valve position. The cooling coil valve sequence is a part of

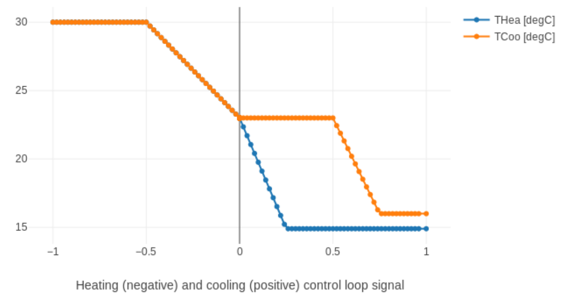


Figure 5: Scatter plots that show the control sequence diagram generated from the simulated sequence.

the ALC EIKON control logic implemented in building 33 on the main LBNL campus in Berkeley, CA. The subsequence is shown in figure 6. It consists of a proportional-integral (PI) controller that tracks the supply air temperature, an upstream subsequence that enables the controller and a downstream output limiter that is active in case of low supply air temperatures.

We created a CDL specification of the same cooling coil valve position control sequence, shown in figure 7, to validate the recorded output. We recorded trend data in 5 second intervals for the supply air temperature, the supply air temperature setpoint, the outdoor air temperature, the VFD fan enable status, and the VFD fan feedback. The output of the subsequence is the cooling coil valve position.

The input and output trends were processed with a script that converts them to the format required by the data readers. The data used in the example begins at midnight on June 7 2018. In addition to the input and output trends, we recorded all parameters, such as the hysteresis offset, shown in figure 6(a), and the controller gains, shown in figure 6(b), to use them in the CDL implementation.

We configured the CDL PID controller parameters such that they correspond to the parameters of the ALC PI controller. The ALC PID controller implementation is described in the ALC EIKON software help section, while the CDL PID controller is described in the info section of the model `Buildings.Controls.OBC.CDL.Continuous.LimPID`. The ALC controller tracks the temperature in degree Fahrenheit, while CDL uses SI units. An additional implementation difference is that for cooling applications, the ALC controller uses direct control action, whereas the CDL controller needs to be configured to use reverse control action, which can be done by setting its parameter `reverseAction` to `true`. Furthermore, the ALC controller outputs the control action in percentages, while the CDL controller outputs a signal between 0 and 1. To reconcile the differences, the ALC controller gains were converted for CDL as follows: The CDL PI controller proportional gain, $k_{p,cdl}$, was set to

$$k_{p,cdl} = u\, k_{p,alc}, \tag{1}$$

where $u = 9/5$ is a ratio of one degree Celsius (or Kelvin) to one degree Fahrenheit of temperature difference, and $k_{p,alc}$ is the proportional gain of the ALC PI controller, as obtained from the settings illustrated at figure 6(b). The CDL integrator time constant was calculated as

$$T_{i,cdl} = \frac{k_{p,cdl}\, I_{alc}}{u\, k_{i,alc}}, \tag{2}$$

where I_{alc} is the ALC controller interval at which the integral error gets updated, and $k_{i,alc}$ is the integral gain of the ALC PI controller, shown in figure 6(b).

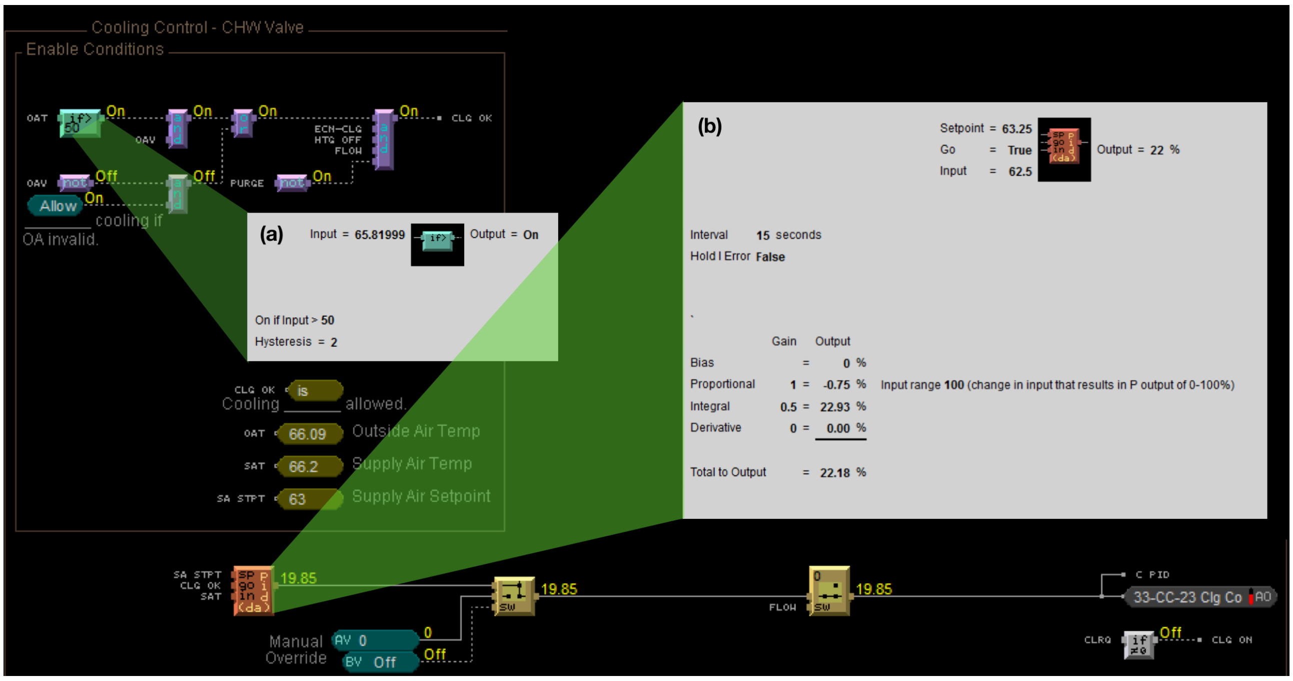
Both controllers were enabled throughout the whole validation time. Figure 8 shows the Modelica model that was used to conduct the verification. On the left hand side are the data readers that read the input and output trends from files. Next are unit converters, and a conversion for the fan status between a real value and a boolean value. These data are fed into the instance labeled `cooValSta`, which contains the control sequence as shown in figure 7. The plotters on the right hand side then compare the simulated cooling coil valve position with the recorded data.

Figure 9, which was produced by the Modelica model using blocks from the `Buildings.Utilities.Plotters` package, shows the trended input temperatures for the control sequence (top plot), the trended and simulated cooling valve control signal for the same time period (middle plot), which are practically on top of each other, and a correlation error between the trended and simulated cooling valve control signal (bottom plot).

The small difference we observed between modeled vs. trended results is due to the following factors:

- ALC EIKON uses a discrete time step for the time integration with a user-defined time step length, whereas CDL uses a continuous time integrator that adjusts the time step based on the integration error.
- ALC EIKON uses a proprietary algorithm for the anti-windup, which differs from the one used in the CDL implementation.

Despite these differences, the computed and the simulated control signals show good agreement, which is also demonstrated by verifying the time series with the *funnel* software, whose output is shown in figure 10. The exceeding error on the control signal is less than 2% and is located during the startup transient after a long shut off period, representing less than 2% of the simulated time period.



(a) Input = 65.81999 Output = On

On if Input > 50
Hysteresis = 2

(b) Setpoint = 63.25
Go = True
Input = 62.5
Output = 22 %

Interval 15 seconds
Hold I Error False

	Gain		Output
Bias		=	0 %
Proportional	1	=	-0.75 %
Integral	0.5	=	22.93 %
Derivative	0	=	0.00 %
Total to Output		=	22.18 %

Input range 100 (change in input that results in P output of 0-100%)

Figure 6: ALC EIKON specification of the cooling coil valve position control sequence. (a) ALC EIKON outdoor air temperature hysteresis to enable/disable the controller. (b) ALC EIKON PI controller parameters.

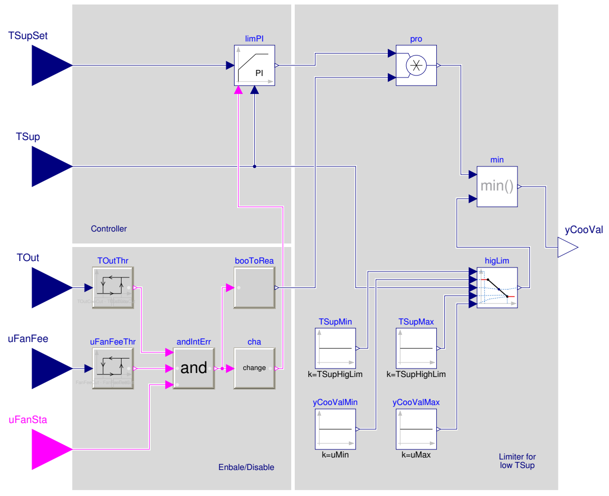


Figure 7: CDL specification of the cooling coil valve position control sequence.

Discussion

The example shows successful verification based on a small test case. In this example, we started with an actual implementation and created its CDL equivalent for the verification. In the workflow shown in figure 1, the sequence in CDL would be a starting point for the implementation of the actual sequence, and therefore this step would not be needed. However, due to differences in the dynamic response of equipment, some control gains may have to be tuned differently in the actual implementation, and some controllers may use autotuning for PI gains. This

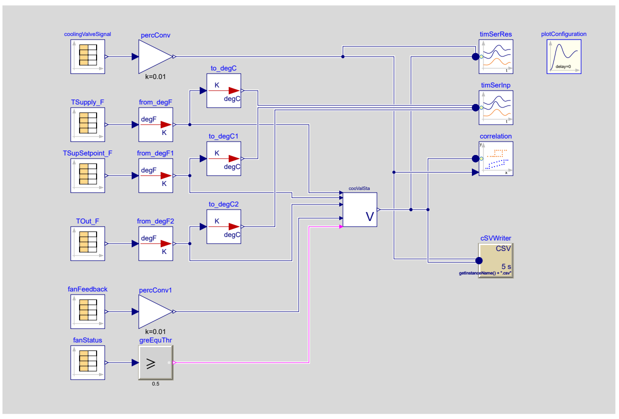


Figure 8: Modelica model that conducts the verification.

may have to be reflected in the CDL implementation used for the verification, unless the differences are within the user-configurable tolerances, which for practical applications can likely be set considerably larger than what we used to generate figure 10.

While we conducted our verification offline for three days, tools could be extended to conduct continuous verification to ensure that sequences are not getting inadvertently changed or disabled by the operator, such as after a manual override. Such continuous verification could contribute to the sustained high performance operation of buildings.

The difficulties that we have encountered while conducting the verification using trended data were related to the temporal resolution of the trended data and to the lack of information about the proprietary control blocks, such as the PI controller. A small sam-

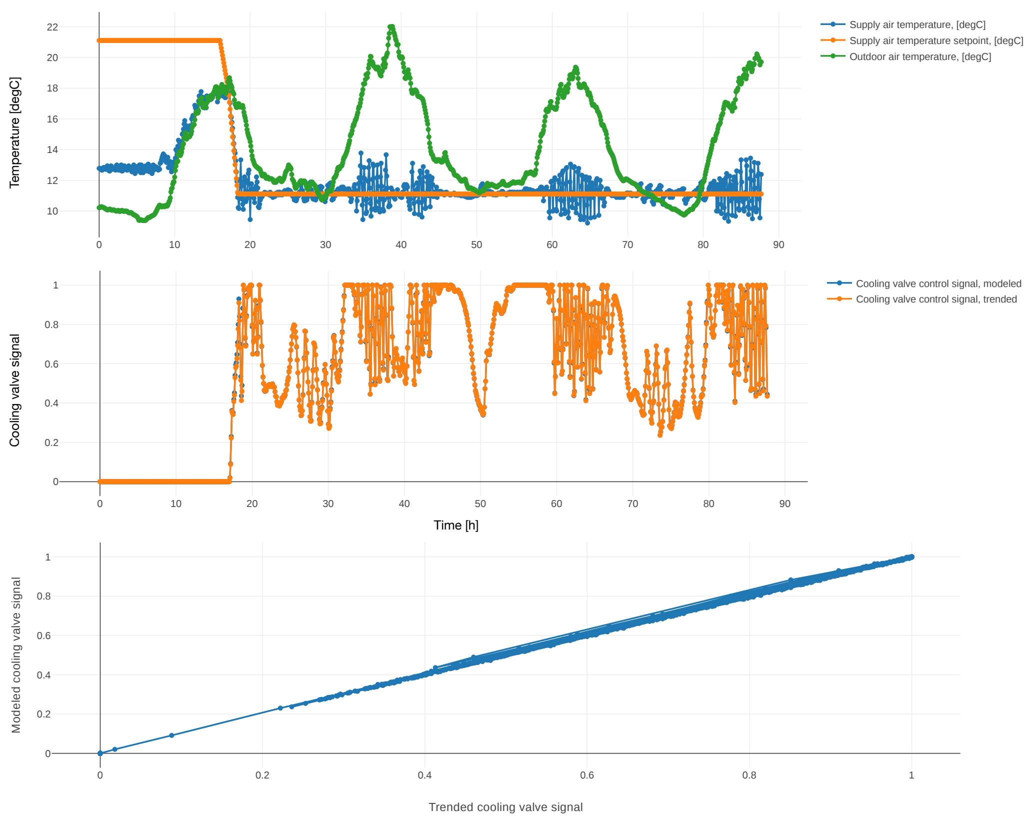


Figure 9: Plot, generated by plotters in the Modelica model shown in figure 8, that compares the cooling valve control signal between ALC EIKON and the simulation. Upper: Trended input signals. Middle: Simulated and trended control signal, which are almost overlapping. Lower: Simulated and trended control signal correlation.

ple time interval is required to capture fast changes in control input and output signals. For large sequences, this may cause problems as today's building control systems have limited capabilities to trend large data streams at high frequency. More field verification is required to see how big the compounding effect is of model mismatch between CDL and vendor-specific implementations of certain control blocks, such as a PI controller, differences in control parameters that may be tuned in the field, such as PI gains, and communication delays of actual hardware.

We also considered in a development version[1] the use of performance-oriented tests such as "Room air temperature shall be within the setpoint ± 0.5 Kelvin for at least 45 min within each 60 minute window and "Damper signal shall not oscillate more than 4 times per hour between a change of ± 0.025 (for a 2 minute sample period)". However, discussions with design engineers and commissioning providers showed that there is currently no accepted threshold that could be used to turn such performance-oriented statements into formal, testable requirements. We believe this is a need that should be addressed, in particular as it can aid commissioning of both, actual implementations as well as simulation models.

Besides these tests, we also considered automatic fault detection and diagnostics methods that were proposed for inclusion in ASHRAE guidelines, and we considered using methods such as described in Veronica (2013) that automatically detect faulty regulation, including excessively oscillatory behavior. However, as it is not yet clear how sensitive these methods are to site-specific tuning, and because field tests are ongoing in a NIST project, we did not implement them.

It is also outside the scope of our verification to test whether the I/O points are connected properly, the mechanical equipment is installed and functions correctly, and the building envelope is meeting its specification.

Conclusions

Based on a small example, we demonstrated successful verification between an actual control response and the response computed by a model expressed in the Control Description Language during three summer days. Having an executable specification of the control sequence would allow the buildings industry to conduct formal verification tests that ensure that the design intent is indeed implemented in the real control system. As programming errors are the leading cause of control related problems (Barwig et al., 2002), and because control sequences can affect

[1]Modelica Buildings Library, commit `https://github.com/lbl-srg/modelica-buildings/commit/454cc7521c0303d0a3f903acdda2132cc53fe45f`.

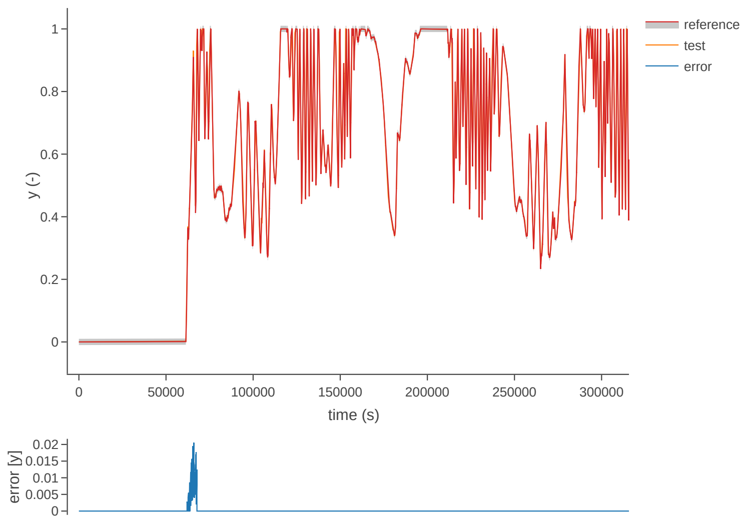


Figure 10: Verification of the cooling valve control signal y with the funnel software. Error computed with an absolute tolerance in time of 1 s *and a relative tolerance in y of* 1%

HVAC annual site energy use by one third (Wetter et al., 2018b), such a formal process seems to be an important contribution to consistently achieving high performance buildings.

Tools could be extended to conduct the verification continuously to ensure that sequences are not getting inadvertently changed or disabled by the operator, such as after a manual override. Such continuous verification could further contribute to the sustained high performance operation of buildings.

Acknowledgments

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 Commissions Electric Program Investment Charge (EPIC) Program.

We thank Chris Weyandt from LBNL for his support with data collection and sequence implementation.

References

ASHRAE 2018. *ASHRAE Guideline 36-2018 – High Performance Sequences of Operation for HVAC systems.* ASHRAE.

Barwig, F. E., House, J. M., Klaassen, C. J., Ardehali, M. M., and Smith, T. F. 2002. The national building controls information program. Summer Study on Energy Efficiency in Buildings, Pacific Grove, CA. ACEEE.

Fernandez, N. E., Katipamula, S., Wang, W., Xie, Y., Zhao, M., and Corbin, C. D. 2017. Impacts of commercial building controls on energy savings and peak load reduction. Technical Report 25985, PNNL.

Gnerre, B. and Fuller, K. 2017. When building controls veer off course. In *Facility Executive*, volume 462707, page 32. Group C Media, Inc., Tinton Falls, NY.

Mattsson, S. E. and Elmqvist, H. 1997. Modelica – An international effort to design the next generation modeling language. In Boullart, L., Loccufier, M., and Mattsson, S. E., editors, *7th IFAC Symposium on Computer Aided Control Systems Design*, pages 1–5, Gent, Belgium.

Veronica, D. A. 2013. Automatically detecting faulty regulation in HVAC controls. *HVAC&R Research*, 19(4):412–422.

Wetter, M., Grahovac, M., and Hu, J. 2018a. Control description language. In *1st American Modelica Conference*, Cambridge, MA, USA.

Wetter, M., Hu, J., Grahovac, M., Eubanks, B., and Haves, P. 2018b. Openbuildingcontrol: Modeling feedback control as a step towards formal design, specification, deployment and verification of building control sequences. In *Proc. of Building Performance Modeling Conference and SimBuild*, pages 775–782, Chicago, IL, USA.

Wetter, M., Zuo, W., and Nouidui, T. S. 2011. Modeling of heat transfer in rooms in the Modelica "Buildings" library. In *Proc. of the 12-th IBPSA Conference*, pages 1096–1103. International Building Performance Simulation Association.

Wetter, M., Zuo, W., Nouidui, T. S., and Pang, X. 2014. Modelica buildings library. *Journal of Building Performance Simulation*, 7(4):253–270.