Modelica Meets ASHRAE: Towards A Digital Standard for Building Control

Michael Wetter[1] Yan Chen[2] Karthik Devaprasad[2] Paul Ehrlich[3] Antoine Gautier[4] Jianjun Hu[1] Anand Prakash[1] Marco Pritoni[1]

[1]Lawrence Berkeley National Laboratory, Berkeley, CA, USA
[2]Pacific Northwest National Laboratory, Richland, WA, USA
[3]Building Intelligence Group, Afton, MN, USA
[4]Solamen, Laval, France

Abstract

Today's process for designing, specifying, installing and testing building HVAC control is not digitalized, leading to expensive manual workflows and missed operational performance. To digitalize the design-build-operate process for building HVAC control, the authors developed a process, associated tools and initiated the voluntary ASHRAE Standard 231P. This paper describes this process and tools, which are both based on Modelica. Standardization through a proposed voluntary ASHRAE Standard and the existing Modelica Language Specification provides a robust technology foundation for industry investment. It is based on declarative specification of the control logic, and allows reuse of existing technologies for open-loop and closed-loop control testing through coupling with an HVAC system or a whole building energy model. It supports control testing using MIL, SIL and HIL, and export of digital twins for operational support. The process and ASHRAE Standard 231P have been designed to accommodate existing Building Automation Systems product lines, while also enabling direct code generation such as by using FMI or eFMI. Control deployment can be digital or manual and conformance to the digital specification can be tested formally and programmatically at each step of the control delivery.

Keywords: Controls, Digitalization, Modelica Buildings Library, ASHRAE Standard 231P

1 Introduction

Design and implementation of control sequences for building and district energy systems has shown to be problematic. Barwig et al. (2002) showed that about one third of the errors in control systems for built-up HVAC systems, those that typically serve medium and large buildings that cover more than half of the floor area of the building stock, are attributed to programming errors. The problem of programming errors remains also today, as more recent literature shows (Crowe et al. 2020; Torabi et al. 2022). HVAC designers usually lack funding and skills needed to develop control logic. Rather than providing the control logic, they write the control intent, typically in the form of a 20 to 50 page untested Word document that has ambiguous formulations. The control provider then interprets this English language text and implements the control logic in software, often by reusing code from previous projects. This process has proven problematic even for the older HVAC systems, which have generally much simpler sequences than today's high performance control sequences. Modern controls are tasked with integrating heat pumps, chillers, and storage to not only meet temperature setpoints but also improve efficiency and shift energy use to periods with lower energy rates. The current control design and delivery process has not kept pace with digitalization. It remains highly manual, based on ambiguous specification, with minimal testing beyond spot checks during commissioning. While control providers have product lines that support communication standards, such as ASHRAE Standard 135 (BACnet, ASHRAE (2024)) or KNX, there is no standard for expressing the control logic.

To address these gaps, the authors have been developing a process and underlying tools for the digitalization of the controls design, delivery, implementation and commissioning. In support of this effort, the authors created a language for expressing control logic, called Control Description Language CDL (Wetter, Grahovac, and Hu 2018). Together with Modelica template models for HVAC systems and tools for configuring and translating CDL, CDL forms the foundation for control digitalization. CDL is defined as a small subset of Modelica, chosen in a way that allows translation to many legacy control languages, yet expressive enough to build configurable control sequences that allow for example specifying the type of sensors (e.g., a room CO_2 sensor), the type of equipment (e.g., water-based cooling coil, direct expansion cooling coil, presence of a heat recovery chiller etc.) and then automatically configure a control logic for the specified system.

Some of the authors also initiated an ASHRAE Standards Committee, and further developed CDL into the proposed ASHRAE Standard 231P[1].

This paper provides an overview of the CDL language,

[1]See https://www.ashrae.org/technical-resources/standards-and-guidelines

the control libraries that are being developed based on CDL, and the tools to configure controls and to translate CDL to target environments, which can be new or legacy Building Automation Systems (BAS) or Energy Information Management Systems (EMIS). As recent EMIS can also provide control capabilities (Pritoni et al. 2022), we collectively call them BAS below. The paper also describes verification of control logic implemented in a BAS to test conformance to the control specification that is expressed in CDL. Additionally, it highlights companies that have been incorporating CDL into their products and processes. Further information on earlier work related to this project can be found at `https://obc.lbl.gov`.

Assuming a widespread adoption of the digital control design-build-operate process presented here, the annual US electricity cost savings would be $8.2*B* and the natural gas savings $1.07*B* (CEC 2021).

2 Methodology

The development of the digital workflow started with the identification of key problems in today's building control design-build-commission-operate process, followed by stakeholder interviews and the development of use case and requirements. For industry adaption, the new workflow needs to reduce time and cost, address the lack of skills and budget among mechanical designers tasked with specifying controls, and ensure that the software solution is easy to support with existing control product lines. Additionally, it must be future-proof to enable more modern software solutions than those currently available in many automation product lines. Further, it has to enable rigorous sequence verification in a formal, programmatic way, which the authors believe is a core requirement that few control providers recognize today.

For this digital workflow, we developed the Control Description Language (CDL) (Wetter, Grahovac, and Hu 2018), which is a subset of Modelica. CDL prescribes basic blocks that are easy to support by legacy control systems, and it prescribes rules to compose such blocks hierarchically to implement control logic. The use of Modelica allows to perform simulations for design and verification of control and mechanical system operation coupled to a whole building or district energy model. It also allows code generation for future control product lines such as via the FMI or eFMI standard (Blochwitz et al. 2011; Lenord et al. 2021).

We validated CDL by creating HVAC control libraries, and using these libraries for closed-loop performance assessment with whole building energy simulation, using either Modelica or Spawn of EnergyPlus to simulate the building envelope model (Wetter, Ehrlich, et al. 2022; Zhang et al. 2022; Wetter, Benne, et al. 2023). Additionally, we developed a new workflow for digital control delivery, and we prototyped each step of the process, from design to code generation for WebCTRL, a commercial control product line from ALC (Automated Logic, a Carrier company) and verification of as-installed control logic in a real building (Wetter, Gautier, et al. 2019). See Wetter, Ehrlich, et al. (2022) for an overview.

To standardize the underlying language for control representation, in 2020, a formal application was submitted to ASHRAE to establish a standards committee, resulting in the proposed ASHRAE Standard 231P "CDL - A Control Description Language for Building Environmental Control Sequences." This standard aims to define a declarative graphical programming language for building environmental control sequences that is both human- and machine-readable, designed for specification, implementation via machine-to-machine translation, documentation, and simulation. The ASHRAE Standard 231P committee held its kick-off meeting in September 2020, and the proposed standard is expected to undergo a second, and potentially final, public review in 2025.

In the meantime, various control libraries and support tools have been implemented, and they are described in the following sections.

3 Workflows

Figure 1 shows a simplified flow chart of the digital workflow enabled by CDL. The figure is based on the more comprehensive workflow shown in Figure 2 in Wetter, Ehrlich, et al. (2022). Depending on the HVAC characteristics, energy code and project requirements, a mechanical engineer uses ctrl-flow (see Section 3.5), a web-based tool, to select and configure the HVAC system and control sequence through menu options. Then, the ctrl-flow software outputs the sequence specification. This consists of the English language sequence description, the digital control sequence conforming to ASHRAE Standard 231P, (essentially a hierarchical block-diagram in Modelica), and the semantic model conforming to ASHRAE Standard 223P. These steps are achieved using the modelica-json software described in Section 3.4.[2] A control provider then implements this control logic in a BAS. This can either be done manually using today's workflow, or preferably through software translation, which we prototyped and explained in Section 4.2 in Wetter, Ehrlich, et al. (2022). The control provider then instantiates the control logic in a controller and links inputs and outputs to physical devices or virtual control points. Regardless of whether the control logic was implemented through software translation or through manual programming, a commissioning agent then trends BAS inputs and outputs, and compares them against outputs computed by the CDL representation for the same control input and parameters. This verification test can be done using the funnel software [3], and prototype code for such verification is described in Section 4.3 in Wetter, Ehrlich, et al. (2022).

[2] modelica-json generates these outputs based on a configured Modelica model. The ability to export the configured Modelica model, as an additional output, is a planned feature for 2025.

[3] See `https://github.com/lbl-srg/funnel`.

DOI 10.3384/ecp218505

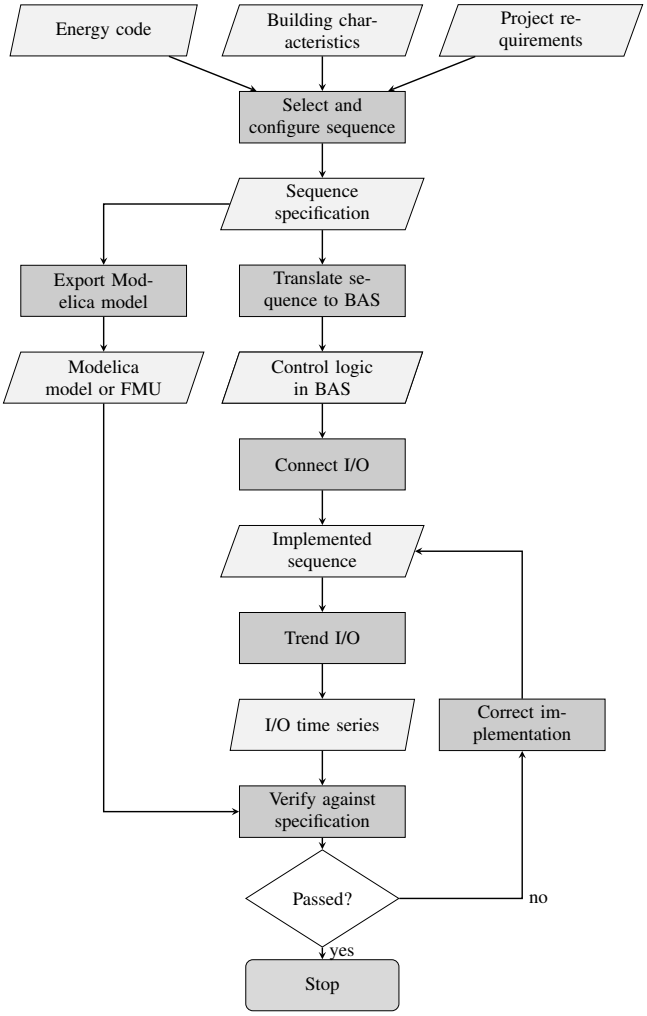


Figure 1. Workflow from control sequence configuration to export of a specification, implementation on a BAS and verification during commissioning. (Simplified representation based on Wetter, Ehrlich, et al. (2022).)

This final step of the workflow provides formal verification ensuring that the as-implemented control logic reproduces the control signal in accordance with the specification that was exported by the mechanical designer. Thus, we give the control vendor the option to use software translation, but don't require to do so, and regardless, we offer the ability to test the implemented control logic for compliance with the CDL specification.

While the above workflow is for an actual building, the above cited publication shows a variant that is applicable for control providers to conduct open-loop tests to verify that they implemented the control logic correctly.

3.1 CDL

CDL is a small subset of Modelica that is needed to implement block diagrams that are sufficient for use in Modelica templates (Gautier et al. 2023), yet simple to implement and modify by non-Modelica experts. CDL defines permissible data types (`Real`, `Integer`, `Boolean`, **`enumeration`**). It also defines immutable, so-called *Elementary Blocks*, such as the `Reals.Add` block that outputs the sum of two real-valued input signals. The language allows single inheritance blocks and the composition of blocks to form a *Composite Block* that may encapsulate a complex hierarchical control logic. It also allows for conditional connectors and instances, enabling the implementation of a room controller that may include, for example, an optional input and associated control logic for the room CO_2 concentration.

To reduce complexity of CDL, and because legacy BAS may not support advanced Modelica concepts, we excluded various Modelica constructs. Excluded constructs are, for example, `Clock`, **`inner`**/**`outer`**, physical ports such as a fluid port, and finite state machines. However, CDL has the concept of an *Extension Block* that allows to encapsulate any Modelica construct. Originally implemented to satisfy the requirement of calling compiled proprietary control logic, an `Extension Block` also allows, for example, the implementation of a finite state machine. ASHRAE Standard 231P requires each *Extension Block* to be exported to an FMU-ME 2.0 when exporting CDL to CXF, which is an exchange format described below. Control vendors are allowed to implement *Elementary Blocks* and *Extension Blocks* in their language of choice, as long as the mapping from inputs, parameters and states produces the same outputs. Thus, these blocks are defined through their inputs, outputs and the mapping from inputs and current states to outputs.

CDL also specifies the Modelica vendor annotation `__cdl`, which is used, among other things, to declare semantic information, or to specify whether an input needs to be hardwired or networked, such as through the BACnet protocol.

3.2 Standardization via ASHRAE Standard 231P

One critical step to digitalizing control delivery is to develop a standard which then provides a solid foundation for industry to adopt and invest in the technology. Approved standards are widely used by government and industry. We chose to develop an ASHRAE/ANSI standard, to facilitate potential future ISO recognition. The ASHRAE/ANSI process is consensus-based and involves various stakeholders. In 2020 the CDL team approached ASHRAE with a proposal and received approval to initiate a standards project. This effort was titled "ASHRAE 231P, Controls Description Language." The project committee consists of 30 members, including a diverse mix of control system manufacturers, HVAC control system designers, and researchers. The committee has been meeting regularly and reviewing and discussing the structure and format of the standard. This has resulted in discussions related to key differences between how control logic is handled in simulation and in building automation product lines. For example, Modelica distinguishes between real and integer data types while many BAS represent integers as integer-valued real data types. Additionally, many BAS lack mechanisms for event iteration, or for synchronizing simultaneous time events. The standards committee discussed and reached an agreement on what blocks should be supported as *Elementary Blocks*.

ASHRAE Standard 231P also specifies the Control eXchange Format (CXF), a representation of CDL in a JSON-LD format that is intended to be used as an intermediate format when importing CDL to a BAS, as JSON-LD is easier to parse than Modelica. For example, a control provider might utilize JSON-LD to import control logic from a design tool and deploy it to their commercial BAS for a particular project. While CDL has language constructs that are used to build library of sequences, CXF was designed to only represent a specifically configured logic. For example, CXF has no mechanism to conditionally remove a block and all its connections, as is provided by Modelica and allowed in CDL. CXF can be exported from CDL using the modelica-json software described below.

In 2024 the committee released an initial draft standard for public review. It is anticipated that a second review will occur in 2025 which, depending on feedback, could result in the publication of an ASHRAE/ANSI Standard in early 2026, which could then be submitted for ISO approval.

3.3 Sequence library and HVAC templates

Concurrently with developing CDL, the authors have created CDL compliant reference implementations of control sequences published in ASHRAE Guideline 36 (ASHRAE 2018), along with the accompanying HVAC system models.[4] These parallel developments created valuable feedback where practical implementation challenges directly informed the CDL specification. One example of such feedback is the introduction of the *Extension Block* concept which provides support for finite state machines (see Section 3.1). This addition helped address one of our main difficulties: using discrete event modeling to represent synchronous logic, as described hereafter.

In sequence documentation, the word "when" (as opposed to "if") typically signals such event-based patterns. Consider the following clauses that handle availability conditions in the staging logic of a plant: "Any unavailable stage is skipped during staging events. *When* the current stage *becomes* unavailable, the transition to the next higher available stage is triggered." We first implemented these clauses by merging them into a single state-based condition: "*If* a stage *is* unavailable, the transition to the next stage is triggered." Now, if the plant is in stage 1 while stage 2 is unavailable and a stage-up command is generated, a simple test shows that, in the model time domain, the plant effectively transitions directly to stage 3. However, issues appear in the event domain where an intermediate step is taken, during which stage 2 is enabled before being disabled at the next event. This intermediate zero-time event triggers all the staging sequence at stage 2, which is unwanted. Refactoring the implementation to be event-safe proved challenging when using only CDL *Elementary Blocks*, but was greatly simplified by using finite state machines. This also improved the block diagram readability, facilitating maintenance tasks. [5]

With our studies involving closed-loop simulations (Wetter, Ehrlich, et al. 2022; Zhang et al. 2022), we realized both the value and the difficulty of integrating equipment models and detailed controls into models that can be easily configured and parameterized. Depending on the system configuration and control options, the set of required actuators and sensors changes. The corresponding components and their connections must be updated consistently, while ensuring that the model remains well formulated for simulation. For example, lumping flow resistances is a good practice that reduces the size of nonlinear algebraic loops when modeling pressure-flow networks (Jorissen, Wetter, and Helsen 2015). But this approach becomes error-prone when the system configuration changes, e.g., when actuated isolation valves are removed, knowing that these are typically the components used to merge the fixed flow resistance of a chiller evaporator and condenser. Parameterizing hydronic system models brings further requirements, such as flow balanc-

[4] Both developments are released as part the Modelica Buildings Library, in the packages `Buildings.Controls.OBC.ASHRAE` and `Buildings.Templates`.

[5] The resulting *Extension Block* `StageIndex` in the Modelica Buildings Library shows the complexity of simply computing the current stage index of a multiple-unit system based on stage change commands. The discussions in the issues #3787, #3952 and #3966 further illustrate the iterations it took to find event-safe implementations for CDL *Elementary Blocks*.

DOI 10.3384/ecp218505

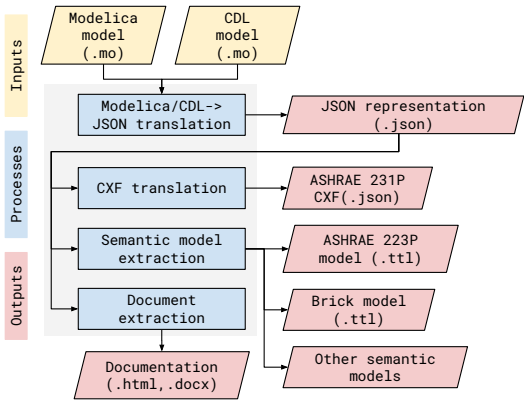


Figure 2. Inputs and outputs supported by modelica-json.

ing between unequally sized equipment in constant primary flow systems. This involves calculating the flow coefficient of balancing valves in case of constant speed pumps, or the pump speed (potentially at each plant stage) in case of variable speed pumps operated at fixed speed. Templates have been developed to programmatically address such tasks, enabling users to configure detailed closed-loop models by simply specifying top-level parameters. The templates handle all the complexity of model creation and parameterization, while embodying modeling best practices and making state-of-the-art control sequences broadly accessible. This is achieved exclusively with Modelica code, leveraging advanced class parameterization features of the language (Gautier et al. 2023).

The templates distributed with version 12.0.0 of the Modelica Buildings Library cover variable-air-volume systems and air-to-water heat pump plants. Templates for boiler and chiller plants are planned for release with the next version of the Library.

3.4 Modelica-json translator

Figure 2 shows modelica-json, a software that can be used to translate Modelica models into a JSON representation (Wetter, Hu, et al. 2021), and from there to generate various other outputs. The JSON representation is used to generate CXF, the representation of the control logic, and to generate a semantic model (see Section 3.4.2). As Modelica and CDL classes support the inclusion of semantic information (Fierro and Pauwels 2022), modelica-json allows the extraction of semantic models for ontologies such as Brick and ASHRAE 223P, as well as other schemas such as Project Haystack. Modelica-json is also used to generate documentation in html or Microsoft Word.

3.4.1 CDL-CXF translation

CXF is a semantic, graph-based representation of a CDL control logic serialized in JSON Linked Data (JSON-LD). Based on JSON-LD and using classes such as `S231:Block`, `S231:Connector` and `S231:Parameter`, and using relationships such as `S231:hasInput`, `S231:containsBlock` and `S231:isConnectedTo`, modelica-json constructs the CXF representation of the CDL control logic. Structurally the content of a logic in CDL and CXF are identical, in that both utilize the same *Elementary Blocks*, *Composite Blocks*, and *Extension Blocks* as well as `Constants`, `Parameters`, `InputConnectors` and `OutputConnectors`. There are, however, a few differences: First, for *Elementary Blocks* in CDL, their CXF representation does not include the implementation (equation section) of the particular block. Second, CXF allows translation tools to either retain array references and expressions present in CDL or to flatten the arrays and evaluate the expressions.

3.4.2 Semantic model extraction

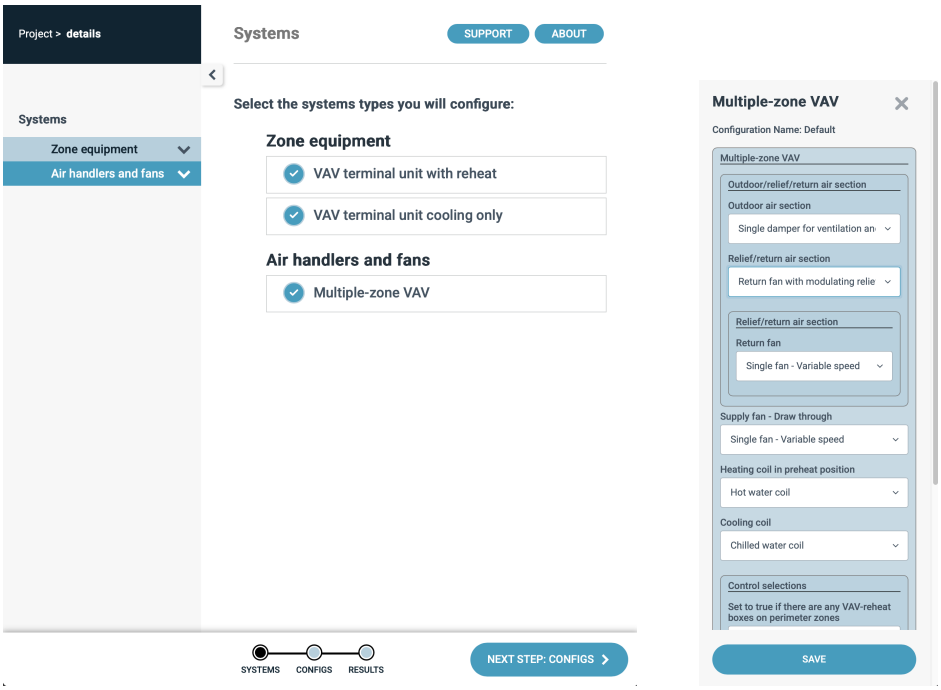
CDL supports the inclusion of semantic information (Fierro and Pauwels 2022) in a control logic using the `__cdl(semantic(...))` Modelica vendor annotation. While, as with other Modelica annotations, these annotations have no influence on the behavior of the control logic, they can be used to add information that is helpful for trending and communication, and for specifying semantic information. These items all support streamlining the configuration of control and communication, connecting or configuring energy analytics programs, and exporting natural language documentation.

Through the `__cdl(semantic(...))` annotation, semantic information can be added to blocks (both the class and the instance definitions), packages, parameters and input and output connectors. Controls developers can include information adhering to a particular semantic ontology or data schema using the `semanticLanguage` keyword. In this case, the included information must conform to the `semanticLanguage` and a supported format, such as `text/turtle`, `application/json` or `application/ld+json`. They can also add documentation in a natural language using the `naturalLanguage` keyword, with the documentation in the `text/plain` format. Similarly, semantic information can also be added to models developed with the Modelica Buildings Library, beyond CDL, by using the `__Buildings(semantic(...))` vendor annotation.

Modelica-json enables the export the semantic information as a separate file that can be shipped along with the CDL (or CXF) control logic or the Modelica model. For example, if a CDL control logic contains semantic shape requirements for the input and output points, described using ASHRAE Standard 223P and SHACL (W3C 2008), modelica-json can export these shapes for each of the connectors. These semantic shapes could then be used to integrate the sequences with BACnet communication points within a BAS.

3.5 ctrl-flow

Figure 3 shows ctrl-flow, a web-based application that aims to facilitate the adoption of the digital workflow by the HVAC designers. The tool is open-source and available at `https://ctrl-flow.lbl.gov`. It solicits user inputs to specify the characteristics of HVAC systems



(a) System selection. **(b)** Configuration panel.

Figure 3. ctrl-flow user interface.

and controls as shown in Figure 3, and outputs standardized specification documents. In its current state of development, the tool enables creating the control sequence documentation for variable-air-volume systems compliant with ASHRAE Guideline 36 (ASHRAE 2018). Ongoing projects are extending both the scope of supported systems and the generated content to include control schematics, point lists, equipment schedules and Modelica models of the specified systems.

At its foundation, ctrl-flow uses templates from the Modelica Buildings Library (see Section 3.3) and can accommodate any Modelica model that conforms to the guidelines available at `https://lbl-srg.github.io/modelica-buildings-templates`. On the server side (*Node Backend* in Figure 4) the tool invokes modelica-json and further processes the JSON representation of the template abstract syntax tree into an intermediate format (*ctrl-flow Schema*) that is more tractable by the React frontend. This schema is then populated by user selections and written back to the *Node Backend*, from where it is returned to the *Python Doc Pipeline*. The *Python Doc Pipeline* creates the sequence documentation using a DOCX template file, and then sends it to the *React Frontend* for user download.

Since templates often rely on advanced Modelica constructs, the tool required an interpreter layer capable of handling Modelica grammar's intricacies such as replaceable elements (instance or class) with optional constraining clauses, conditional declarations or composite bindings with `record` instances. Additionally, the frontend needed an integrated expression evaluation engine to process Boolean expressions used in user interface annotations that control the visibility of input fields.

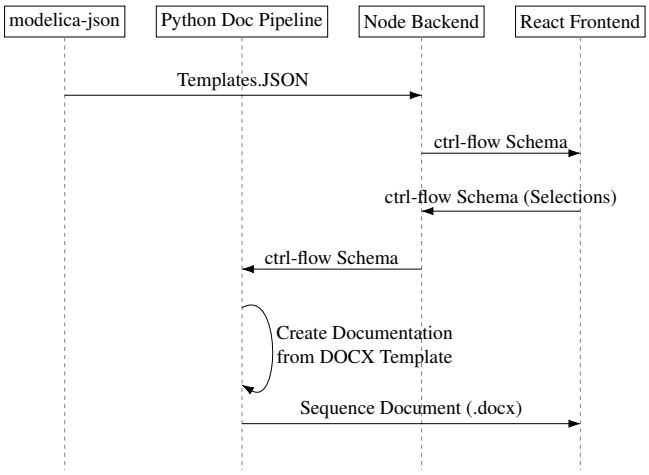


Figure 4. Data flow diagram of ctrl-flow.

In practice, this translates into handling an object of roughly 20 MB in size that gets regenerated at each user interaction while maintaining both responsiveness in the interface and simplicity in the user experience.

3.6 Import of CDL in BAS

Today's BAS generally do not accommodate generating C code and executing the compiled code, as is done, e.g., in embedded systems. Therefore, CDL was designed so that its mathematical mapping of inputs and states to outputs can be replicated by a BAS, and the declarative specification of the block diagram be translated to the programming

DOI
10.3384/ecp218505

language of the target environment. Hence, supporting CDL can be accomplished either through code generation, or through translation of CDL. The translation of CDL consists of mapping the CDL block diagrams to a representation in the target platform that given the same inputs and states, produces the same outputs. Our ongoing collaboration with various BAS providers indicate that only minor changes if any are needed in the target BAS. This requirement for compatibility with existing BAS product lines was the reason for excluding Modelica constructs such as `inner`/`outer`, `expandable connector` and `Modelica.Clocked`.

For a translation from CDL to a BAS, control providers seem to favor the JSON-LD representation rather the CDL representation, as it avoids having to implement a Modelica parser, although we encountered a company that started doing so. If one designs a new control product line, we recommend exploring the use of the eFMI Standard (Lenord et al. 2021) and the SSP standard (Modelica Association 2019) for using code in a BAS.

3.7 Verification

The proposed digital workflow supports the verification of the control logic relative to the original specification, which is expressed as a model in the CDL language. The verification compares the difference between the outputs of the controller minus the outputs computed by the specification, both using the same control inputs, parameters and constants.[6] The verification uses the funnel software to verify that the differences are within user-specified tolerances in time and in signal value. If all differences are within the tolerances, the tested control logic conforms to the specification.

Note that this verification process only tests the control logic implementation relative to the specification. It neither evaluates the suitability of the control intent as specified in the control specification, nor does it test the performance of the closed loop system or the proper operation of the HVAC components. Such functional tests and performance tests have been deliberately excluded from the control logic verification, and should be done separately.

We will now describe two distinct use cases, the first being the verification of a control logic implemented in a building system, and the second being the testing of a control logic as part of product development during which a controller can be run open loop. Both use cases are also described in more detail in Wetter, Gautier, et al. (2019) and Wetter, Ehrlich, et al. (2022)

3.7.1 Controller installed in a building system

If a controller is installed in a building system, it may not be possible to arbitrarily change the control inputs without detrimentally affecting the safe operating of the building system. Therefore, in this workflow, a commissioning agent can run the building system through acceptable operational scenarios and trend the inputs and outputs of the controller. Next, the CDL model is simulated with the same parameters and inputs, and its outputs are recorded. Using the funnel software, tests can be run to verify that the differences between trended and simulated outputs are within a user-set tolerance.

3.7.2 Open loop testing of a controller

Prior to installation in a building system, a controller could also be executed open loop to test its response to different sets of parameters and inputs. Therefore, in this workflow, a control technician executes the CDL model for a variety of parameters and inputs, and records the output. Next, the actual controller is executed for the same parameters and inputs, and its outputs are recorded. As in the previous use case, the outputs of the simulated and actual controller are compared.

4 Industry collaboration

The control industry has shown significant interest in developing and adopting standards for digitalization of control as described above. All of the major US BAS vendors have been actively participating in the creation of ASHRAE Standard 231P, with representatives being voting members of the committee. The authors have been collaborating with Carrier's Automated Logic to prototype a translator from CDL to the EIKON language, as described in Wetter, Ehrlich, et al. (2022), which is now being further developed. Additionally, another BAS vendor, Tridium, and two EMIS vendors, Skyfoundry and Normal inc, have recently begun developing parsers and translators from CDL or CXF to their product lines. Furthermore, the authors have been working with KTC, a controls and energy optimization company, to incorporate ASHRAE Standard 231P into their design-build-operate workflow via a translation to a PLC controller using IEC 61131-10 XML. Notably, Meta has been using the Modelica Buildings Library and CDL to design and operate hyperscale data centers and optimize their operation using Modelica-based digital twins (Rivalin et al. 2023).

5 Outlook

We expect that in 2025, the ASHRAE Standards 231P and 223P undergo a formal vote for approval. Once accepted, this will provide a robust foundation for BAS and EMIS providers for the digitalization of control and energy information system delivery. Through the ESTCP project "DoD Building Controls Design Tool"[7] that is led by the US Army Construction Engineering Research Laboratory, the authors are working on adopting the ctrl-flow software to the needs of DoD who owns around 550,000 buildings and facilities and thus provides an opportunity for scaled

[6] Typically, one also has to set initial states. However, many BAS do not allow setting initial states or recording them. In this case, the verification should start once initial transients disappeared.

[7] See `https://serdp-estcp.mil/projects/details/a964da95-b45e-404a-b7e1-d8dd6fcc246f/dod-building-controls-design-tool`.

deployment. The authors are also working with DOE and DoD/ESTCP in exporting Modelica template models for combined heat pump and chiller plants for buildings and district energy systems, and including them in ctrl-flow as part of a design guide for the robust, scalable deployment of such systems. In addition the authors are working on a California Energy Commission project to implement new demand-flexible control sequences in four California campuses using CDL in collaboration with two BAS and three EMIS providers.

6 Conclusion

Our proof of concept translation and various follow on work showed that a small subset of Modelica that is simple to use and support suffices for many building control representation in support of a digital design-deploy-verify process, with a notable exception of certain complex equipment staging sequences that are much simpler to implement in finite state machines, for which CDL provides a means for including such encapsulated Modelica constructs. With CDL, which is a Modelica subset, and its CXF representation, which is in JSON-LD, languages were developed that allows controls design, closed loop performance assessment with HVAC and building models in the loop, creation of template libraries of HVAC and associated control logic, and digital deployment to control product lines.

The next challenge is to scale its use in the buildings industry to make workflows cheaper, more robust and repeatable based on a highly digitalized workflow that allows formal validation and verification at every step along the design, build, and operate process. With the inherent higher control complexity that we see for energy efficient, grid-responsive and resilient systems, we believe such a new process is essential to control costs and meet the technical performance goals.

7 Data Availability

The CDL reference implementation, the control sequence libraries and the HVAC and control templates are available in the Modelica Buildings Library (`https://github.com/lbl-srg/modelica-buildings`). The modelica-json translator is available at `https://github.com/lbl-srg/modelica-json`. The guide for how to implement Modelica templates for use with ctrl-flow is available at `https://lbl-srg.github.io/modelica-buildings-templates`. The ctrl-flow source code is at `https://github.com/lbl-srg/ctrl-flow-dev` and the online web version is at `https://ctrl-flow.lbl.gov/`. A specification of CDL and CXF, together with accompanying process descriptions and publications, is available at `https://obc.lbl.gov/`. The ASHRAE Standards 231 and 223, once approved, will be available at `https://www.ashrae.org/technical-resources/standards-and-guidelines` (access to draft versions is restricted to committee members) and an unofficial specification of CDL together with other information is available at `https://obc.lbl.gov`.

Acknowledgments

This work 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 under Contract No. DE-AC05-76RL01830.

References

ASHRAE (2018-06). *ASHRAE Guideline 36-2018 – High Performance Sequences of Operation for HVAC systems*. ASHRAE.

ASHRAE (2024). *ANSI/ASHRAE Standard 135-2024, BACnet, A Data Communication Protocol for Building Automation and Control Networks*.

Barwig, Floyd E. et al. (2002-08). "The National Building Controls Information Program". In: Summer Study on Energy Efficiency in Buildings. ACEEE. Pacific Grove, CA.

Blochwitz, T. et al. (2011-03). "The Functional Mockup Interface for Tool independent Exchange of Simulation Models". In: *Proc. of the 8-th International Modelica Conference*. Modelica Association. Dresden, Germany. DOI: 10.3384/ecp11063105.

CEC (2021-02). *Open Building Control*. Tech. rep. CEC 500-2021-012. Sacramento, CA: California Energy Commission. URL: https://www.energy.ca.gov/sites/default/files/2021-05/CEC-500-2021-012.pdf.

Crowe, Eliot et al. (2020). "Building commissioning costs and savings across three decades and 1500 North American buildings". In: *Energy and Buildings* 227, p. 110408. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2020.110408. URL: https://doi.org/10.1016/j.enbuild.2020.110408.

Fierro, Gabe and Pieter Pauwels (2022). "Survey of metadata schemas for datadriven smart buildings (Annex 81)". In: URL: https://annex81.iea-ebc.org/Data/publications/IEA%20Annex%2081%20Survey%20of%20Metadata%20Schemas.pdf.

Gautier, Antoine et al. (2023-12). "HVAC and control templates for the Modelica Buildings Library". In: *Proceedings of the 15th International Modelica Conference 2023*. Ed. by Dirk Müller, Antonello Monti, and Andrea Benigni. Linköping Electronic Conf. Proc. 204. Aachen, Germany: Modelica Association and Linköping University Electronic Press, pp. 217–227. ISBN: 978-91-8075-505-4. DOI: 10.3384/ecp204217. URL: https://doi.org/10.3384/ecp204217.

Jorissen, Filip, Michael Wetter, and Lieve Helsen (2015-09). "Simulation speed analysis and improvements of Modelica models for building energy simulation". In: *Proceedings of the 11th International Modelica Conference 2015*. Ed. by Peter Fritzson and Hilding Elmqvist. Linköping Electronic Conf. Proc. 118. Versailles, France: Modelica Association and Linköping University Electronic Press, pp. 59–69. ISBN: 978-91-7685-955-1. DOI: 10.3384/ecp1511859. URL: https://doi.org/10.3384/ecp1511859.

Lenord, Oliver et al. (2021-09). "eFMI: An open standard for physical models in embedded software". In: *Proc. of the 14th International Modelica Conference*. Ed. by Martin Sjölund et

al. Linköping Electronic Conf. Proc. 181. Linköping, Sweden: Modelica Association and Linköping University Electronic Press, pp. 57–71. ISBN: 978-91-7929-027-6. DOI: 10.3384/ecp2118157. URL: https://doi.org/10.3384/ecp2118157.

Pritoni, Marco et al. (2022). "From fault-detection to automated fault correction: a field study". In: *Building and Environment* 214. DOI: 10.1016/j.buildenv.2022.108900. URL: https://doi.org/10.1016/j.buildenv.2022.108900.

Rivalin, Lisa et al. (2023-05). *Predicting Temperature and Differential Pressure in Data Centers Using Physical Modeling*. Tech. rep. Meta Platforms, Menlo Park, CA. URL: https://research.facebook.com/file/752832263209106/IAQVEC-2023_full-paper.pdf.

Modelica Association (2019-03). *System Structure and Parameterization*. Tech. rep. URL: https://ssp-standard.org/.

Torabi, Narges et al. (2022). "Common human errors in design, installation, and operation of VAV AHU control systems - A review and a practitioner interview". In: *Building and Environment* 221, p. 109333. ISSN: 0360-1323. DOI: 10.1016/j.buildenv.2022.109333. URL: https://doi.org/10.1016/j.buildenv.2022.109333.

W3C (2008). *SPARQL 1.1 overview*. en. https://www.w3.org/TR/sparql11-overview/. Accessed: 2025-3-5.

Wetter, Michael, Kyle Benne, et al. (2023). "Spawn: coupling Modelica Buildings Library and EnergyPlus to enable new energy system and control applications". In: *Journal of Building Performance Simulation*, pp. 1–19. DOI: 10.1080/19401493.2023.2266414. URL: https://doi.org/10.1080/19401493.2023.2266414.

Wetter, Michael, Paul Ehrlich, et al. (2022-01). "OpenBuildingControl: Digitizing the control delivery from building energy modeling to specification, implementation and formal verification". In: *Energy* 238.A. DOI: 10.1016/j.energy.2021.121501. URL: https://doi.org/10.1016/j.energy.2021.121501.

Wetter, Michael, Antoine Gautier, et al. (2019-09). "Verification of Control Sequences within OpenBuildingControl". In: *16-th IBPSA Conference*. International Building Performance Simulation Association. Rome, Italy, pp. 885–892. DOI: 10.26868/25222708.2019.210722. URL: http://simulationresearch.lbl.gov/wetter/download/2019-ibpsa-OpenBuildingControl.pdf.

Wetter, Michael, Milica Grahovac, and Jianjun Hu (2018-08). "Control Description Language". In: *1st American Modelica Conference*. Cambridge, MA, USA. DOI: 10.3384/ecp1815417. URL: https://doi.org/10.3384/ecp1815417.

Wetter, Michael, Jianjun Hu, et al. (2021-09). "Modelica-json: Transforming energy models to digitize the control delivery process". In: *17-th IBPSA Conference*. Intern. Building Performance Simulation Assoc. Brugge, Belgium, pp. 1–8. DOI: 10.26868/25222708.2021.30141. URL: https://doi.org/10.26868/25222708.2021.30141.

Zhang, Kun et al. (2022-01). "Estimating ASHRAE Guideline 36 Energy Savings for Multi-zone Variable Air Volume Systems Using Spawn of EnergyPlus". In: *Journal of Building Performance Simulation* (2), pp. 215–236. DOI: 10.1080/19401493.2021.2021286. URL: https://doi.org/10.1080/19401493.2021.2021286.