A brief introduction to Modelica

Michael Wetter, LBNL, Berkeley, CA

October 17, 2016

Lawrence Berkeley National Laboratory

Purpose and approach

The purpose is to

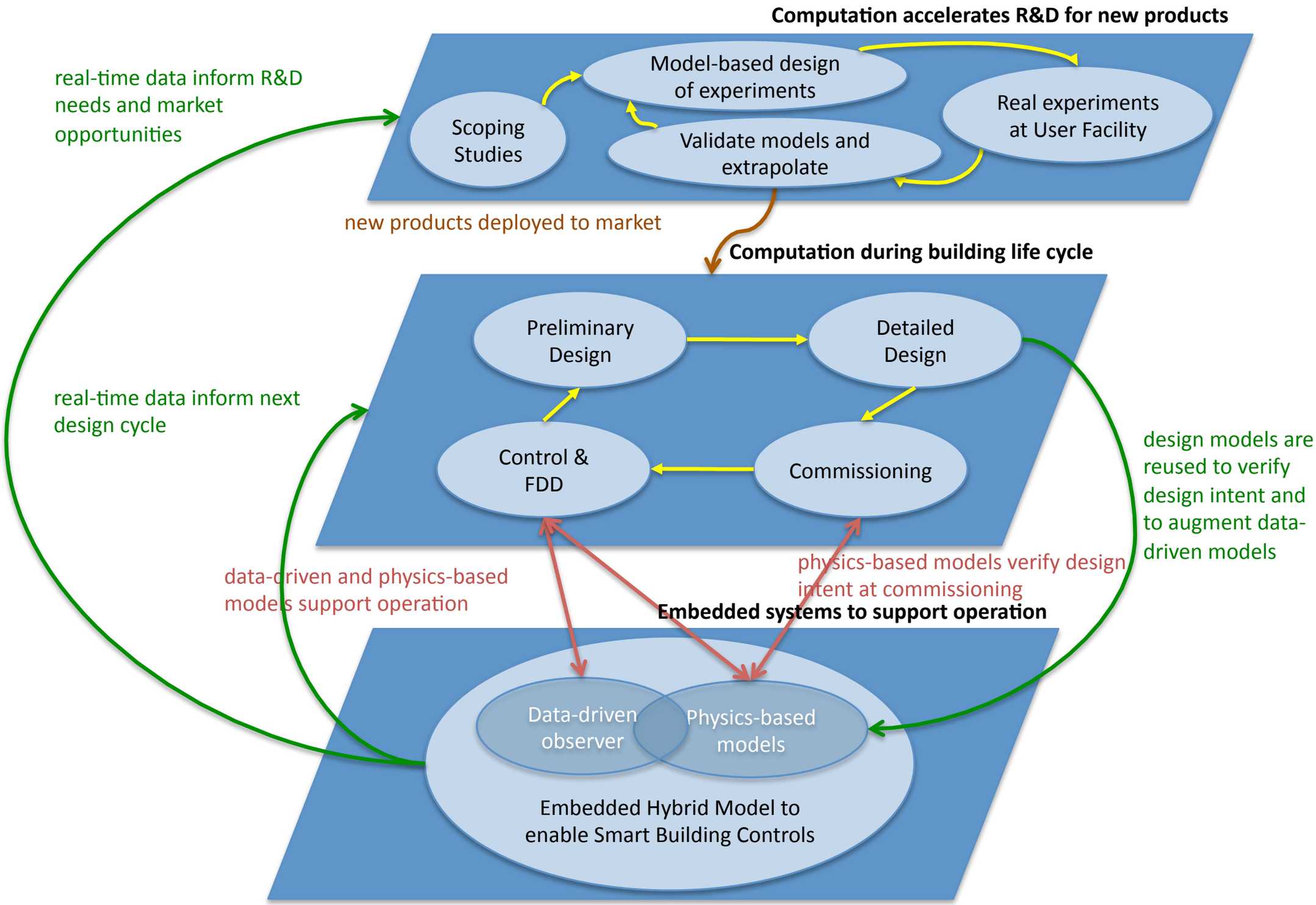
- introduce Modelica
- explain how it differs from other building simulators
- present tool ecosystem

(Applications will be shown afterwards.)

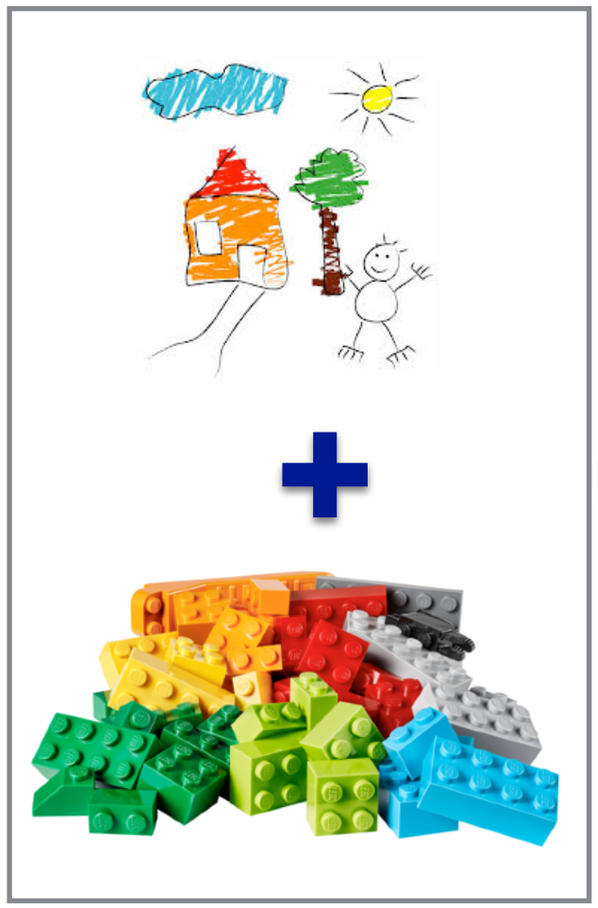
Background material includes:

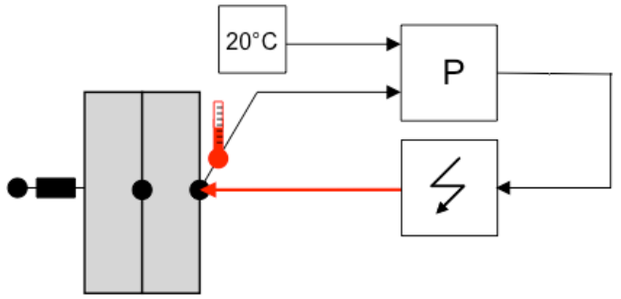
- The online book from Michael Tiller:
  http://book.xogeny.com
- Modelica reference: http://modref.xogeny.com/
- Interactive tour: http://tour.xogeny.com
- Other references:
  http://simulationresearch.lbl.gov/modelica/userGuide/gettingStarted.html

Simulation has applicability beyond building design

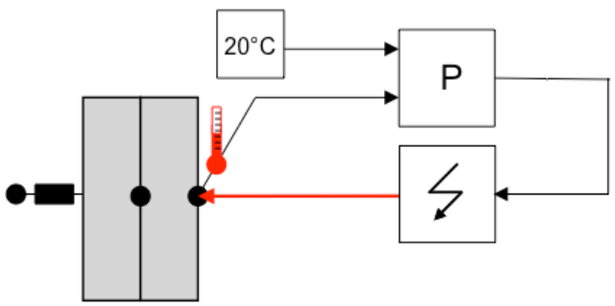


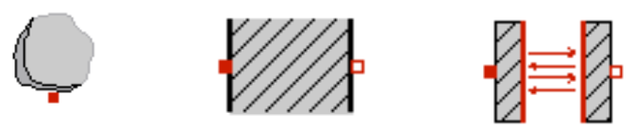
Modeling vs. simulation

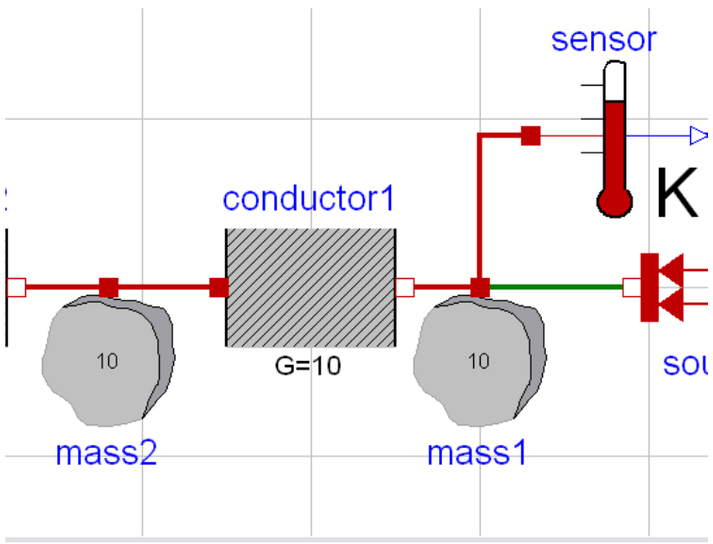


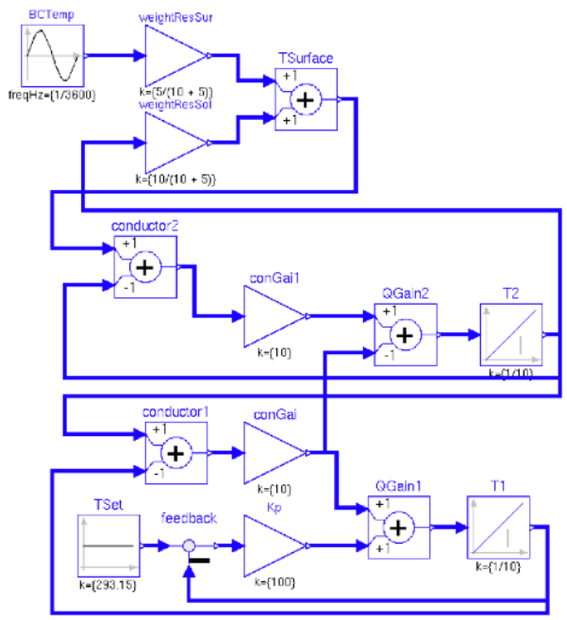



```
integer :: i                                    ! Loop counter
integer, parameter :: lun = 6                   ! logical unit number

double precision, parameter :: amp = 5          ! Amplitude
double precision :: time                        ! Simulation time

character(LEN=*), parameter  :: FMT = "(4F14.7)"

! Initialize variables
dt = tFin/N
T1 = T0
T2 = T0
time = 0
iCom = 1
open (lun, FILE='results.txt')
! Perform integration
do i = 1, N, 1
   TBC = T0 + amp * dsin(2*3.14159*time/3600)
   TSur = T2 * ( G / (G+h) ) + TBC * ( h / (G+h) )
   QSou = Kp * (TSet - T1)
   QCon1 = G * (T2 - T1)
```

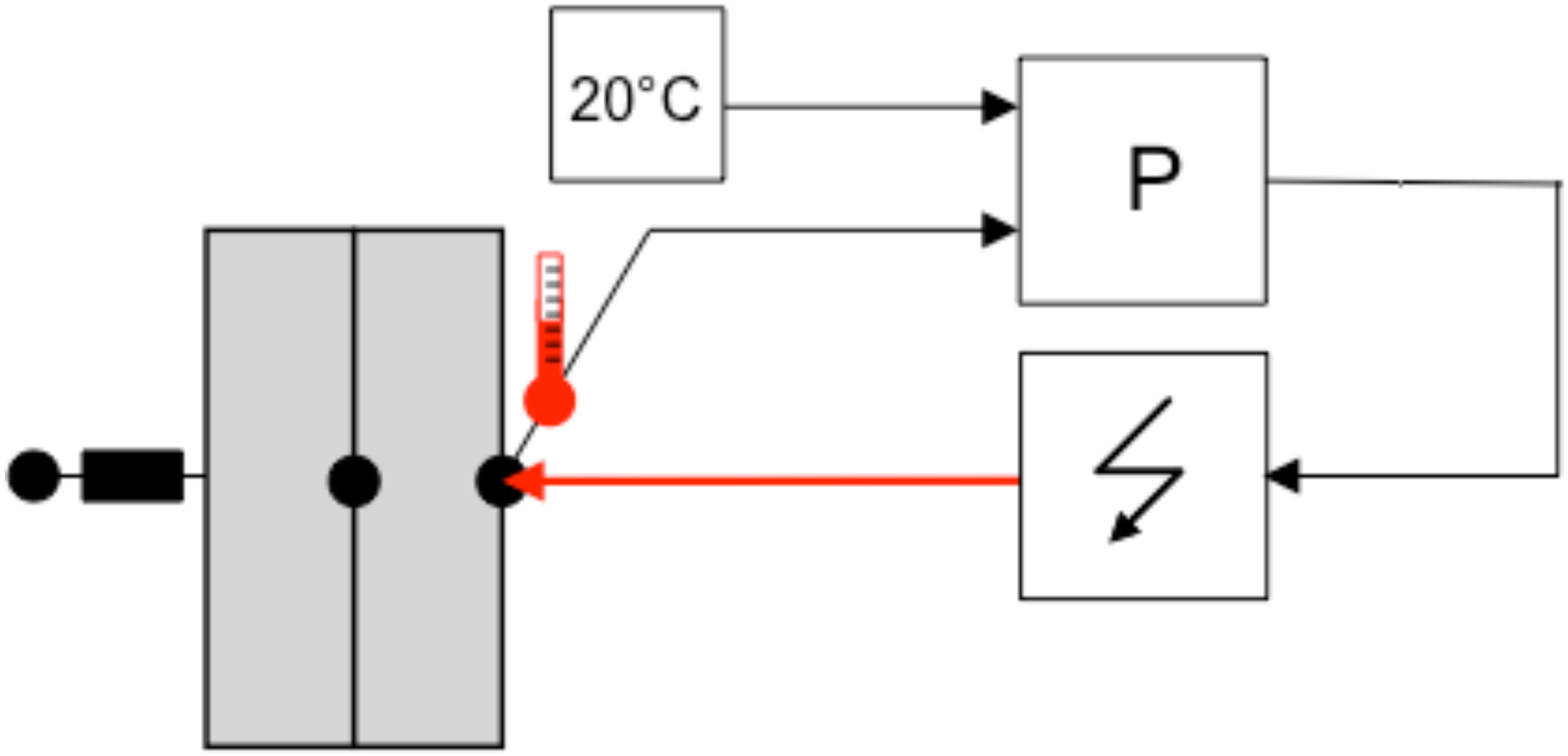








Small demo



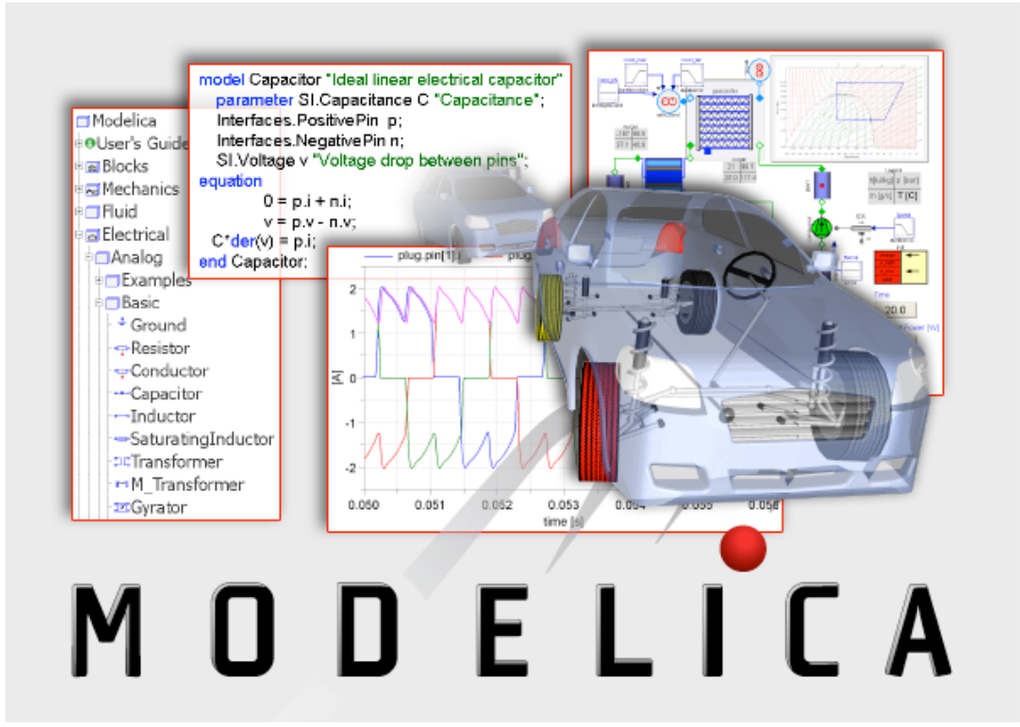
What is Modelica?

Modelica, an open standard that specifies an equation-based, object-oriented modeling language

A modeling, not a programming, language.

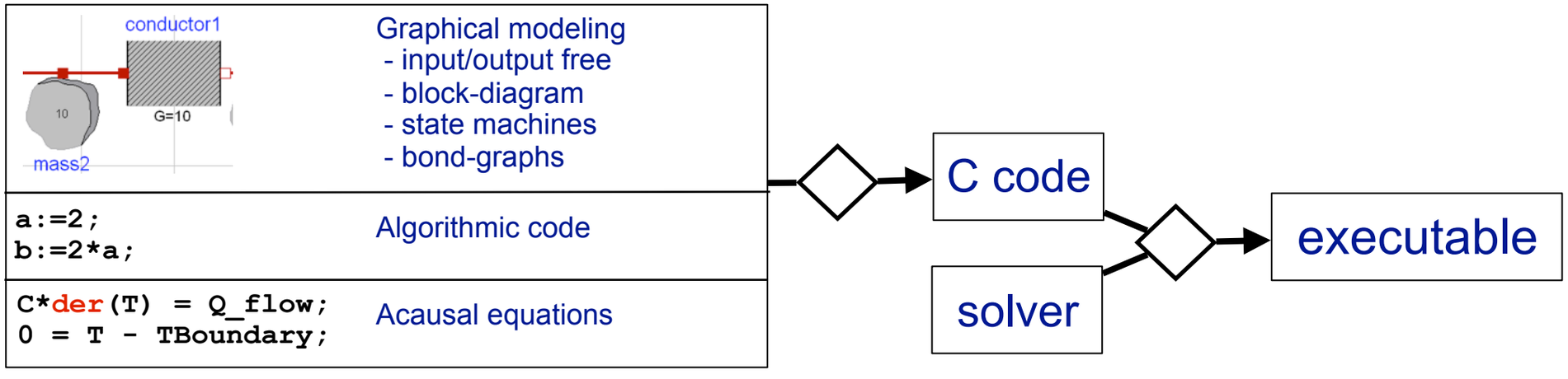
Open, industry-driven standard for modeling multi-physics, engineered systems

Developed since 1996 because conventional approach for modeling was inadequate for integrated engineered systems



Large number of free and commercial libraries, https://modelica.org/libraries

Needs a tool to translate models to an executable.



Separation of equation, solvers, and data input/output yields readable and maintainable models with high reusability

```
model HeatCapacitor "Lumped thermal element storing heat"
```

declare

```
  parameter Modelica.SIunits.HeatCapacity C
     "Heat capacity of element (= cp*m)";
  Interfaces.HeatPort_a port
     "Port with (Q_flow, T)";
```

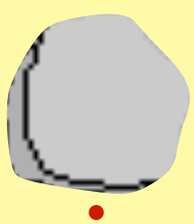
```
equation
```

model

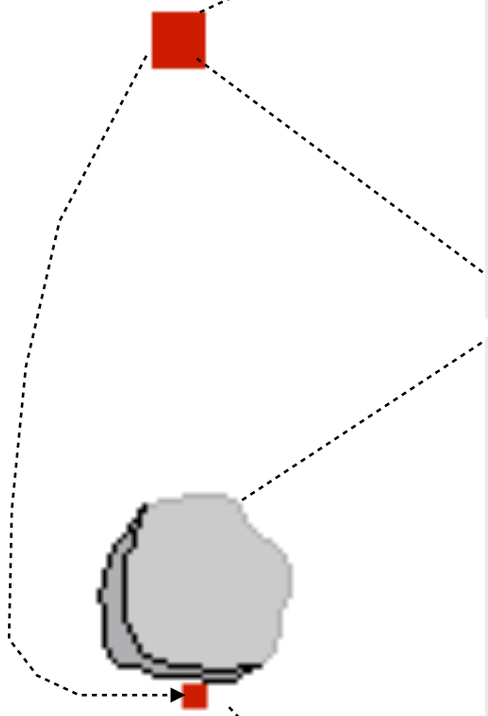
```
  C*der(port.T) = port.Q_flow;
```

```
end HeatCapacitor;
```

iconify

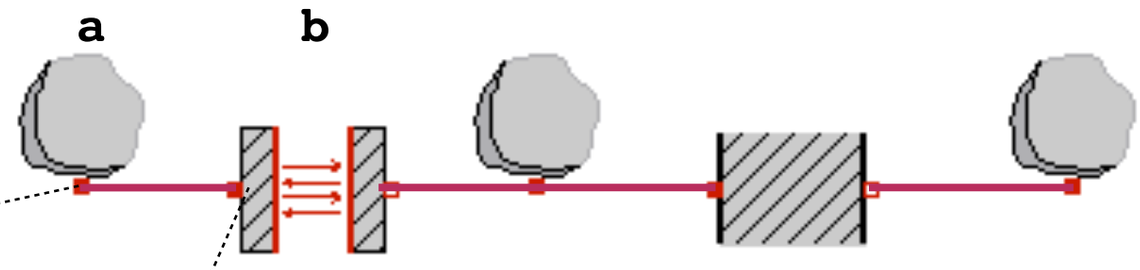


Modularization in object-oriented modeling supports creation of transparent models



```
connector HeatPort_a
  "Thermal port for 1-dim. heat transfer"
  Modelica.SIunits.Temperature T "Port temperature";
  flow Modelica.SIunits.HeatFlowRate Q_flow
    "Heat flow rate (positive if flowing from
     outside into the component)";
end HeatPort_a;
```

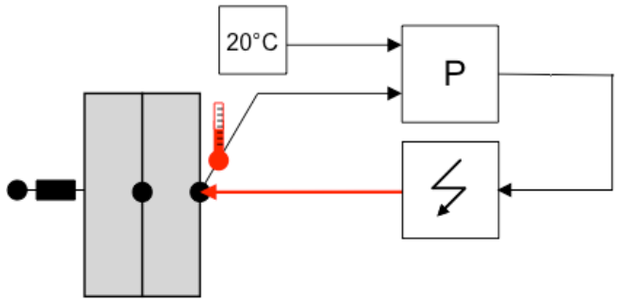
```
model HeatCapacitor "Lumped thermal element storing heat"

  parameter Modelica.SIunits.HeatCapacity C "Heat capacity";
  Modelica.SIunits.Temperature T "Temperature of element";
  Interfaces.HeatPort_a port;

equation
  T = port.T;
  C*der(T) = port.Q_flow;
end HeatCapacitor;
```



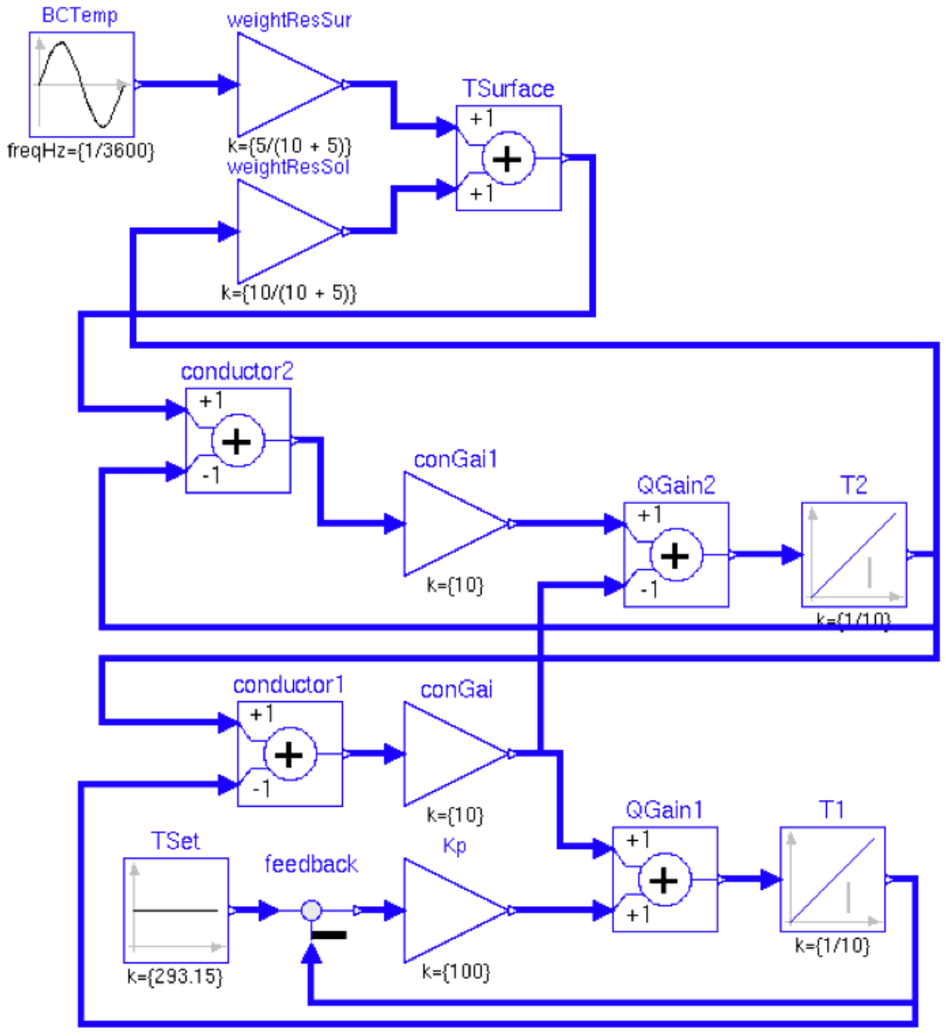

```
connect(a.port, b.port);
```

```
a.port.T = b.port.T;
0 = a.port.Q_flow + b.port.Q_flow;
```

Acausal connectors are used to enable assembling models schematically

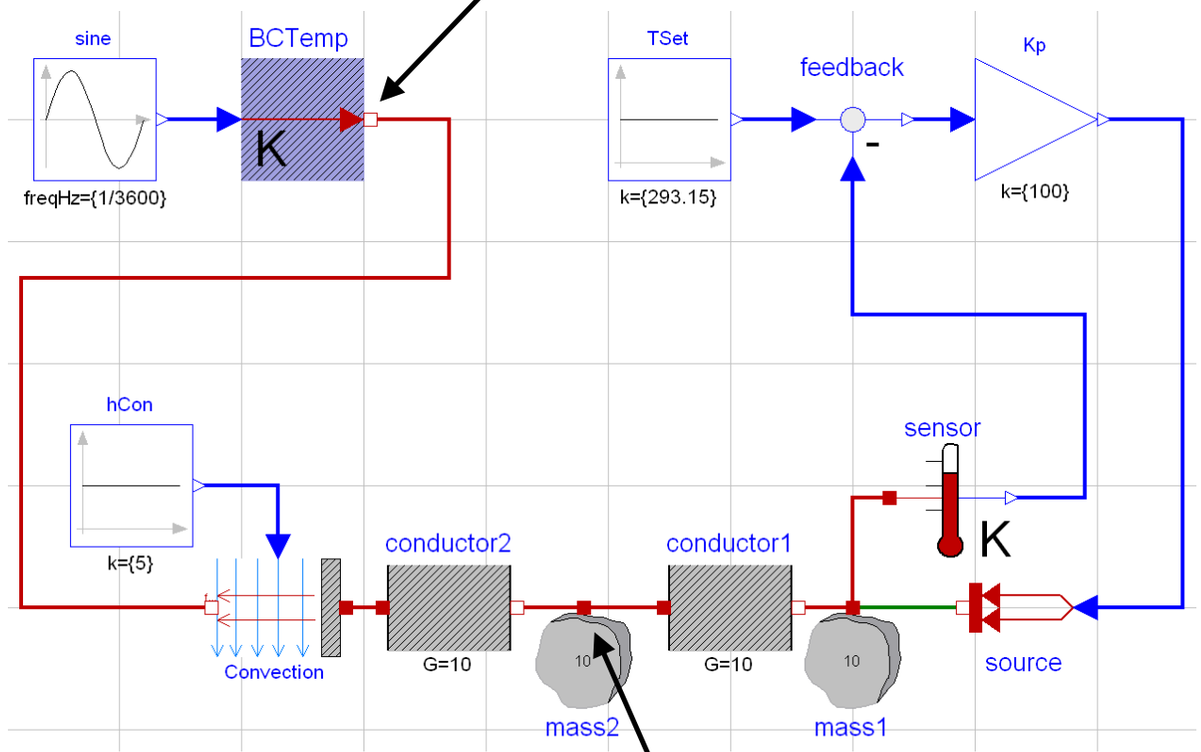


Block Diagram Modeling



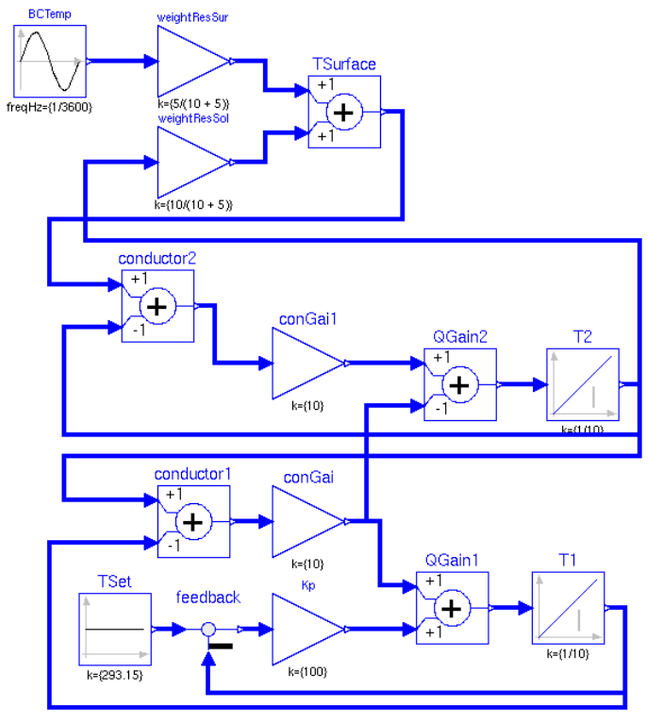
Acausal Modeling

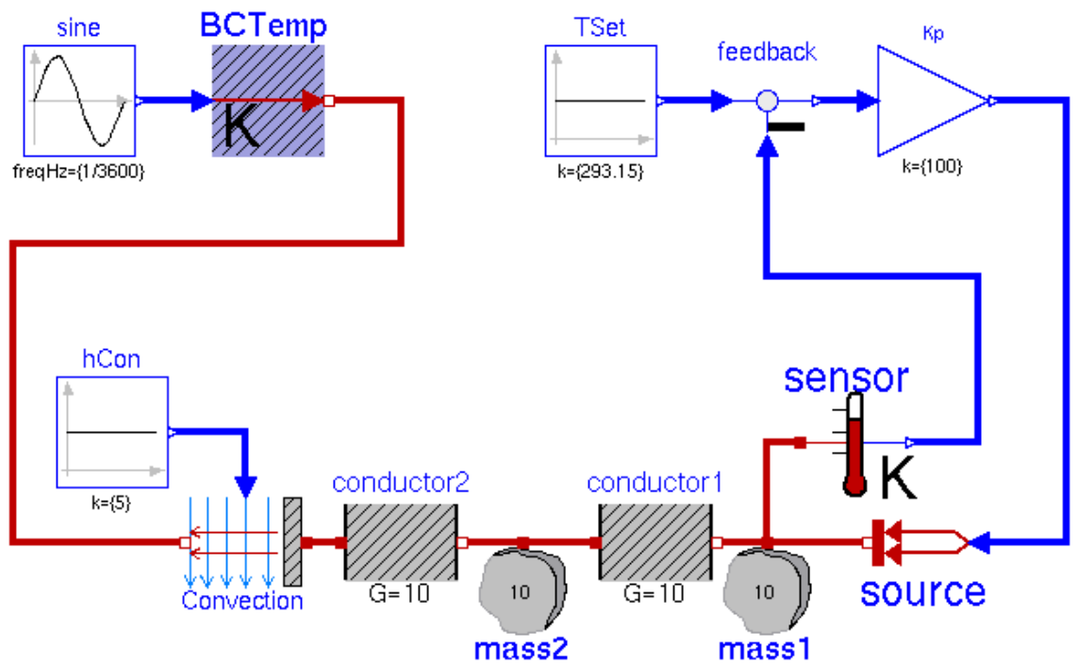
This port carries temperature and heat flow rate. Hence, an electrical pin cannot be connected to it

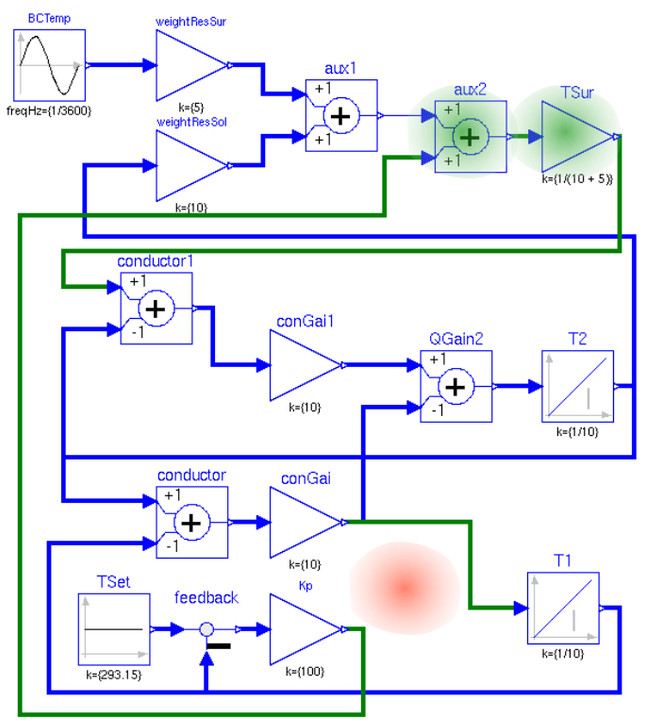


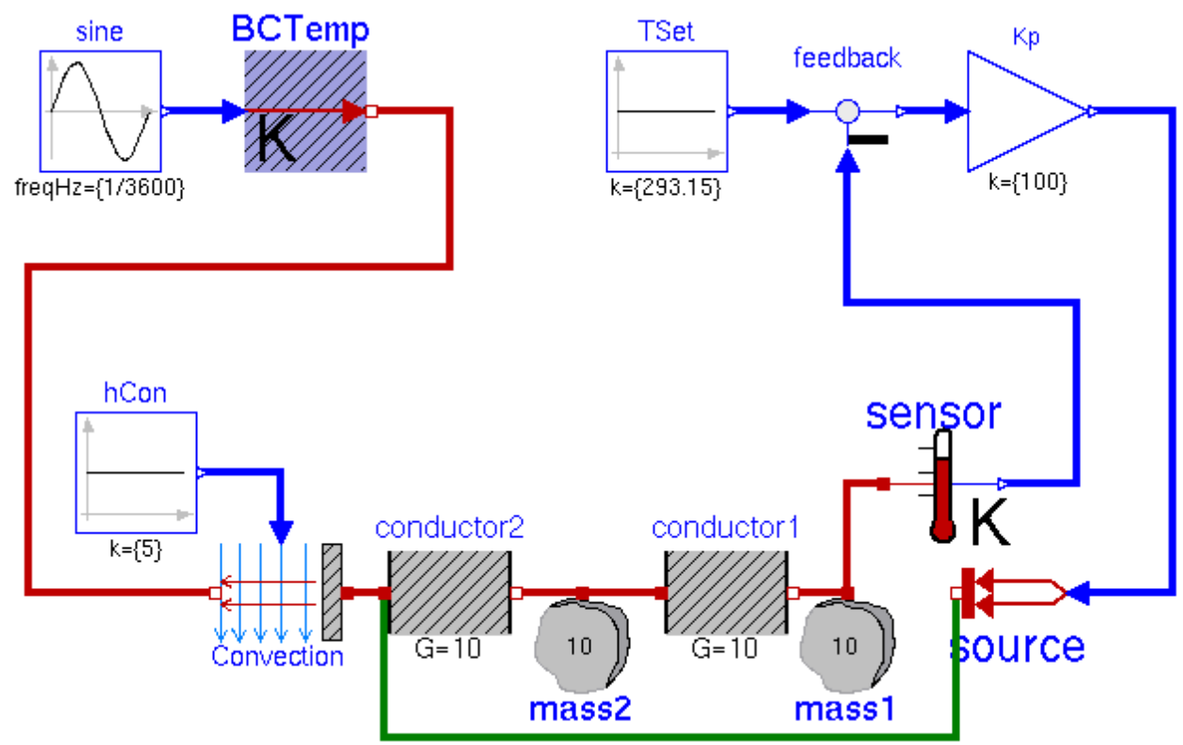
What does it mean to connect three ports?

Acausal components leads to much higher readability and reusability as they allow to declare the system architecture









Physical connectors and balanced models

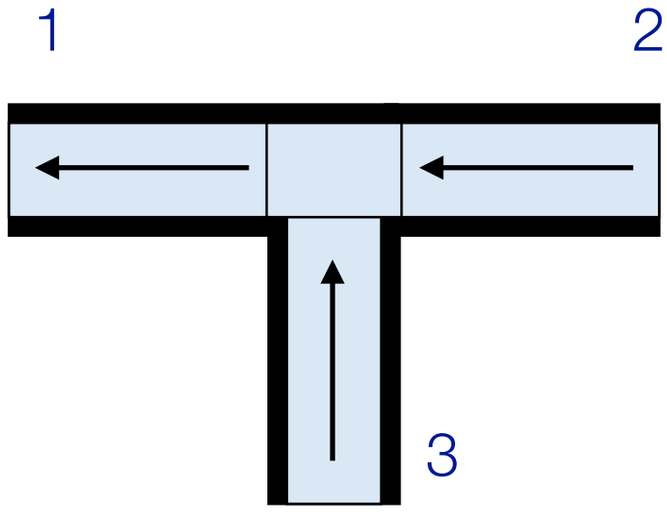
Acausal connectors

- input/output determined at compilation time
- can connect none or multiple components to a port
- A connector should contain all information required to uniquely define the boundary condition

Domain	Potential	Flow	Stream
Heat flow	T	Q_flow	
Fluid flow	p	m_flow	h_outflow X_outflow C_outflow
Electrical	V	I	
Translational	x	F	

Requirement of locally balanced models

- # of equations = # of variables, at each level of model hierarchy.



$$\sum_{i=1}^{3} \dot{m}_i = 0$$

$$p_1 = p_2 = p_3$$

$$h_{1,out} = -\frac{h_{2,in}\,\dot{m}_2 + h_{3,in}\,\dot{m}_3}{\dot{m}_1}$$

Composability

Connectors

- Designed for *physical* compatibility, not causal compatibility
- No *a-priori* knowledge is needed to connect components

Multi-physics

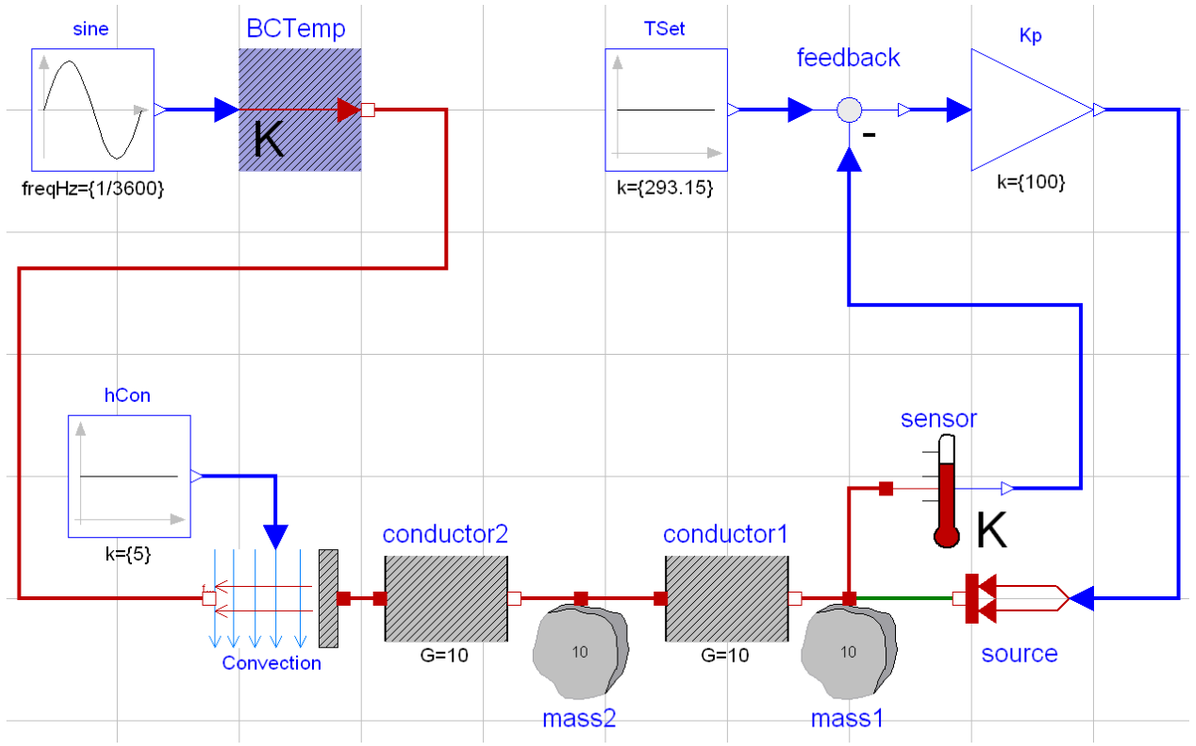
- Components can have a heat port and a fluid port (and a control input signal, …)

Multi-domain

- Can combine schematic diagrams, block diagrams (and state machines, …)

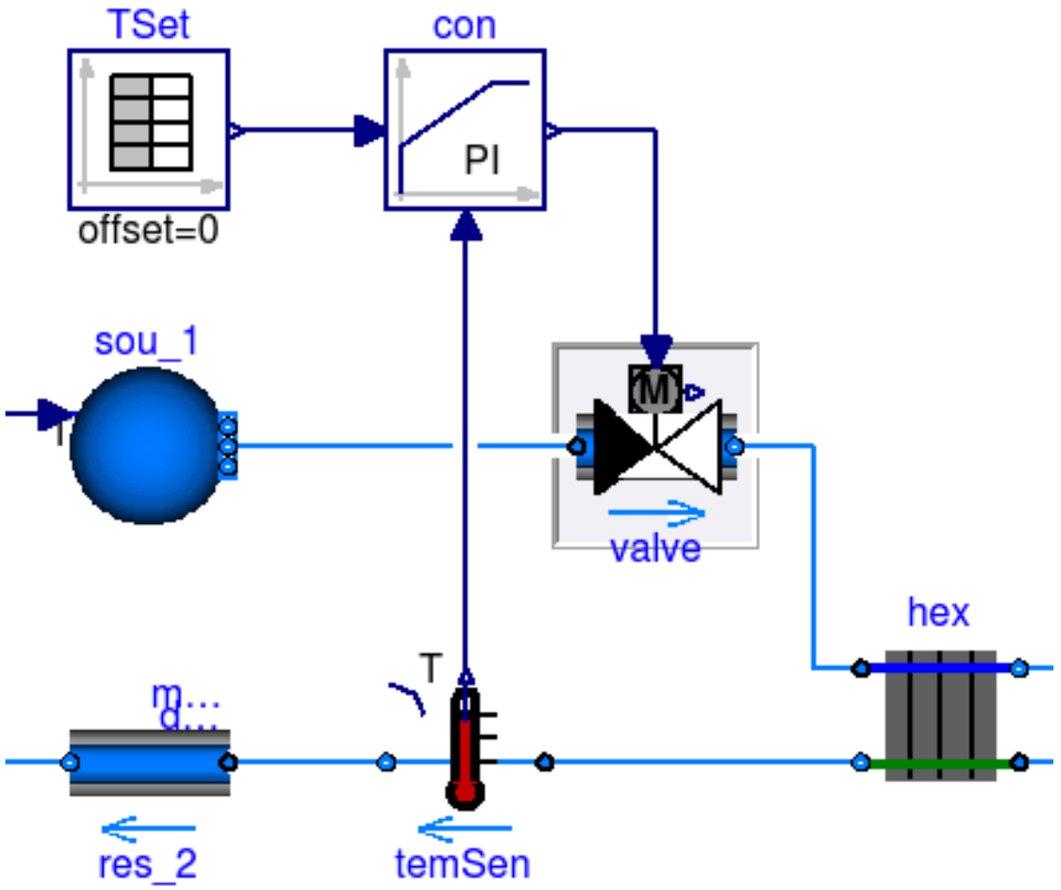
Reusability

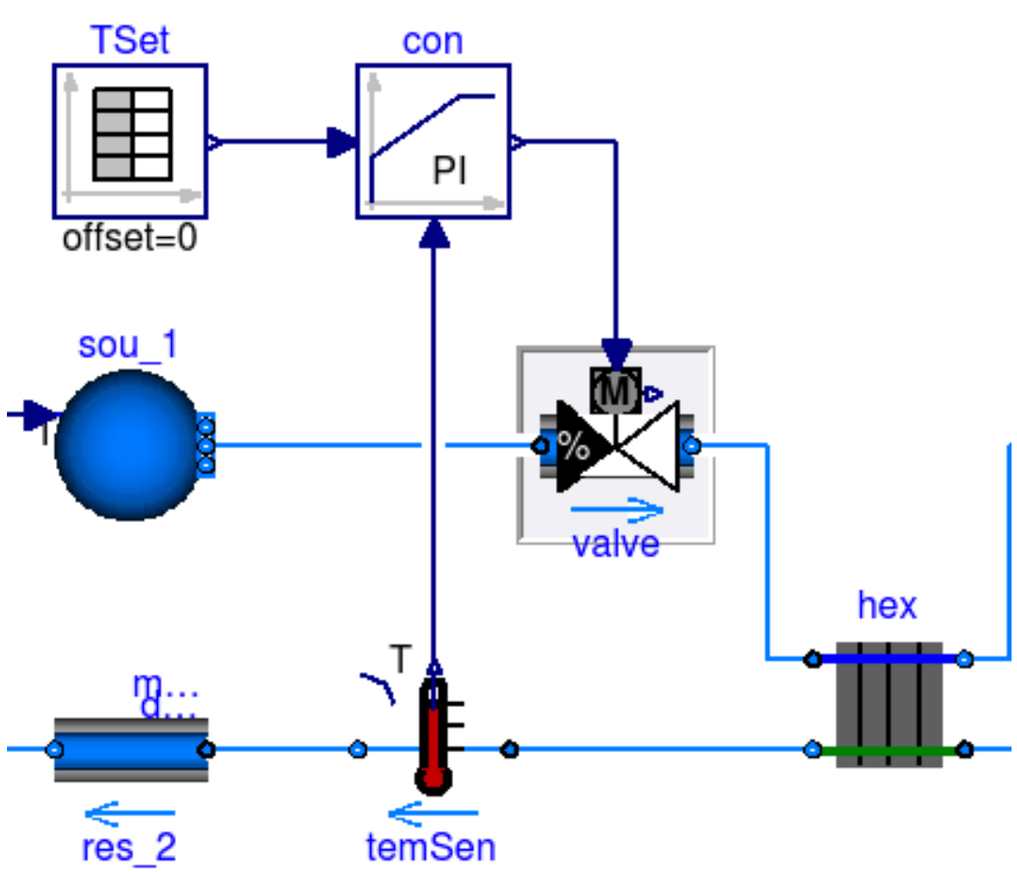
- Change the system architecture by deleting and dragging components from a library (in block diagrams, profound changes would be required)



Architecture-driven modeling

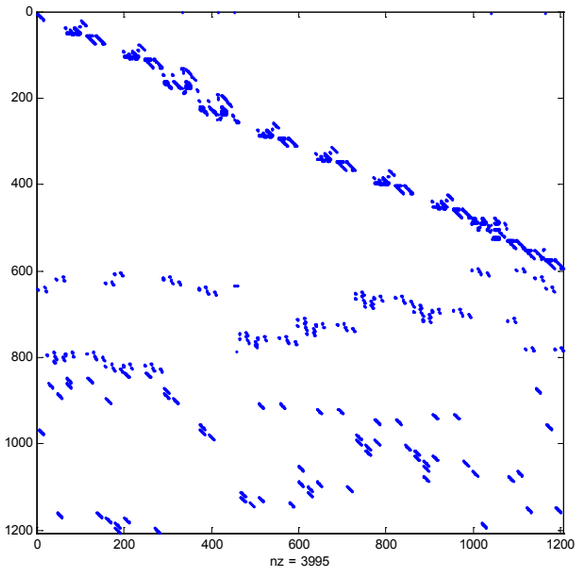
Keep system architecture, but replace “plug-compatible” subsystems



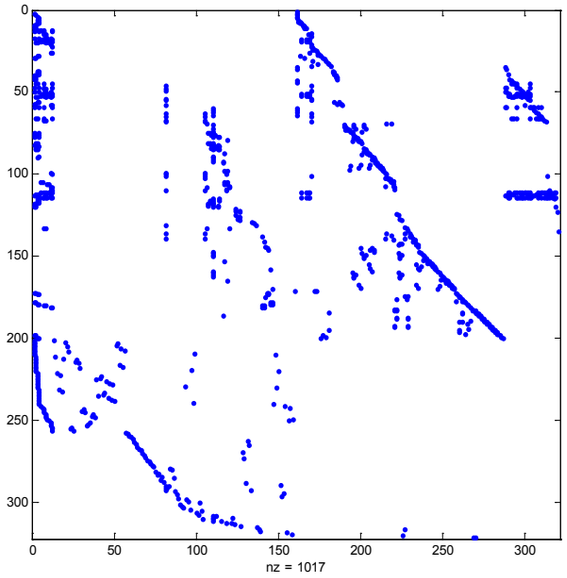


See DataCenterDiscreteTimeControl for a model that uses this construct to change the controls, or http://book.xogeny.com/components/architectures/thermal_control/ for a simple application.

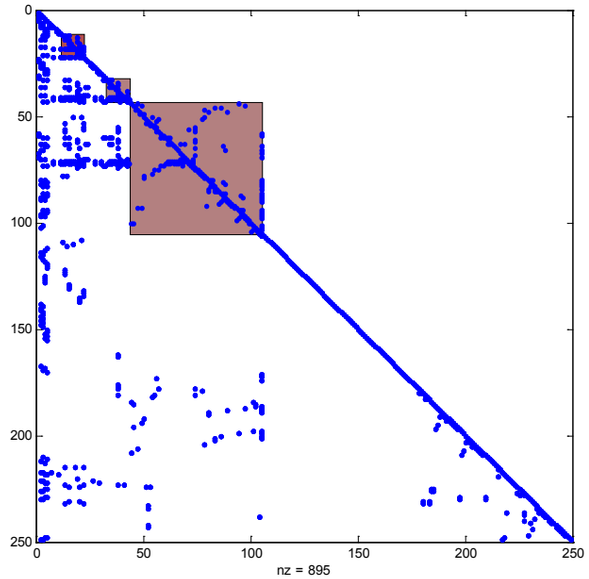
Symbolic manipulations significantly reduce problem size



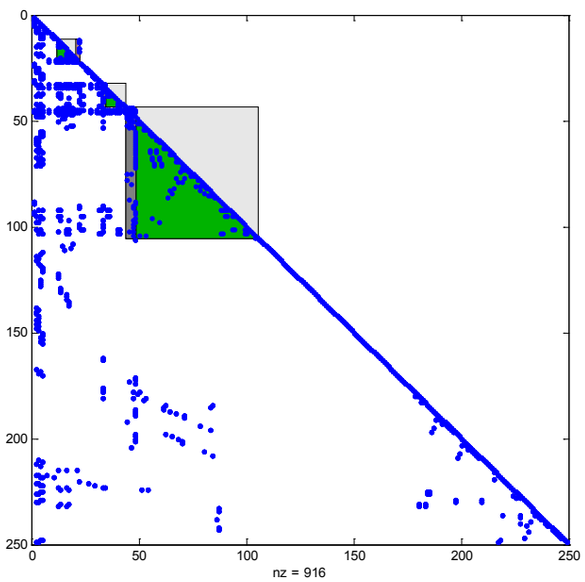
Incidence matrix of the original problem (1200x1200)



Incidence matrix after elimination of alias variables (330x330)



After simplifications and BLT (250x250)



Note: Many numerical algorithms are $O(n^3)$

Figures from Dymola 2016 user manual for mechanical model with kinematic loop.
For description of method, see Cellier and Kofman, Continuous System Simulation, Springer, 2006.

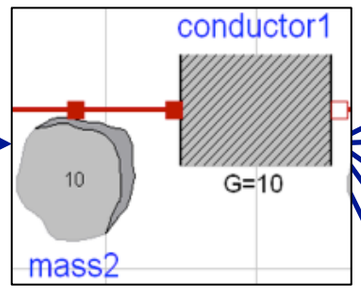
Tearing reduces nonlinear 12 to 2, linear 11 to 2 and linear 57 to 5.

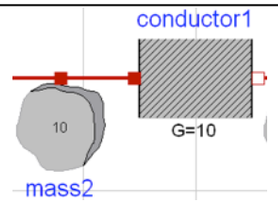
How does it differ from other building simulators?

Separation of concern

Modeling

Specifies the system



	Graphical modeling - input/output free - block-diagram - state machines - bond-graphs
`a:=2;` `b:=2*a;`	Algorithmic code
`C*der(T) = Q_flow;` `0 = T - TBoundary;`	Acausal equations
`external "C"` `y=someCFunction(x);`	C interface

Computation

Solves the equations

Code for time-domain simulation

Code for real-time operation

Limited memory and storage.

Constraints on computing time.

Code for optimization

Differentiation for gradient.

Symbolic processing for collocation.

Code for co-simulation as FMU

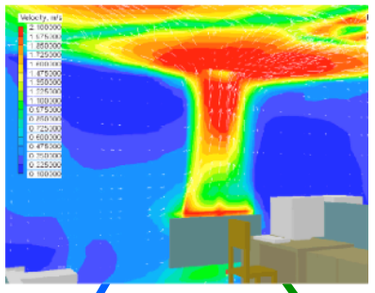
Provide API for model discovery and that returns $x_{k+1}=f(x_k, t_k)$

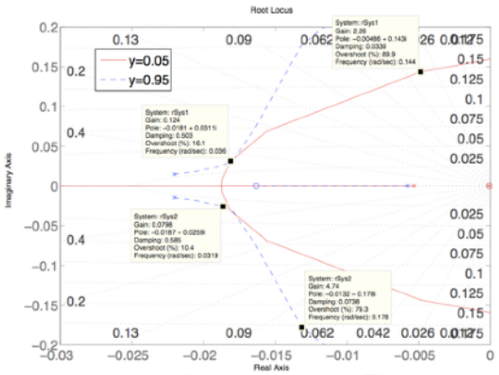
Code for model exchange as FMU

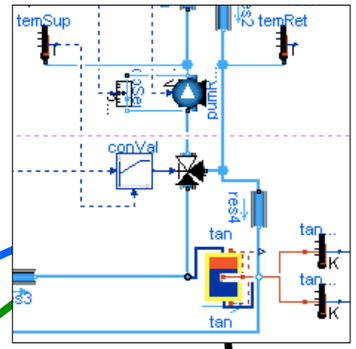
Provide API for model discovery and that expose right-hand-side of $dx(t)/dt=f(x(t),t)$

Related efforts within the building simulation community include ENET (Low and Sowell, 1982), SPANK (Sowell et al., 1986), SPARK (Buhl et al., 1993) and the Neutral Model Format NMF (Sahlin and Powell, 1989).

Context: Models need to be compatible with up-stream data-formats and serve down-stream applications

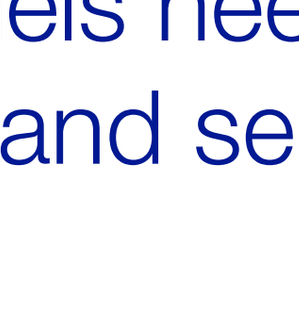




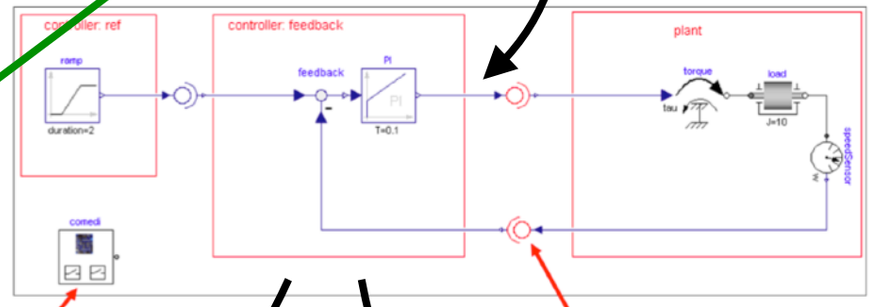


data

dynamics

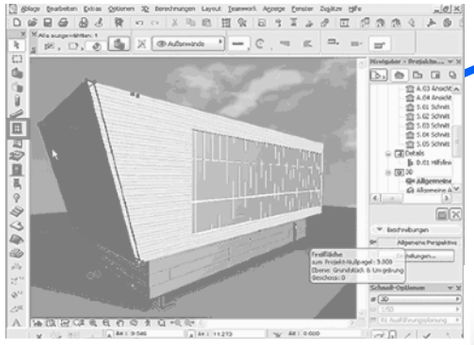


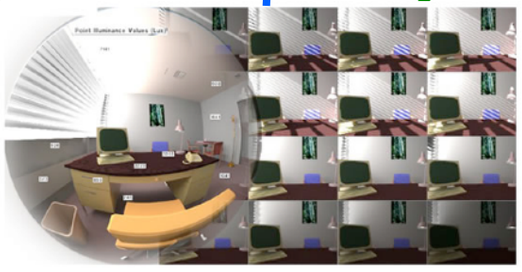
$$\dot{x} = f(x)$$

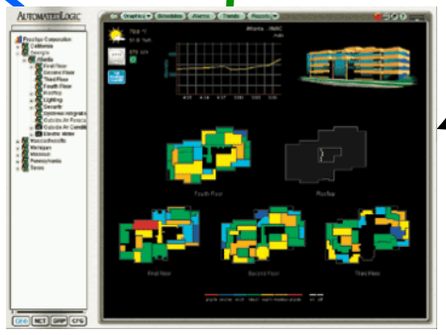


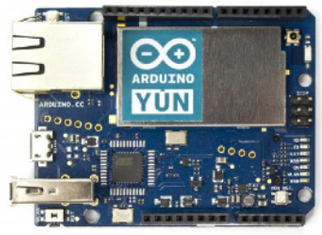
defines configuration of embedded system
(sampling, ECUs, initialization of device drivers, ...)

Defines
- splitting in task/sub-task,
- device drivers,
- references configuration



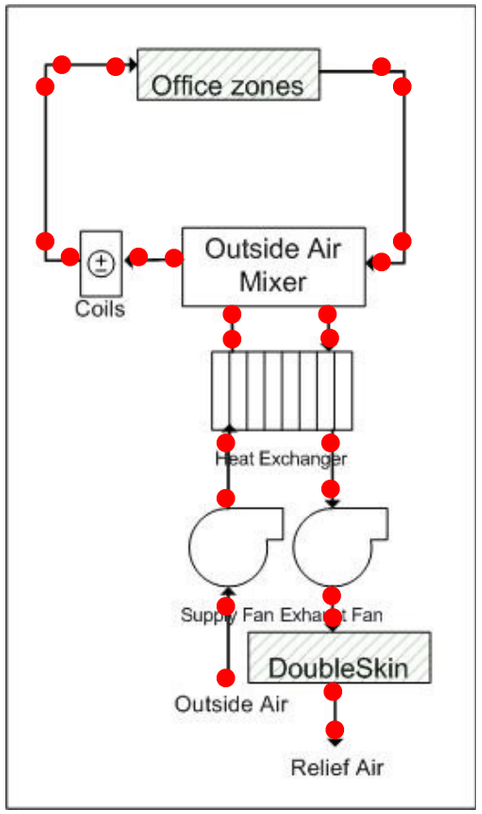






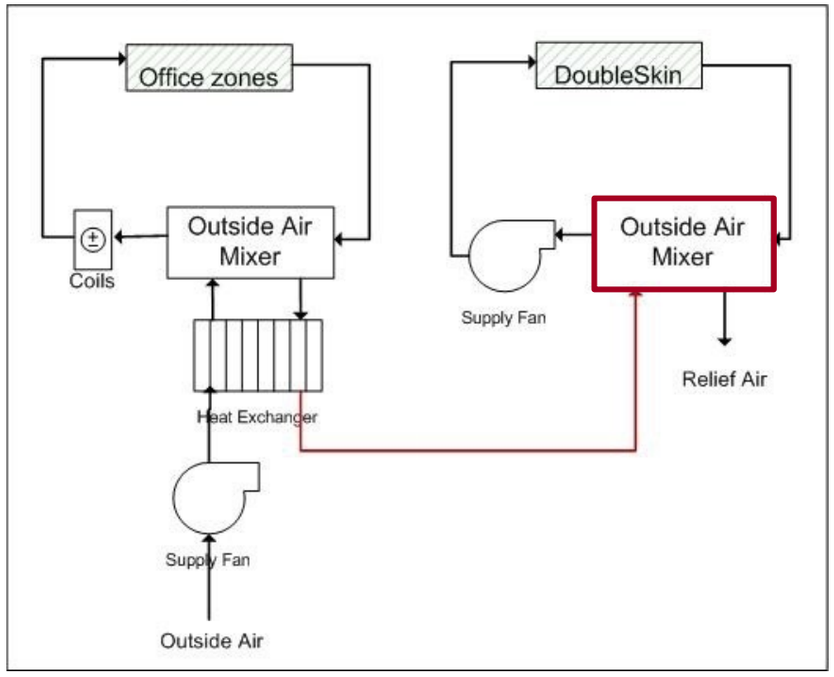
Case study:
Building in which the fixed schematic of the simulator required a model to be used that is different from the actual system

Actual system and implementation in BIM
Granularity & connectivity defined by product offerings and physical connections.



These are different systems

EnergyPlus system schematics
Granularity & connectivity is constraint by solution algorithms (e.g., fluid loops) and model causality.



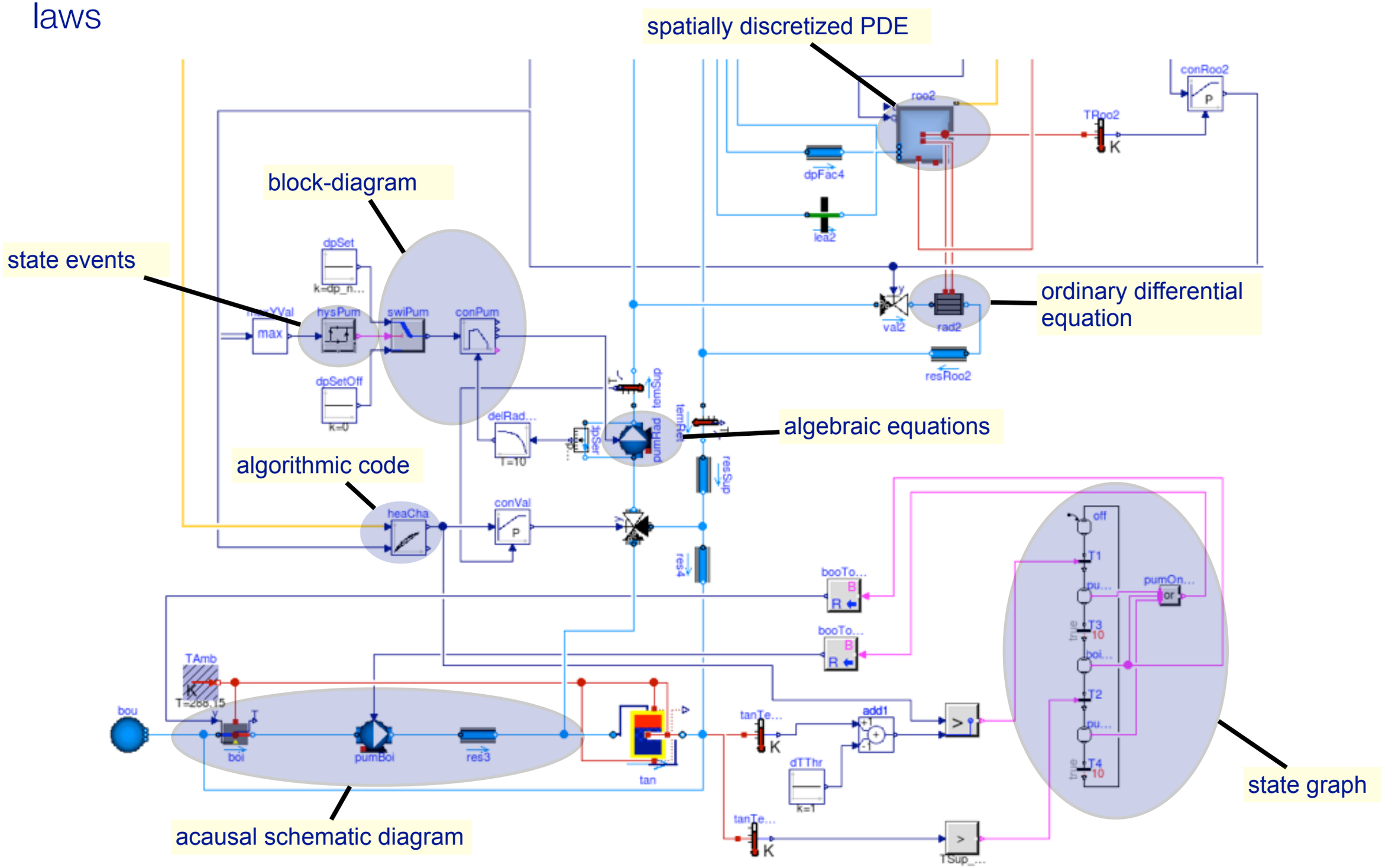
The red objects had to be inserted by the user due to the system context in which the other components have been used. In the actual system and in the BIM view, there is only one outside air mixer needed; this component was required merely because of the simulation program architecture.

To enable a 1-to-1 mapping, an acausal modeling language is needed that allows to connect components in any way that is possible in the real world.

Michael Wetter.
A View on Future Building System Modeling and Simulation.
In Building Performance Simulation for Design and Operation.
Jan L. M. Hensen and Roberto Lamberts (editors), Routledge, UK, ISBN: 978-0-415-47414-6, 2011.
Also available as LBNL Technical Report LBNL-4792E.pdf.

Schematic modeling of complex systems

Complex systems are packaged intuitively: icons represent components and subsystems that evolve differently in time, and lines equate their interface variables *and* impose conservation laws



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

Support for Optimization - JModelica

Extensible Modelica-based open source platform for optimization,

simulation and analysis of complex dynamic systems.

Main objective:

Create an industrially viable open source platform for optimization of Modelica models.

Offer a flexible platform serving as a virtual lab for algorithm development and research.

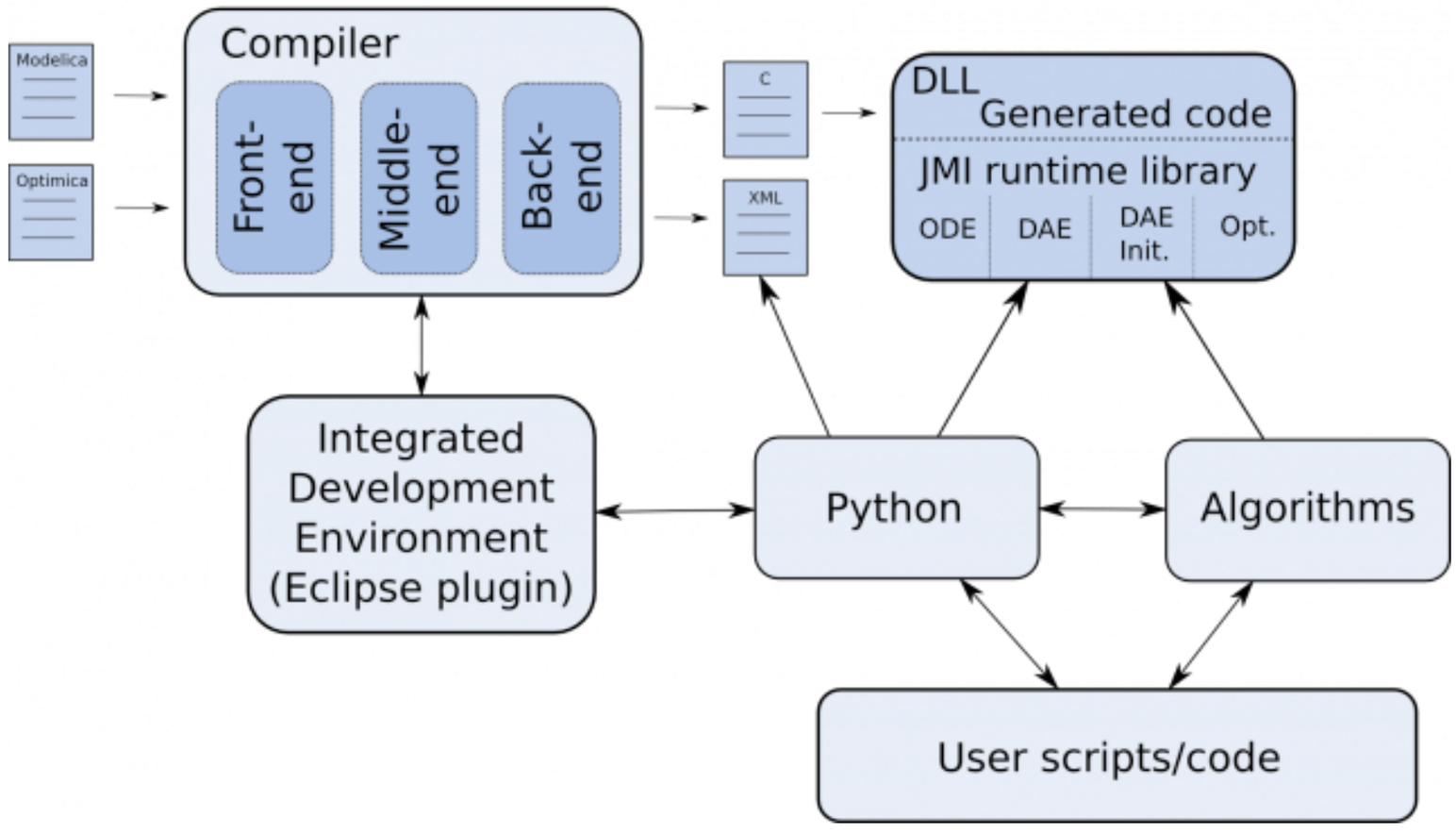
Main features

Optimal control

Parameter estimation

Simulation

Source: http://www.jmodelica.org



Modelica for building and district energy systems (see https://modelica.org/libraries for complete list)

Developed through Annex 60

AixLib (RWTH Aachen)

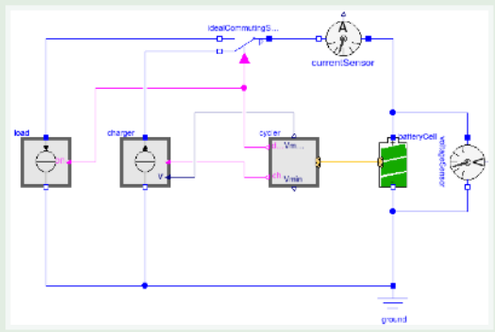
Buildings (LBNL)

BuildingSystem (UdK Berlin)

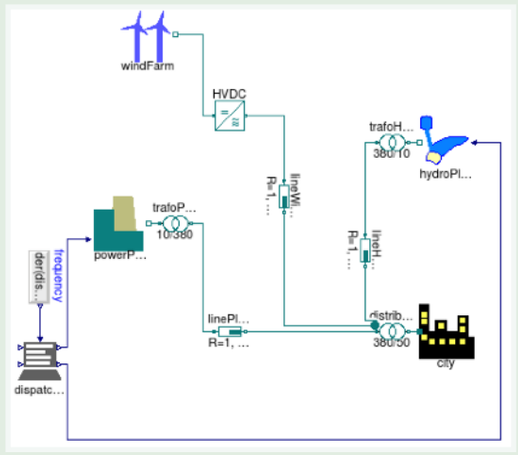
IDEAS (KU Leuven)

Free 3rd party libraries

Energy storage library (batteries)



Power systems library (transmission)



Commercial libraries

Electric power systems

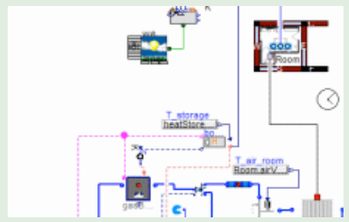
Batteries

Human Comfort

Heat exchanger

Air conditioning

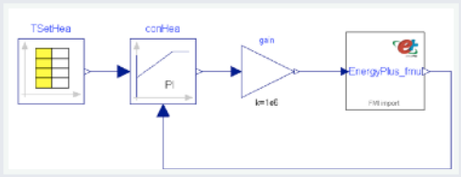
HVAC



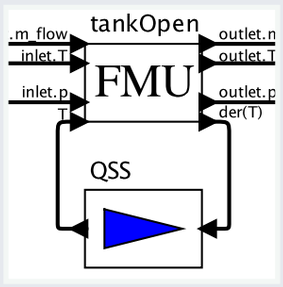
FMI

Simulator deployment and interoperability

FMI export & import in EnergyPlus, Niagara and BCVTB



Co-simulation



More than 80 tools, see https://www.fmi-standard.org/

Further literature

Online book with interactive examples of Michael Tiller at http://book.xogeny.com/ .

Modelica web reference http://modref.xogeny.com/

Interactive Modelica lessons at http://tour.xogeny.com/

The books by Michael Tiller (2001) and Peter Fritzson (2011 and 2004).

The tutorials at https://www.modelica.org/publications.

The Modelica language specification at https://modelica.org/documents.

Although the Modelica Language Tutorial is for an older version (Modelica 1.4), it is still instructive and relevant to understand the concepts of the language.

?