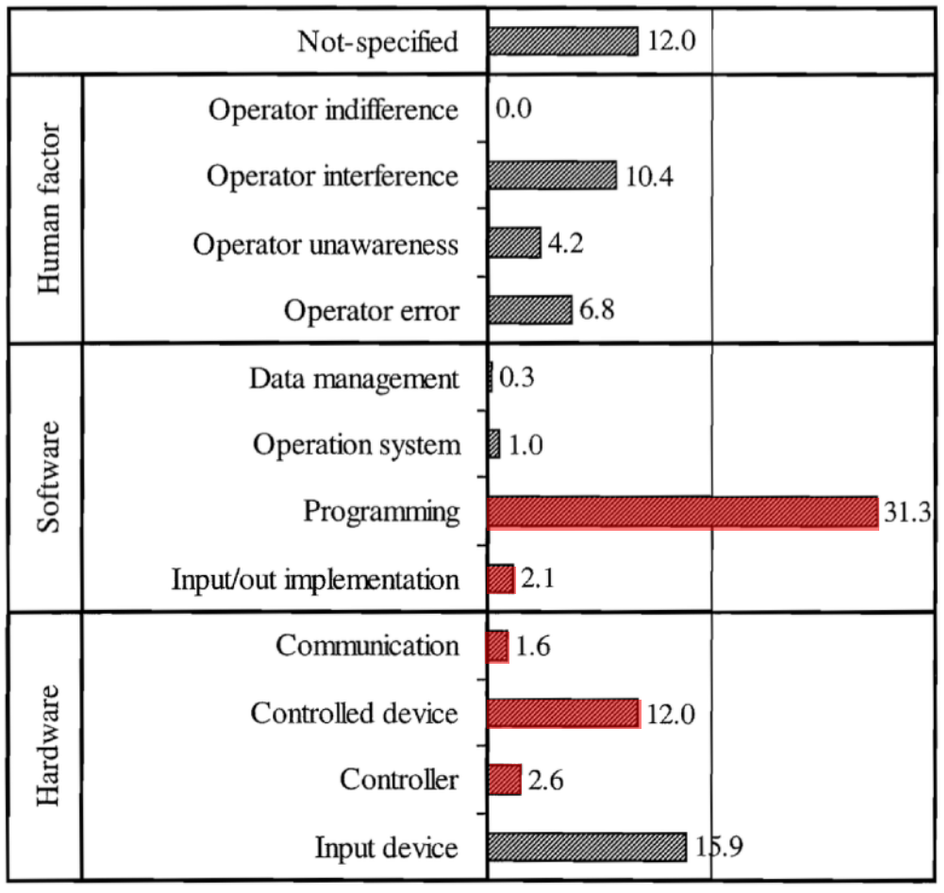
OpenBuildingControl & Control Description Language

Michael Wetter, Antoine Gautier, Milica Grahovac, Philip Haves, Jianjun Hu, Lisa Rivalin, Kun Zhang

April 4, 2019

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.

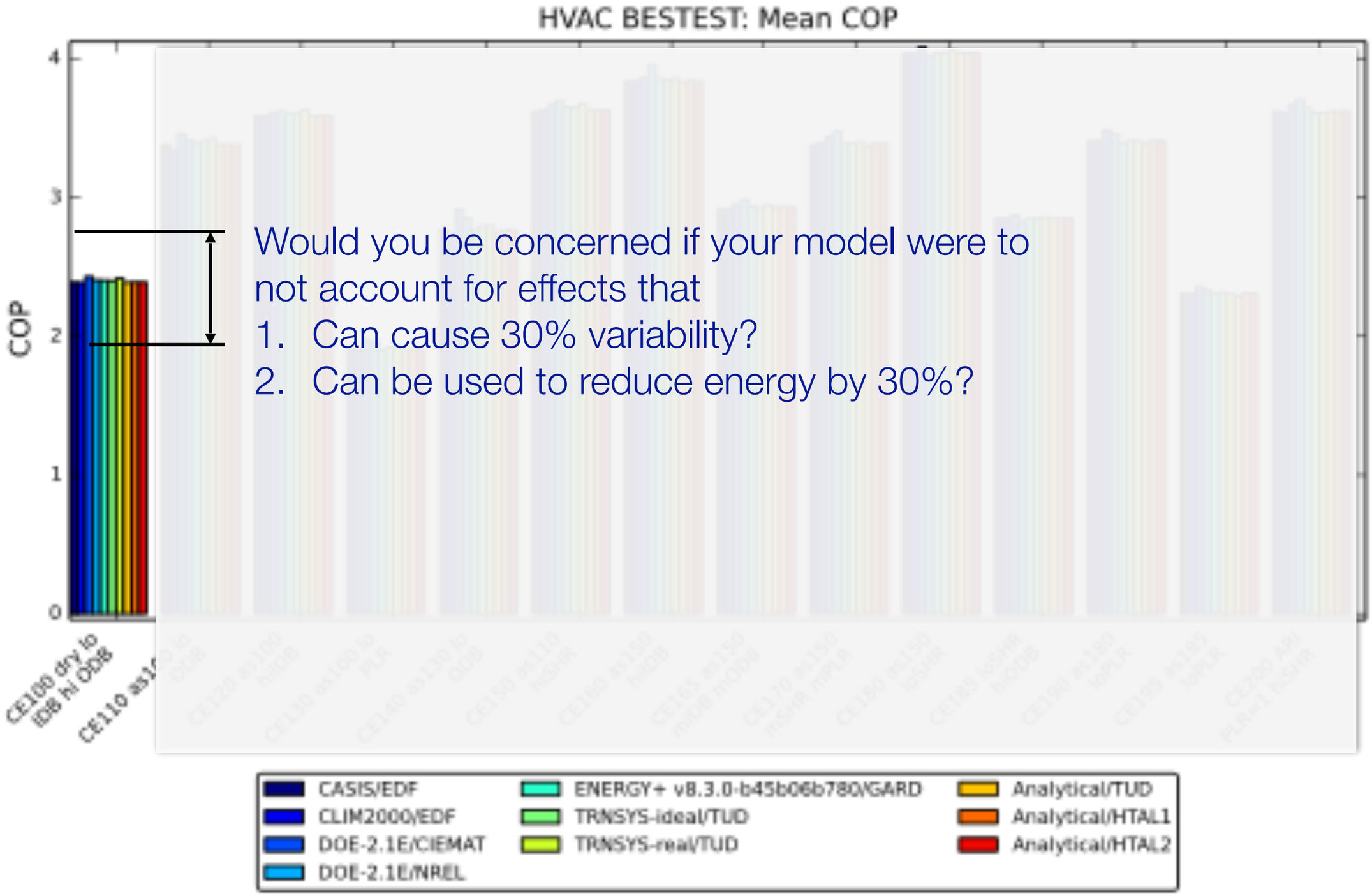
More than 1 quad/yr of energy is wasted in the US because control sequences are poorly specified and implemented in commercial buildings.

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

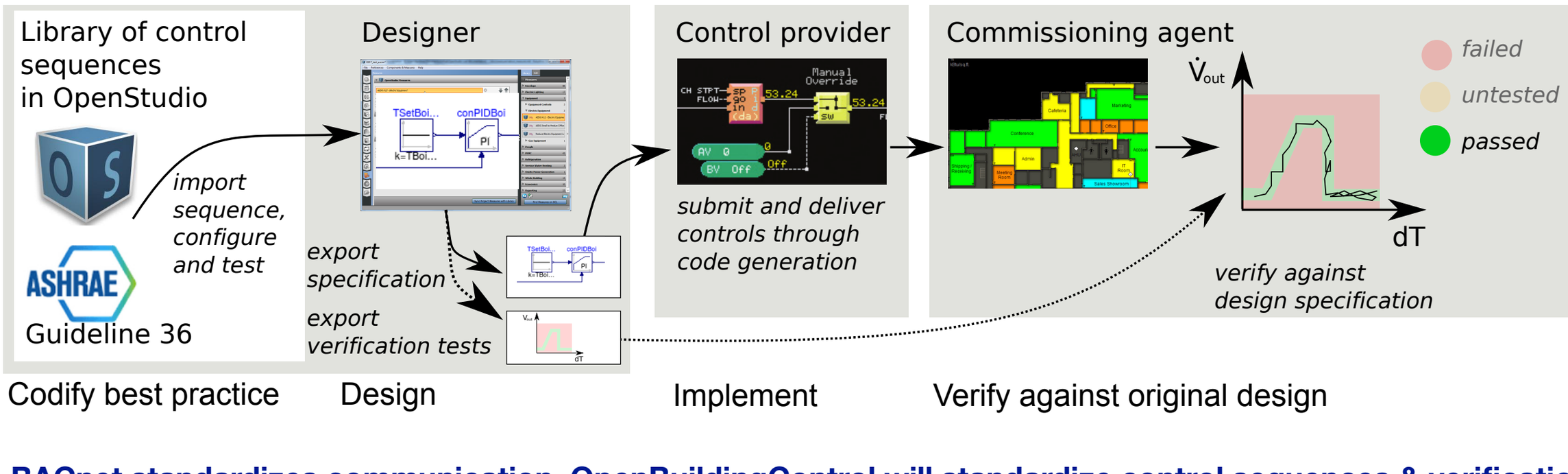
1. mechanical designers can import in building energy modeling tools best-in-class control sequences from ASHRAE-vetted guidelines?

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

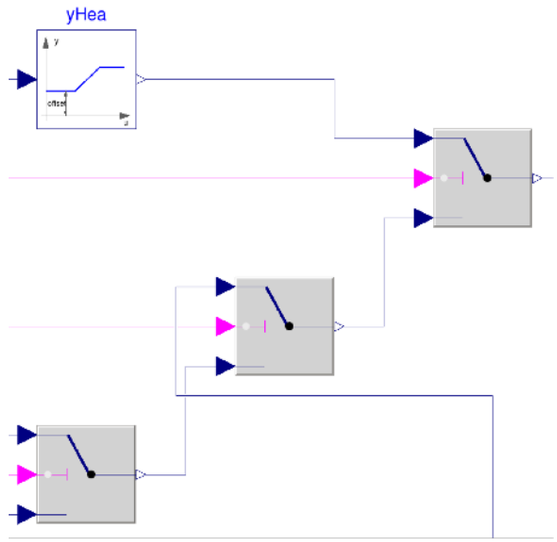
Digital, executable control specification, called Control Description Language (CDL), enabling

- 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

What is the Control Description Language?

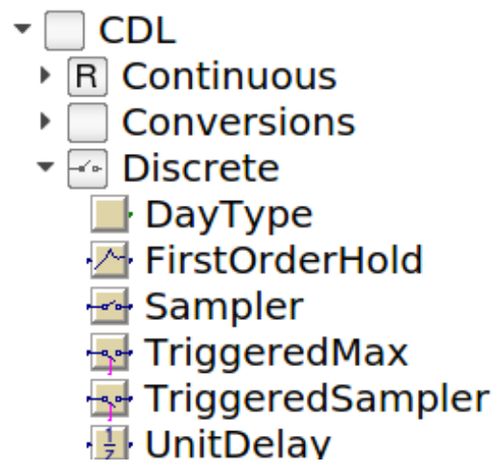
A declarative language for expressing block-diagrams for controls (and requirements)

A graphical language for rendering these diagrams.

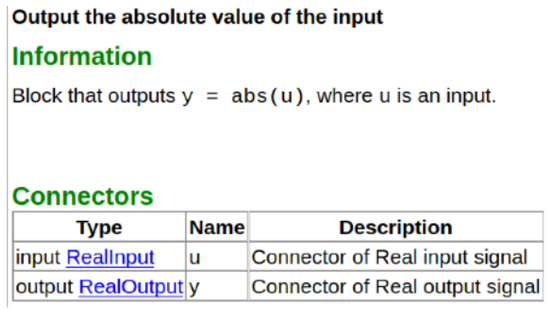


A library with elementary input/output blocks that should be supported [through a translator] by CDL-compliant control providers

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

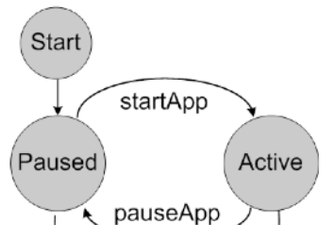
Information

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.



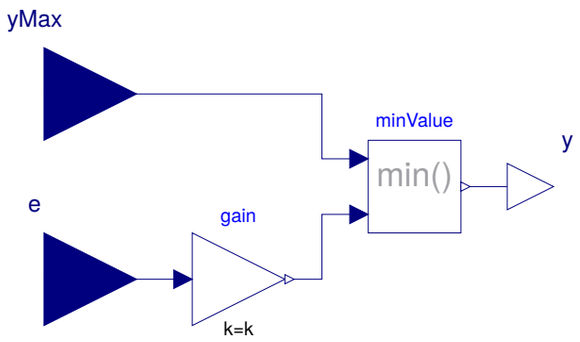
What is the Control Description Language?

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**
- **ASHRAE**
- **G36**

  Sequences that come out of ASHRAE projects and can be shared with community.

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

- **GSA**

  GSA preferred sequences, made available through a CDL-complaint implementation.

- **ARUP**

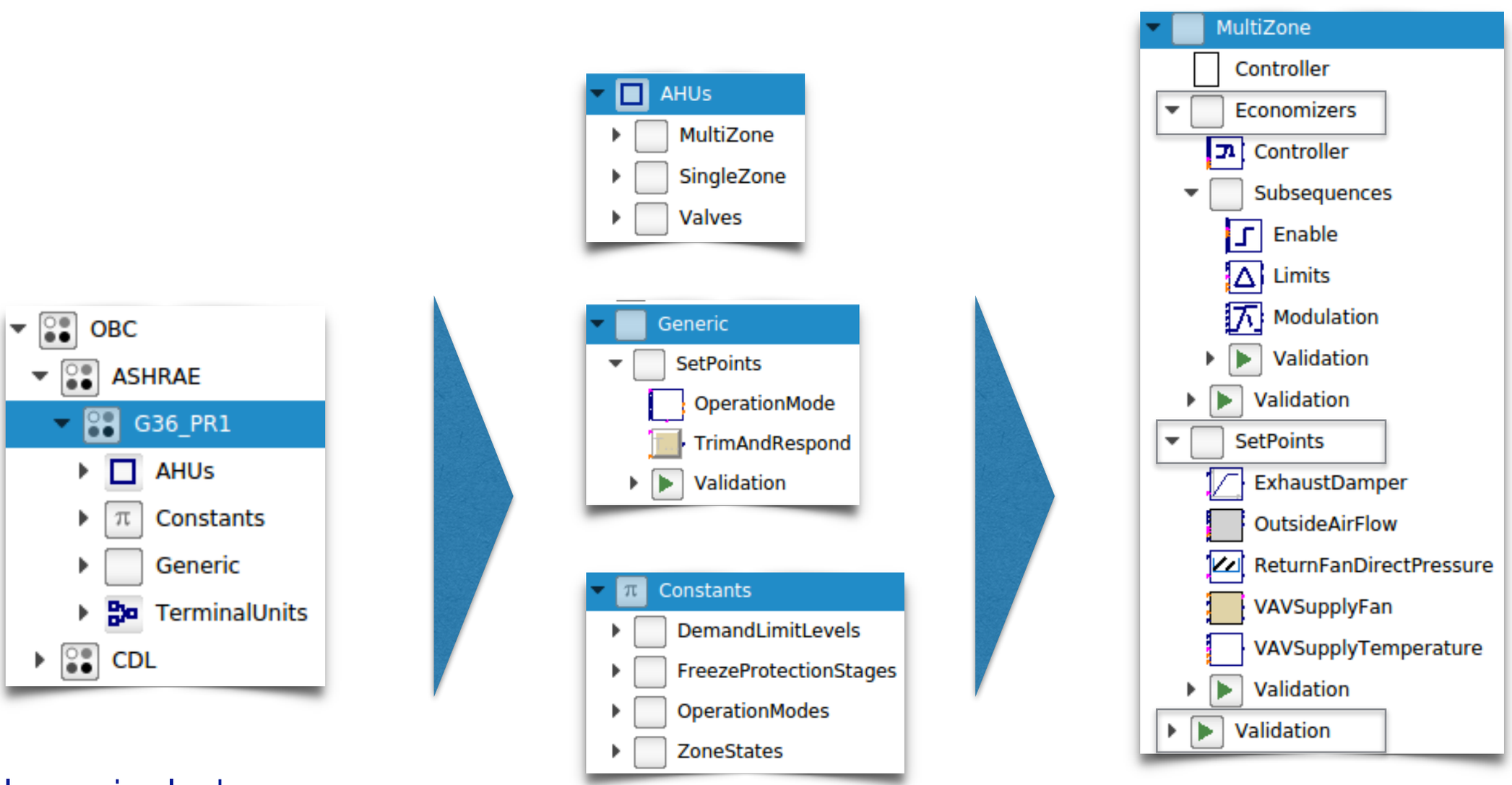
  Design firms can share their own (proprietary) implementation across their offices.

- **ALC**

  Control vendors can provide their own specialized sequences, either as open-source, or as compiled (proprietary) I/O blocks.

Companies are illustrative

Released Guideline 36 sequences with Buildings library 5.0.0

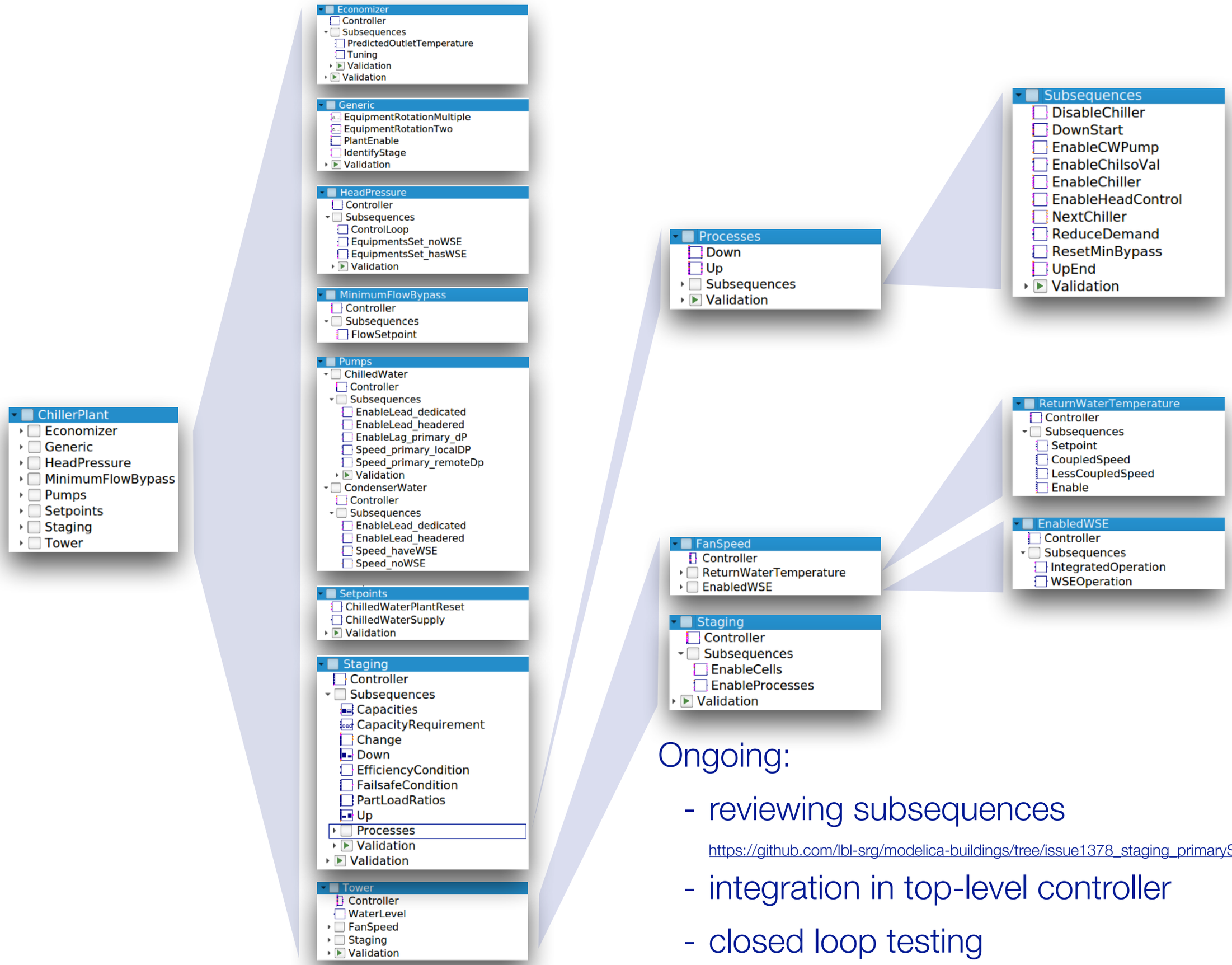


Release includes

- VAV models
- 34 test cases/examples

http://simulationresearch.lbl.gov/modelica/

Primary sequence implementation



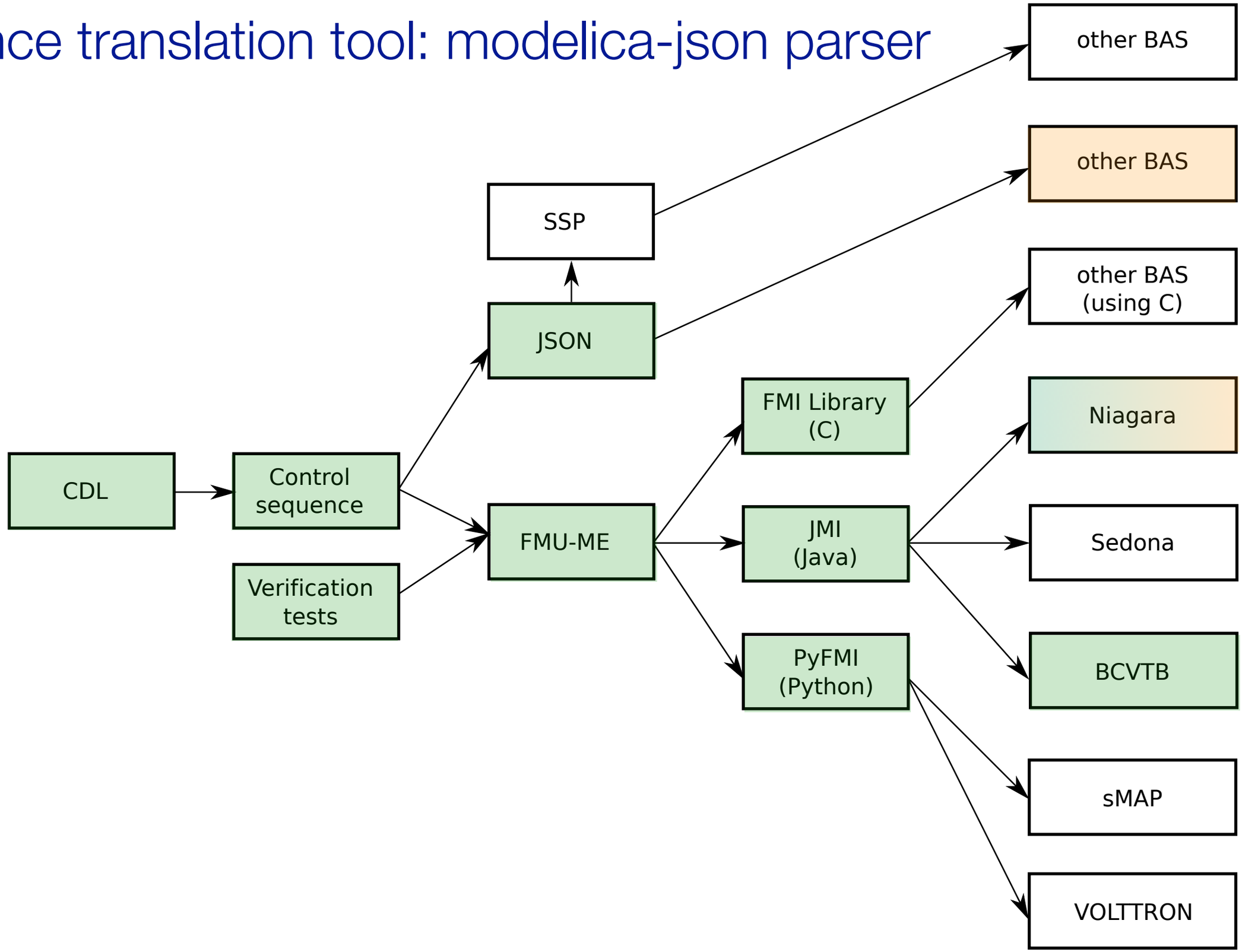
Ongoing:

- reviewing subsequences

  https://github.com/lbl-srg/modelica-buildings/tree/issue1378_staging_primarySequences

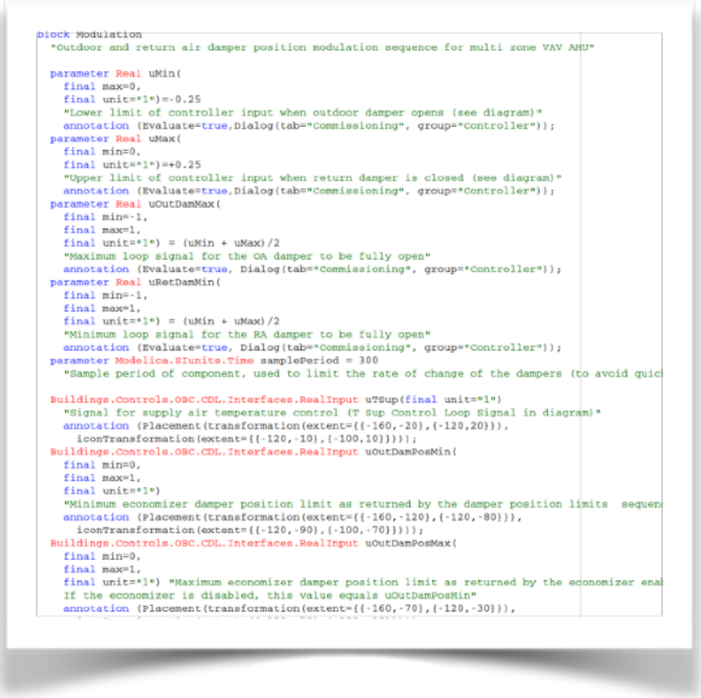
- integration in top-level controller
- closed loop testing

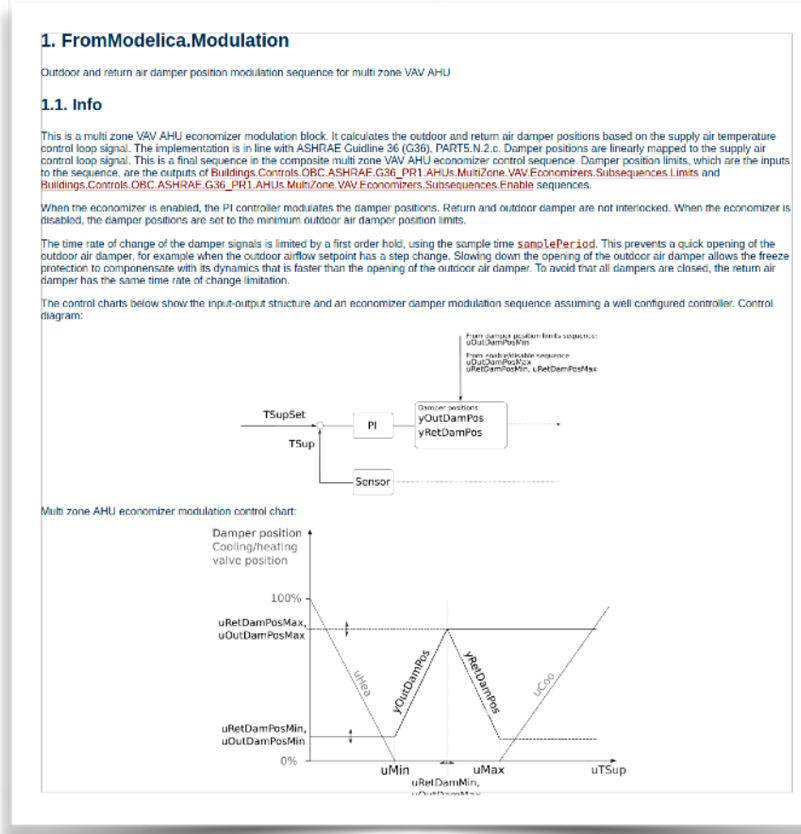
Sequence translation tool: modelica-json parser

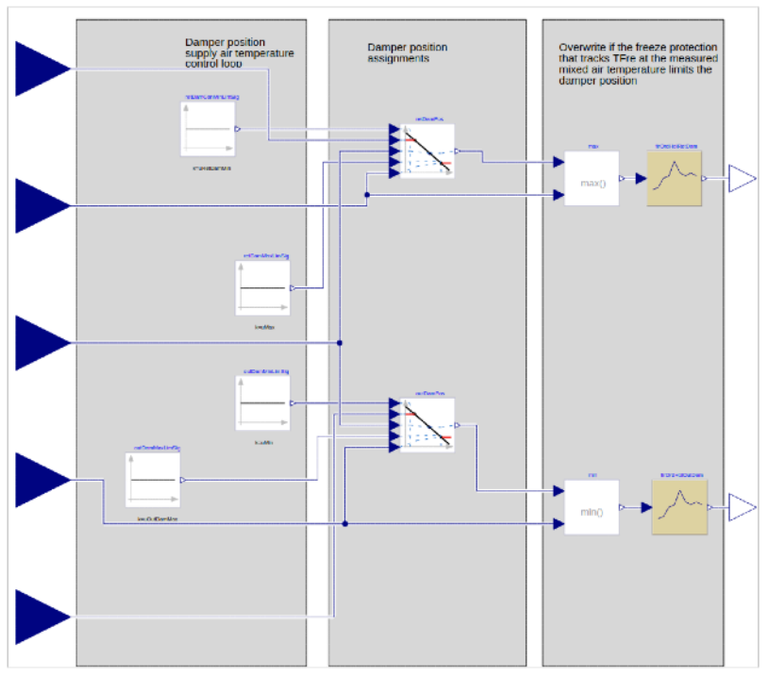


http://obc.lbl.gov/specification/codeGeneration.html#use-of-control-sequences-or-verification-tests-in-realtime-applications

Sequence translation tool: modelica-json parser







Parse modelica models and/or control sequences

Output formats:

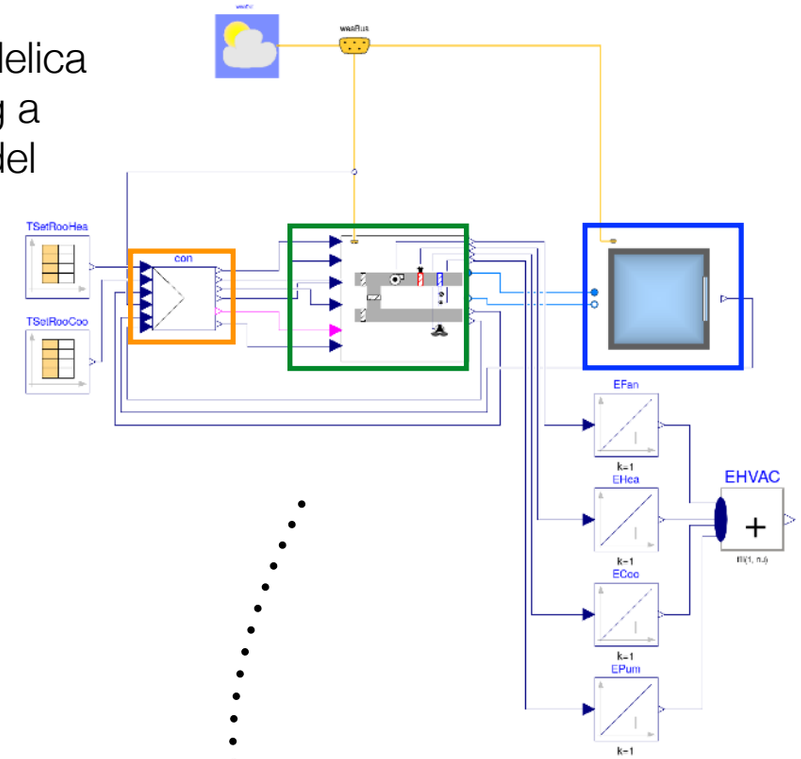
- json intermediate format
- validate format by JSON schema
- svg for graphics
- html & svg for documentation
- 2019/20: prototype translator to a commercial control product line
- 2020+: Basis for HVAC & control design tool

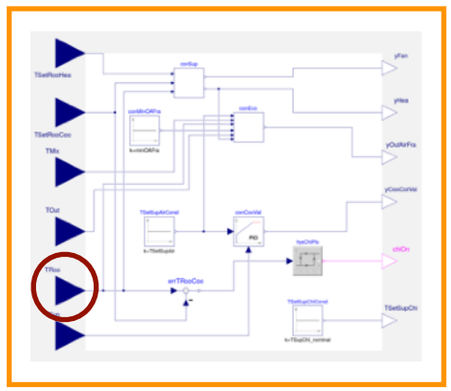
https://github.com/lbl-srg/modelica-json

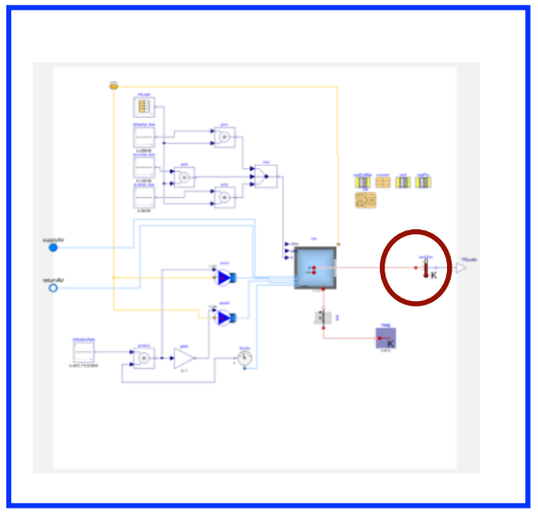
Modelica to BRICK

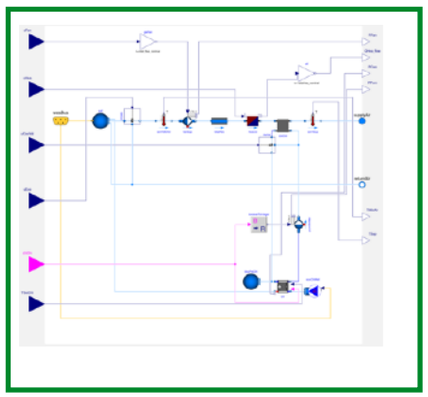
Context : From design to operation

Design stage
Construction of the Modelica model by instantiating a template system model



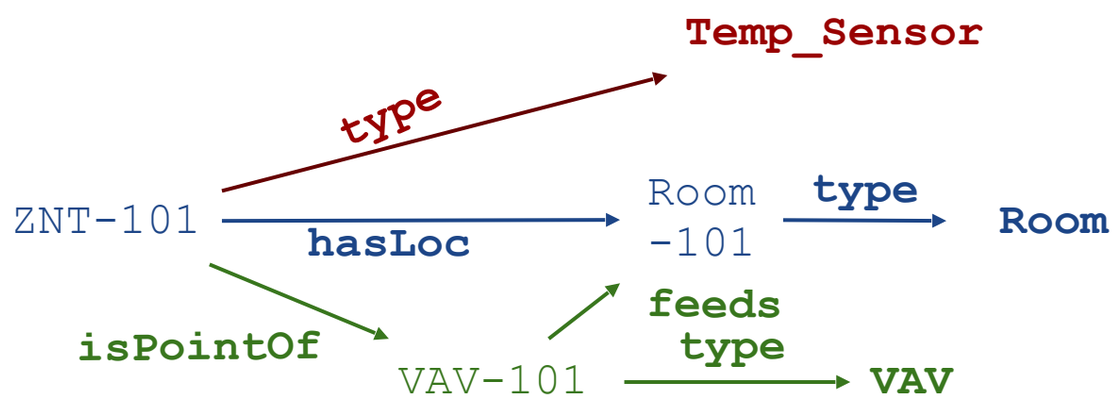






Execution phase
Export the data from the Modelica model to a Json-file and create a BRICK model.

Operation
Use the BRICK model to retrieve the data and conduct analyses, optional: combine the BRICK model with an FMU of mechanical system and/or controls



Control verification tool: `funnel`

A cross-platform C-based software for comparing two (x, y) data sets given tolerances along x and y directions

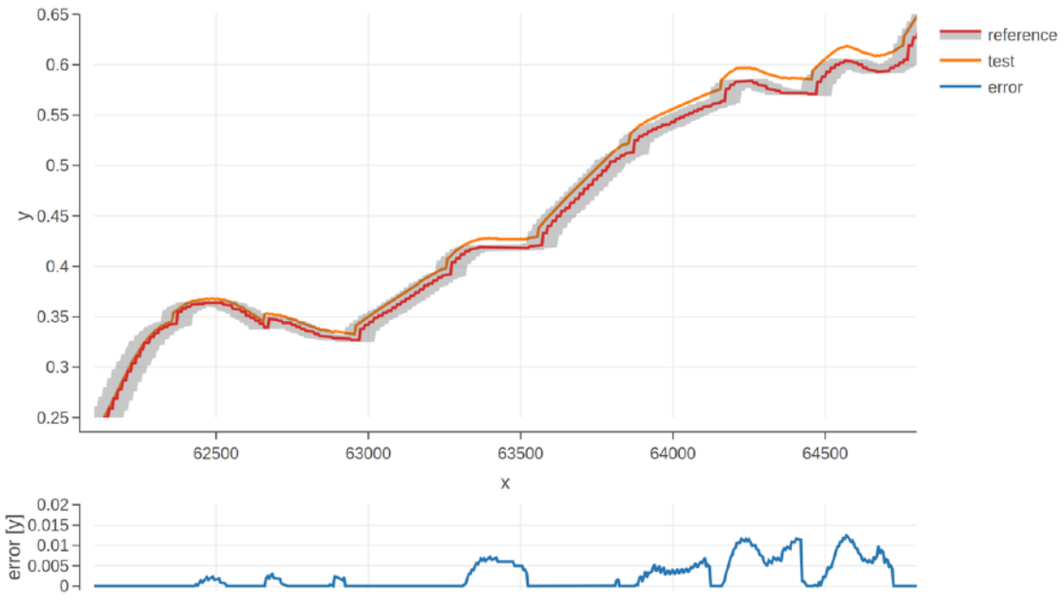
- Validation of control sequences by comparing time series from real operation vs simulation
- Main principles and features:
  - Available as a Python module with HTML interactive plot for enhanced error analysis
  - To be released: multiple plots and HTML summary report

(https://github.com/lbl-srg/funnel)

Simulation | Translation | Comparison

Show 10 entries | Search:

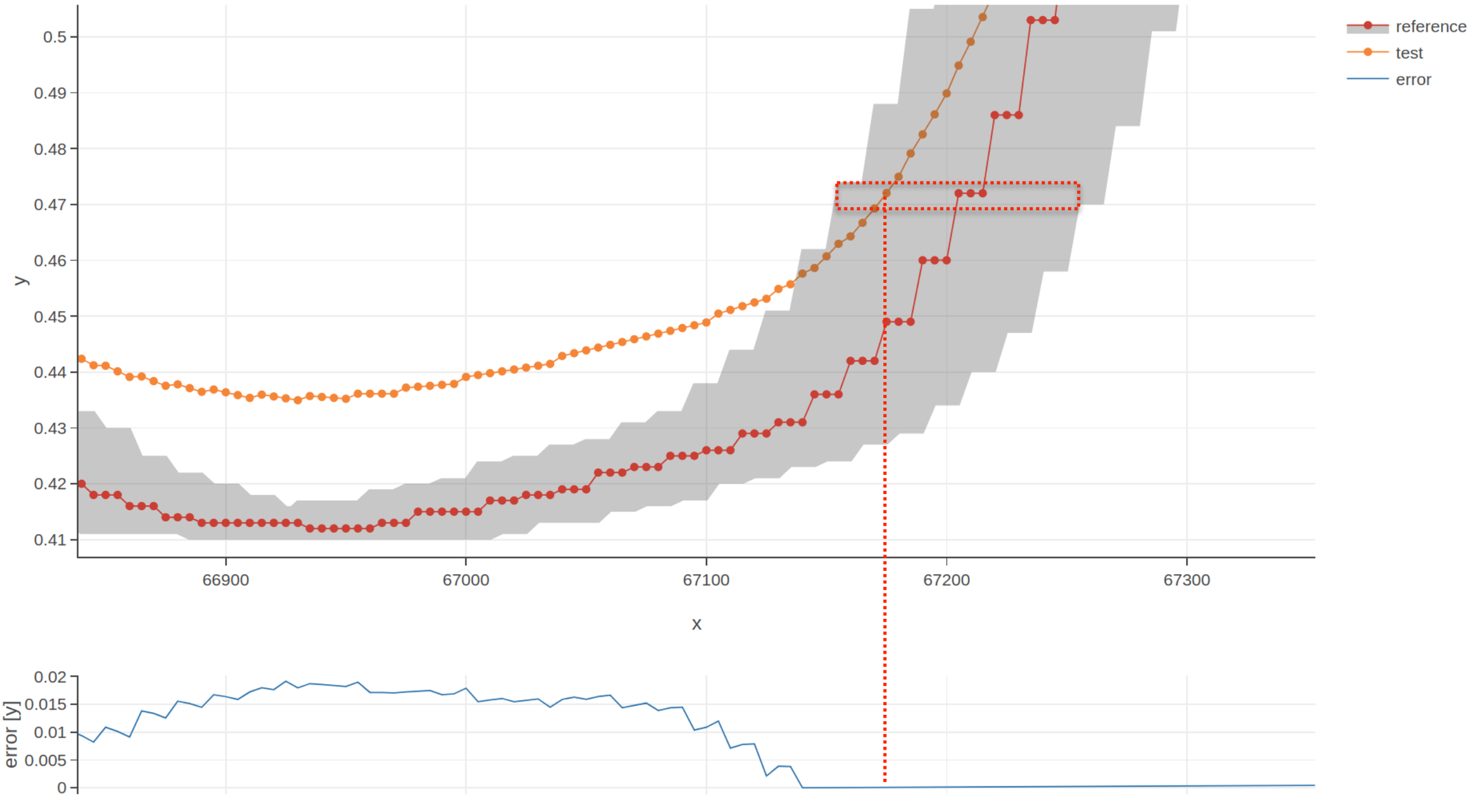
Model	Variables	Success
Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.MultiZone.VAV.Economizers.Subsequences.Validation.Enable_FreProSta	enaDis.yRetDamPosMax, enaDis.yOutDamPosMax, enaDis.yRetDamPosMin, enaDis.truFalHol.y, enaDis.andEnaDis.y, freProSta1.y	100%
Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.MultiZone.VAV.Economizers.Subsequences.Validation.Enable_TOut_hOut	TOutCut.y, TOut.y, enaDis.truFalHol.y, enaDis.yRetDamPosMax, enaDis.yRetDamPosMin, enaDis.yOutDamPosMax, hOutCut1.y, hOut.y, enaDis1.truFalHol.y, enaDis1.yRetDamPosMin, enaDis1.yRetDamPosMax, enaDis1.yOutDamPosMax, TOutCut.y, TOut.y, enaDis2.truFalHol.y, enaDis2.yRetDamPosMax, enaDis2.yRetDamPosMin, enaDis2.yOutDamPosMax	100%
Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.MultiZone.VAV.Economizers.Subsequences.Validation.Limits_LoopDisable	fanSta.y, VOut_flow.y, VOutMinSet_flow.y, damLim.yOutDamPosMin, damLim.yOutDamPosMax, damLim.yRetDamPosMax, damLim.yRetDamPosMin, damLim.damLimCon.y, opeMod1.y, VOut1_flow.y, VOutMinSet1_flow.y, damLim1.yOutDamPosMax, damLim1.yOutDamPosMin, damLim1.yRetDamPosMax, damLim1.yRetDamPosMin, damLim1.damLimCon.y, freProSta2.y, VOut2_flow.y, VOutMinSet2_flow.y, damLim2.yOutDamPosMin, damLim2.yOutDamPosMax, damLim2.yRetDamPosMin, damLim2.yRetDamPosMax, damLim2.damLimCon.y	100%
Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.MultiZone.VAV.Economizers.Subsequences.Validation.Limits_VOut_flow	VOut_flow.y, VOutMinSet_flow.y, damLim.yOutDamPosMin, damLim.yOutDamPosMax, damLim.yRetDamPosMax, damLim.yRetDamPosMin, damLim.damLimCon.y	100%
Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.MultiZone.VAV.Economizers.Subsequences.Validation.Modulation_TSup	mod.uMin, mod.uTSup, mod.uMax, mod.yOutDamPos, mod.yRetDamPos, modFre.uTSup, modFre.uMax, modFre.uMin, modFre.yRetDamPos, modFre.yOutDamPos	100%
Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.MultiZone.VAV.Economizers.Validation.Controller_Disable	economizer.VOutMinSet_flow_normalized, economizer.VOut_flow_normalized, economizer.enaDis.andEnaDis.y, economizer.uFreProSta, economizer.damLim.yRetDamPosMax, economizer.damLim.yOutDamPosMin, economizer.yOutDamPos, economizer.yRetDamPos, economizer1.VOut_flow_normalized, economizer1.VOutMinSet_flow_normalized, economizer1.uFreProSta, economizer1.enaDis.andEnaDis.y, economizer1.damLim.yOutDamPosMin, economizer1.damLim.yRetDamPosMax, economizer1.yRetDamPos, economizer1.yOutDamPos, economizer2.freProTMix.TFreSet, economizer2.TMix, economizer2.yOutDamPos, economizer2.yRetDamPos, economizer2.VOutMinSet_flow_normalized, economizer2.VOut_flow_normalized, economizer2.enaDis.andEnaDis.y, economizer2.damLim.yOutDamPosMin, economizer2.damLim.yRetDamPosMax	100%
Buildings.Controls.OBC.ASHRAE.G36_PR1.AHUs.MultiZone.VAV.Economizers.Validation.Controller_Mod_DamLim	economizer.enaDis.andEnaDis.y, economizer.VOutMinSet_flow_normalized, economizer.VOut_flow_normalized, economizer1.uTSup, economizer.yOutDamPos, economizer.damLim.yOutDamPosMin, economizer.damLim.yOutDamPosMax, economizer.damLim.yRetDamPhyPosMax, economizer.damLim.yRetDamPosMin, economizer.damLim.yRetDamPosMax, economizer.yRetDamPos, economizer1.enaDis.andEnaDis.y, economizer1.VOut_flow_normalized, economizer1.VOutMinSet_flow_normalized, economizer1.uTSup, economizer1.damLim.yOutDamPosMin, economizer1.yOutDamPos,	100%



Control verification tool: `funnel`

Detailed principles

- L1-norm based comparison
- Trajectory comparison (as opposed to point-to-point): handles time events & different time scales



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

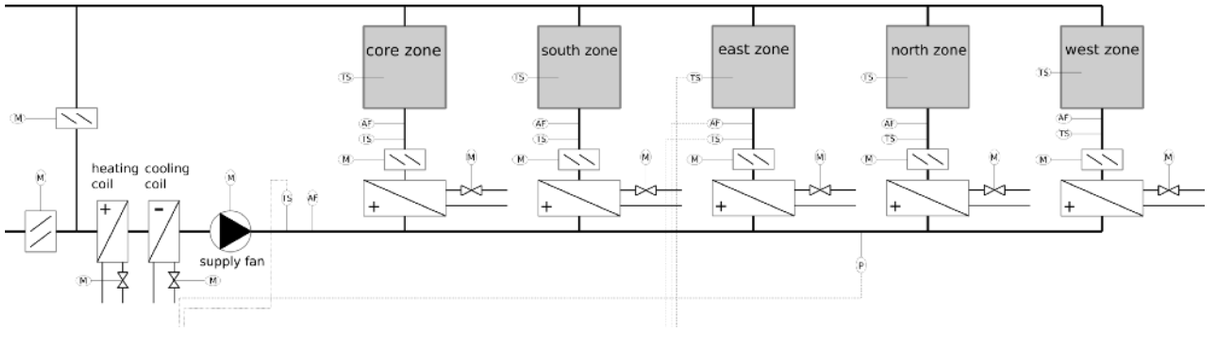
BEM should have tools and use a process that

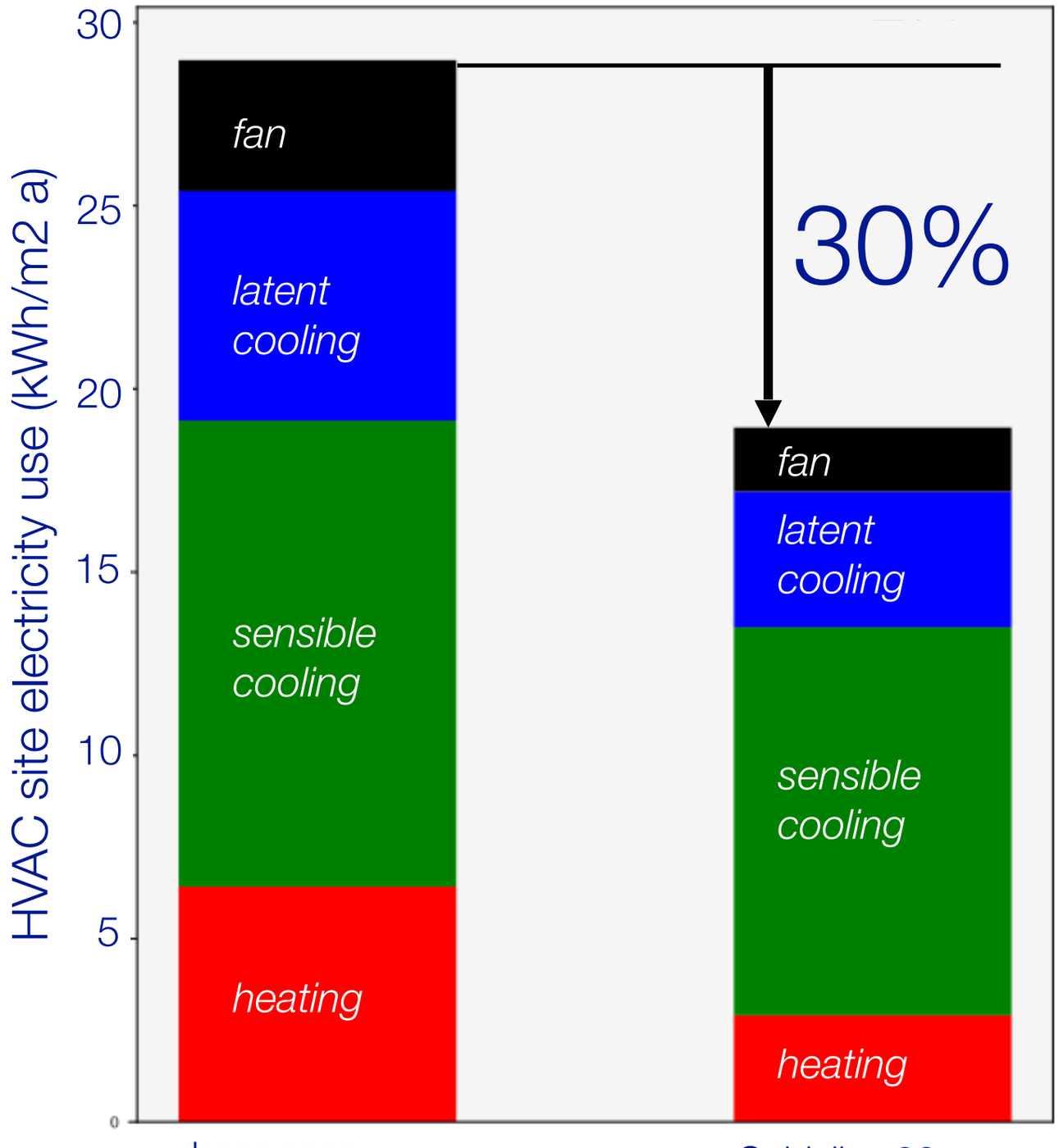
- identifies and closes this 30% performance gap,
- yields better control sequences, and
- ensures that savings are realized

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

Questions