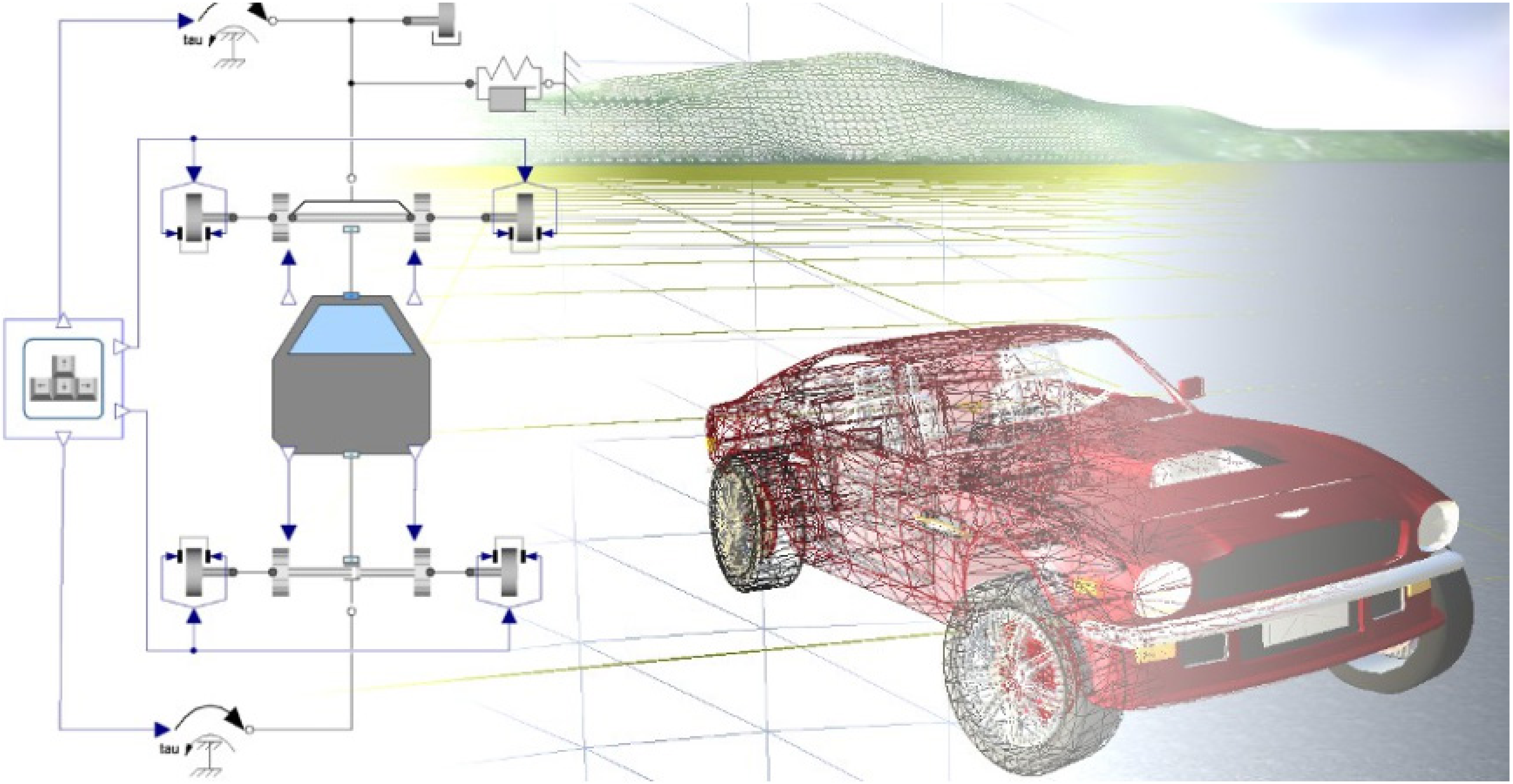
Modelica Modelling Tutorial

Dr. Dirk Zimmer
German Aerospace Center (DLR)

DLR

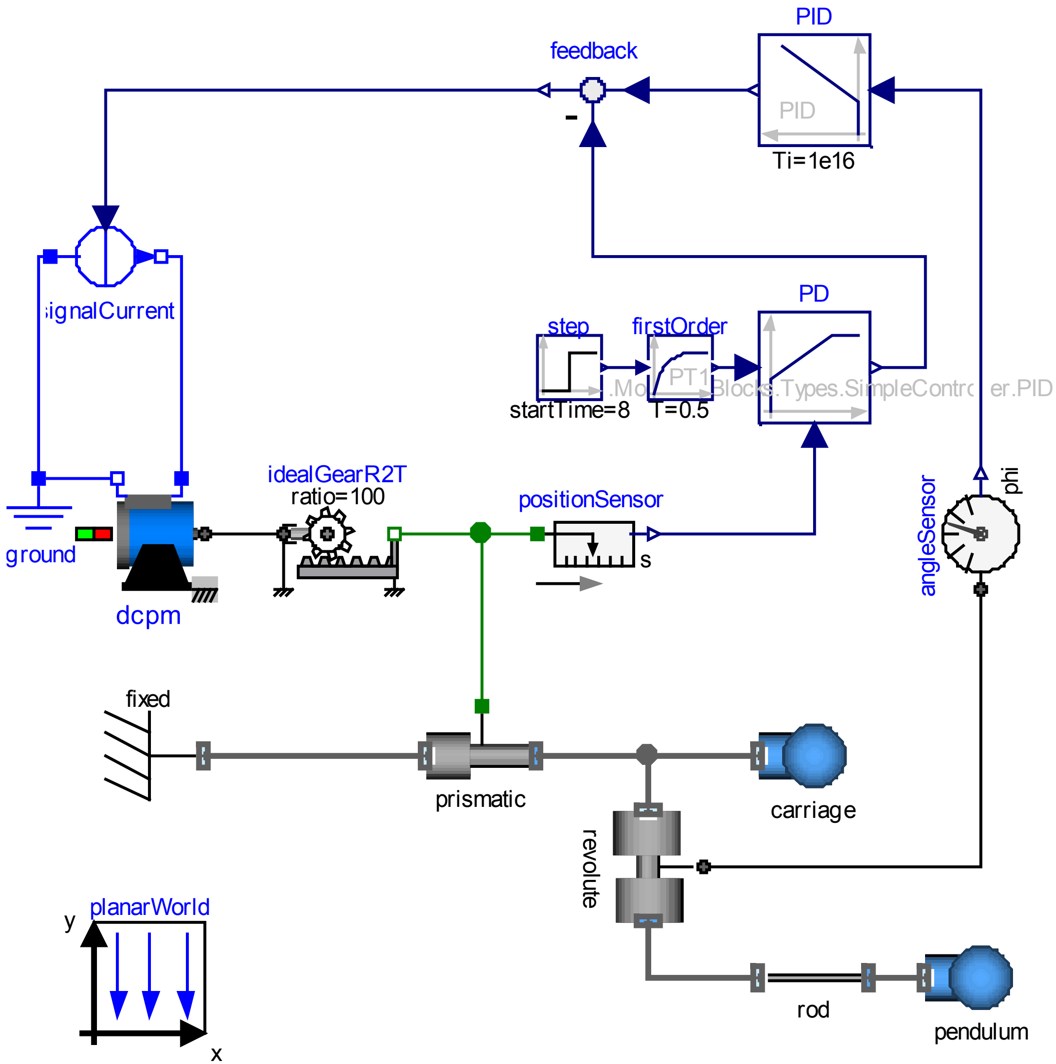
Teaching Modelica



http://www.robotic.de/Dirk.Zimmer

DLR

Demonstration of Modelica



5 Basic Principles of Modelica

- Equation-Based
- Acausality
- Physical Connectors
- Object Orientation
- Graphical Modeling

DLR

5 Basic Principles of Modelica

- Equation-Based
- Acausality
- Physical Connectors
- Object Orientation
- Graphical Modeling

DLR

Principle 1: Equation-Based Modelling

For this simple circuit, we can still derive all equations by hand, even in a very compact form.

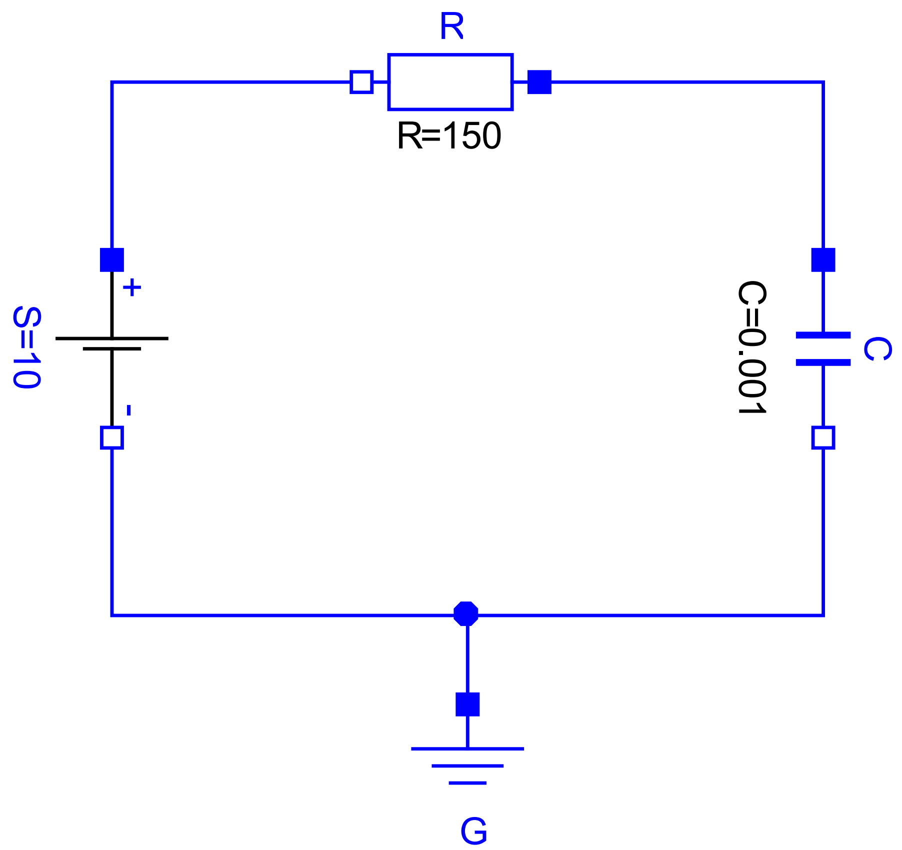
- The current is determined by:

$$10 - u_C = R \cdot i$$

- And the capacitor voltage is state of the system:

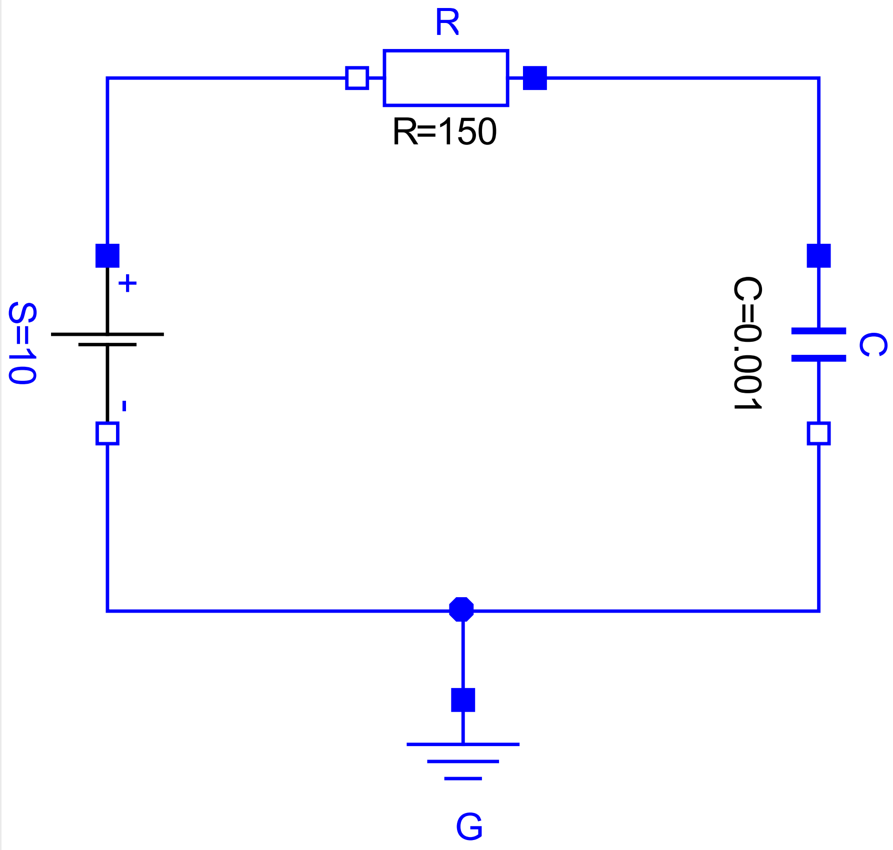
$$du_C/dt \cdot C = i$$

- Let us punch that into the computer by using Modelica



Principle 1: Equation-Based Modelling

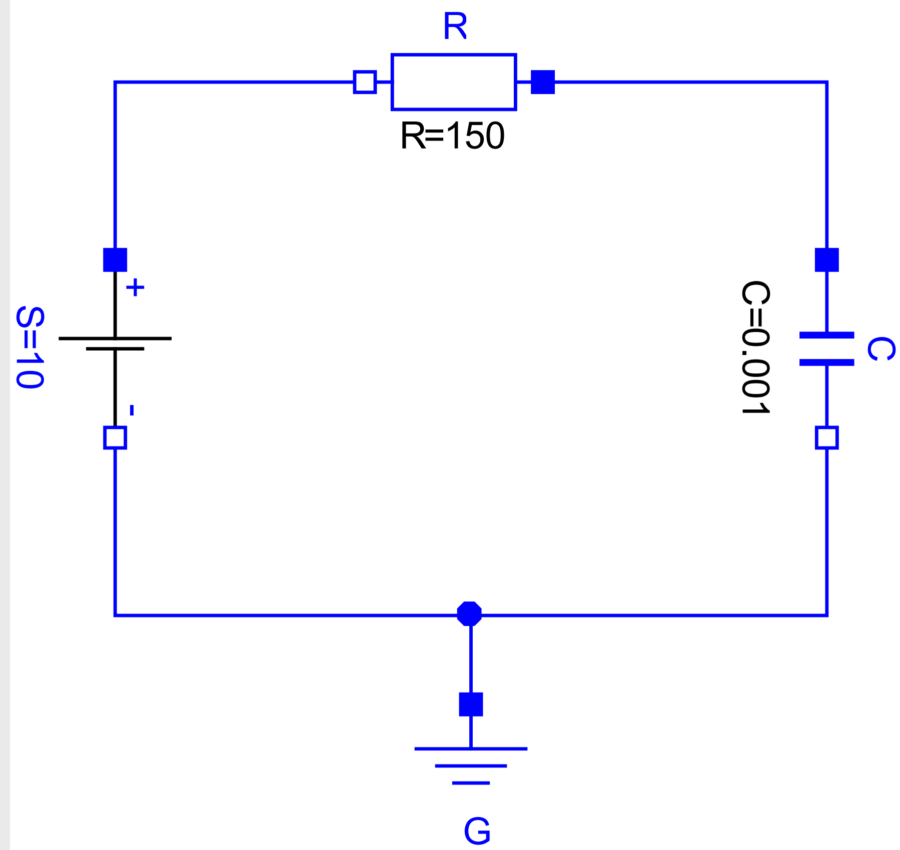
```
model SimpleCircuit

  parameter Real C;
  parameter Real R;
  parameter Real V0;

  Real i;
  Real uC;

equations

  V0-uC = R*i;
  der(uC)*C = i;

end SimpleCircuit;
```



DLR

Principle 1: Equation-Based Modelling

```
model SimpleCircuit
  "A simple RC circuit"
  import SI = Modelica.SIunits;
  parameter SI.Capacitance C=0.001
  "Capacity";
  parameter SI.Resistance R = 100
  "Resistance";
  parameter SI.Voltage V0 = 10
  "Source Voltage";
  SI.Current i "Current" ;
  SI.Voltage uC "Capacitor Voltage";

initial equation
   uC = 0;
equations
   V0-uC = R*i;
   der(uC)*C = i;

end SimpleCircuit;
```



5 Basic Principles of Modelica

- Equation-Based
- Acausality
- Physical Connectors
- Object Orientation
- Graphical Modeling

DLR

Principle 2: Acausality

```
model SimpleCircuit
  "A simple RC circuit"
  import SI = Modelica.SIunits;
  parameter SI.Capacitance C=0.001
  "Capacity";
  parameter SI.Resistance R = 100
  "Resistance";
  parameter SI.Voltage V0 = 10
  "Source Voltage";
  SI.Current i "Current" ;
  SI.Voltage uC "Capacitor Voltage";

initial equation
   uC = 0;
equations
   V0-uC = R*i;
   der(uC)*C = i;

end SimpleCircuit;
```

Equations are non-causal:

I can write:

```
V0-uC = R*i;
```

or

```
uC + R*i = V0;
```

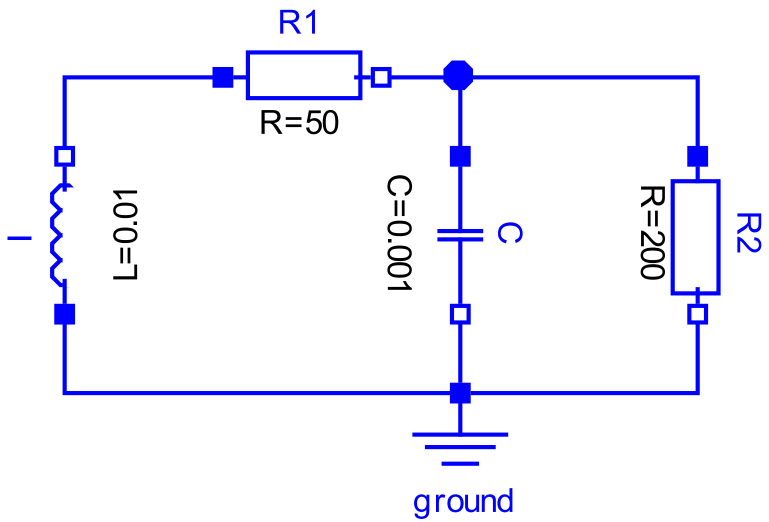
or

```
(uC-V0)/R = -i;
```

It expresses the same relation between uC and i

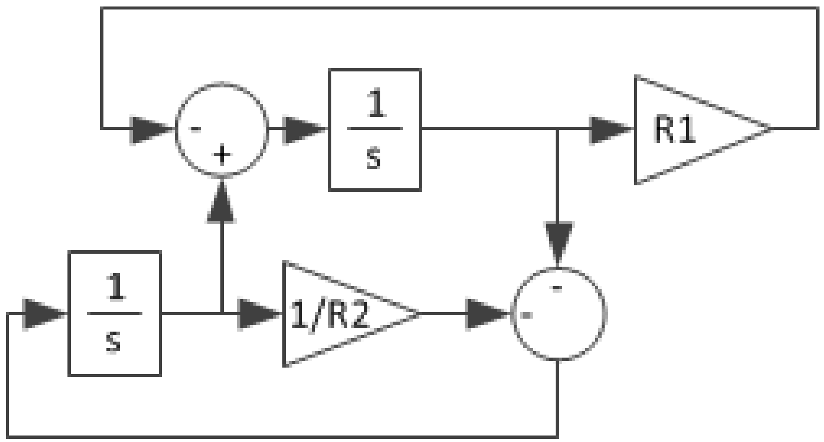
Principle 2: Acausality

Modelica



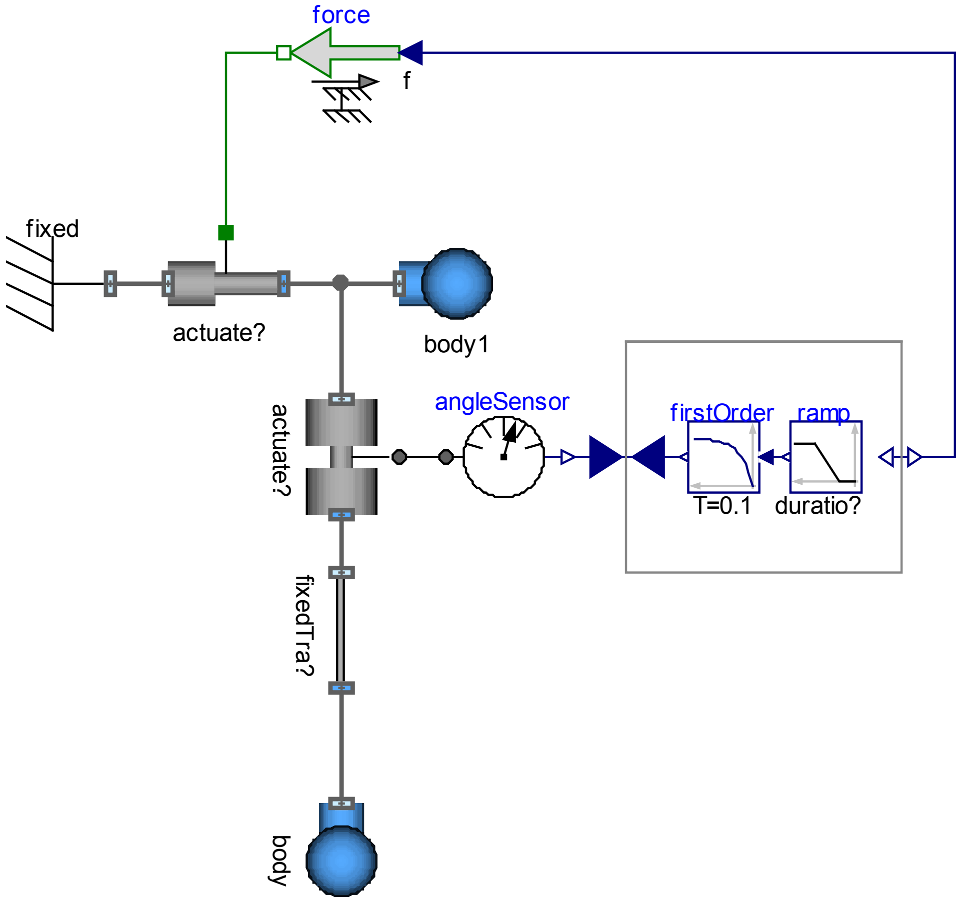
- In R1, the causality is: u := R*i
- In R2, the causality is: i := u/R
- In Modelica, one can use the same, non-causal equation u = R*i for both resistor components.

Simulink



- R1 is a gain factor
- R2 is a divisor
- In Simulink, I have causal signal flow and have to use different assignments for differently used resistors

Principle 2: Acausality



- Acausality is very powerful
- Instead of prescribing the forces and computing the motion…
- …I can prescribe the motion and compute the required force.

Principle 2: Acausality

- A Modelica Compiler typically transforms the model-equations from the implicit DAE Form:

$$\mathbf{0} = \mathrm{F}(\boldsymbol{x}_{\mathrm{p}},\mathrm{d}\boldsymbol{x}_{\mathrm{p}}/\mathrm{d}t,\boldsymbol{u},t)$$

- to the explicit ODE form

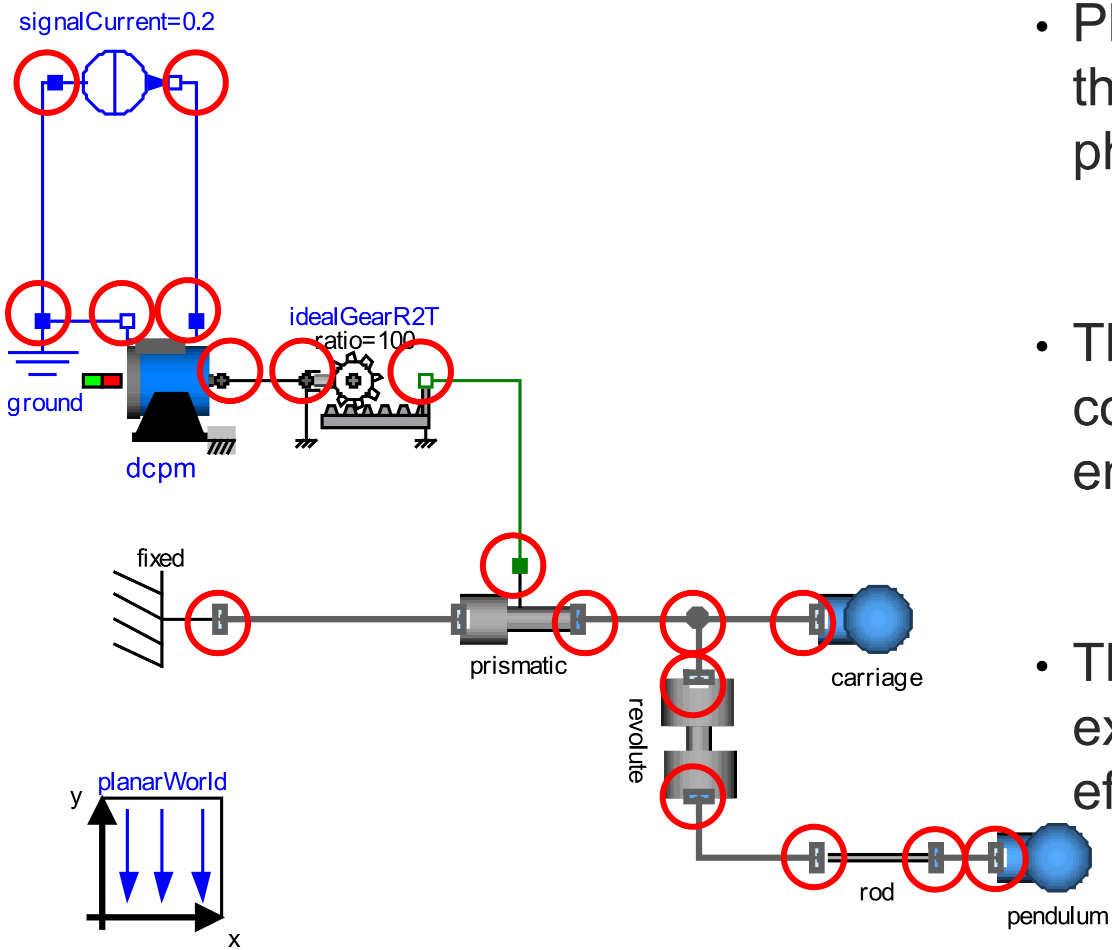
$$\mathrm{d}\boldsymbol{x}/\mathrm{d}t = f(\boldsymbol{x},\boldsymbol{u},t) \text{ with } \mathbf{x} \subseteq \boldsymbol{x}_{\mathrm{p}}$$

- This transformation is called index-reduction and involves
  - Differentiation of constraint equation (Pantelides)
  - Generation of BLT Form (Maximum Matching, Tarjan)
  - Identification of iteration (aka tearing) variables (Heuristics…)
  - …

5 Basic Principles of Modelica

- Equation-Based
- Acausality
- Physical Connectors
- Object Orientation
- Graphical Modeling

Principle 3: Physical Connectors



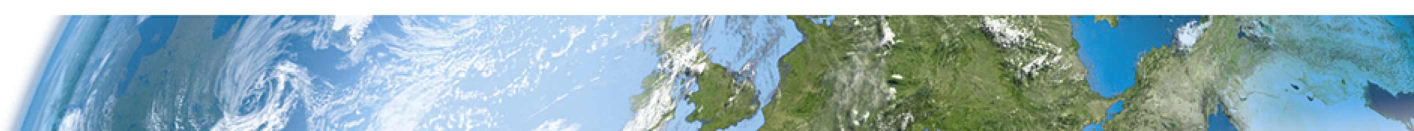
- Physical Connectors express the boundaries between physical components
- Through each of the connectors there is a flow of energy.
- This flow of energy can be expressed as a product of an effort and a flow variable….

Principle 3: Physical Connectors

For each physical domain, there is a specific pair of effort / flow variables

Domain	Potential	Flow
Translational Mechanics	Velocity: v [m/s]	Force: f [N]
Rotational Mechanics	Angular Velocity: ω [1/s]	Torque: τ [Nm]
Electrics	Voltage Potential v [V]	Current i [A]
Magnetics	Magnetomotive Force: Θ [A]	Time-derivative of Magnetic Flux: Φ [V]
Hydraulics	Pressure p [Pa]	Volume flow rate $\dot{V}$ [m³/s]
Thermal	Temperature T[K]	Entropy Flow Rate $\dot{S}$ [J/Ks]
Chemical	Chemical Potential: µ [J/mol]	Molar Flow Rate $\dot{\nu}$ [mol/s]

DLR



Principle 3: Physical Connectors

```
connector Pin

  SI.Voltage v "Potential at the pin";
  flow SI.Current i "Current flowing into the pin";

end Pin;
```

- This is the definiton of the corresponding connector.
- It consists in a set of variables.
- These variables can be declared to be…
  - potential variables: `SI.Voltage v`
  - flow variables: `flow SI.Current i`

DLR

Principle 3: Physical Connectors

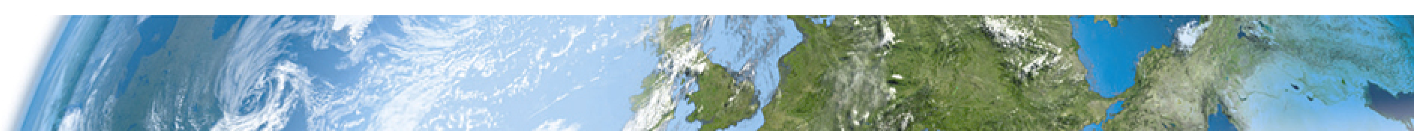
```
connector Pin

  SI.Voltage v "Potential at the pin";
  flow SI.Current i "Current flowing into the pin";

end Pin;
```

- We can link two ore more pins by using the connect statement.

```
connect(pin1, pin2)
connect(pin1, pin3)
```

```
pin1.v = pin2.v
pin1.v = pin3.v
pin1.i + pin2.i +pin3.i = 0
```

- The equations are generated in dependence on the declaration
- Connections form a graph that represents a wood and that is component relevant and structure irrelevant.



5 Basic Principles of Modelica

- Equation-Based
- Acausality
- Physical Connectors
- Object Orientation
- Graphical Modeling

Principle 4: Object Orientation

```
model Resistor
 "Resistor Model"

  parameter SI.Resistance R;
  Pin n;
  Pin p;

  SI.Current i;
  SI.Voltage u;

equations

  u = p.v - n.v;
  n.i + p.i = 0;
  i = p.i;
  u = R*i;

end Resistor;
```

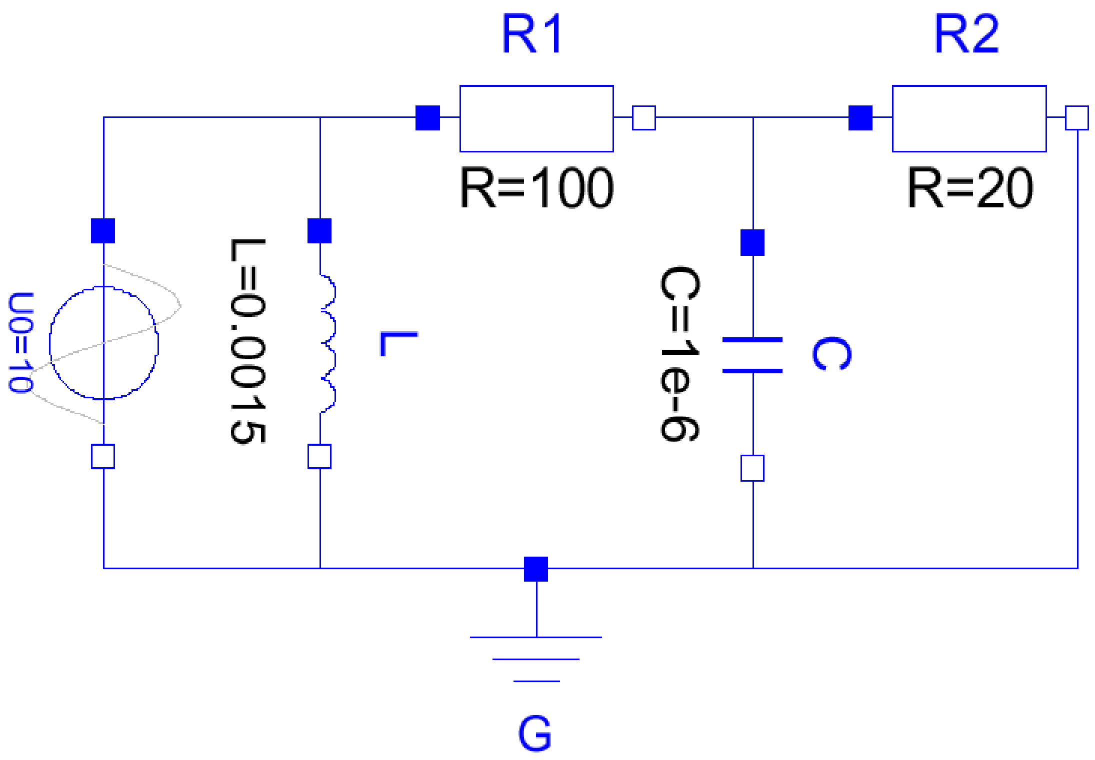
- Models are created for components
- These contain an incomplete set of equations.
- These components models can then be composed to a complete system.

Principle 4: Object Orientation

```
package Electrics
 ”Basic Electric Elements”

  model Ground
    …
  end Ground;

  model Resistor
    …
  end Resistor;

  model Capacitor
    …
  end Capacitor;

  …

end Electrics;
```

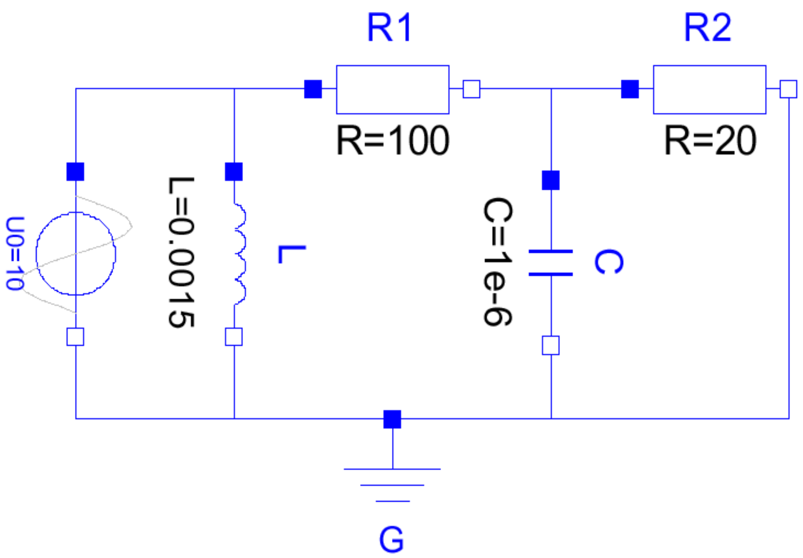
- All these models can be collected in a Modelica package
- A package can contain arbitrary classes, also sub-packages.

Principle 4: Object Orientation



Principle 4: Object Orientation

```
model Circuit
  Resistor R1(R=100);
  Resistor R2(R=20);
  Capacitor C(C=1e-6);
  Inductor L(L=0.0015);
  SineVSource S(Ampl=15, Freq=50);
  Ground G;

equations
  connect(G.p,S.n)
  connect(G.p,L.n)
  connect(G.p,R2.n)
  connect(G.p,C.n)

  connect(S.p,R1.p)
  connect(S.p,L.p)

  connect(R1.n,R2.p)
  connect(R1.n,C.p)
end Circuit;
```



Principle 4: Object Orientation

```
model Circuit
  Resistor R1(R=100);
  Resistor R2(R=20);
  Capacitor C(C=1e-6);
  Inductor L(L=0.0015;
  SineVSource S(Ampl=15, Freq=50);
  Ground G;

equations
  connect(G.p,S.n)
  connect(G.p,L.n)
  connect(G.p,R2.n)
  connect(G.p,C.n)

  connect(S.p,R1.p)
  connect(S.p,L.p)

  connect(R1.n,R2.p)
  connect(R1.n,C.p)
end Circuit;
```

$5 \cdot 4 + 1 \cdot 1 = 21$

component equ.

$5 \cdot 6 + 1 \cdot 2 = 32$

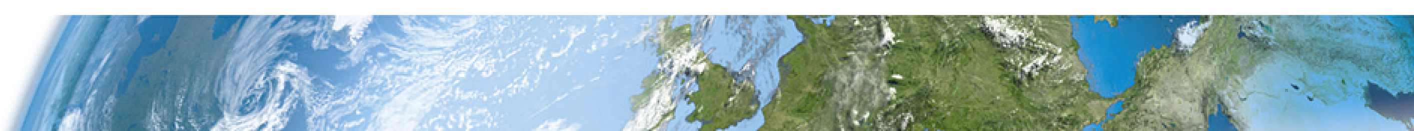
unknowns

32 equations

32 unknowns

8 potential equations

3 flow equations



5 Basic Principles of Modelica

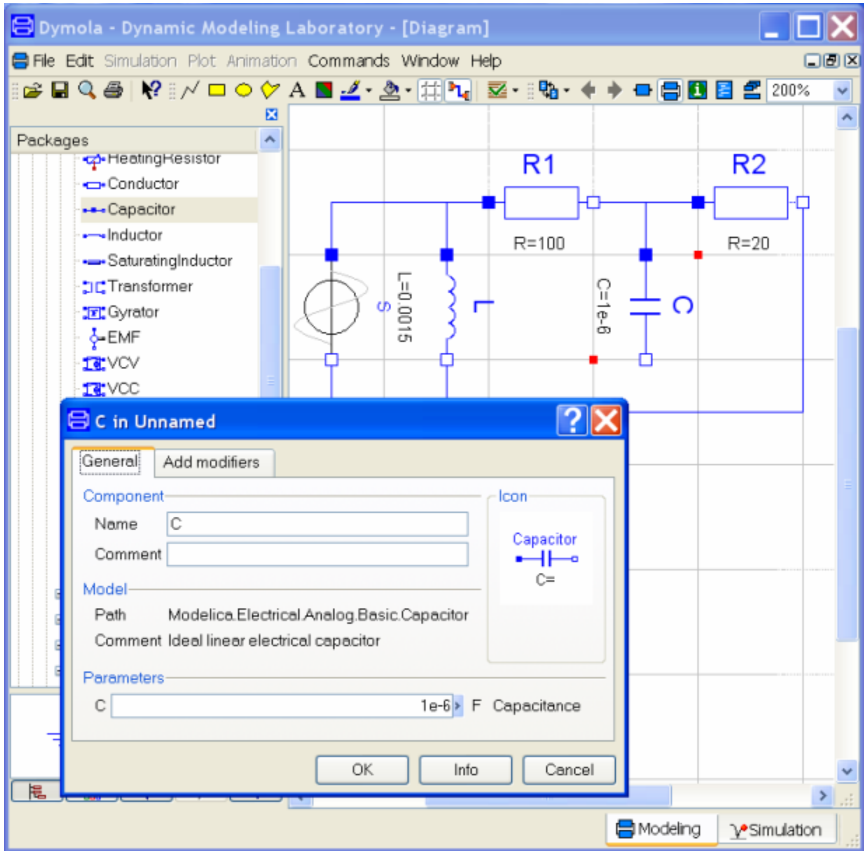
- Equation-Based
- Acausality
- Physical Connectors
- Object Orientation
- Graphical Modeling

DLR

Graphical Modeling

So far, we have only looked at the textual side of modeling.

- Using a modern modeling environment like Dymola, most modeling is performed graphically.
- Textual modeling is only done for the lower level tasks.

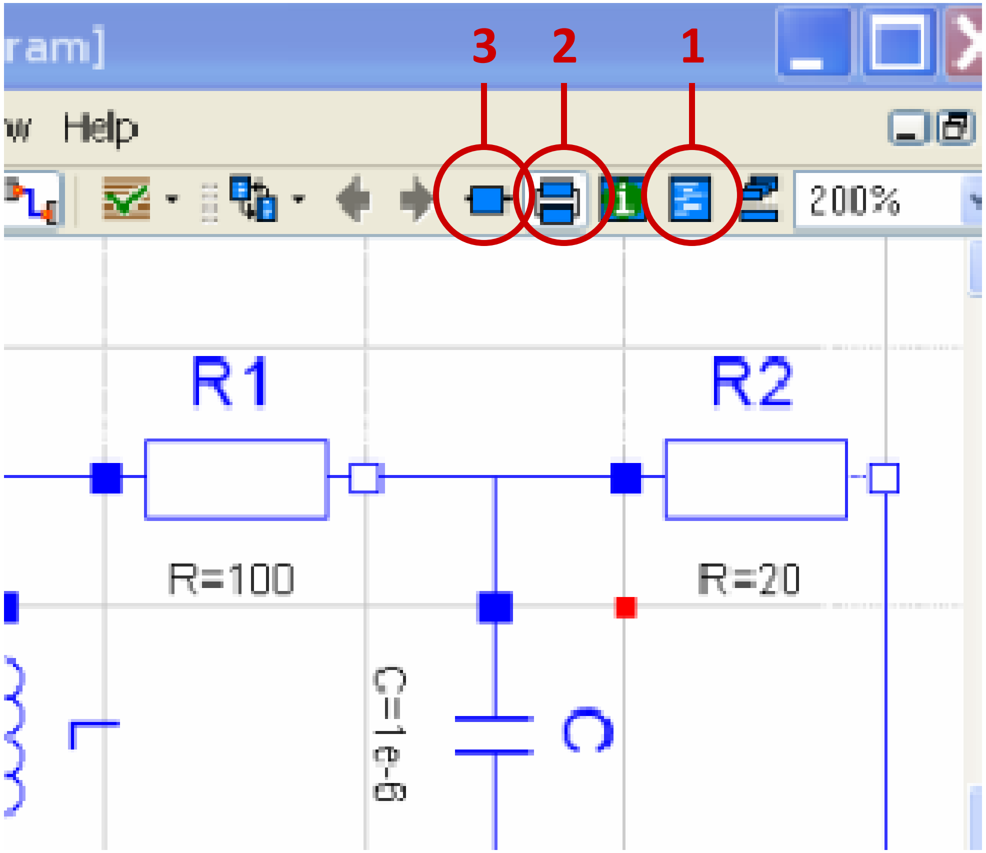


DLR

Graphical Modeling

To this end, Dymola offers three distinct modeling layers.

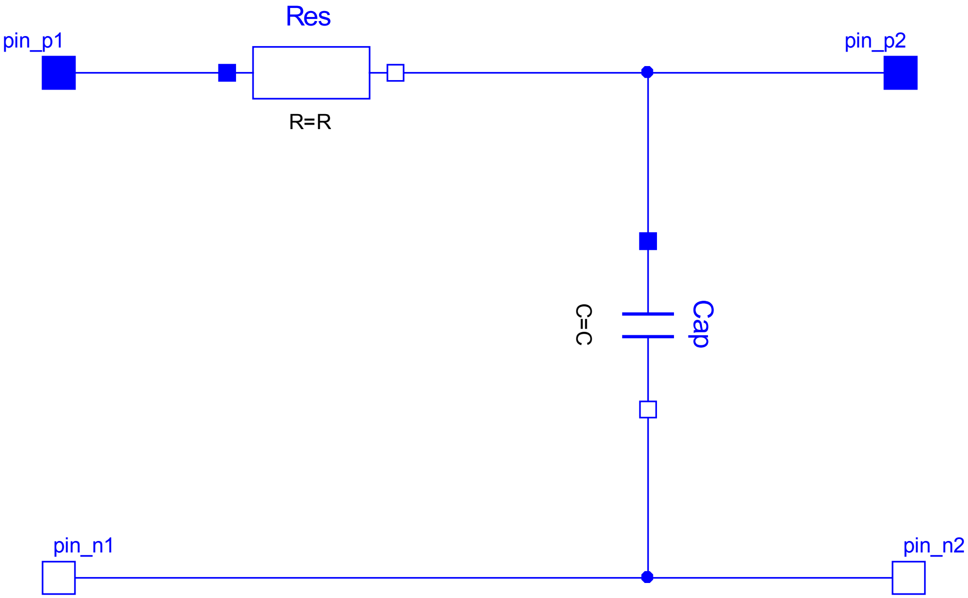
- The inner textual representation (1)
- The inner graphical representation (2)
- The outer graphical representation (3)



Graphical Modeling

Let us model an RC-Filter.

- We start with the inner graphical representation.
- Here we model the actual sub-circuit



Graphical Modeling

Let us model an RC-Filter.

- On the textual layer, we provide two parameters for the resistor and the capacitor

```
model RCFilter
  import SI = Modelica.SIunits;
  parameter SI.Resistance R = 100;
  parameter SI.Capacitance C = 1e-3;

  Modelica…Resistor Res(R=R);
  Modelica…Capacitor Cap(C=C);
  Modelica…NegativePin pin_n1;
  Modelica…NegativePin pin_n2;
  Modelica…PositivePin pin_p1;
  Modelica…PositivePin pin_p2;
equation
  connect(pin_p1, Res.p);
  connect(Res.n, pin_p2);
  connect(Cap.p, Res.n);
  connect(Cap.n, pin_n2);
  connect(pin_n1, pin_n2);
end RCFilter;
```

Graphical Modeling

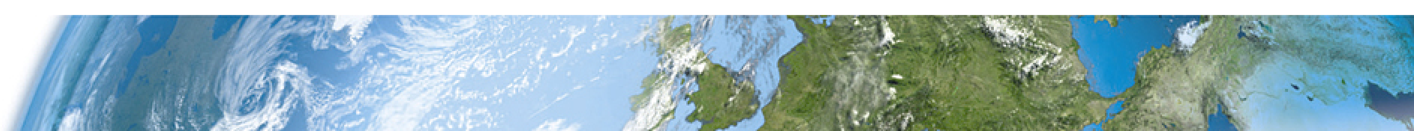
```
model RCFilter
   import SI = Modelica.SIunits;
   parameter SI.Resistance R = 100;
   parameter SI.Capacitance C = 1e-3;

   Modelica…Resistor Res(R=R) a;
   Modelica…Capacitor Cap(C=C) a;
   Modelica…NegativePin pin_n1 a;
   Modelica…NegativePin pin_n2 a;
   Modelica…PositivePin pin_p1 a;
   Modelica…PositivePin pin_p2 a;
equation
   connect(pin_p1, Res.p) a;
   connect(Res.n, pin_p2) a;
   connect(Cap.p, Res.n) a;
   connect(Cap.n, pin_n2) a;
   connect(pin_n1, pin_n2) a;
end RCFilter;
```

- How is the graphical information stored within the model.
- Modelica uses annotations for this purpose.
- Dymola typically hides annotations and represents them by the symbol: a
- The visibility of annotations can be enabled in the Dymola Editor.

Graphical Modeling

```
annotation(Icon(graphics={
  Rectangle(
    extent={{-80,80},{80,-80}},
    lineColor={0,0,255},
    fillColor={255,255,255},
    fillPattern=FillPattern.Solid),
  Line(
    points={{-90,60},{-60,60},
            {-60,-60},{-90,-60}},
    color={0,0,255},
    smooth=Smooth.None),
  Line( points={{90,60},{60,60},
                {60,-60},{90,-60}},
    color={0,0,255},
    smooth=Smooth.None),
  Text(extent={{-60,60},{60,2}},
     lineColor={0,0,255},
     textString="Low"),
   …
```

- How is the graphical information stored within the model.
- Modelica uses annotations for this purpose.
- Dymola typically hides annotations and represents them by the symbol: **a**
- The visibility of annotations can be enabled in the Dymola Editor.



5 Basic Principles of Modelica

- Equation-Based
- Acausality
- Physical Connectors
- Object Orientation
- Graphical Modeling

Declarative Modelling Language with Self-contained models

The holy trinity of Modelica

The Modelica Language

The Modelica Standard Library

The Modelica Association
www.modelica.org

Questions?