KU LEUVEN

Modelica, Dymola and IDEAS

Crash Course 2023

Javier Arroyo, Jelger Jansen,
Louis Hermans, Lucas Verleyen

THERMAL SYSTEMS SIMULATION

Who are we?

Jelger Jansen

Javier Arroyo

Lucas Verleyen

Louis Hermans

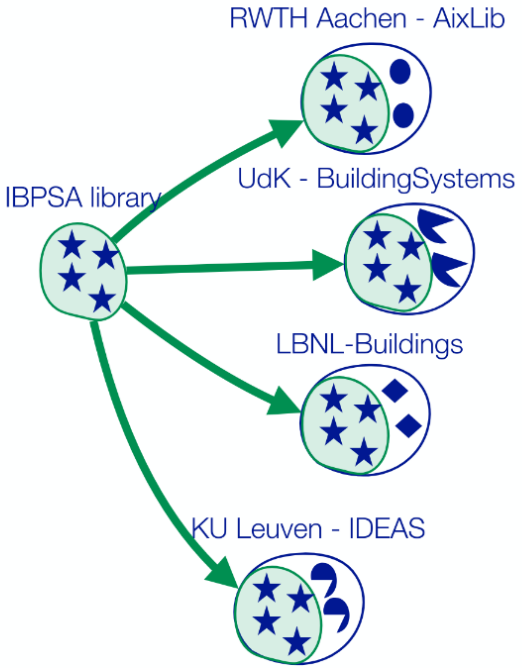
The SySi Team

- Led by Professor Lieve Helsen
- Our Mission
  To sustainably use resources through integration and optimization of thermal systems performance in the built environment, including other energy vectors and sectors.

Motivation

- Why a crash course?
  - Introduction for our students and others who are interested in Modelica
  - Broaden the user base

- About IDEAS
  - Modelica users and library development since 2010
  - IDEAS v3.0, BaseClasses inherited from IBPSA project 1
    - https://github.com/open-ideas/IDEAS
    - https://github.com/ibpsa/modelica-ibpsa
  - Many models are validated in academic research
  - Main user base: researchers, students



- Builtwins: startup for sustainable control of buildings, using IDEAS

Agenda

Morning: Dymola and Modelica

- 9:30 - 10:00 Lecture 1
  - What is Modelica? What is Dymola? What is OpenModelica?
  - Modelica/Dymola basics
- 10:00 - 10:15 Exercise 1
- 10:15 - 10:30 Break
- 10:30 - 11:00 Lecture 2
  - Create new models/packages
  - Modelling with several components
  - Use connectors
  - Set parameters/propagate parameters
- 11:00 - 12:15 Exercise 2
- 12:15 - 13:15 Lunch break

Afternoon: IDEAS

- 13:15 - 13:45 Lecture 3
  - What is IDEAS?
  - IDEAS building components
  - IDEAS workflow
- 13:45 - 15:45 Exercise 3
- 15:45 - 16:00 Break
- 16:00 - 16:15 Lecture 4
  - IDEAS HVAC components
  - Hydronic models
  - Discrete control logic
- 16:15 - 18:00 Exercise 4

Part 1: Introduction to Modelica and Dymola

Javier Arroyo

Modelica

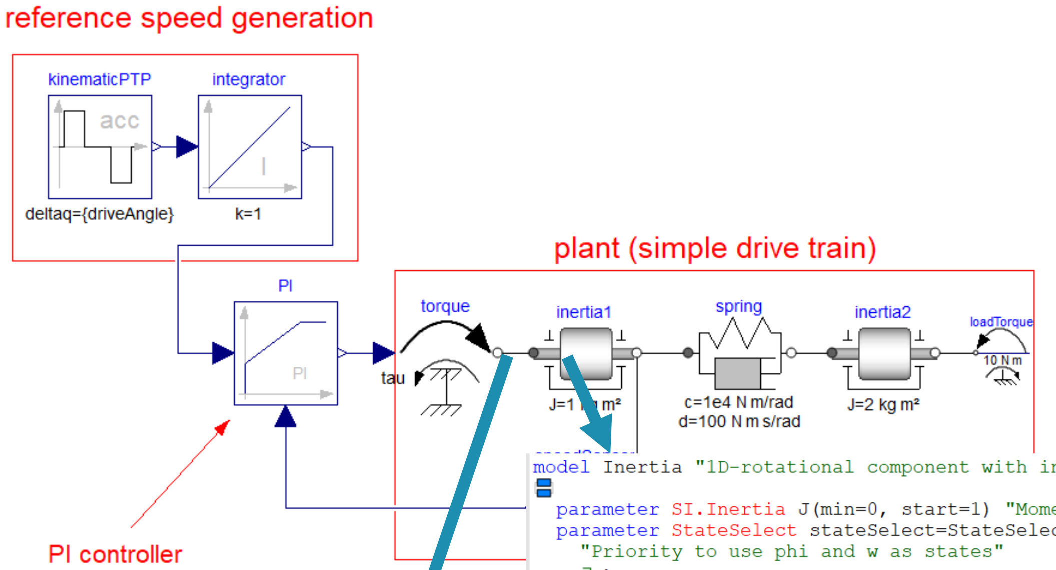
Modelica is a **modelling language** for modelling physical systems

- Language specification is open source
- object oriented
- Acausal modeling (equation-based)
- Multi-domain
- Primarily for simulation, but usable for optimization
- Small and large models (> 100 000 equations)
- Large community with many model libraries, especially in automotive industry (free and commercial)
- Textual and graphical modelling

Modelica

Object-oriented physical equation-based modelling

- A model represents a physical component
- Component is composed of sub-components and/or is described by equations
  → hierarchical structure
- Components can be connected to each other using connectors (=physical coupling)
- To simulate Modelica models, a Modelica simulation environment is needed




```
model Inertia "1D-rotational component with inertia"

  parameter SI.Inertia J(min=0, start=1) "Moment of inertia";
  parameter StateSelect stateSelect=StateSelect.default
    "Priority to use phi and w as states"
    ;
  SI.Angle phi(stateSelect=stateSelect)
    "Absolute rotation angle of component"
    ;
  SI.AngularVelocity w(stateSelect=stateSelect)
    "Absolute angular velocity of component (= der(phi))"
    ;
  SI.AngularAcceleration a
    "Absolute angular acceleration of component (= der(w))"
    ;

equation
  phi = flange_a.phi;
  phi = flange_b.phi;
  w = der(phi);
  a = der(w);
  J*a = flange_a.tau + flange_b.tau;

end Inertia;
```

```
connector Flange_a
  "1-dim. rotational flange of a shaft (filled square icon)"
  SI.Angle phi "Absolute rotation angle of flange";
  flow SI.Torque tau "Cut torque in the flange";

end Flange_a;
```


THERMAL SYSTEMS SIMULATION

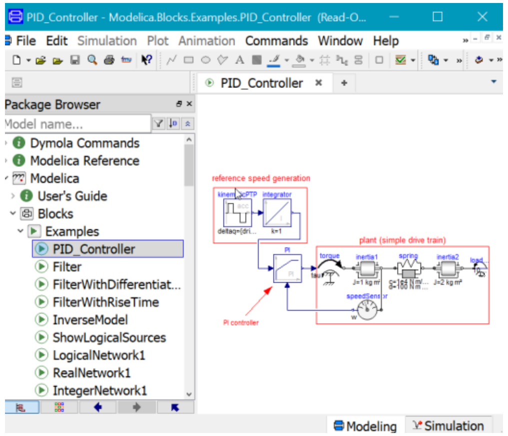
KU LEUVEN

Dymola

Dymola is a **commercial Modelica simulation environment**

Live demo of features:

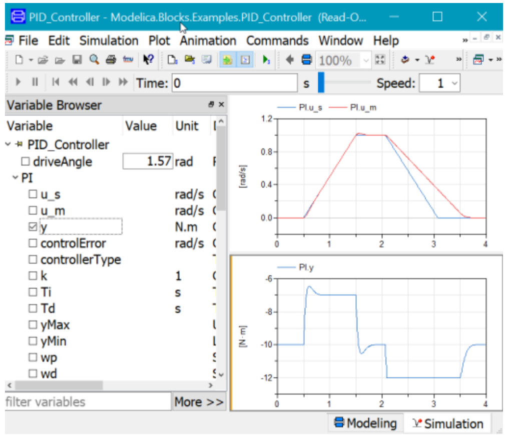
- Icon, Diagram, Editor, Info
- Package browser, modelling, simulation
- Set up (compiler), run
- Adapt parameter
- Load libraries
- Look at simulation results: plot, zoom, filter variable, plot as a function of other variable.
- Try Simulate and plot (IDEAS library)
- Open sub-components
- Documentation



- Graphical editor
- Modelica simulation environment

```
model PID_Controller
  "Demonstrates the usage of a Continuous.LimPID controller"
  extends Modelica.Icons.Example;
  parameter Modelica.SIunits.Angle driveAngle=1.57
    "Reference distance to move";
  Modelica.Blocks.Continuous.LimPID PI(
    k=100,
    Ti=0.1,
    yMax=12,
    Ni=0.1,
    initType=Modelica.Blocks.Types.InitPID.SteadyState,
    limitsAtInit=false,
    controllerType=Modelica.Blocks.Types.SimpleController.PI,
    Td=0.1) a ;
  Modelica.Mechanics.Rotational.Components.Inertia inertia1(
    phi(fixed=true, start=0),
    J=1,
    a(fixed=true, start=0)) a ;

  Modelica.Mechanics.Rotational.Sources.Torque torque a ;
```

- Textual description (Modelica language)



- Translation of Modelica code into executable C-code
- Coupling with a solver
- Visualization of results

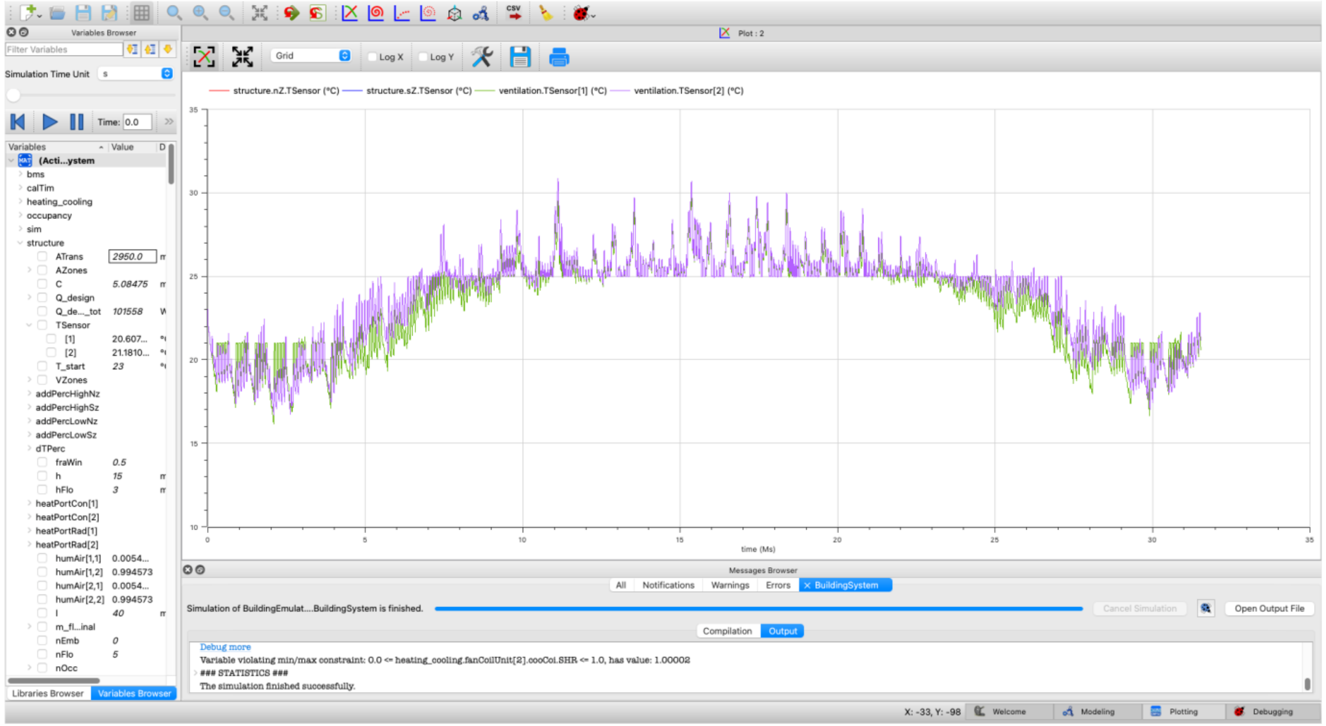
THERMAL SYSTEMS SIMULATION

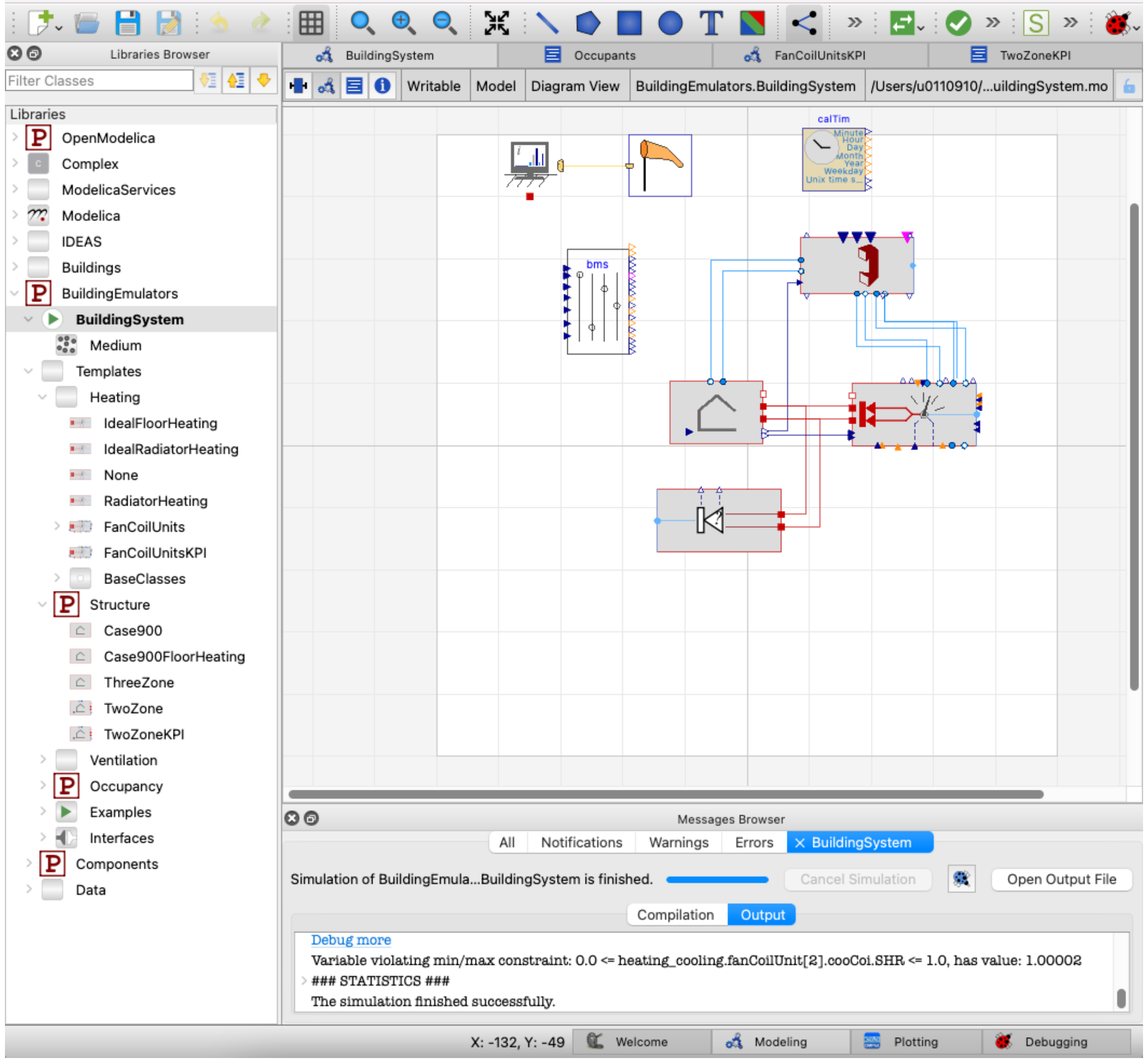
KU LEUVEN

OpenModelica

OM is a **free Modelica simulation environment**

Has very similar features than Dymola, like those we have just seen

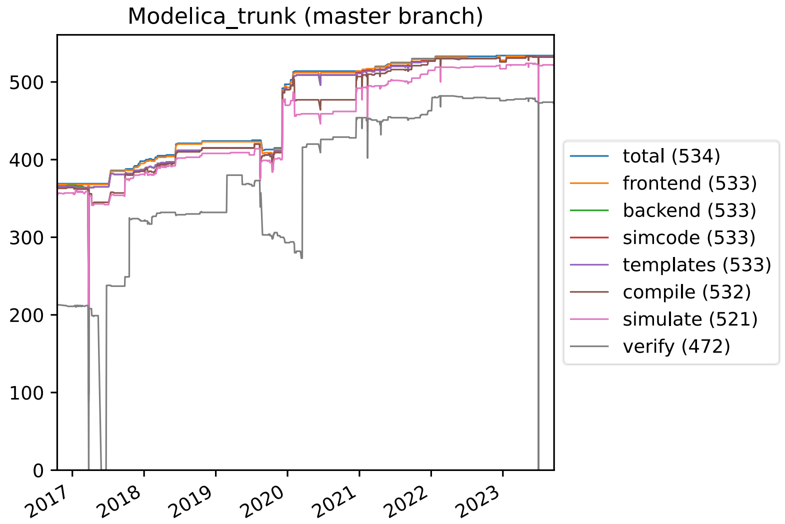




OpenModelica

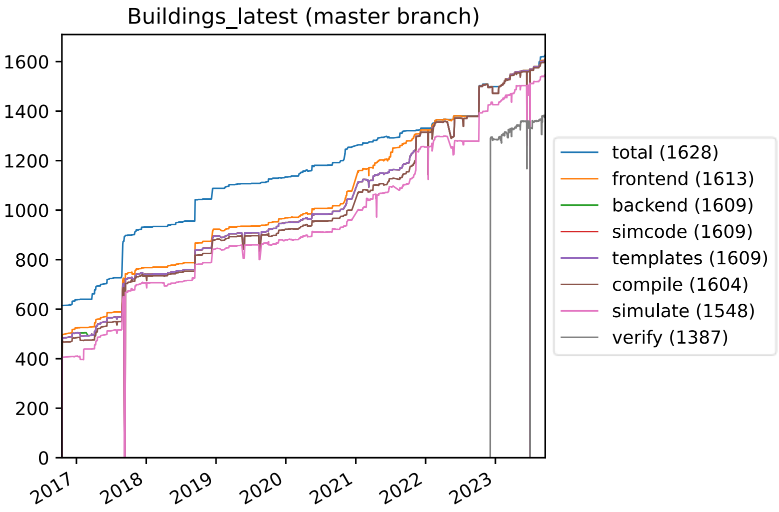
Historically less performant than Dymola, but has radically improved over the last years:

Modelica Standard Library (MSL) coverage



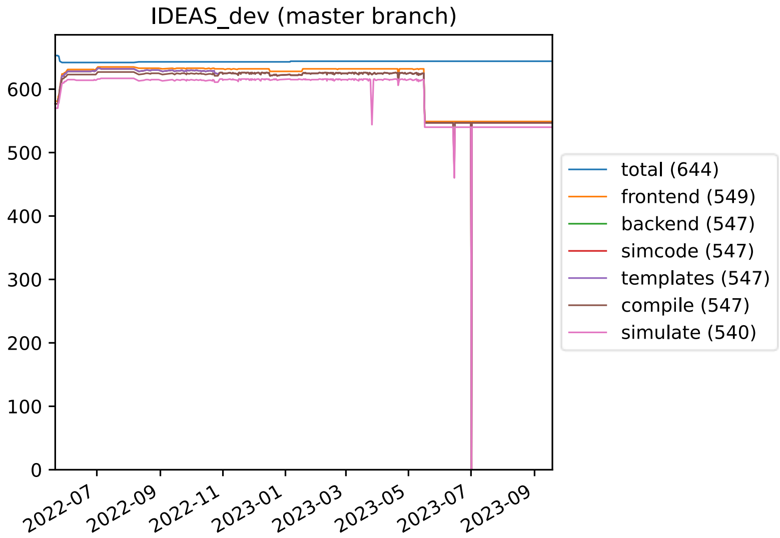
https://libraries.openmodelica.org/branches/history/master/Modelica_trunk.svg

Buildings library coverage



https://libraries.openmodelica.org/branches/history/master/Buildings_latest.svg

IDEAS coverage (recently added!)



https://libraries.openmodelica.org/branches/history/master/IDEAS_dev.svg

Reasons not to use it in the crash course: Dymola is still more performant…

Useful links

General

- www.modelica.org
- www.openmodelica.org
- www.jmodelica.org DISCONTINUED
- http://www.claytex.com/tech-blog/

Modelica language

- http://book.xogeny.com/
- http://doc.modelica.org
- http://specification.modelica.org/

Libraries:

- IDEAS
  https://github.com/open-ideas
- Buildings
  https://simulationresearch.lbl.gov/modelica
  (look at Buildings.Examples.Tutorial)
- IBPSA Project 1
  https://github.com/ibpsa/modelica-ibpsa

Dymola user guide

- Online
- Via Dymola > help

Exercise 1

- See exercise sheet on Github
  https://github.com/open-ideas/__CrashCourse__/blob/master/Exercises/Exercise%201/Latex/Exercise1.pdf

Part 2: Modelling and simulating in Dymola

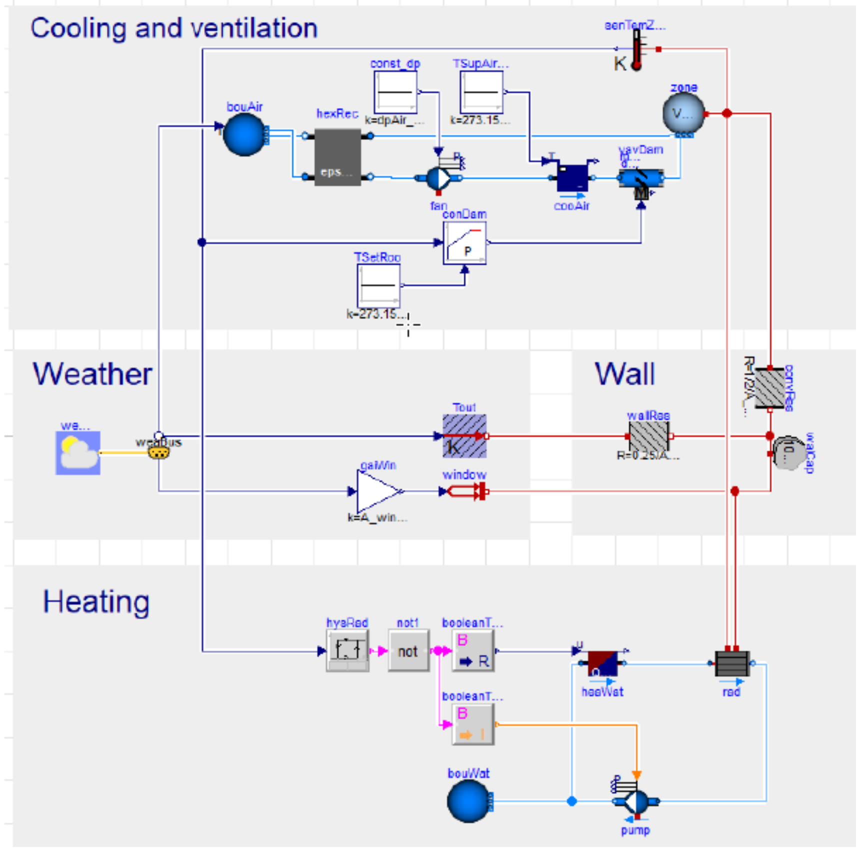
Louis Hermans

Live demonstration

- Create package, create model
- Units
- Search, drag and drop subcomponents. Instantiate model convention
- Simulation tab and adapt parameters
- Connect components
- Propagate parameters
- Use check/translate in Dymola and debug:
    - Syntax error
    - Modeling error: singularity
    - Model with external input

Exercise 2 – Simple house model

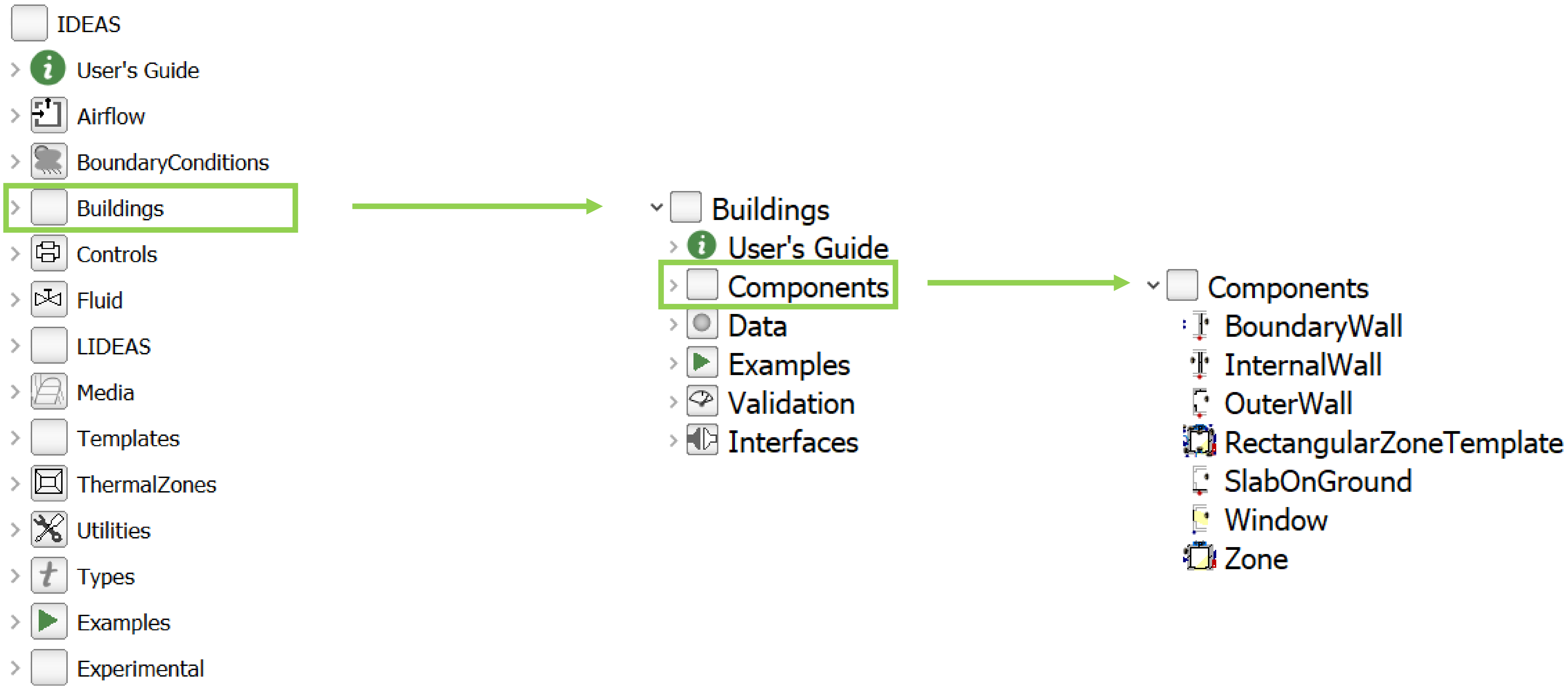
- See exercise sheet on Github
  https://github.com/open-ideas/__CrashCourse__/blob/master/Exercises/Exercise%202/Latex/Exercise2.pdf



Part 3: IDEAS – Building

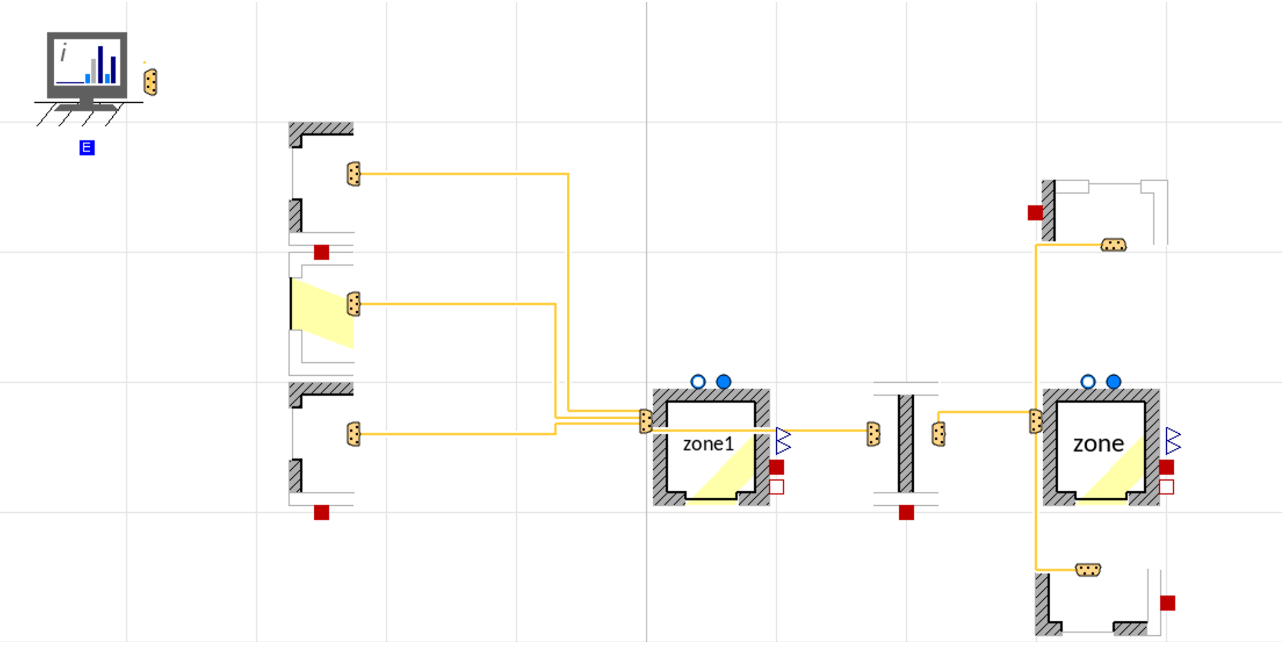
Jelger Jansen

IDEAS – Overview

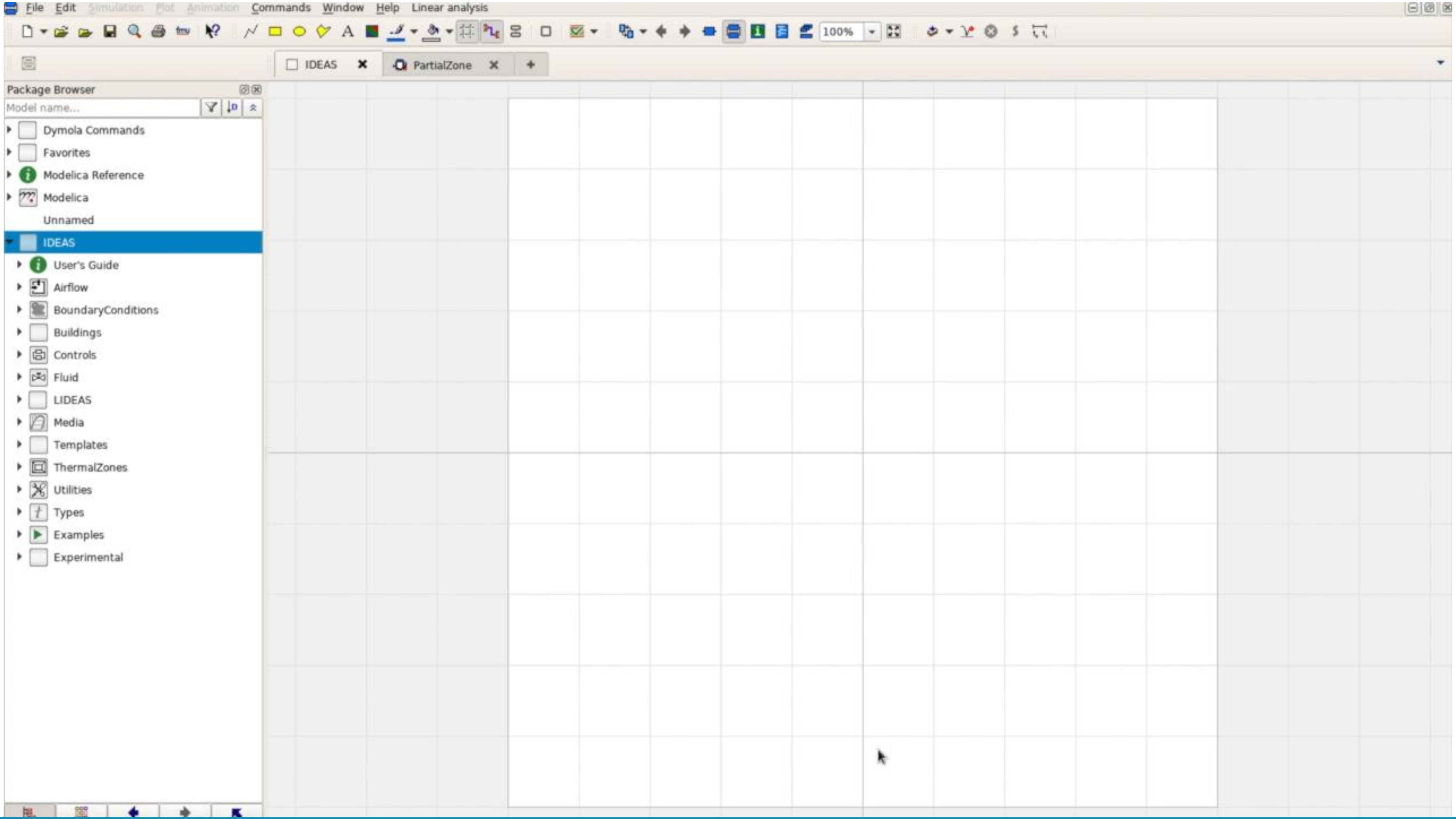


IDEAS – Philosophy

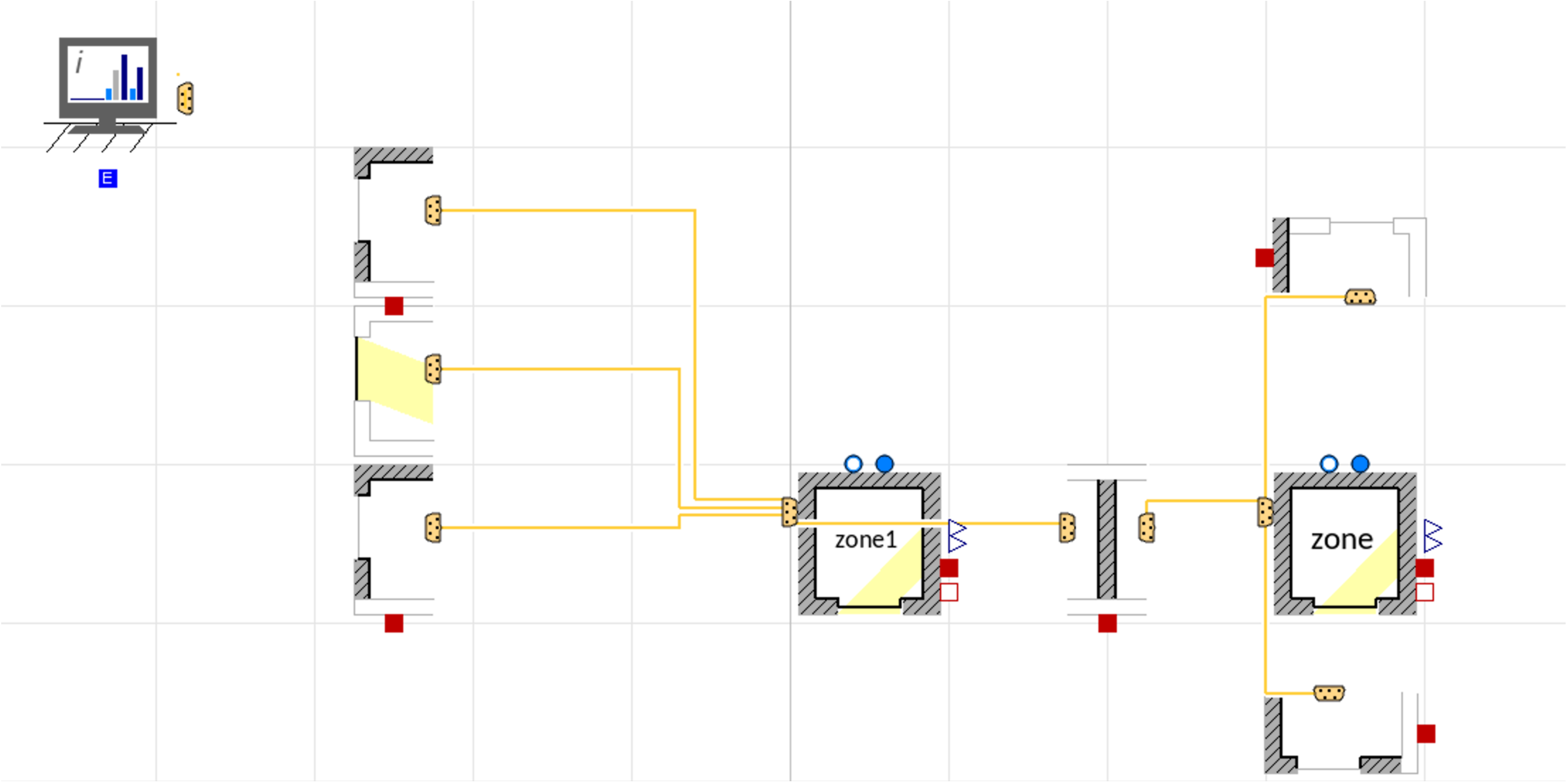
- Philosophy
  - Direct mapping between physical objects and components
  - “What you see is what you get”
  - Exception: SimInfoManager



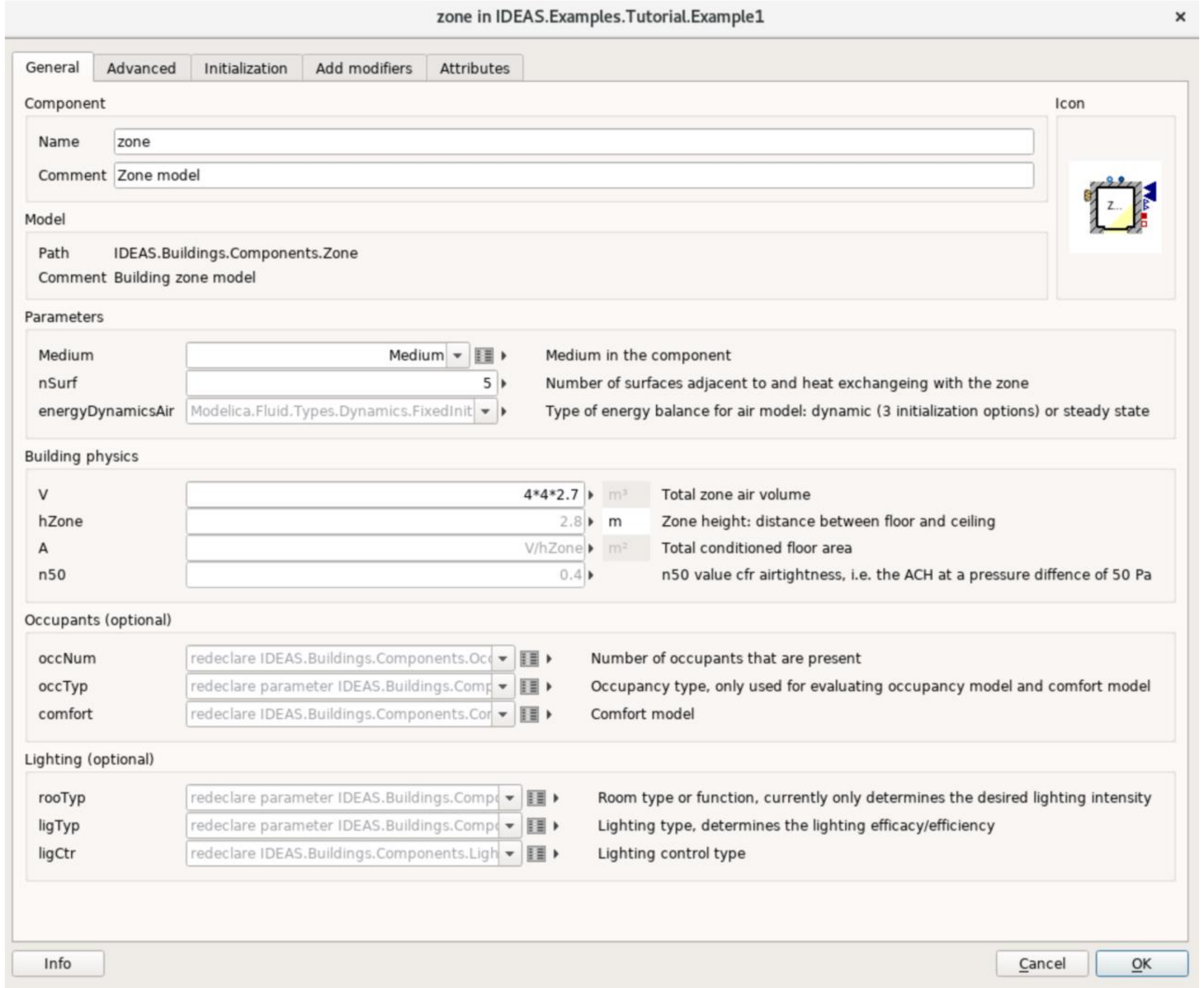
IDEAS – Workflow



IDEAS – Multizone

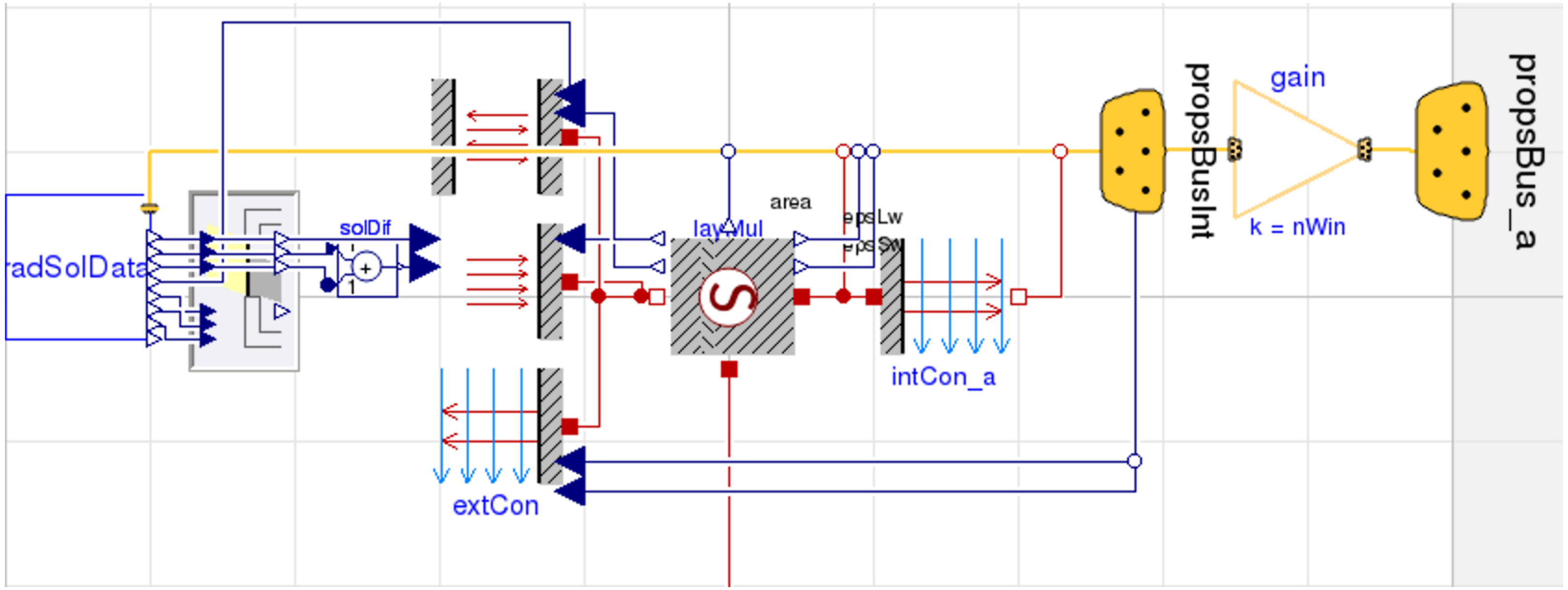


IDEAS – Parameters



IDEAS – Main building physics

- Conduction, thermal mass
- Convective heat transfer
- Radiative heat transfer
- Shortwave heat gains (incl. shading)
- Internal heat gains (occupants, lighting)



THERMAL SYSTEMS SIMULATION

KU LEUVEN

Advanced Modelica concepts – ‘extend’

- Imports all equations from the extended model
- Allows modifications/extensions on top of that model

Valve 1

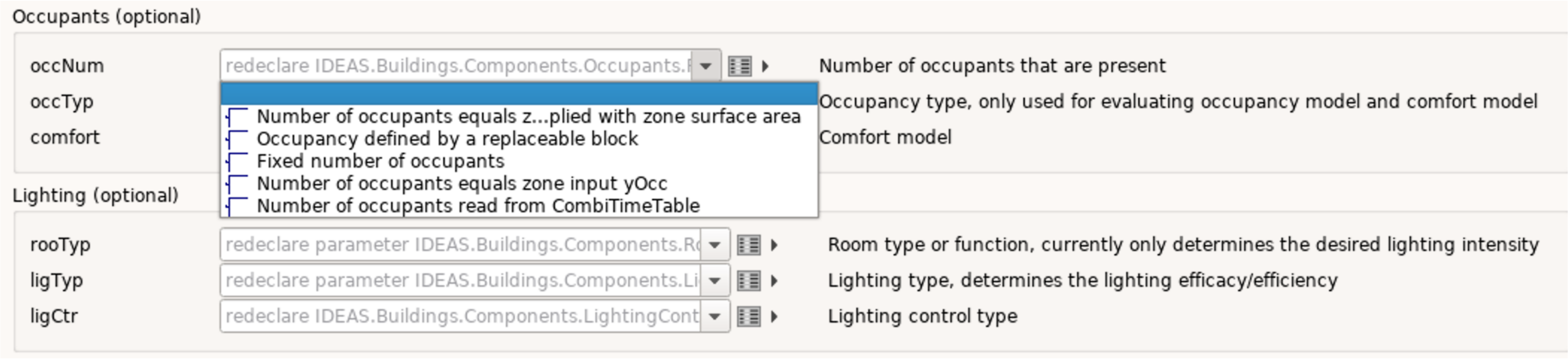
```
model TwoWayLinear "Two way valve with linear flow characteristics"
  extends BaseClasses.PartialTwoWayValveKv(phi=l + y_actual*(1 - l));
```

Valve 2

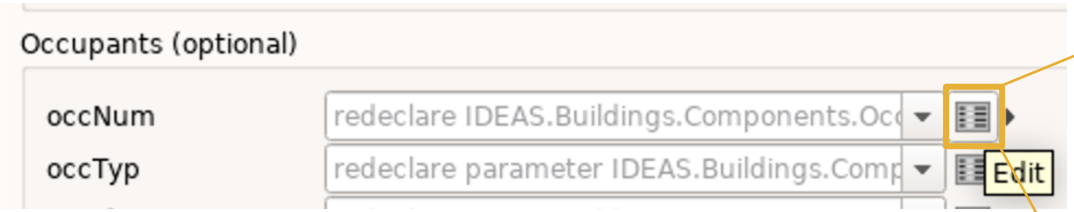
```
model TwoWayPolynomial "Two way valve with polynomial characteristic"
  extends IDEAS.Fluid.Actuators.BaseClasses.PartialTwoWayValveKv(
    phi=l + pol_y*(1 - l));

  parameter Real[:] c
    "Polynomial coefficients, starting with fixed offset";

protected
  constant Integer nP = 100
    "Number of points for initial algorithm test";
  Real pol_y = sum(c.*{y_actual^i for i in 0:size(c,1)-1})
    "Polynomial of valve control signal";
```

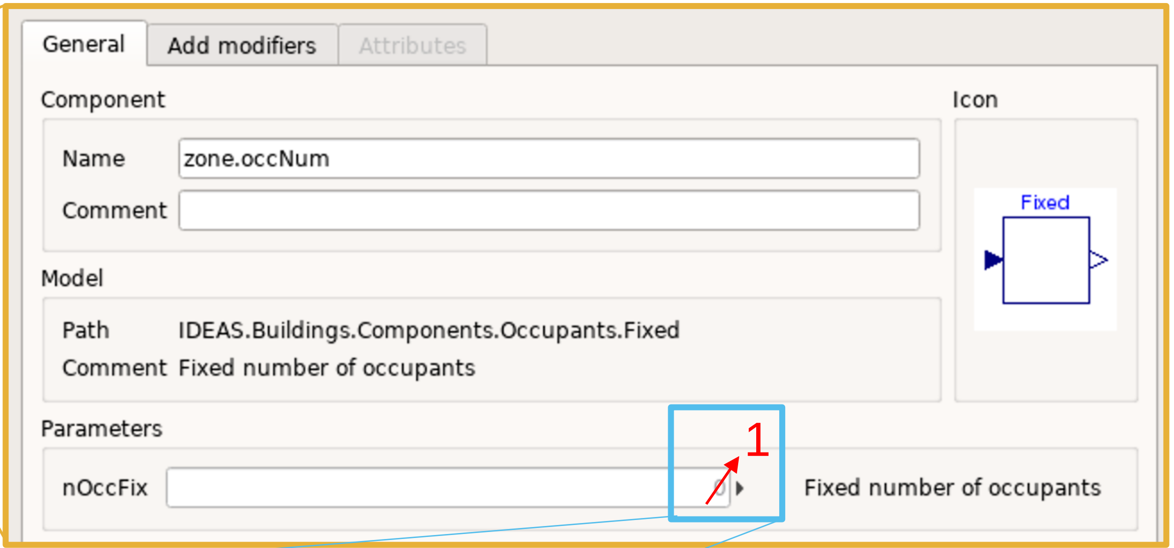
Advanced Modelica concepts – ‘redeclare’

- ‘replaceable’: a component whose type can be changed
- ‘constrainedby’: specify a constraining type of a ‘replaceable’
- ‘redeclare’: changing the type of a replaceable component
- Example in IDEAS: zone model



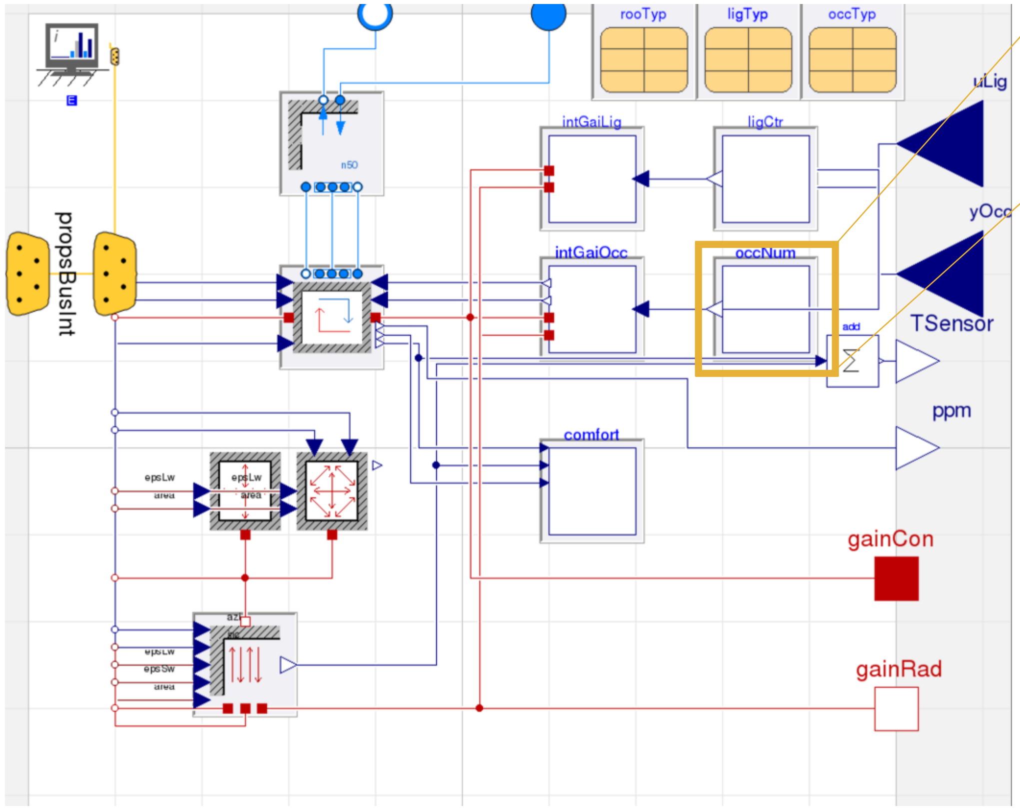
Advanced Modelica concepts – ‘redeclare’






```
model Example3 "Adding occupant and lighting"
  extends Example2(zone(
      redeclare replaceable Buildings.Components.Occupants.Fixed occNum(nOccFix=1),
      redeclare Buildings.Components.OccupancyType.OfficeWork occTyp,
      redeclare Buildings.Components.RoomType.Office rooTyp,
      redeclare Buildings.Components.LightingType.LED ligTyp,
      redeclare Buildings.Components.LightingControl.OccupancyBased ligCtr));
end Example3;
```

Advanced Modelica concepts – ‘redeclare’

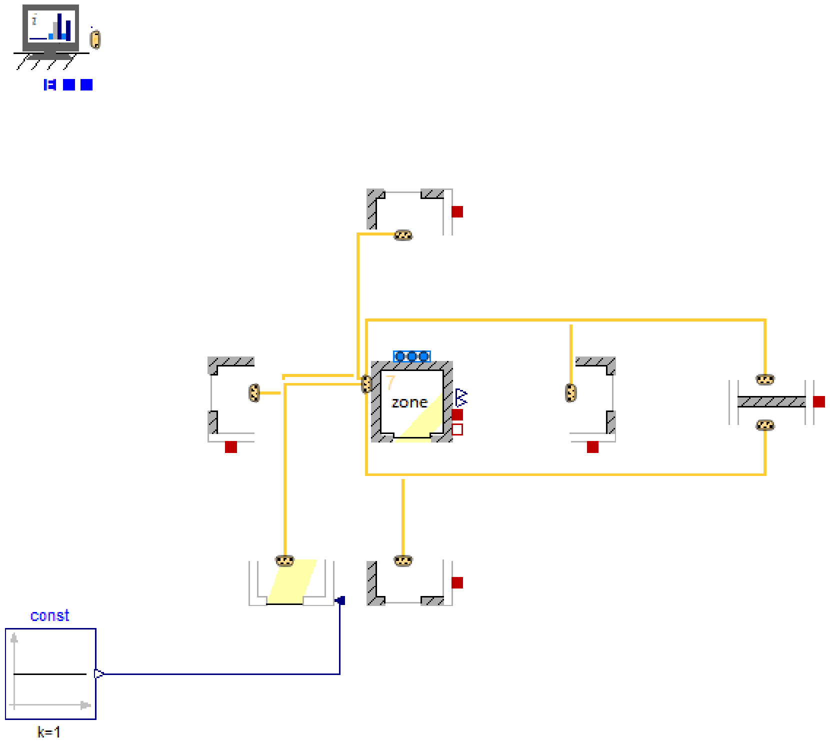



```
replaceable IDEAS.Buildings.Components.Occupants.Fixed occNum
  constrainedby Occupants.BaseClasses.PartialOccupants(
    final A=A,
    final linearise = sim.lineariseDymola)
  "Number of occupants that are present" ə ;
```

```
partial block PartialOccupants "Partial for defining the number of occupants"
  extends Modelica.Blocks.Icons.Block;
  parameter Boolean useInput= false
    "=true to use external input";
  parameter Boolean linearise
    "For linearisation checks";
  parameter Modelica.SIunits.Area A
    "Zone surface area";
  Modelica.Blocks.Interfaces.RealOutput nOcc "Number of occupants"
    ə ;
  Modelica.Blocks.Interfaces.RealInput yOcc if   useInput
    "Input for number of occupants"
    ə ;
  ə ;
end PartialOccupants;
```

Exercise 3 – Building envelope model

- See exercise sheet on Github
  https://github.com/open-ideas/__CrashCourse__/blob/master/Exercises/Exercise%203/Latex/Exercise3.pdf



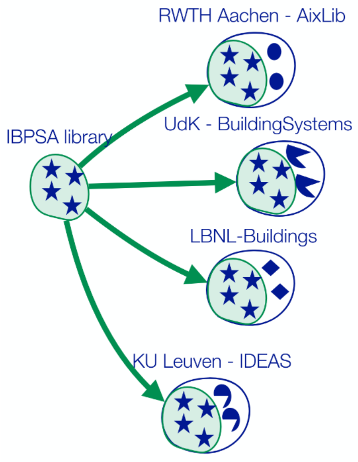
Part 4: IDEAS – HVAC

Lucas Verleyen

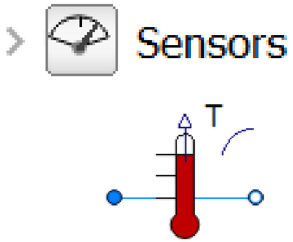
IDEAS – HVAC overview

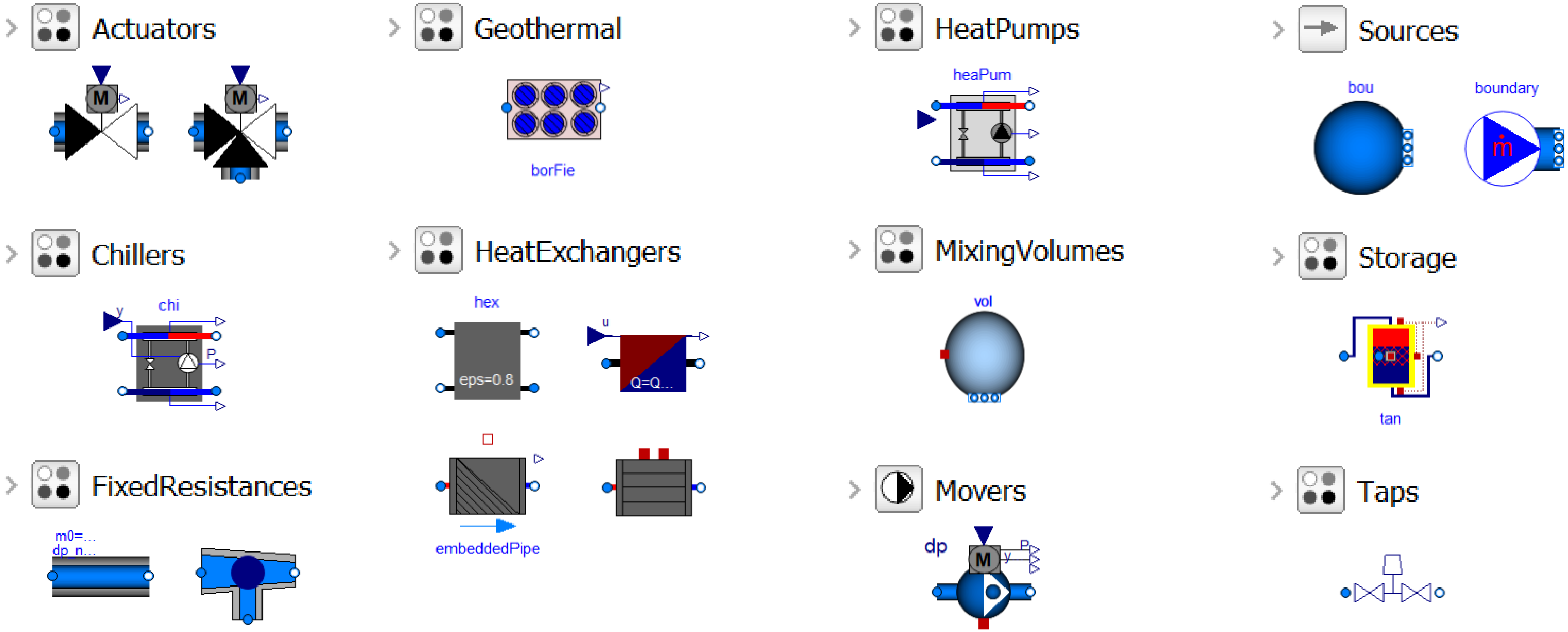
- IDEAS
  - User's Guide
  - Airflow
  - BoundaryConditions
  - Buildings
  - Controls
  - Fluid
  - LIDEAS
  - Media
  - Templates
  - ThermalZones
  - Utilities
  - Types
  - Examples
  - Experimental

- Fluid
  - UsersGuide
  - Actuators
  - Chillers
  - Delays
  - FMI
  - FixedResistances
  - Geothermal
  - HeatExchangers
  - HeatPumps
  - Humidifiers
  - MassExchangers
  - MixingVolumes
  - Movers
  - Sensors
  - Sources
  - Storage
  - Taps
  - Types
  - Examples
  - Interfaces
  - BaseClasses



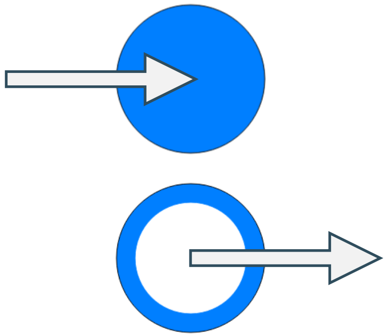
IDEAS – HVAC overview





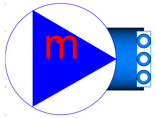
Modelica.Fluid.Interfaces.FluidPort

- Potential variable: pressure → unique value
- Flow variable: mass flow rate → Σ = 0
- Stream variable: enthalpy → characteristic of flow



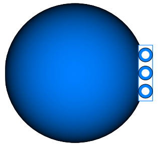
```
connector FluidPort
  "Interface for quasi one-dimensional fluid flow in a piping network (incompressible or comp

  replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
    "Medium model" ;

  flow Medium.MassFlowRate m_flow
    "Mass flow rate from the connection point into the component";
  Medium.AbsolutePressure p "Thermodynamic pressure in the connection point";
  stream Medium.SpecificEnthalpy h_outflow
    "Specific thermodynamic enthalpy close to the connection point if m_flow < 0";
  stream Medium.MassFraction Xi_outflow[Medium.nXi]
    "Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0";
  stream Medium.ExtraProperty C_outflow[Medium.nC]
    "Properties c_i/m close to the connection point if m_flow < 0";
end FluidPort;
```

Basic circuit



IDEAS.Fluid.Sources.MassFlowSource

→ Sets mass flow rate

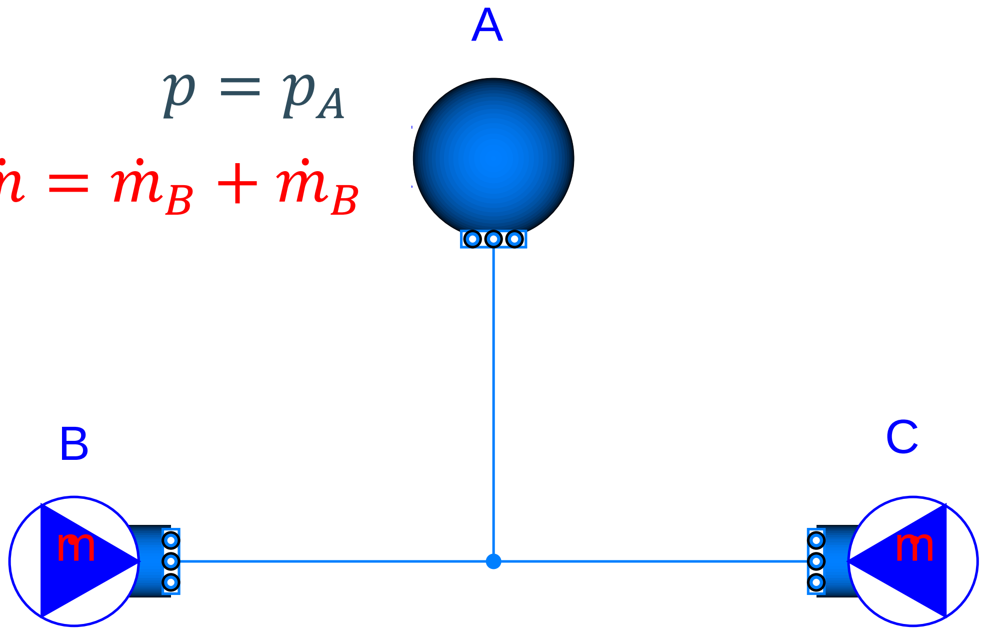


IDEAS.Fluid.Sources.Boundary_pT

→ Sets absolute pressure

$p = p_A$

$\dot{m} = \dot{m}_B + \dot{m}_B$



$p = p_A$

$\dot{m} = \dot{m}_B$

$p = p_A$

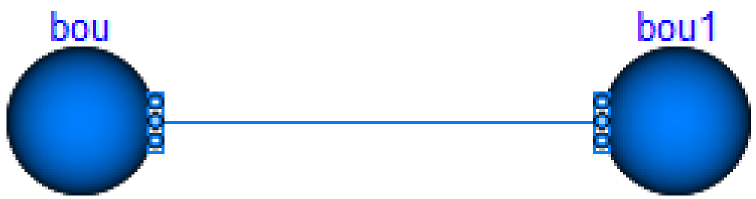
$\dot{m} = \dot{m}_C$

Pressure equals pressure set by Boundary A

Flow rate equals sum of flow rates set by Source B and Source C

Illegal circuits

The problem is structurally singular

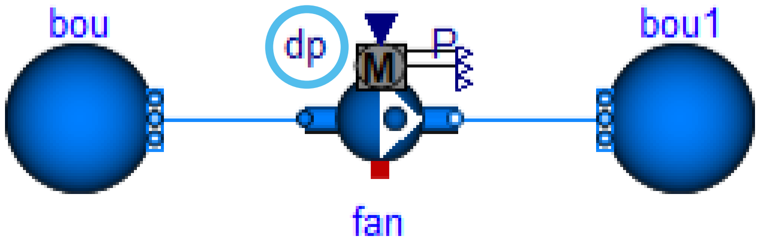


No mass flow rate

& problem when $p_{bou} \neq p_{bou1}$

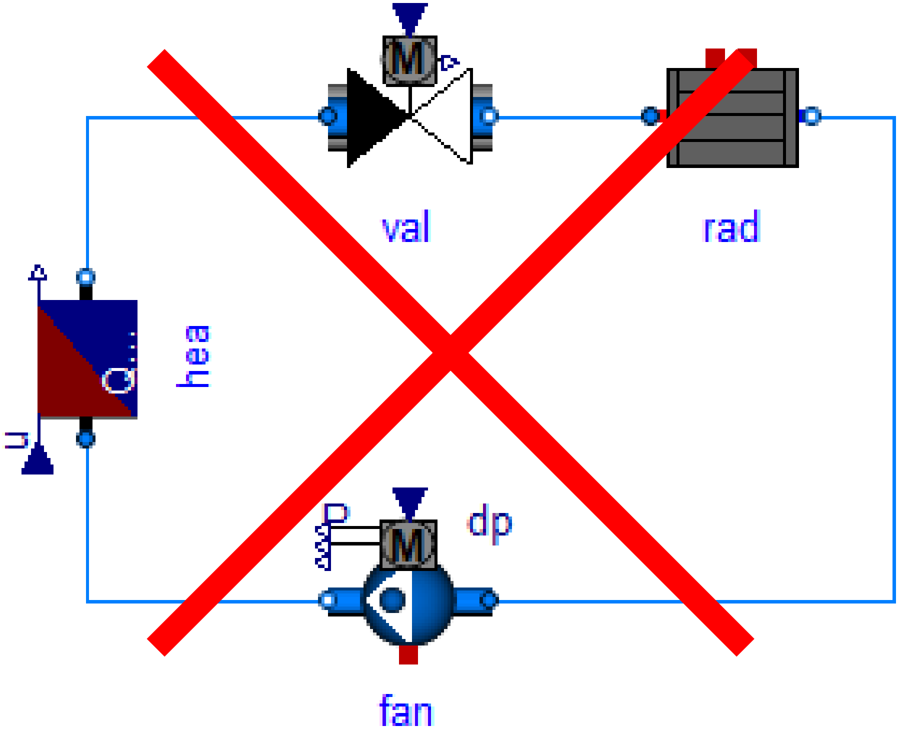
No absolute pressure

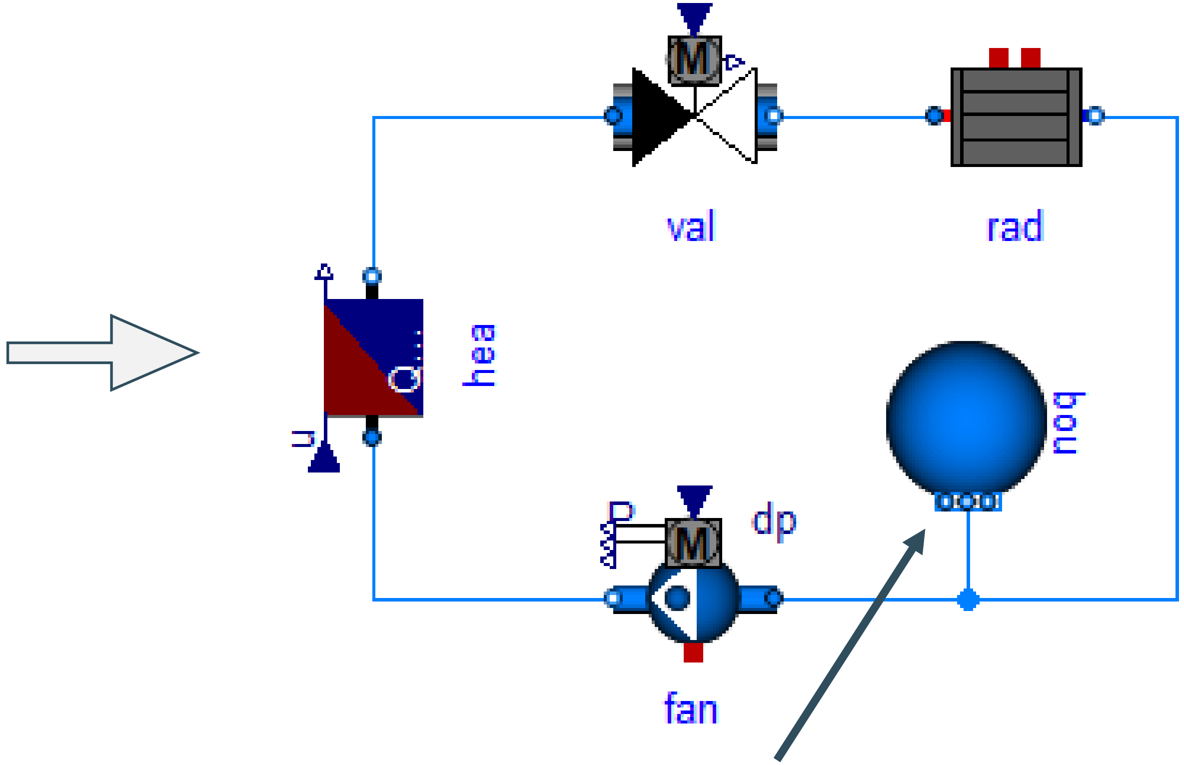
& problem when $\dot{m}_{boundary} \neq -\dot{m}_{boundary1}$



Problem when $p_{bou} + \Delta p_{fan} \neq p_{bou1}$

Simple HVAC circuit





Only pressure drops

Sets absolute pressure

Singularity error

- Modelica is a generic modelling language → errors are general (unclear)

  i.e. not tailored to buildings / HVAC

- Important language requirements:
  - # equations = # variables
  - Equations should not contradict each other (e.g. adding both $x = 1$ and $x = 0$)
  - Solution to equations should exist (e.g. don't try $x^2 = -1$)
- These requirements are abstract and are hidden from the user by library developers. However, they pop up sometimes, leading to unclear errors like:

The model IDEAS.Examples.PPD12.Heating is structurally singular.

The problem is structurally singular for the element type Real.

The number of scalar Real unknown elements are 4114.
The number of scalar Real equation elements are 4114.

Singularity error

The model IDEAS.Examples.PPD12.Heating is structurally singular.
The problem is structurally singular for the element type Real.
The number of scalar Real unknown elements are 4114.
The number of scalar Real equation elements are 4114.

- You don't have to worry about component models when dragging and dropping
- Singularity error occurs, when:
  - Dangling connectors / more than 1 connection
  - # equations ≠ # variables
  - Conflicting equations, equations without real solutions
  - Infinite number of solutions (e.g. no absolute pressure set)
- IDEAS.Buildings is fairly robust as long as each zone propsBus connector is connected to exactly one surface propsBus connector.
- IDEAS.Fluid pressure drop circuits can require some experience:
  - Set absolute pressure in flow circuits
  - Don't oversimplify pressure drops

Further reading

- F. Jorissen, G. Reynders, R. Baetens, D. Picard, D. Saelens, and L. Helsen. Implementation and Verification of the IDEAS Building Energy Simulation Library. *Journal of Building Performance Simulation*, **11** (6), 669-688, 2018. doi: 10.1080/19401493.2018.1428361.
- F. Jorissen, M. Wetter, and L. Helsen. Simulation Speed Analysis and Improvements of Modelica Models for Building Energy Simulation. In 11th International Modelica Conference, pages 59–69, Paris, 2015. doi: 10.3384/ecp1511859.
- F. Jorissen, M. Wetter, and L. Helsen. Simplifications for Hydronic System Models in Modelica. *Journal of Building Performance Simulation*, **11** (6), 639-654, 2019.
- F. Jorissen. *Toolchain for Optimal Control and Design of Energy Systems in Buildings*. PhD thesis, Arenberg Doctoral School, KU Leuven, April 2018

Exercise 4 – HVAC model

- See exercise sheet on Github
  https://github.com/open-ideas/__CrashCourse__/blob/master/Exercises/Exercise%204/Latex/Exercise4.pdf
- Start from building envelope of step 5 (*see IDEAS.Examples.Tutorial*):
  1. Add a geothermal heat pump heating system
  2. Add a heat pump controller
  3. Compute the energy use and export it in a json file
  4. Add a CO_2-controlled ventilation system