Control Description Language

Michael Wetter

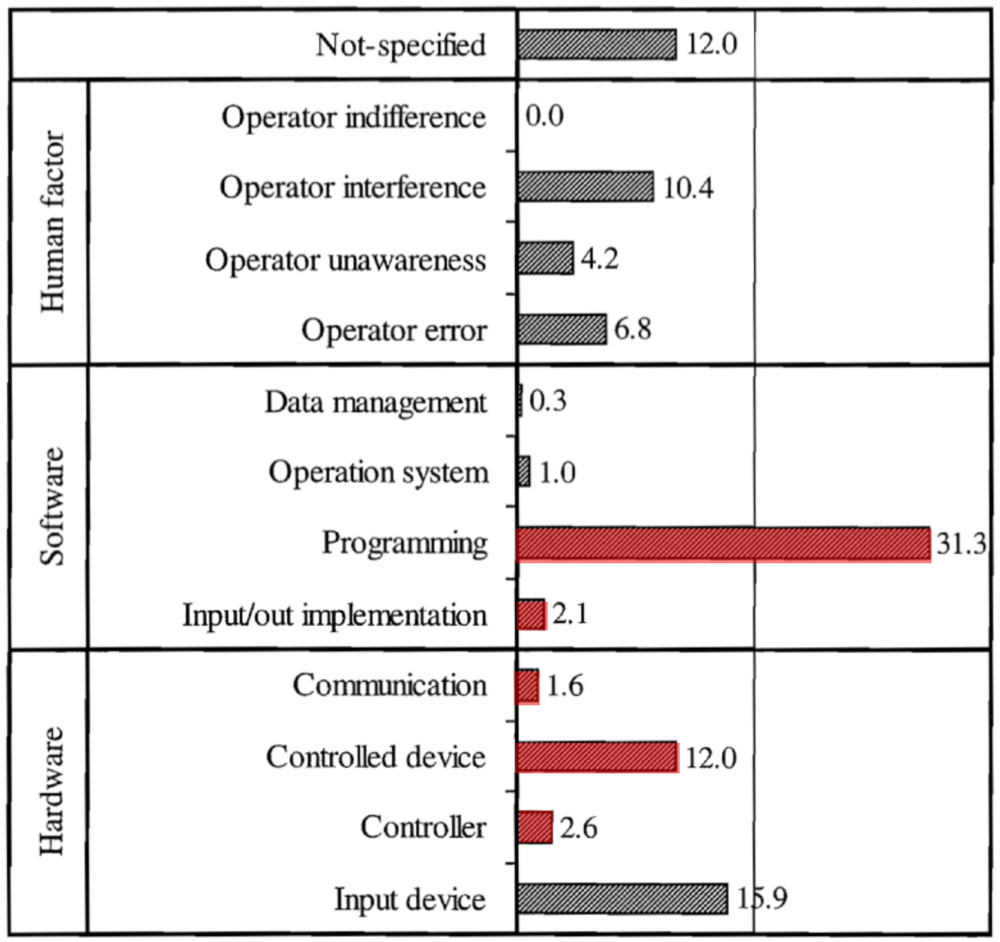
Milica Grahovac

Michael Wetter

October 9, 2018

BERKELEY LAB Lawrence Berkeley National Laboratory

Controls are the Achilles heel of commercial buildings, because there is no end-to-end quality control, and no standardization for control logic



Control-related problems (Ardehali, Smith 2002). While the study is not recent, discussions with mechanical designers and operators of large buildings confirmed that correct implementation of the control intent remains a problem.

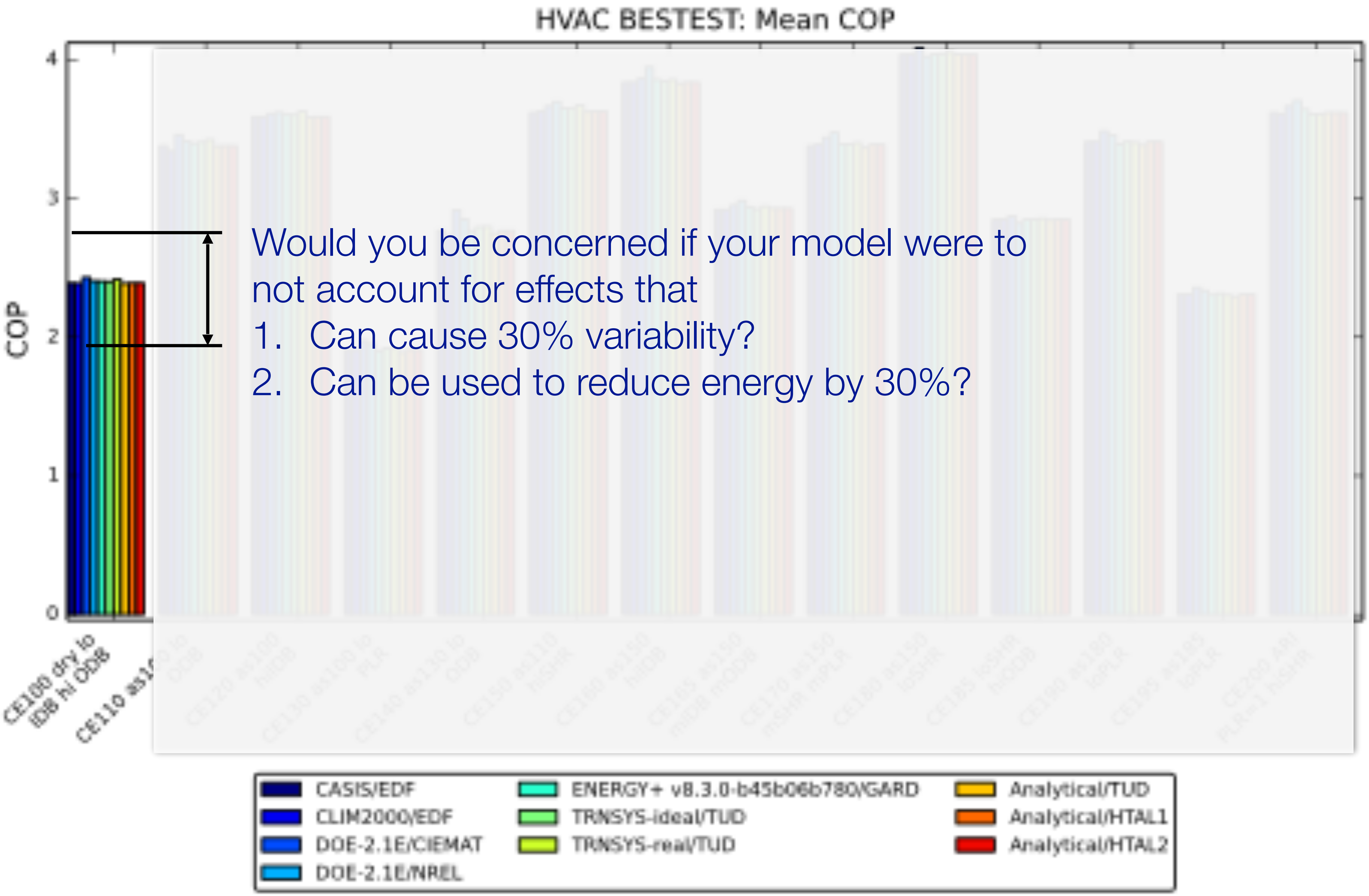
More than 1 quad/yr of energy is wasted in the US because control sequences are poorly specified and implemented in commercial buildings.

The process to specify, implement and verify controls sequences is often only partially successful, with efficiency being the most difficult part to quantify and realize.

This limits adoption of advanced control sequences as

- anticipated energy savings are not achieved,
- their expected ROI may be missed, and
- engineers are exposed to risk due to malfunctioning system integration, often leading to oversized or overengineered systems.

Impact



Would you be concerned if your model were to not account for effects that

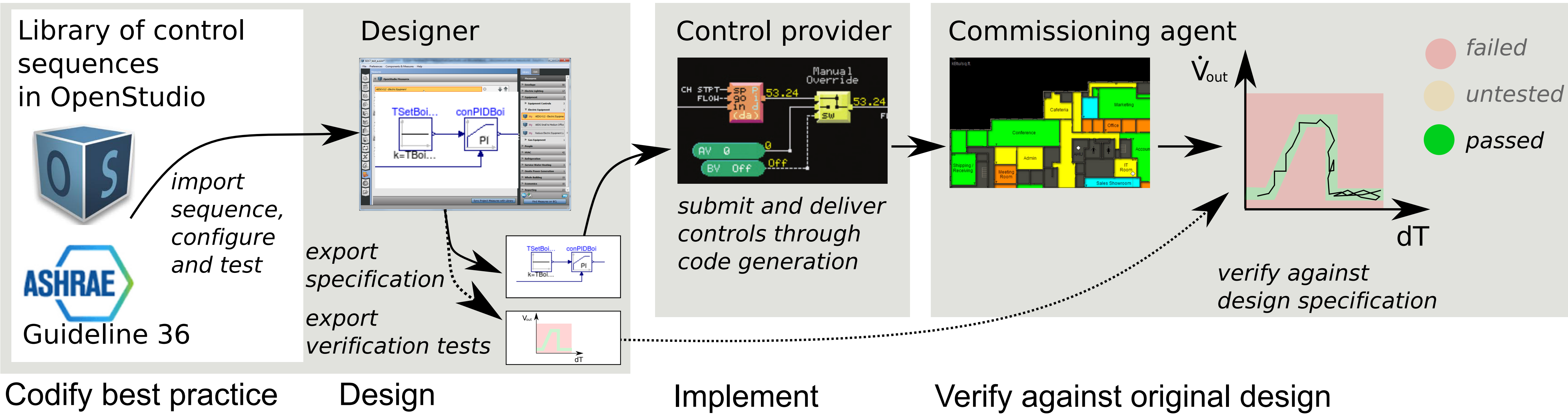
1. Can cause 30% variability?
2. Can be used to reduce energy by 30%?

Vision

What if

1. mechanical designers can import in building energy modeling tools best-in-class control sequences from ASHRAE-vetted guidelines?
2. mechanical designers can adapt these sequences to their project, and then exported them digitally for bidding and implementation, together with verification tests?
3. control providers could automatically implement these sequences in their building automation systems?
4. commissioning agents could verify formally that the sequences are implemented as specified?

OpenBuildingControl: Bridge silos between BEM and controls, and realize energy savings of advanced controls



BACnet standardizes communication, OpenBuildingControl will standardize control sequences & verification tests:

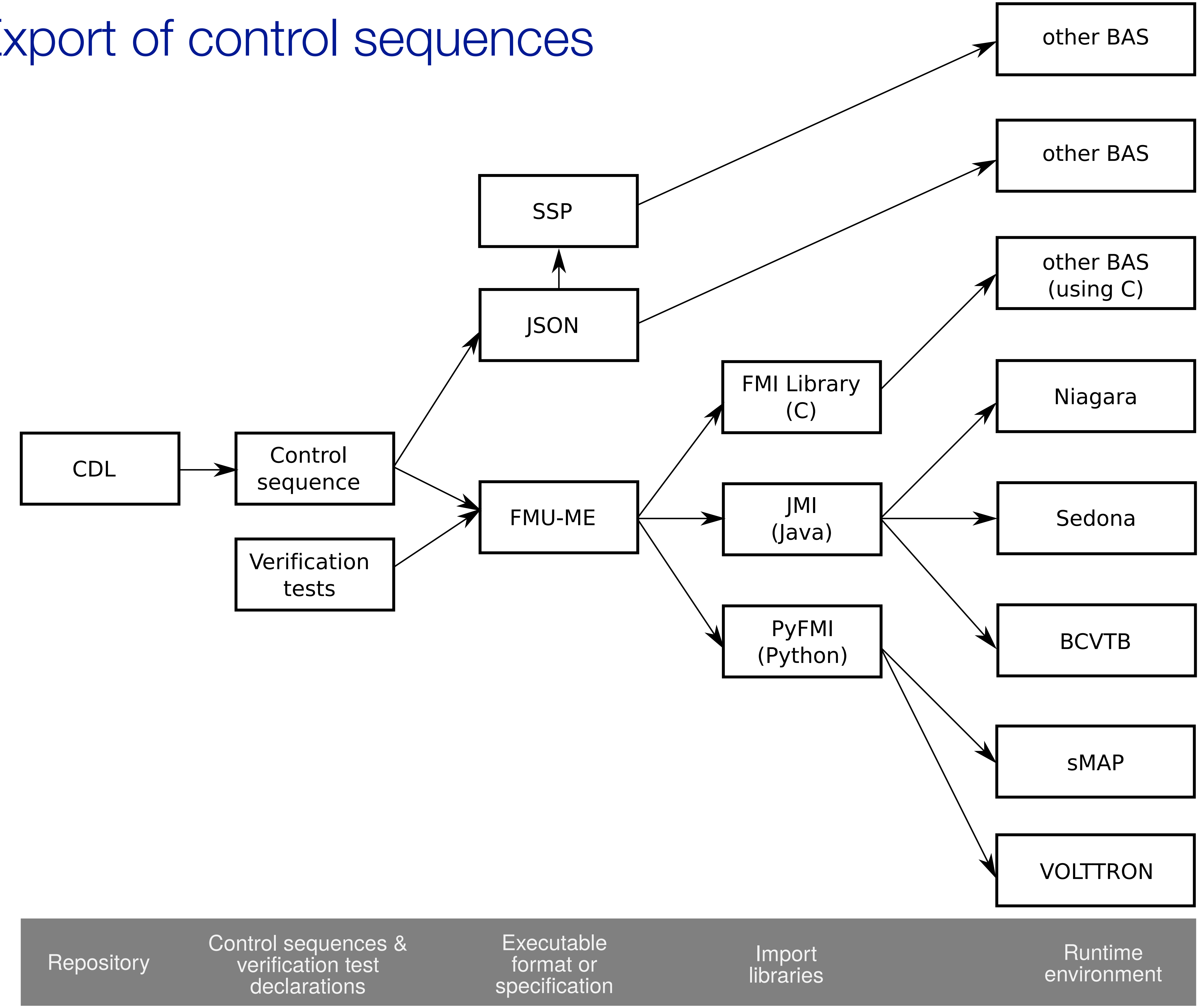
- basic functional building blocks
- composition rules for control sequences, and
- for bidding and automatic implementation
- declaration of functional verification tests criteria.

Key Innovations

Digital, executable control specification, called Control Description Language (CDL), enabling

- Sharing of best-practice, e.g., ASHRAE Guideline 36
- Error-free implementation of the specified control sequence
- Formal process that connects design to operation
- Formal verification of design intent

Export of control sequences

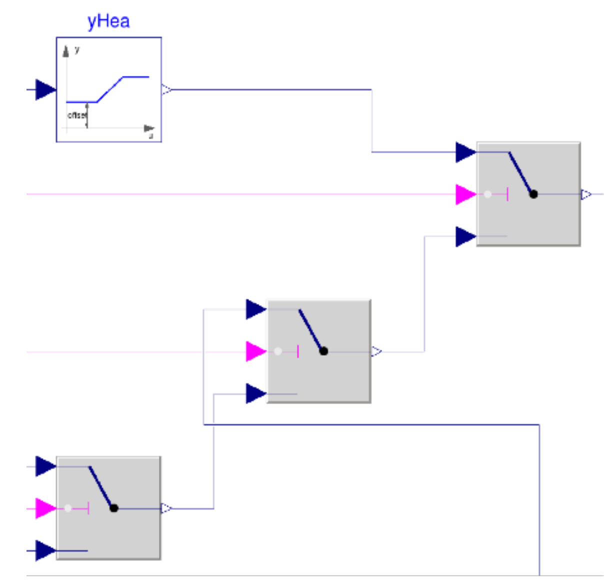


http://obc.lbl.gov

What is the Control Description Language?

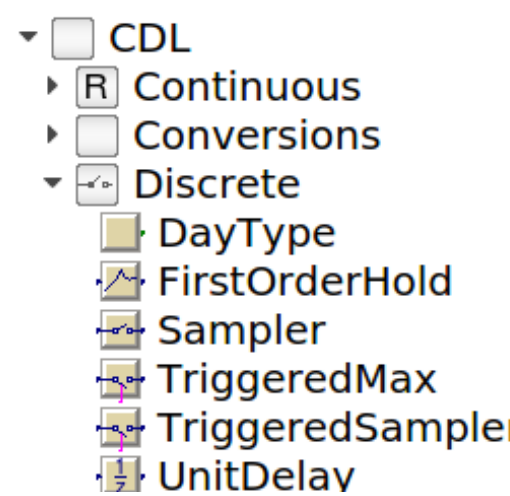
A declarative language for expressing block-diagrams for controls (and requirements)

A graphical language for rendering these diagrams.

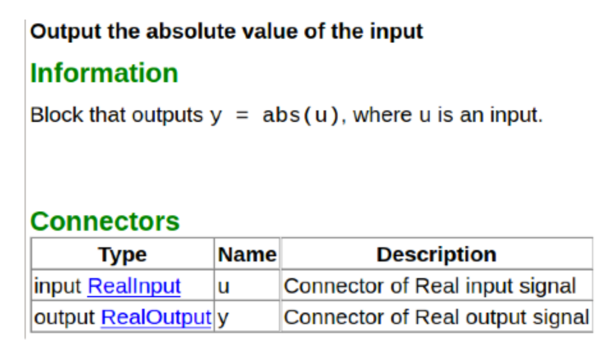


A library with elementary input/output blocks that should be supported [through a translator] by CDL-compliant control providers

Example: CDL has an adder with inputs `u1` and `u2`, gains `k1` and `k2`, and output `y = k1*u1 + k2*u2`.



A syntax for documenting the control blocks and diagrams.



Output the absolute value of the input

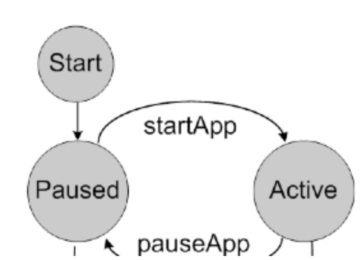
Information

Block that outputs `y = abs(u)`, where u is an input.

Connectors

Type	Name	Description
input RealInput	u	Connector of Real input signal
output RealOutput	y	Connector of Real output signal

A model of computation that describes the interaction among the blocks.



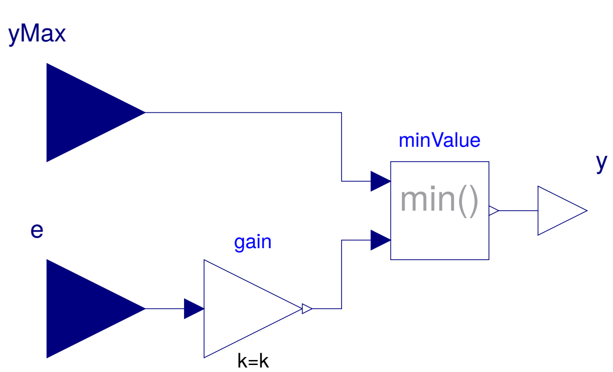
What is the Control Description Language?

Allowed constructs include

- parameters
- connect statements
- hierarchical models
- basis math operations when assigning parameters

```
CDL.Logical.Hysteresis hys(
 uLow  = pRel-25,
 uHigh = pRel+25)
 "Hysteresis for fan control";
```

- Composite models



Not allowed are

- acausal connectors
- variables
- equations (except in parameter assignments)
- anything other than “connect” statements in equation section
- initial equation, initial algorithm and algorithm
- use of blocks other than
  - from OBC.CDL,
  - composite blocks built using blocks from OBC.CDL
- State machines
- Clocks

CDL can be used to implement open or proprietary sequences

The standard to be supported by vendors

- CDL

Custom implementations can be built using the CDL language, and CDL blocks

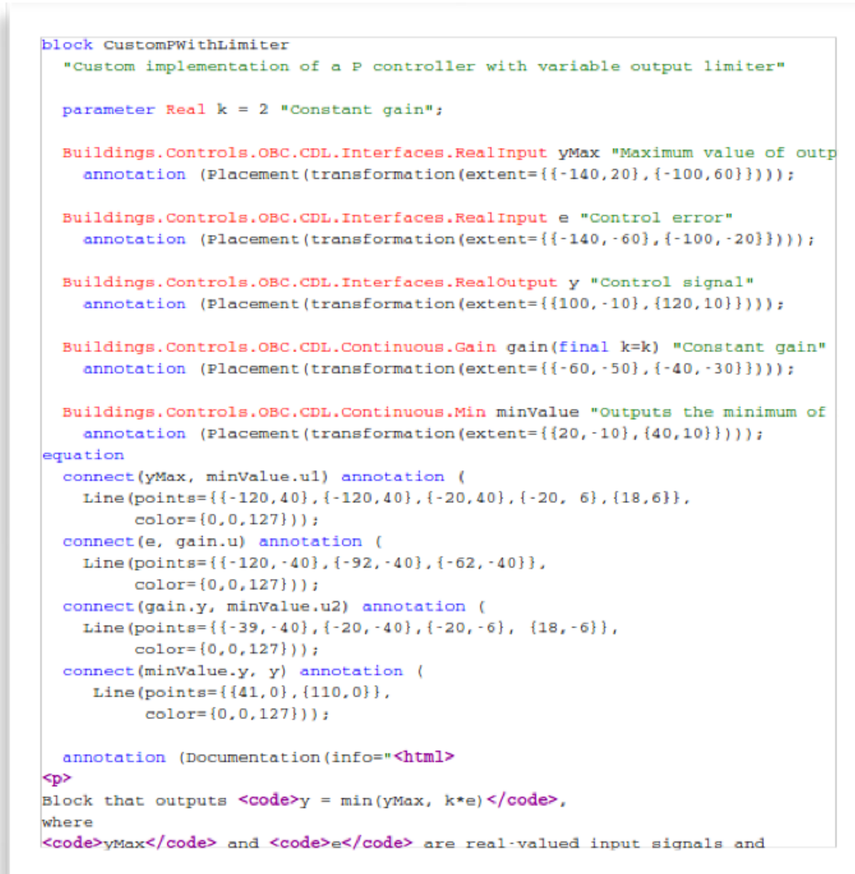
- ASHRAE
  - G36 — Sequences that come out of ASHRAE projects and can be shared with community.
- GSA — GSA preferred sequences, made available through a CDL-complaint implementation.
- ARUP — Design firms can share their own (proprietary) implementation across their offices.
- ALC — Control vendors can provide their own specialized sequences, either as open-source, or as compiled (proprietary) I/O blocks.

Companies are illustrative

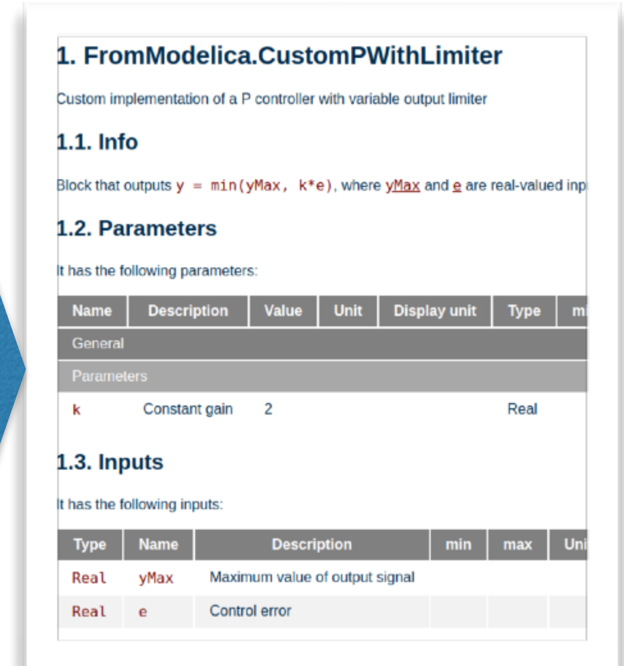
Control sequence translation tool

"modelica-json": parse control sequences written in Modelica to JSON, and from JSON to other format, such as html.

- different parse mode:
  - "cdl": ensure models following cdl syntax
  - "modelica": general modelica syntax
- graphical annotation
  - provide graphical layout for display in block diagram editors (Modelica or actual control platforms)
  - generate graphical diagram for inclusion in documentation (in svg format)



```
block CustomPWithLimiter
  "Custom implementation of a P controller with variable output limiter"

  parameter Real k = 2 "Constant gain";

  Buildings.Controls.OBC.CDL.Interfaces.RealInput yMax "Maximum value of outp
    annotation (Placement(transformation(extent={{-140,20},{-100,60}})));

  Buildings.Controls.OBC.CDL.Interfaces.RealInput e "Control error"
    annotation (Placement(transformation(extent={{-140,-60},{-100,-20}})));

  Buildings.Controls.OBC.CDL.Interfaces.RealOutput y "Control signal"
    annotation (Placement(transformation(extent={{100,-10},{120,10}})));

  Buildings.Controls.OBC.CDL.Continuous.Gain gain(final k=k) "Constant gain"
    annotation (Placement(transformation(extent={{-60,-50},{-40,-30}})));

  Buildings.Controls.OBC.CDL.Continuous.Min minValue "Outputs the minimum of
    annotation (Placement(transformation(extent={{20,-10},{40,10}})));
equation
  connect(yMax, minValue.u1) annotation (
    Line(points={{-120,40},{-120,40},{-20,40},{-20, 6},{18,6}},
         color={0,0,127}));
  connect(e, gain.u) annotation (
    Line(points={{-120,-40},{-92,-40},{-62,-40}},
         color={0,0,127}));
  connect(gain.y, minValue.u2) annotation (
    Line(points={{-39,-40},{-20,-40},{-20,-6}, {18,-6}},
         color={0,0,127}));
  connect(minValue.y, y) annotation (
     Line(points={{41,0},{110,0}},
          color={0,0,127}));

  annotation (Documentation(info="<html>
<p>
Block that outputs <code>y = min(yMax, k*e)</code>,
where
<code>yMax</code> and <code>e</code> are real-valued input signals and
```



1. FromModelica.CustomPWithLimiter

Custom implementation of a P controller with variable output limiter

1.1. Info

Block that outputs y = min(yMax, k*e), where yMax and e are real-valued inp

1.2. Parameters

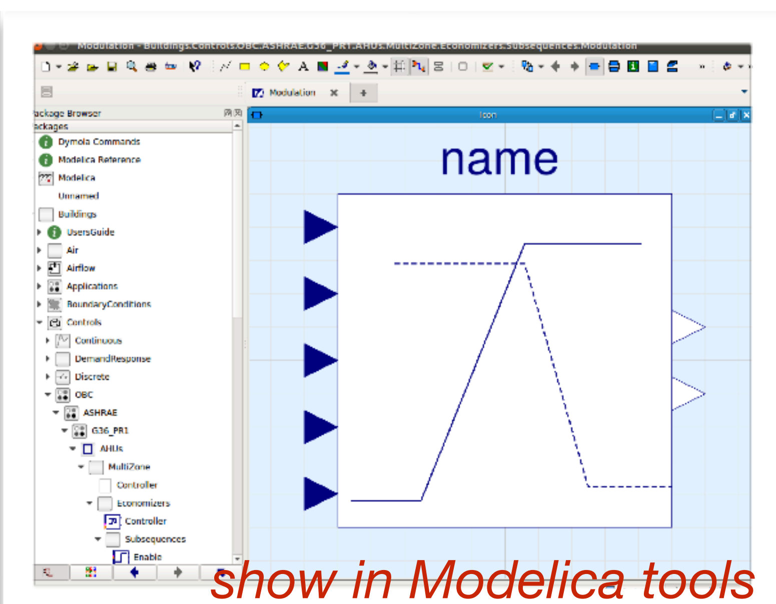
It has the following parameters:

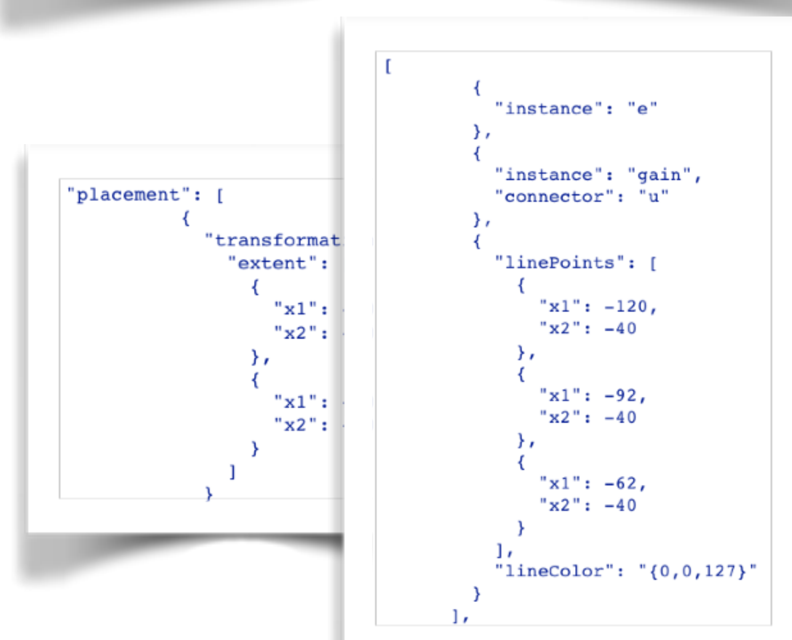
Name	Description	Value	Unit	Display unit	Type	m
General						
Parameters						
k	Constant gain	2			Real	

1.3. Inputs

It has the following inputs:

Type	Name	Description	min	max	Uni
Real	yMax	Maximum value of output signal			
Real	e	Control error			

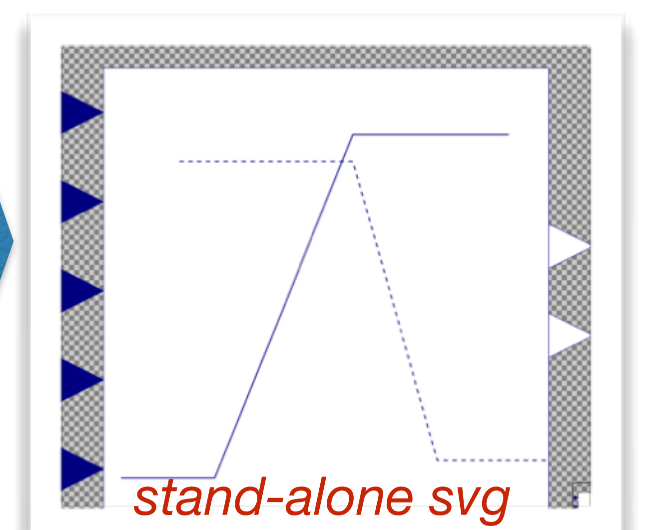




```
"placement": [
    {
        "transformat
         "extent":
          {
            "x1":
            "x2":
          },
          {
            "x1":
            "x2":
          }
        ]
    }
```

```
[
     {
       "instance": "e"
     },
     {
       "instance": "gain",
       "connector": "u"
     },
     {
       "linePoints": [
         {
           "x1": -120,
           "x2": -40
         },
         {
           "x1": -92,
           "x2": -40
         },
         {
           "x1": -62,
           "x2": -40
         }
       ],
       "lineColor": "{0,0,127}"
     }
   ],
```

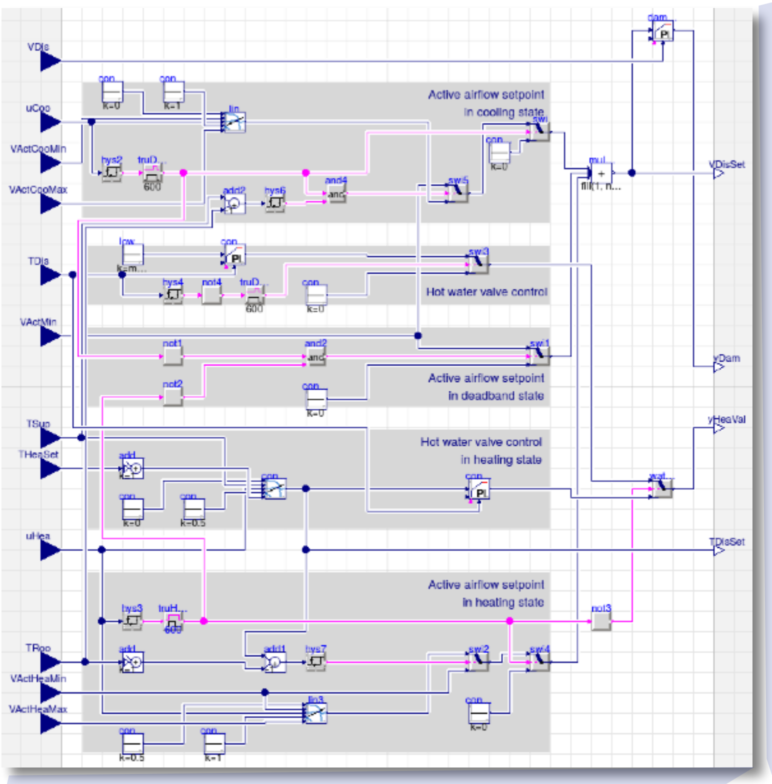
graphical annotation

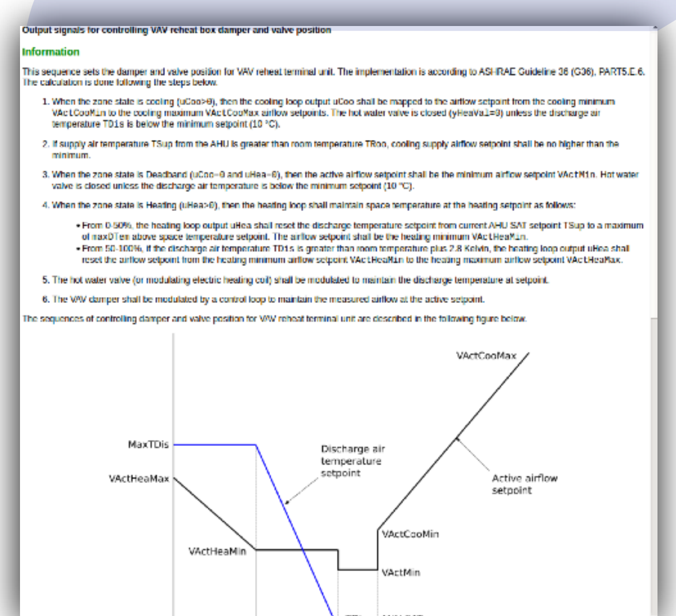


https://github.com/lbl-srg/modelica-json

Example implementation of an ASHRAE Guideline 36 sequence

Block-diagram view.





Output signals for controlling VAV reheat box damper and valve position

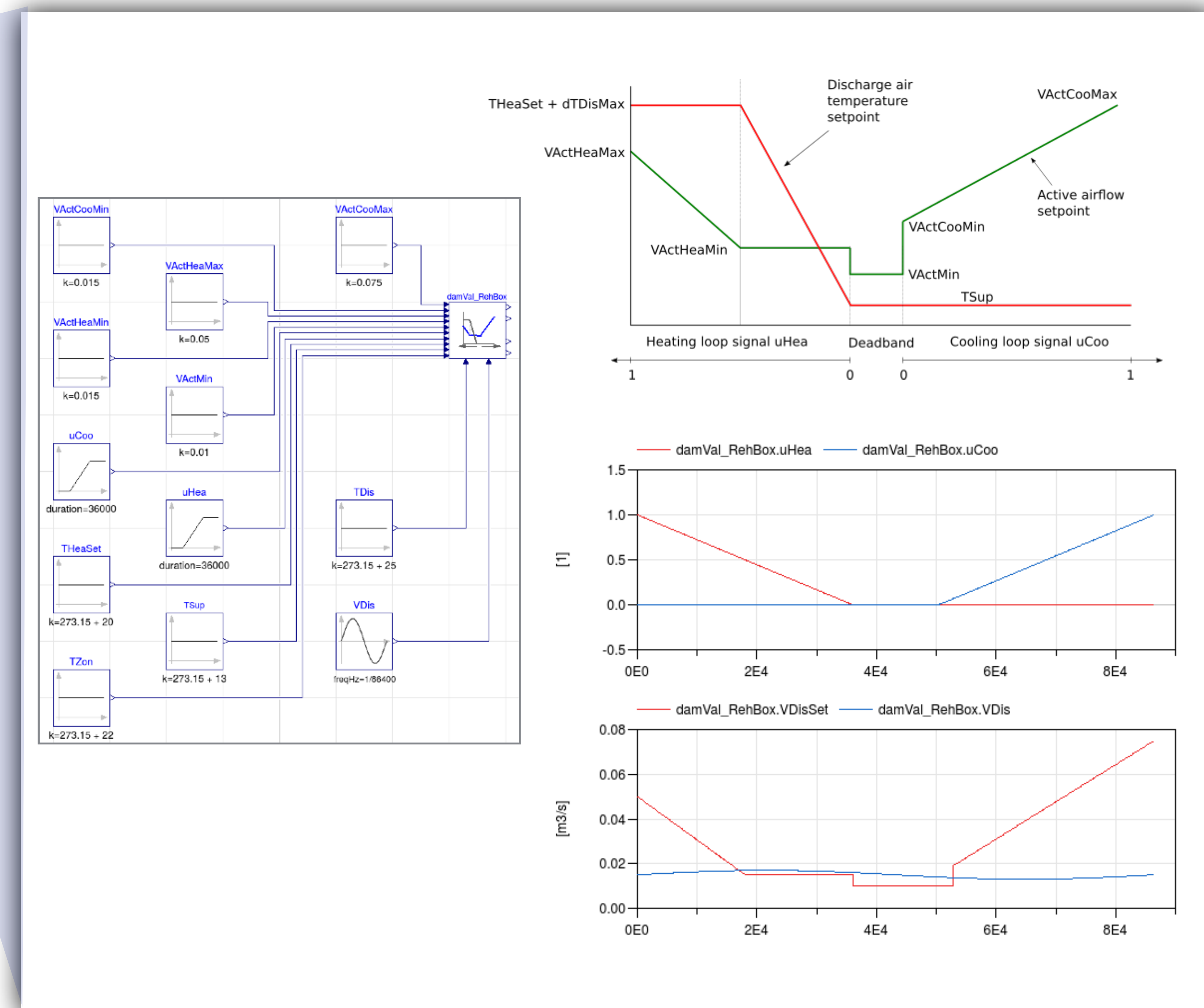
Information

This sequence sets the damper and valve position for VAV reheat terminal unit. The implementation is according to ASHRAE Guideline 36 (G36), PART5.E.6. The calculation is done following the steps below.

1. When the zone state is cooling (uCoo>0), then the cooling loop output uCoo shall be mapped to the airflow setpoint from the cooling minimum VActCooMin to the cooling maximum VActCooMax airflow setpoints. The hot water valve is closed (yHeaVal=0) unless the discharge air temperature TDis is below the minimum setpoint (10 °C).
2. If supply air temperature TSup from the AHU is greater than room temperature TRoo, cooling supply airflow setpoint shall be no higher than the minimum.
3. When the zone state is Deadband (uCoo=0 and uHea=0), then the active airflow setpoint shall be the minimum airflow setpoint VActMin. Hot water valve is closed unless the discharge air temperature is below the minimum setpoint (10 °C).
4. When the zone state is Heating (uHea>0), then the heating loop shall maintain space temperature at the heating setpoint as follows:
   - From 0-50%, the heating loop output uHea shall reset the discharge temperature setpoint from current AHU SAT setpoint TSup to a maximum of maxDTem above space temperature setpoint. The airflow setpoint shall be the heating minimum VActHeaMin.
   - From 50-100%, if the discharge air temperature TDis is greater than room temperature plus 2.8 Kelvin, the heating loop output uHea shall reset the airflow setpoint from the heating minimum airflow setpoint VActHeaMin to the heating maximum airflow setpoint VActHeaMax.
5. The hot water valve (or modulating electric heating coil) shall be modulated to maintain the discharge temperature at setpoint.
6. The VAV damper shall be modulated by a control loop to maintain the measured airflow at the active setpoint.

The sequences of controlling damper and valve position for VAV reheat terminal unit are described in the following figure below.

Autogenerated documentation.



Examples and validation tests.

Buildings.Controls.OBC.ASHRAE.G36_PR1.TerminalUnits.Reheat.DamperValve

Impact: Bridge silos between BEM and controls to realize energy savings of advanced control sequences

Two similar ASHRAE-published VAV sequences yield 30% different HVAC energy use

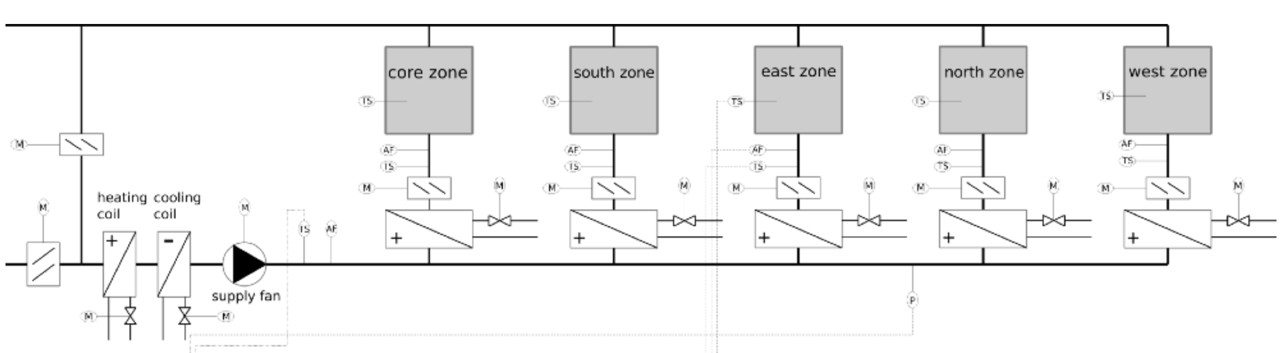
BEM should have tools and use a process that

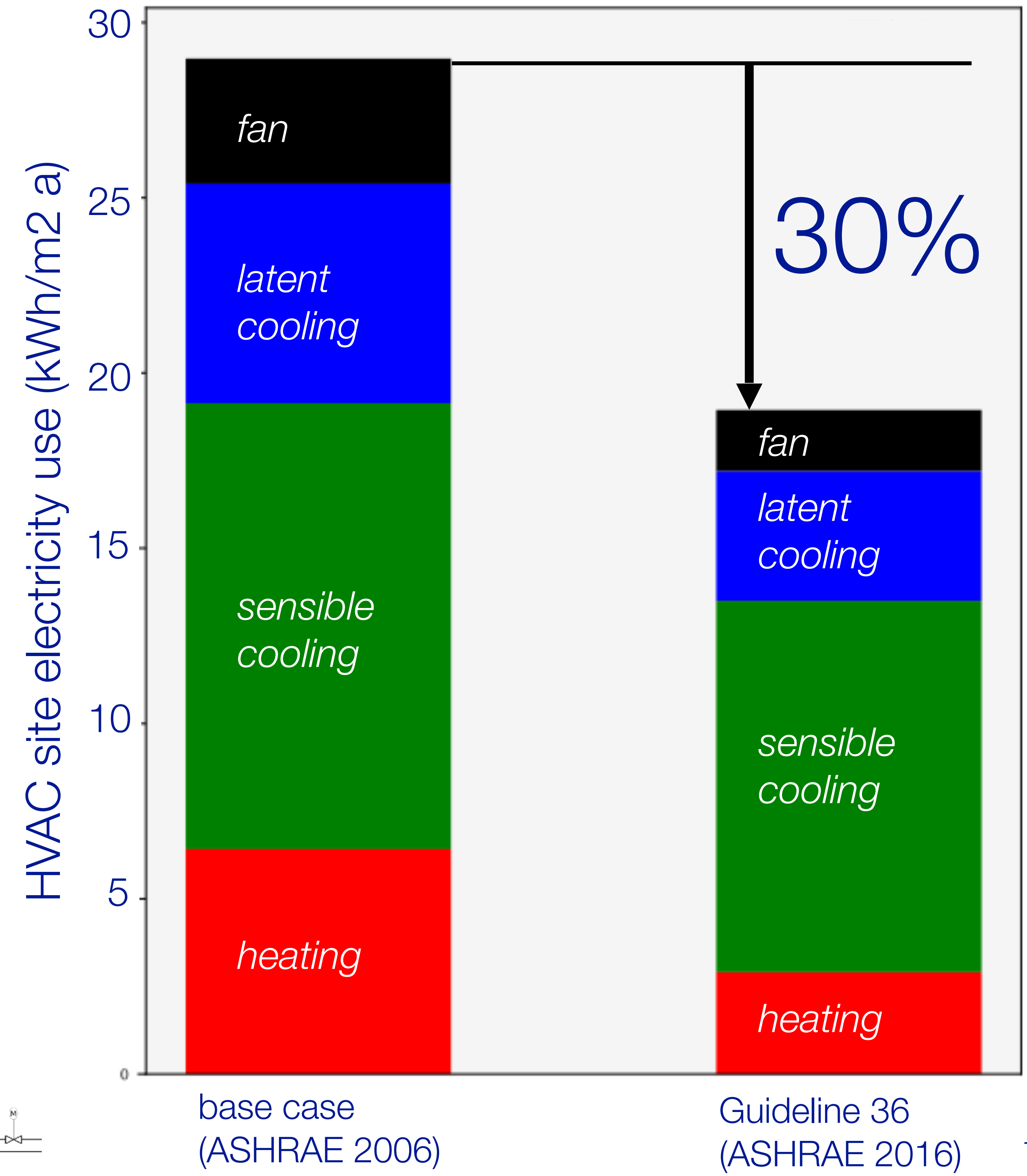
- identifies and closes this 30% performance gap,
- yields better control sequences, and
- ensures that savings are realized

Can you tell which of these VAV sequences your BEM tool uses?

How do you ensure your simulation uses the VAV sequence that will be implemented in the building?

What do you mean if you tell a customer that radiant systems are 20% more efficient than VAV?





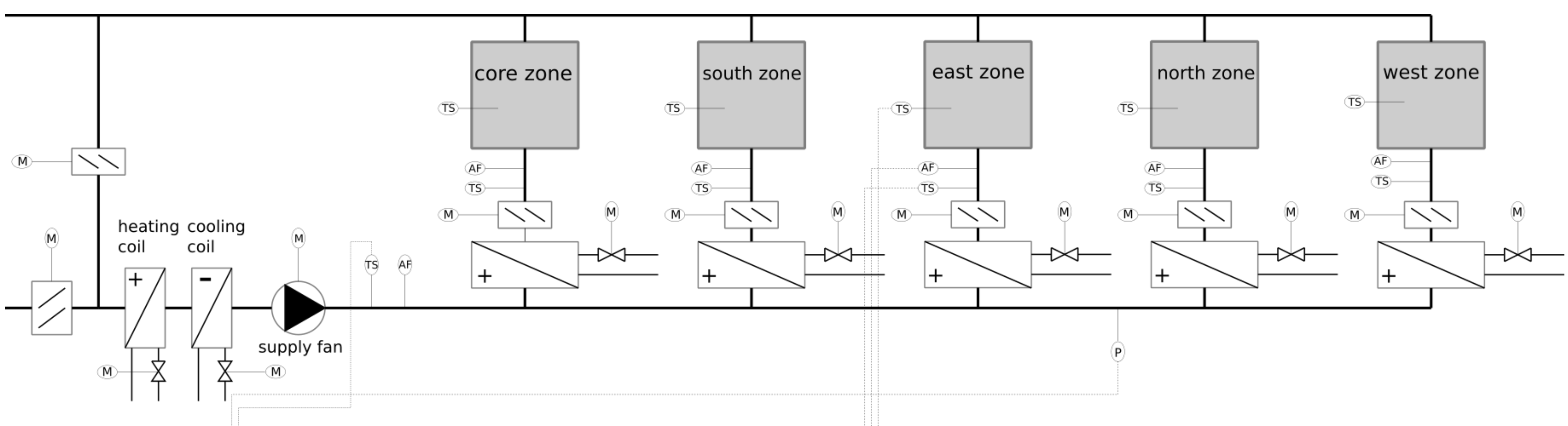
See http://obc.lbl.gov/specification/example.html

Lessons learned regarding simulation

Approach for HVAC & building model:

- Full airflow network.
- Wind pressure driven infiltration.
- All flows based on flow friction, damper positions and fan curves.
- 4,000 components, 40,000 variables (generated from high-level declarations)
- adaptive time step based on error control, state- and time-events.

Simulation using Modelica Buildings Library 5.0.0 and Dymola 2018FD01.



Lessons learned regarding simulation

Oversimplifying physics can lead to problems

Fan model

- Fan with prescribed mass flow rate: Not a good idea, as fan head was 4 kPa (16 inch H20), with 10 K temperature raise across fan because dampers opened slowly.
- Fan with prescribed head: Not a good idea as flow rates in return duct were unrealistic large (due to small flow friction).

Using a fan with speed as input worked fine.

Heating coil

Initial simulations had very small flow rates at night when the fan was off, caused by wind pressure on the building. This caused the heating coil to freeze as the HVAC system entrained -20 degC outside air.

Adding heat diffusion worked fine.

Properly handling hard switches

All switches can chatter due to numerical noise (or, in reality, sensor noise).

Adding hysteresis or a timer worked fine.

Control verification tool “funnel”

Compares time series within certain threshold.
Allows verifying correctness of installed sequence.

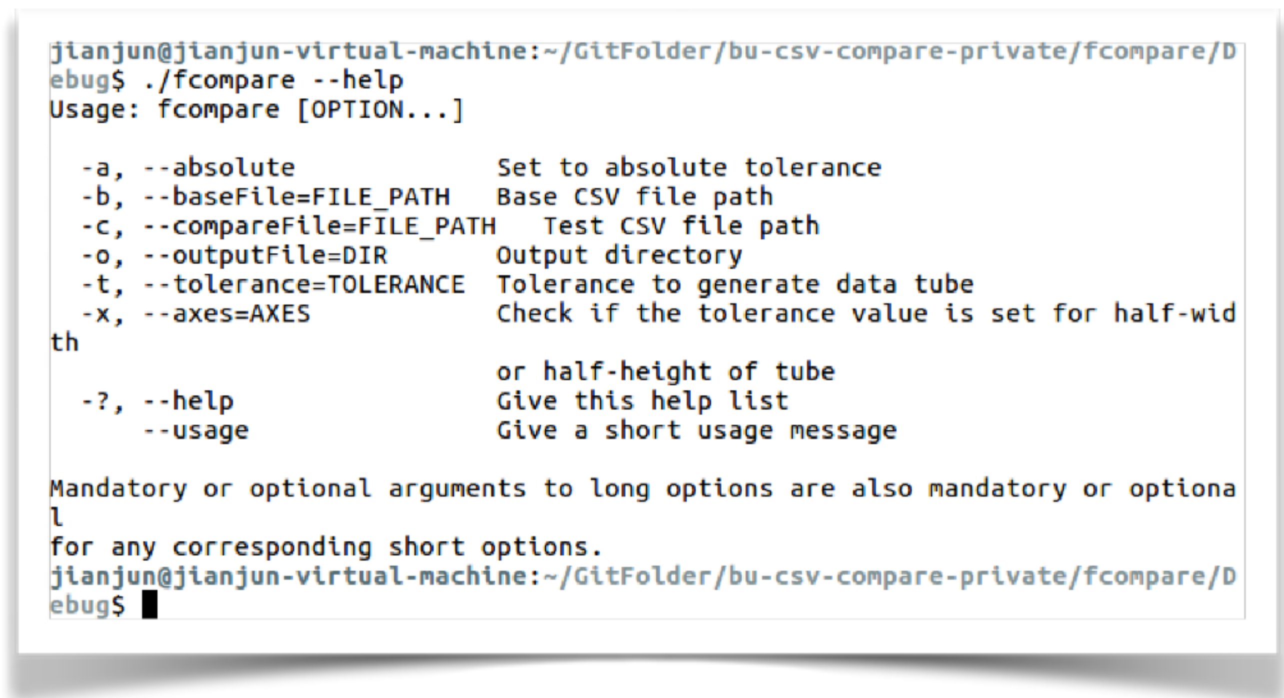
Based on C (Python binding to be added), BSD licensed.

— creates funnel around reference curves, i.e. curves around simulated trajectory

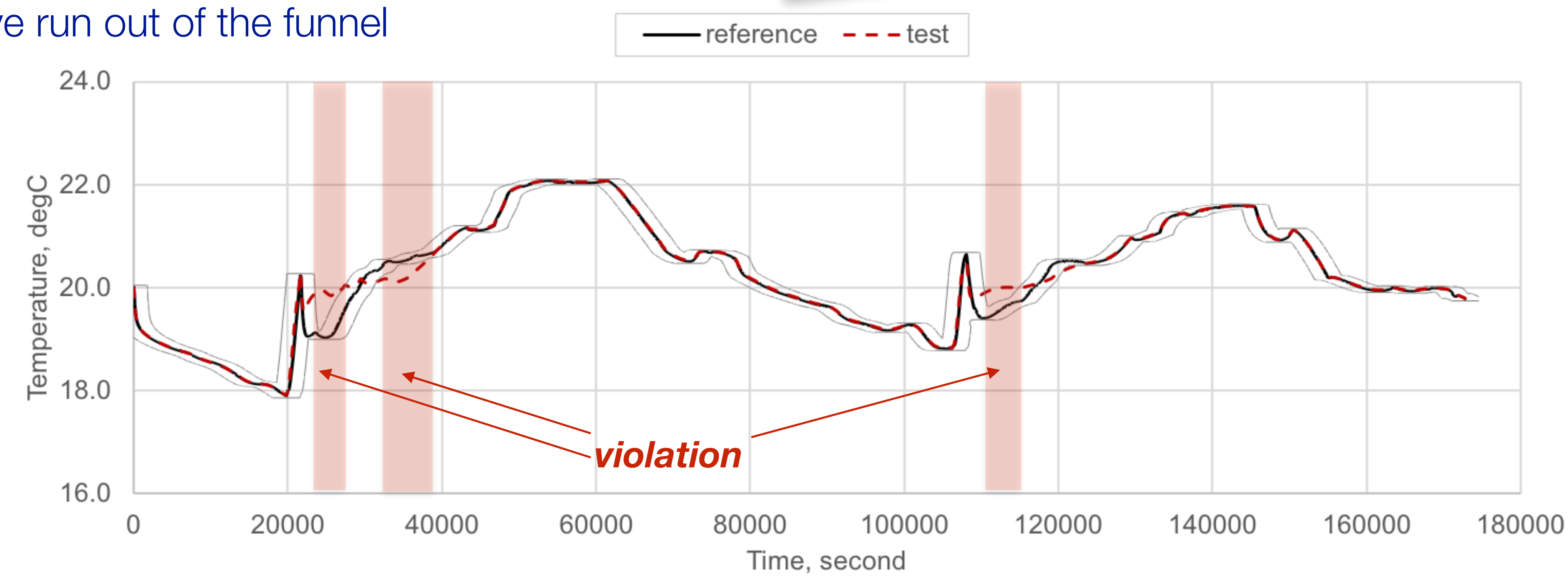
- user giving tolerance for specifying funnel size

— check control simulation outputs to see if the curves across funnel

- identify whether simulation profile is out of funnel, when it happens
- count how long times the simulation output curve run out of the funnel



```
jianjun@jianjun-virtual-machine:~/GitFolder/bu-csv-compare-private/fcompare/D
ebug$ ./fcompare --help
Usage: fcompare [OPTION...]

  -a, --absolute             Set to absolute tolerance
  -b, --baseFile=FILE_PATH   Base CSV file path
  -c, --compareFile=FILE_PATH   Test CSV file path
  -o, --outputFile=DIR       Output directory
  -t, --tolerance=TOLERANCE  Tolerance to generate data tube
  -x, --axes=AXES            Check if the tolerance value is set for half-wid
th
                             or half-height of tube
  -?, --help                 Give this help list
      --usage                Give a short usage message

Mandatory or optional arguments to long options are also mandatory or optiona
l
for any corresponding short options.
jianjun@jianjun-virtual-machine:~/GitFolder/bu-csv-compare-private/fcompare/D
ebug$ 
```



Other usage: CI testing, similar to csv-compare https://github.com/lbl-srg/funnel

Benefits

DOE/BTO:

Potential to reduce HVAC energy by 20% to 30%, solely due to better control sequences

Have tools for dynamic assessment of energy/peak load reduction through integrated systems (HVAC, façade, grid), including path towards hardware-in-the-loop and control deployment

Path towards development & publication of more sophisticate control sequences, such as for energy-aware, grid-flexible buildings

Mechanical designer:

Adapt, test and specify control sequences (and verification tests) for particular building

Reduce risk that building does not meet energy target due to control discrepancies

Control provider:

Faster, higher quality, error-free automated implementation

Get non-ambiguous control specification from designer

Commissioning provider:

Semi-automated verification of compliance with design intent, using formal tests from designer

ASHRAE Committee:

- Guideline 36: Formal way to test, compare and publish sequences in product-neutral way that can be digitally processed and simulated
- Advanced Energy Design Guides: Can include energy-saving sequences in product-neutral way.

Questions