Workshop

Introduction to the Modelica Buildings Library

David Blum, Michael Wetter, and Ettore Zanetti

October 12, 2026



Lawrence Berkeley National Laboratory

Workshop

Introduction to the Modelica Buildings Library

1. Overview of Modelica Buildings Library (~15 min)
2. Best Practices and Modeling Hints (~15 min)
3. Exercise Part I: Modeling a Simple Thermofluid System (~20 min)
4. Exercise Part II: Space Cooling Part 1 (~25 min)
5. Break (~10 min)
6. Exercise Part II: Space Cooling Parts 2-4 (~1 hr 30 min)
7. Wrap-Up (~5 min)

Overview of Modelica Buildings Library

Primary Use of Modelica Buildings Library

- Model repository for building and district energy simulation.
- Home page: https://simulationresearch.lbl.gov/modelica/
- Latest Version: 13.0.0

Example Applications

- Model-based design process.
  - Spawn of EnergyPlus
    see https://lbl-srg.github.io/soep/.
  - Development of 5th generation district heating and cooling and URBANopt,
    see https://www.nrel.gov/buildings/urbanopt.html.
- Repository of control sequences in the Control Description Language (CDL).
  - OpenBuildingControl, see https://obc.lbl.gov/.
- Controls design and performance evaluation.
  - Building Optimization Testing Framework (BOPTEST)
    see https://github.com/ibpsa/project1-boptest
- Development and testing of FDD algorithms.

License

- All development is open-source under BSD.

Primary Use of Modelica Buildings Library

For Spawn of EnergyPlus

- modular models for controls and HVAC

For building designers and manufacturers

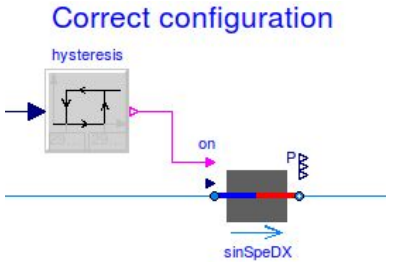
- open-source, free library of component and system models
- collection of case studies and demonstrations

For researchers and manufacturers

- library and tools for rapid virtual prototyping and model-based design

For simulation tool developers

- collaborative environment
- software components with liberal open-source license, vetted by experts from around the world

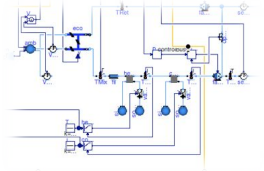


User guide with best practice.

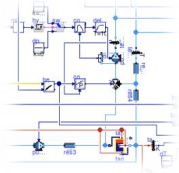
Model Repository

Open-source repository of 3000+ models and functions.

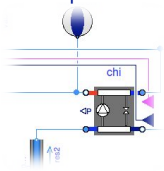
Air-based HVAC



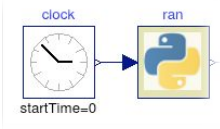
Hydronic heating



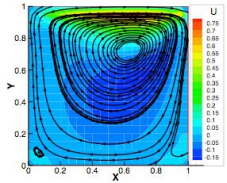
Chiller plants



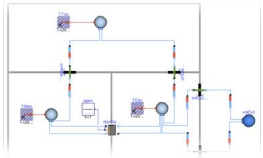
Embedded Python



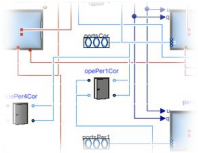
Room air flow



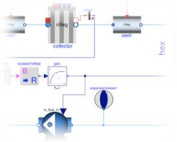
Natural ventilation,
multizone air exchange,
contaminant transport



Room heat transfer,
incl. window (TARCOG)



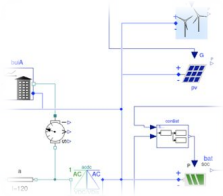
Solar collectors



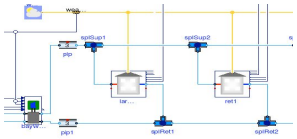
FLEXLAB



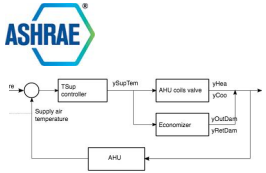
Electrical systems



District heating
and cooling systems



Control design & deployment,
including ASHRAE G36



Co-develop with **IBPSA Modelica Library** as core, including district heating and cooling systems

IBPSA Modelica Working Group
https://ibpsa.github.io/modelica-working-group/

Michael Wetter, Wangda Zuo, Thierry S. Nouidui and Xiufeng Pang. Modelica Buildings library. Journal of Building Performance Simulation, 7(4):253-270, 2014.

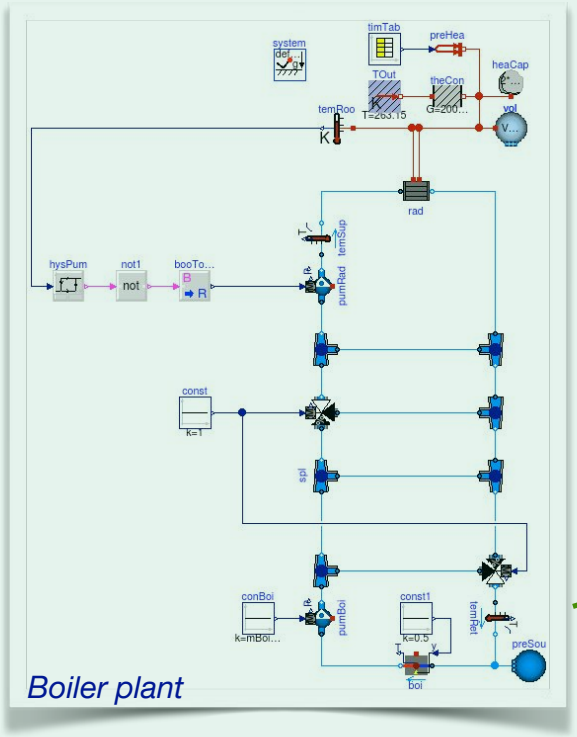
Model Repository

Package Content

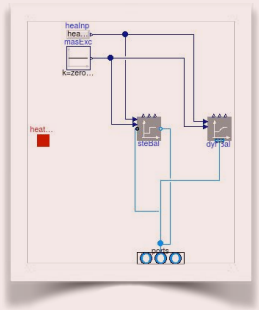
Name	Description
UsersGuide	User's Guide
Air	Package with models that are configured to use with air-based HVAC systems
Airflow	Package to compute airflow and contaminant transport between rooms
Applications	Package with models for different application domains
BoundaryConditions	Package with models for boundary conditions
Controls	Package with models for controls
DHC	Models for district heating and cooling systems
Electrical	Package with models for electrical systems
Fluid	Package with models for fluid flow systems
HeatTransfer	Package with heat transfer models
Media	Package with medium models
Occupants	Package with models to simulate building occupant behaviors
Templates	Package with templates for various HVAC systems
ThermalZones	Models for BuildingPhysics
Utilities	Package with utility functions such as for I/O
Types	Package with type definitions
Examples	Collection of models that illustrate model use and test models
BaseClasses	Package with base classes for the Buildings library
Obsolete	Classes that are obsolete and will be removed in later versions

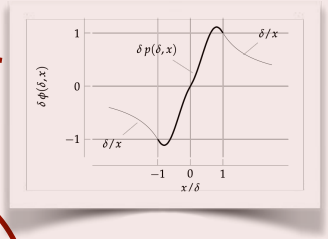
Usage

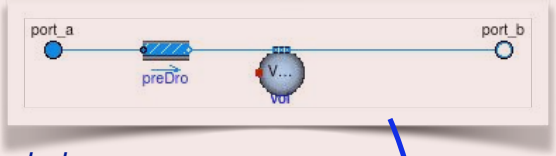
Separation between library developer, component developer, and end user



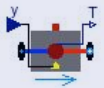
Boiler plant



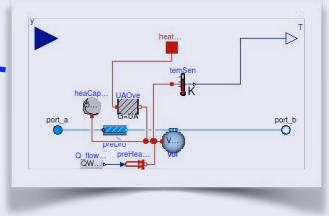




Mass and energy balance



Boiler



Legend:

- **Library developer:** Base class implementations (e.g. stable mass and energy balance)
- **Component developer:** Use base classes to develop component models (e.g. equipment)
- **End user:** Use component models to develop system model (e.g. hydronic heating system)

Main Modeling Assumptions

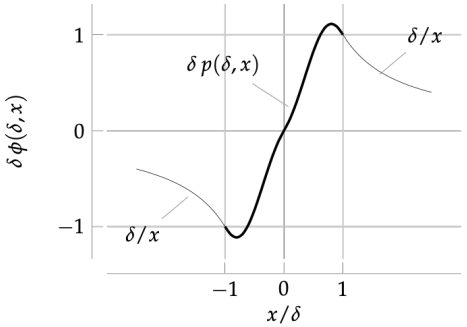
Media	• Can track moisture (X) and contaminants (C).
HVAC equipment	• Most equipment based on performance curve, or based on nominal conditions and similarity laws. • Vapor compression cycle mostly not modeled. • Most equipment either steady-state or 1st order transient.
Flow resistances	• Based on m_flow_nominal and dp_nominal plus similarity law. • Optional flag to linearize or to set dp=0.
Room model	• Any number of rooms and constructions are possible. • Layer-by-layer window model (similar to Window 6). • Optional flag to linearize radiation and/or convection. • Spawn of EnergyPlus uses EnergyPlus envelope model
Electrical systems	• DC. • AC 1-phase and 3-phase (dq, dq0). • Quasi-stationary or dynamic phase angle (but not frequency).
Special Consideration	• Numerical stability around zero-flow.

Special modeling approach

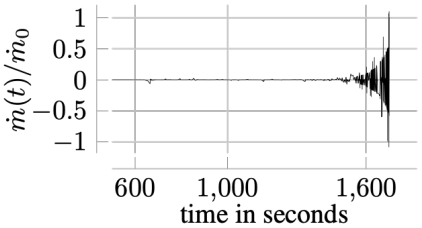
All equations of physical systems are once continuously differentiable.

$$\Delta h = \frac{\dot{Q}}{\dot{m}} \approx \dot{Q}\,\phi(\dot{m}_\delta, \dot{m})$$

Special treatments to avoid numerical problems if m_flow is in neighborhood around 0.



Fan/pump model for which we can prove existence of unique solution.
See paper at Building Simulation 2013.



Validation

Room model: ANSI/ASHRAE 140.

Window model: Window 6 plus full-scale experiments.

Comparative model validation for

- Window model (Window 6)

- Multizone air exchange (CONTAM)

- DX coils (EnergyPlus)

- Solar collectors (TRNSYS)

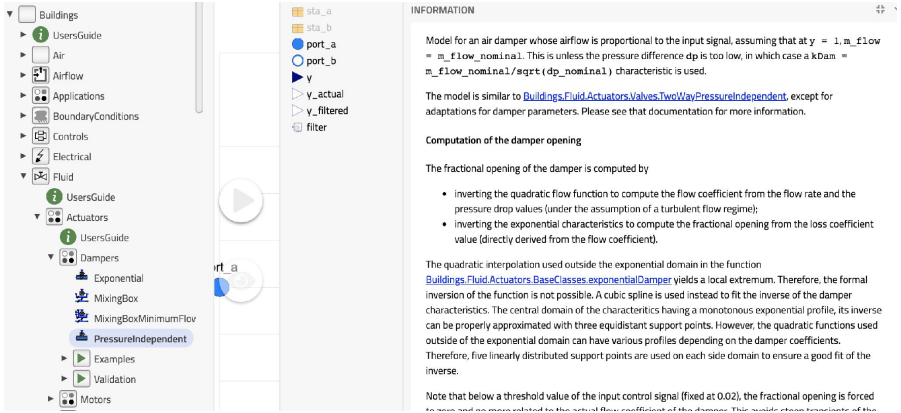
Where possible, all components were verified with analytical solutions.

500+ regression tests compare results to reference results as part of development,
see https://github.com/lbl-srg/modelica-buildings/wiki/Unit-Tests

Documentation and Distribution

Documentation

- General user guide (getting started, best practice, developer instructions, ...).
- User guides for individual packages.
- 4 tutorials with step-by-step instructions.
- All models contain an “info” section.
- Example models for all classes and many validation cases.
- Example models for system-level templates.

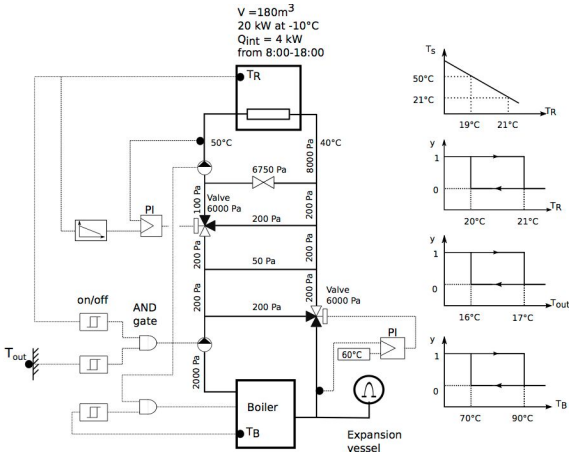


Distribution

- For users: http://simulationresearch.lbl.gov/modelica
- For developers: https://github.com/lbl-srg/modelica-buildings

Continuous Integration Testing

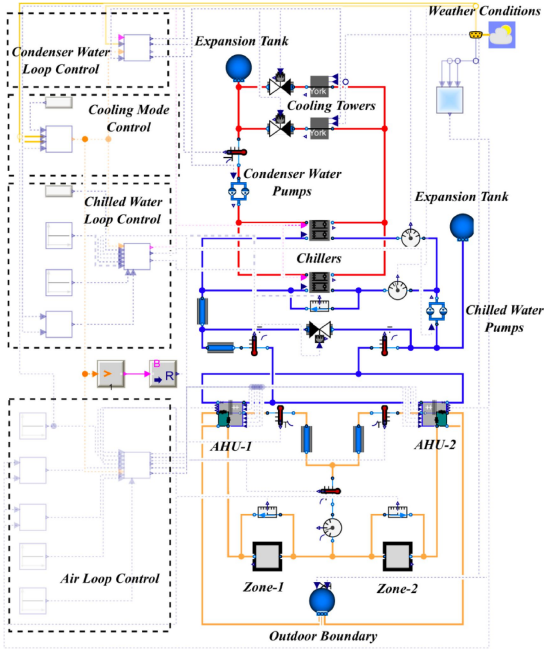
- Small tests for all classes and larger “smoke tests”
- Testing with Dymola, OPTIMICA (Impact), and OpenModelica (see https://simulationresearch.lbl.gov/modelica/environments.html)

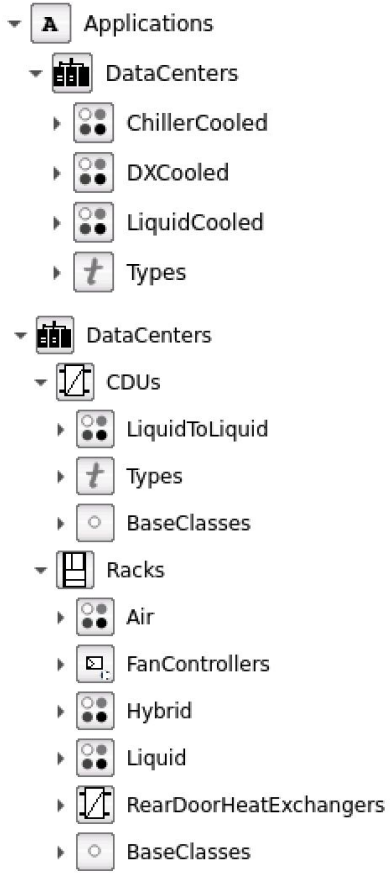


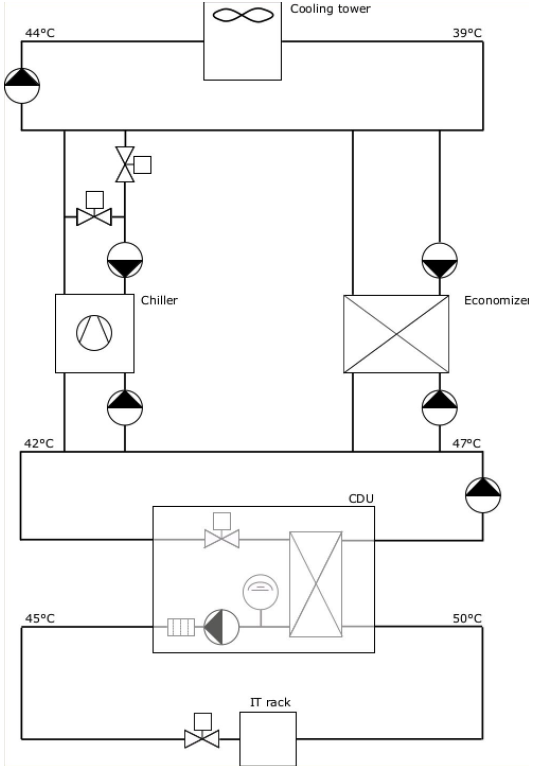
Example Applications

Cooling Plant for Data Center

Fu et al. 2019. Equation-based object-oriented modeling and simulation for data center cooling: A case study. *Energy and Buildings 186.* https://doi.org/10.1016/j.enbuild.2019.01.018

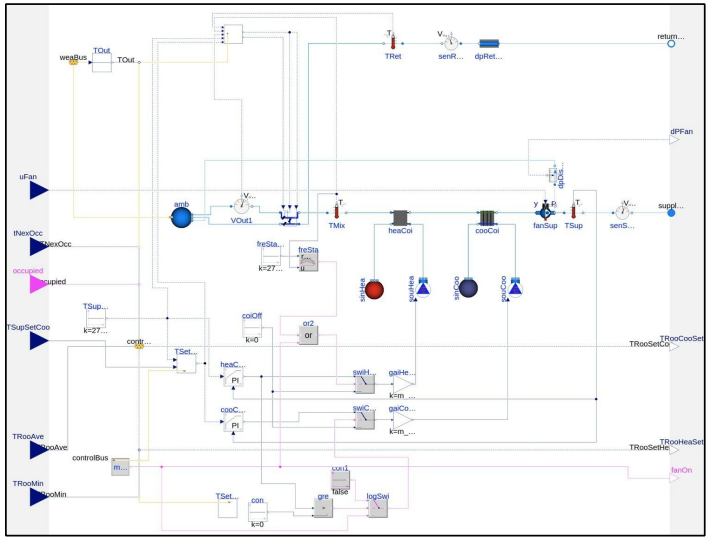




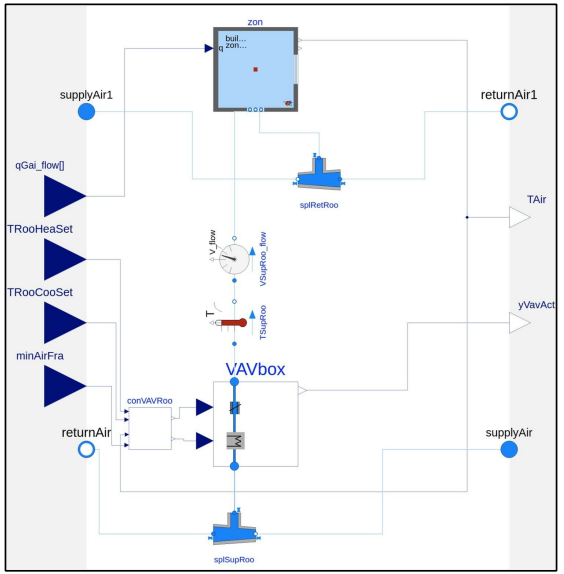


Implemented system diagram

Evaluating Building Controls

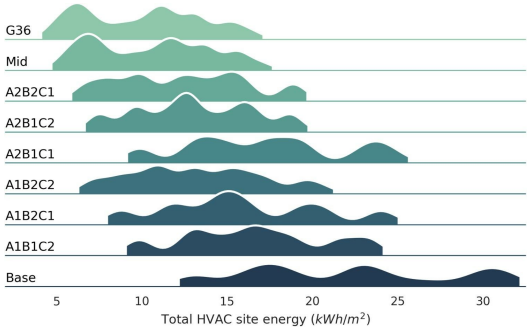


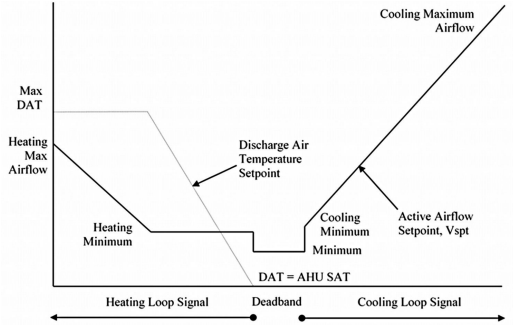
Air Handling Unit



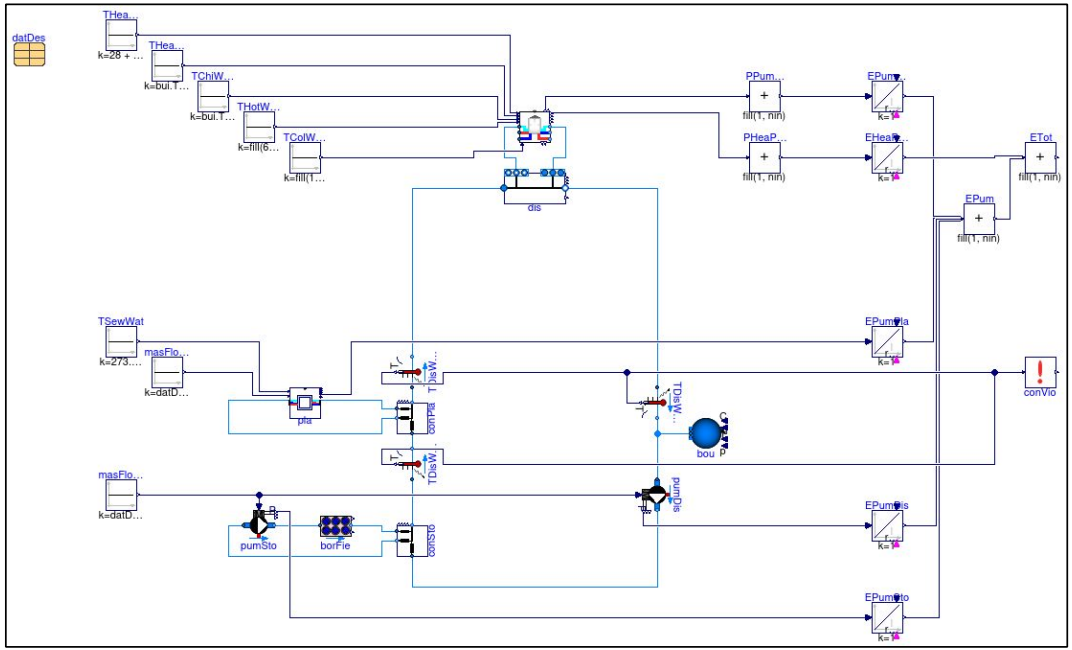
Terminal Unit and Zone

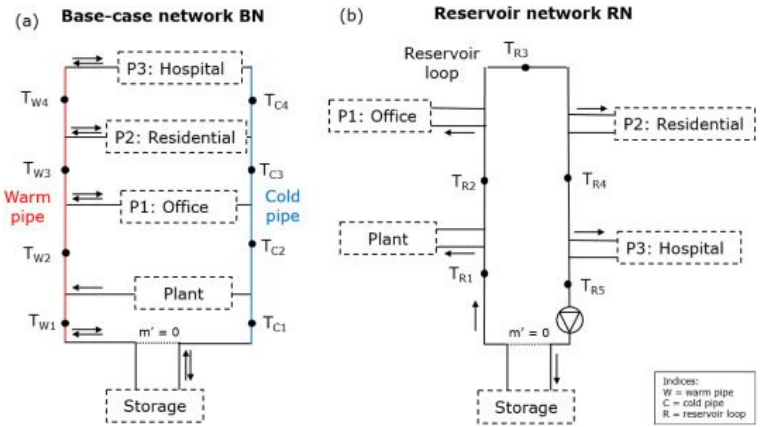
Zhang et al. 2022. Estimating ASHRAE Guideline 36 energy savings for multi-zone variable air volume systems using Spawn of EnergyPlus. ***Journal of Building Performance Simulation*** 15(2). https://doi.org/10.1080/19401493.2021.2021286





5th Generation District Energy Systems



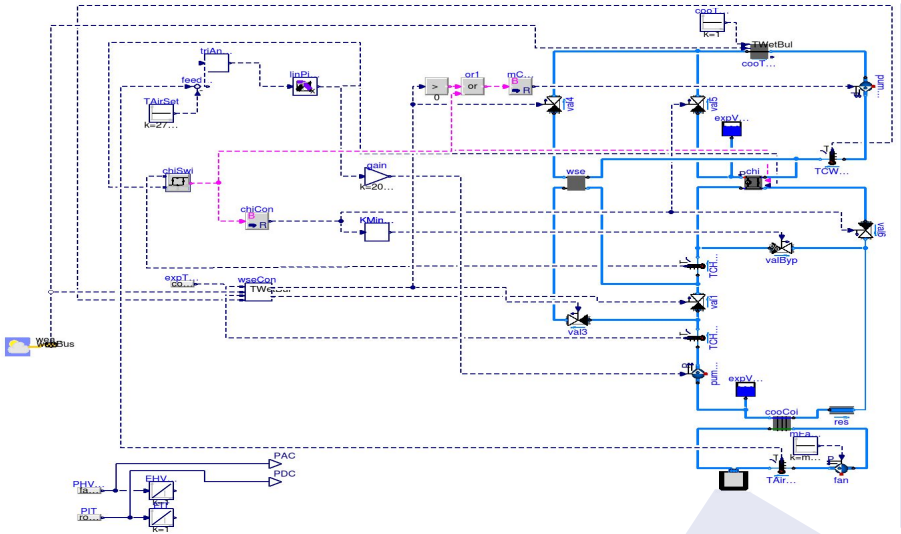


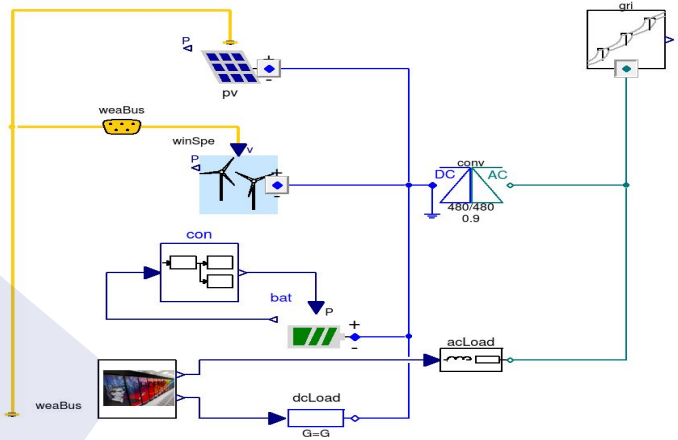
Sommer et al. 2021. The reservoir network: A new network topology for district heating and cooling. ***Energy*** 199. https://doi.org/10.1016/j.energy.2020.117418

Virtual prototyping through graphical modeling of multi-physics systems

Feedback control

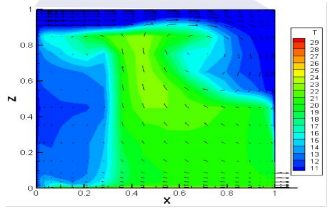
Cooling plant





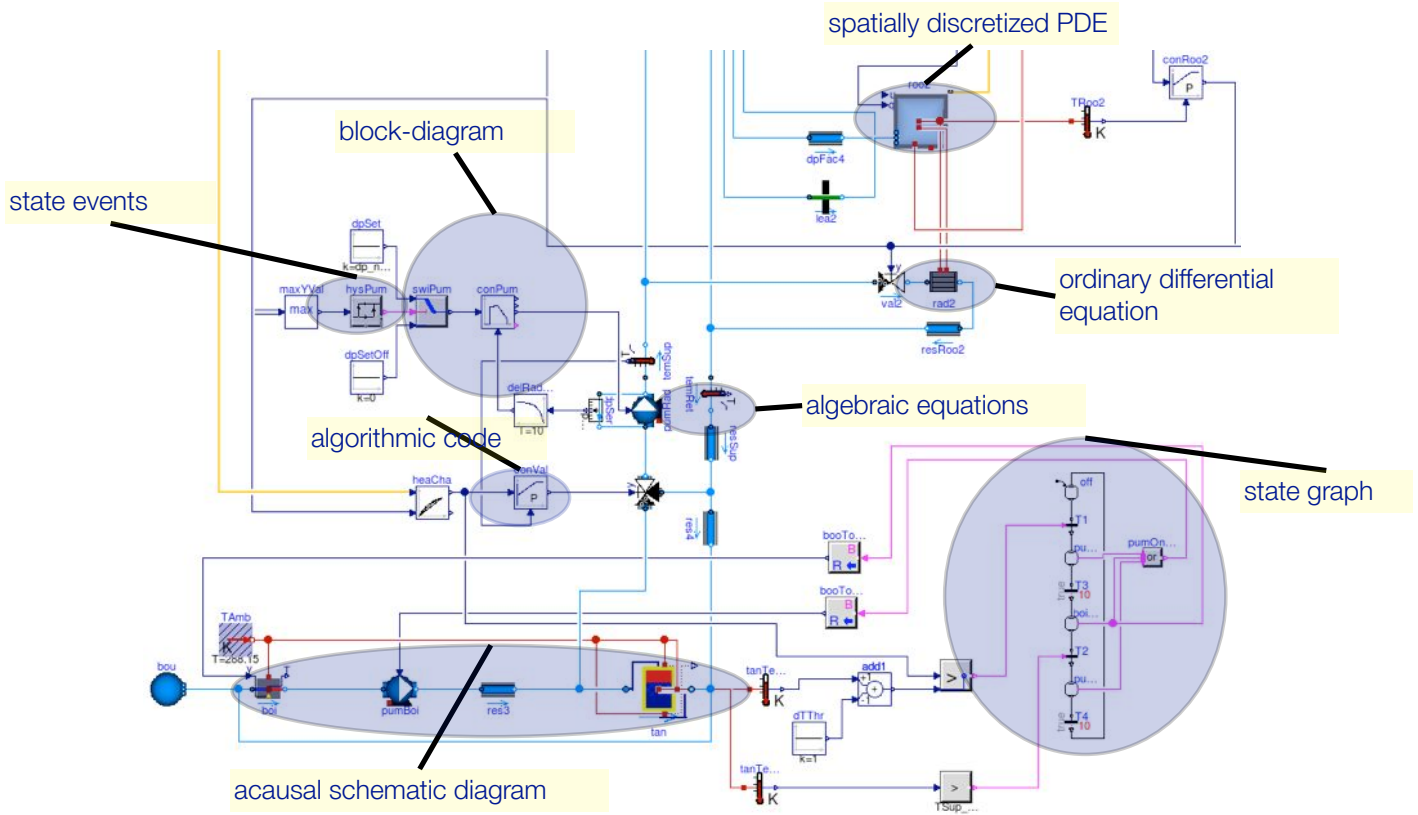
DC renewable plant

AC grid



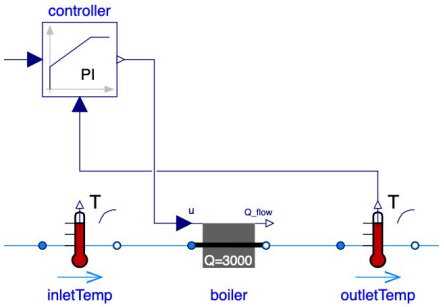
Temperature stratification based on CFD

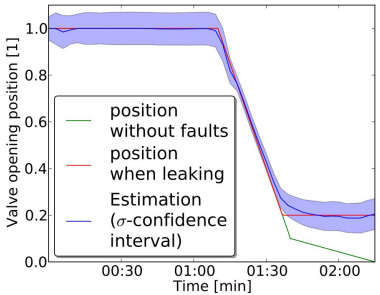
Modeling of hydronic systems

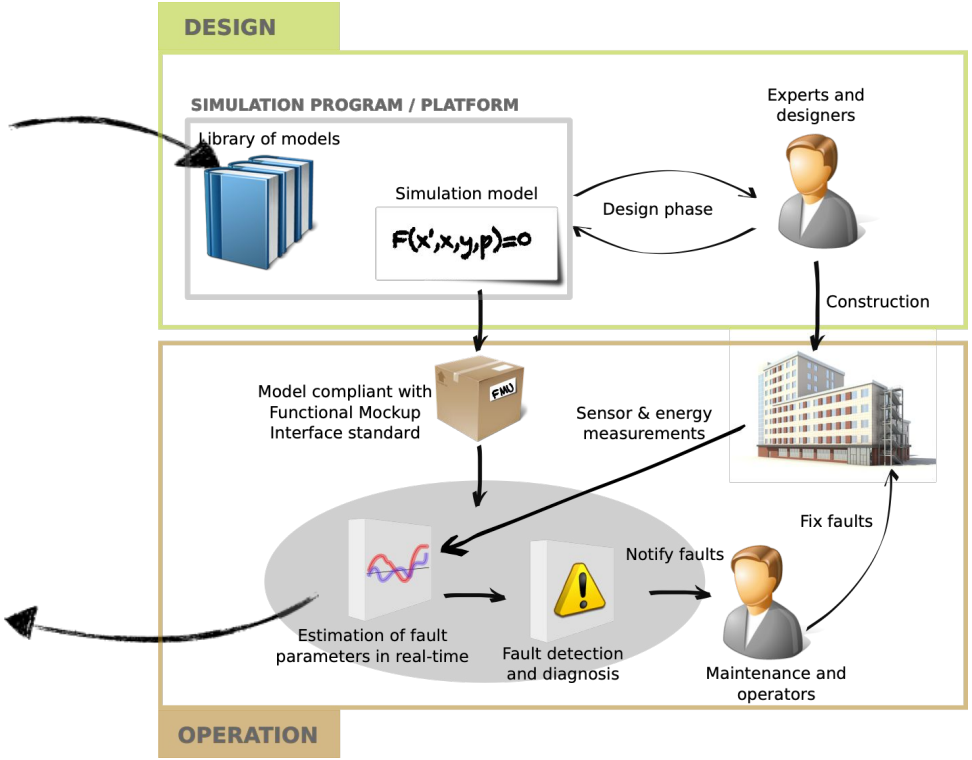


The above model is composed hierarchically, partially through automatic generation, of 1700 component models.

Fault detection based on design models under consideration of uncertainties

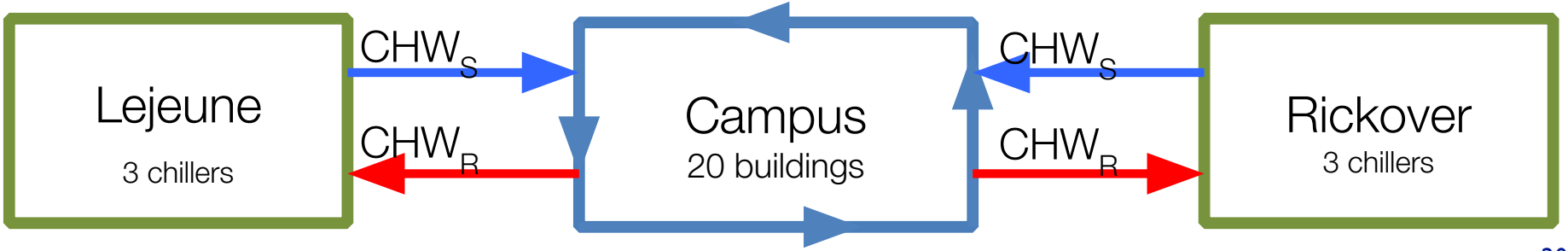




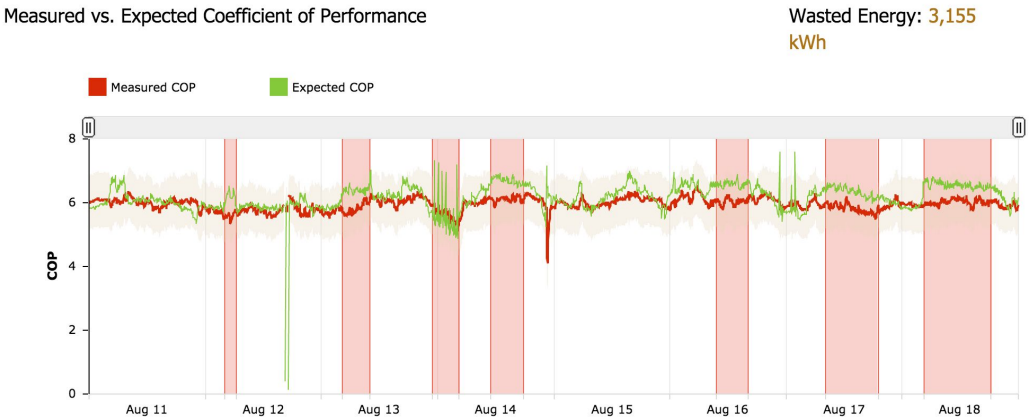


FDD applied to district chilled water plant

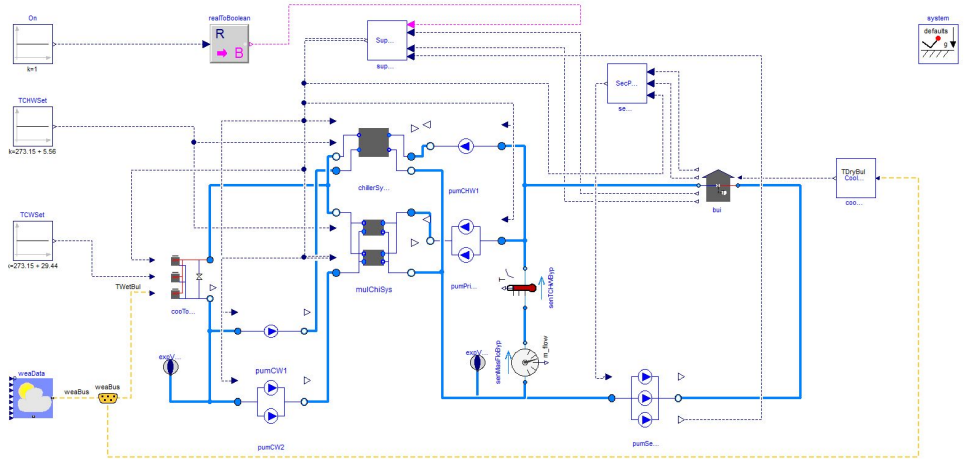




FDD applied to district chilled water plant



User-interface



Underlying analysis model

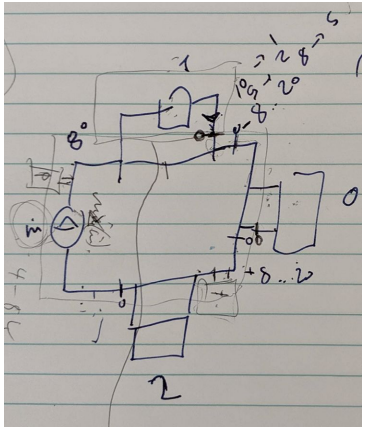
Best Practice and Modeling Hints

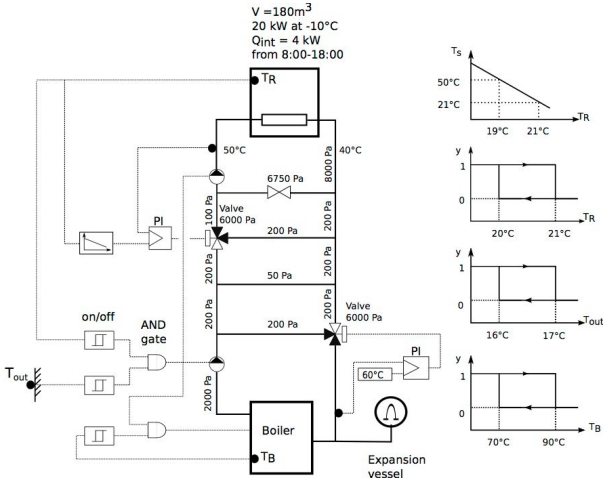
See also https://simulationresearch.lbl.gov/modelica/userGuide/bestPractice.html

Building large system models

1. **Understand the problem**:
    1. What question do you want to answer?
    2. Know what you want to model.
        1. Draw system schematics.
        2. Identify control input.
        3. Draw the control loops.
        4. Determine the control sequences.
2. **Compartmentalize**: Split the system into subcomponents that can be tested in isolation.
3. **Implement**: Now, and only now, start implementing in software.
    1. Document and build test cases as you go along.

       Errors are easy to detect in small models, but hard in large models. If you add unit tests, you make sure what has been tested remains intact as the model evolves.
    2. Assemble the subcomponents to build the full model.





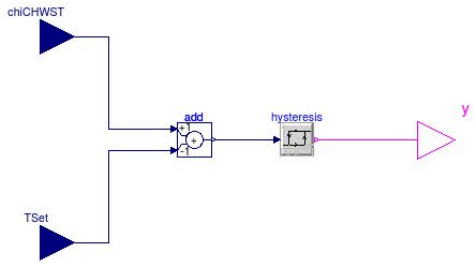
Building large system models

How do you debug a large system model?

1. Split the model into small models — or better, architect the large model from the beginning to be based on smaller models

2. Test the smaller models for well known conditions.

3. Add smaller models to unit tests.



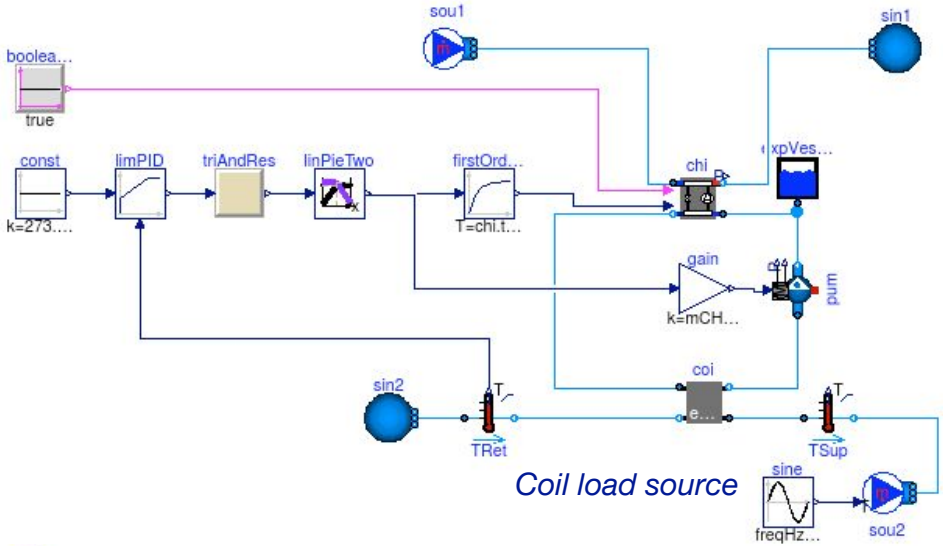
For example, see Chiller Plant,
in which each small models contains a simple unit test.

Building large system models

Chiller plant with well-defined boundary conditions

Condenser water source

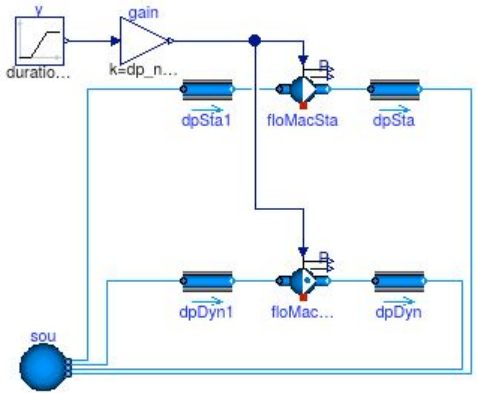
Condenser water sink



Coil load source

Coil load sink

Pumps



Don't **R**epeat **Y**ourself: Propagate common parameters

Don't assign the same values to multiple parameters:

```
Pump pum(m_flow_nominal=0.1) "Pump";
TemperatureSensor sen(m_flow_nominal=0.1) "Sensor";
```

Instead, propagate parameters and assign the value once:

```
Modelica.Units.SI.MassFlowRate m_flow_nominal = 0.1"Nominal mass flow rate";
Pump pum(final m_flow_nominal=m_flow_nominal) "Pump";
TemperatureSensor sen(final m_flow_nominal=m_flow_nominal) "Sensor";
```

Note: For arrays of parameters, use the **each** keyword, as in

```
TemperatureSensor sen[2]( each final m_flow_nominal=m_flow_nominal) "Sensor";
```

Assignments can include computations, such as

```
Modelica.Units.SI.HeatFlowRate QHea_nominal = 3000 "Nominal heating power";
Modelica.Units.SI.TemperatureDifference dT = 10  "Nominal temperature difference";
Modelica.Units.SI.MassFlowRate m_flow_nominal = QHea_nominal/dT/4200
  "Nominal mass flow rate";...
```

Don’t **R**epeat **Y**ourself: Define the media at the top-level

Top-level system-model

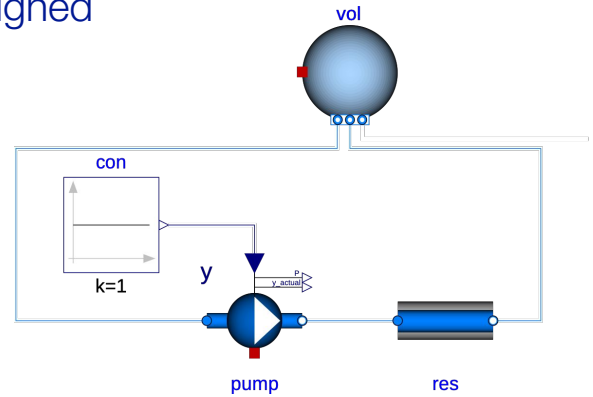
```
replaceable package Medium = Buildings.Media.Air "Medium model";
```

Propagate medium to instance of model

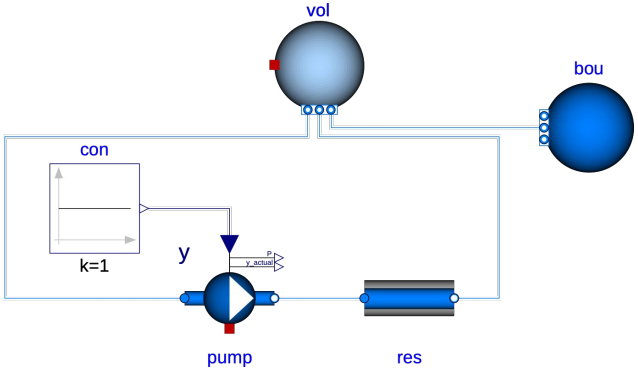
```
TemperatureSensor sen(
  redeclare final package Medium = Medium,
  final m_flow_nominal=m_flow_nominal) "Sensor";
```

All system models must have a reference pressure

Underdetermined model as no pressure state is assigned

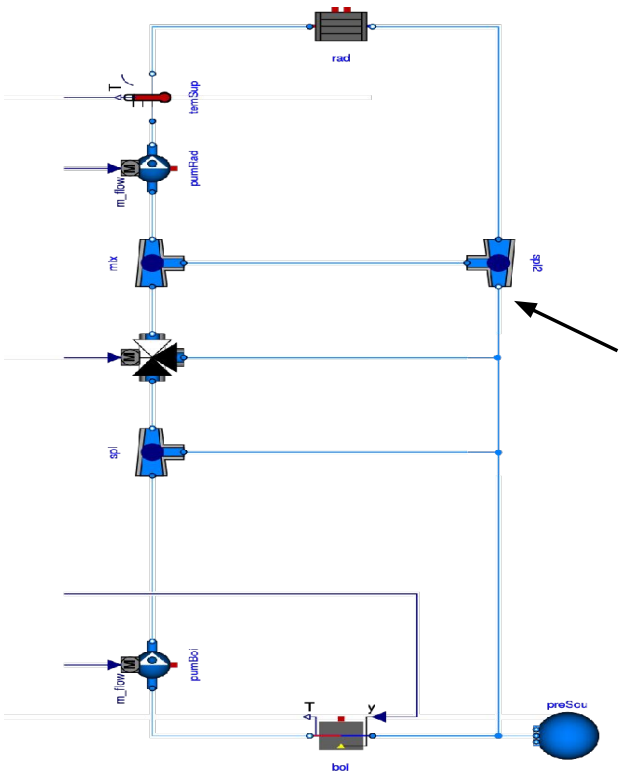


Model that provides a reference presssure through the instance **bou**.

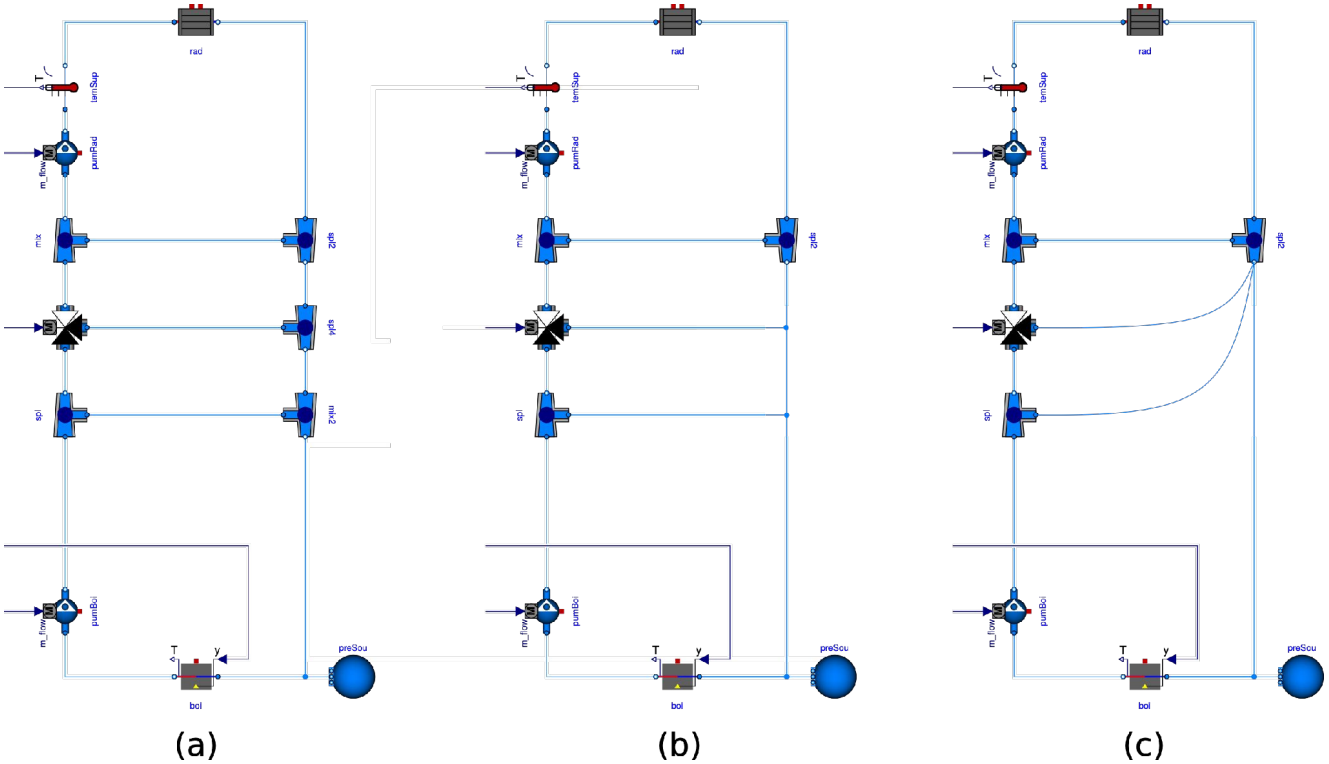


Modeling of fluid junctions

What is wrong with this model?



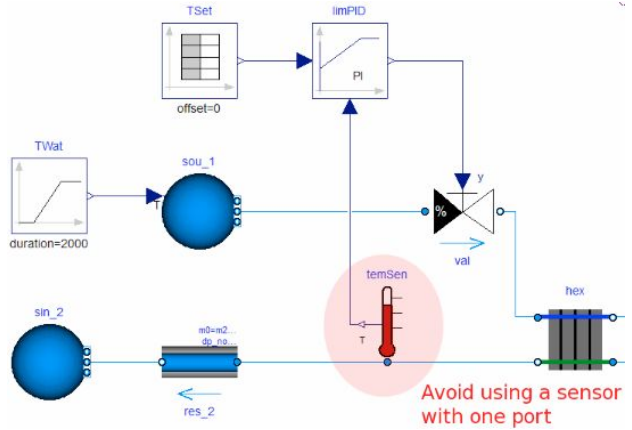
Modeling of fluid junctions



Avoid oscillations of sensor signal: Use two-port sensors

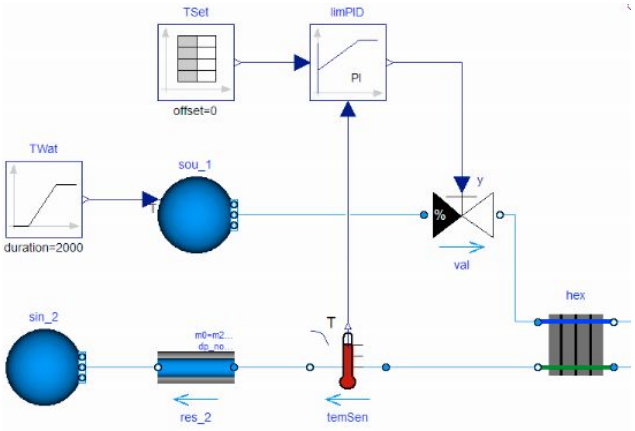
Incorrect, as sensor output oscillates if mass flow rate changes sign.
This happens for example if the mass flow rate is near zero and approximated by a solver.

See also User Guide.



Correct use because

$$\tau \frac{dT}{dt} = \frac{|\dot{m}|}{\dot{m}_0} (\theta - T)$$



Nature is smooth, so avoid events

This triggers events:

```
h = if port_a.m_flow > 0
  then inStream(port_a.h_outflow) else port_a.h_outflow;
```

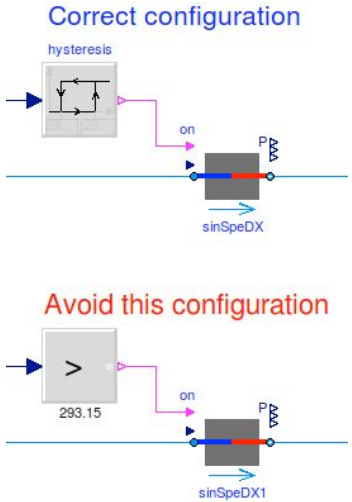
Avoid events using regularization:

```
h = Modelica.Fluid.Utilities.regStep(
      x = port_a.m_flow,
      y1 = inStream(port_a.h_outflow),
      y2 = port_a.h_outflow,
      x_small = m_flow_nominal*1E-4);
```

See also User Guide.

Beware of oscillating control, and guard against noise

If the control error oscillates around zero, then this model stalls due to numerical noise and fast switching



Similarly, this model will also stall when x reaches 0.

```
model Test
  Real x(start=0.1);
equation
  der(x) = if x > 0 then -1 else 1;
end Test;
```

Setting nominal values is important for scaling solver residuals

If pressure is around 1E5 Pa, set **p(nominal=1E5)**.

Nominal values are used to scale residuals.

Modelica simulation tools typically control the local integration error as

$$\epsilon \leq t_{rel} |x^i| + t_{abs}$$

where the absolute tolerance is scaled with the nominal value as

$$t_{abs} = t_{rel} |x_{nom}{}^i|.$$

Most component models in Buildings Library already have the nominal value set.

Exercise 1: Modeling a Simple Thermofluid System

(See accompanying tutorial pdf)

Exercise: Modeling of a simple thermofluid flow system

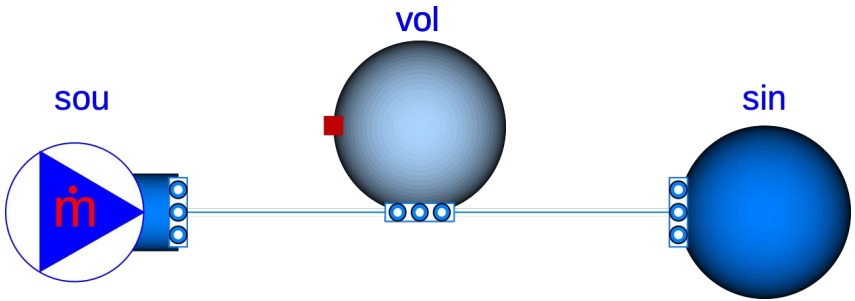
Implement a model with

1. a flow source of *1 kg/s* of water at *30°C*,
2. a well stirred tank with no heat loss, volume of *2 m^3*, and 20*°C* initial temperature, and
3. an infinite sink that is at atmospheric pressure?



Exercise: Modeling of a simple thermofluid flow system

1. Make instances using models from **Buildings.Fluid.Sources** and **Buildings.Fluid.MixingVolumes**.
2. Assign the parameters.
3. Check and simulate the model.



Further resources

Tutorials

- Buildings.Examples.Tutorial

User guides

- User guides for specific packages of models.
- User guide with general information.

Exercise 2: Modeling a Space Cooling System

From Space Cooling Tutorial

(See accompanying tutorial pdf)

Developer Guide

See also https://simulationresearch.lbl.gov/modelica/userGuide/prePostProcessing.html

Overview

Main topics

- Coding style and conventions
- Requirements
- Organization of the library
- Adding a new model
- Adding regression tests

Further literature

- User Guide -> Development
- Style guide

Coding style and conventions

Based on Modelica Standard Library.

Most variables are 3 letter camel case to avoid too long names, for example,
TOut, yDamMax, …

Code duplication avoided where practical.

Additional information at
https://simulationresearch.lbl.gov/modelica/userGuide/development.html#style-guide
and
https://simulationresearch.lbl.gov/modelica/releases/latest/help/Buildings_UsersGuide.html#Buildings.UsersGuide.Conventions

Requirements

Physical requirements

Mathematical requirements

Organization of individual packages

Packages are typically structured as shown on the right.

To add a new class, look first at **Interfaces** and **BaseClasses**.

You probably will never implement a component without extending a base class, such as from **Buildings.Fluid.Interfaces**

Tutorial
UsersGuide

Any other classes (models, functions etc.)

Data
Types
Examples
Validation
Benchmarks
Experimental
Interfaces
BaseClasses
Internal
Obsolete

Implementing new thermofluid flow devices

Buildings.Fluid.Interface provides base classes.

Buildings.Fluid.Interface.UsersGuide describes these classes.

Alternatively, simple models such as the models below may be used as a starting point for implementing new models for thermofluid flow devices:

Buildings.Fluid.HeatExchangers.HeaterCooler_u
For a device that adds heat to a fluid stream.

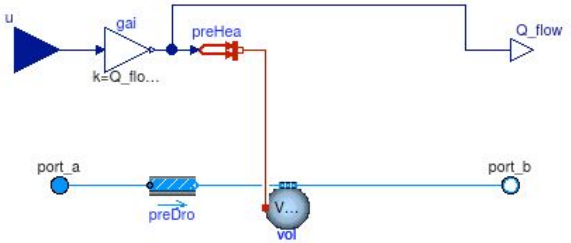
Buildings.Fluid.MassExchangers.Humidifier_u
For a device that adds humidity to a fluid stream.

Buildings.Fluid.Chillers.Carnot
For a device that exchanges heat between two fluid streams.

Buildings.Fluid.MassExchangers.ConstantEffectiveness
For a device that exchanges heat and humidity between two fluid streams.

Adding a heat exchanger

See HeaterCooler_u




```
within Buildings.Fluid.HeatExchangers;
model HeaterCooler_u "Heater or cooler with prescribed heat flow rate"
  extends Buildings.Fluid.Interfaces.TwoPortHeatMassExchanger(
    redeclare final Buildings.Fluid.MixingVolumes.MixingVolume vol(prescribedHeatFlowRate=true));
  parameter Modelica.SIunits.HeatFlowRate Q_flow_nominal "Heat flow rate at u=1, positive for
heating";
  Modelica.Blocks.Interfaces.RealInput u "Control input";
  Modelica.Blocks.Interfaces.RealOutput Q_flow(unit="W") "Heat added to the fluid";
protected
  Buildings.HeatTransfer.Sources.PrescribedHeatFlow preHea  "Prescribed heat flow";
  Modelica.Blocks.Math.Gain gai(k=Q_flow_nominal) "Gain";

equation
  connect(u, gai.u); ... // other connect statements
  annotation (...);      // documentation
end HeaterCooler_u;
```

Add examples and validations to unit testing framework

1. Add validation and stress tests for different model configurations.

2. Validate results and add main outputs to plot script. These variables become part of the regression tests.

2. Run
   modelica-buildings/bin/runUnitTests.py

3. Update **Buildings/package.mo** release notes.

4. Issue pull request on
   https://github.com/lbl-srg/modelica-buildings.

See Unit Test documentation.

