# Modelica and Optimica

Johan Åkesson

# What is Modelica?

- A language for modeling of complex heterogeneous physical systems
  - Open language
    - Modelica Association (www.modelica.org)
  - Several tools supporting Modelica
    - Dymola
    - OpenModelica (free)
    - MosiLab
    - Scilab/Scicos (free)
  - Extensive (free) standard library
    - Mechanical, electrical, thermal etc.

# Key Features of Modelica

- Declarative equation-based modeling
  - Text book style equations
- Multi-domain modeling
  - Heterogeneous modeling
- Object oriented modeling
  - Inheritance and generics
- Software component model
  - Instances and (acausal) connections
- Graphical and textual modeling

# A Simple Modelica model

**Differential equation**

$$\frac{dx(t)}{dt} = ax(t) + bu(t) \quad x(0) = x_0$$

**Class definition**

**Parameter declaration**

**Variable declaration**

**Initialization**

**Derivative operator**

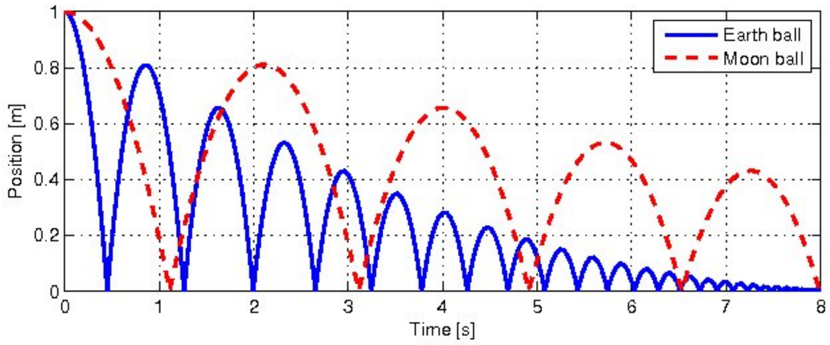
**Equation**

```
model FirstOrder
  input Real u;
  parameter Real b = 1;
  parameter Real a = -1;
  Real x(start=1);
equation
  der(x)  = a*x + b*u;
end FirstOrder;
```

# Hybrid modeling

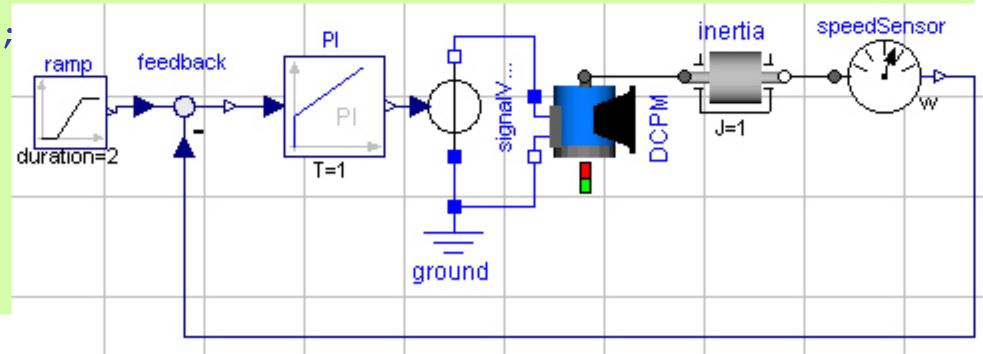
```
class BouncingBall //A model of a bouncing ball
  parameter Real g = 9.81; //Acceleration due to gravity
  parameter Real e = 0.9; //Elasticity coefficient
  Real pos(start=1); //Position of the ball
  Real vel(start=0); //Velocity of the ball
equation
  der(pos) = vel;   // Newtons second law
  der(vel) = -g;
  when pos <=0 then
    reinit(vel,-e*pre(vel));
  end when;
end BouncingBall;
```

```
class BBex
  BouncingBall eBall;
  BouncingBall mBall(g=1.62);
end BBex;
```



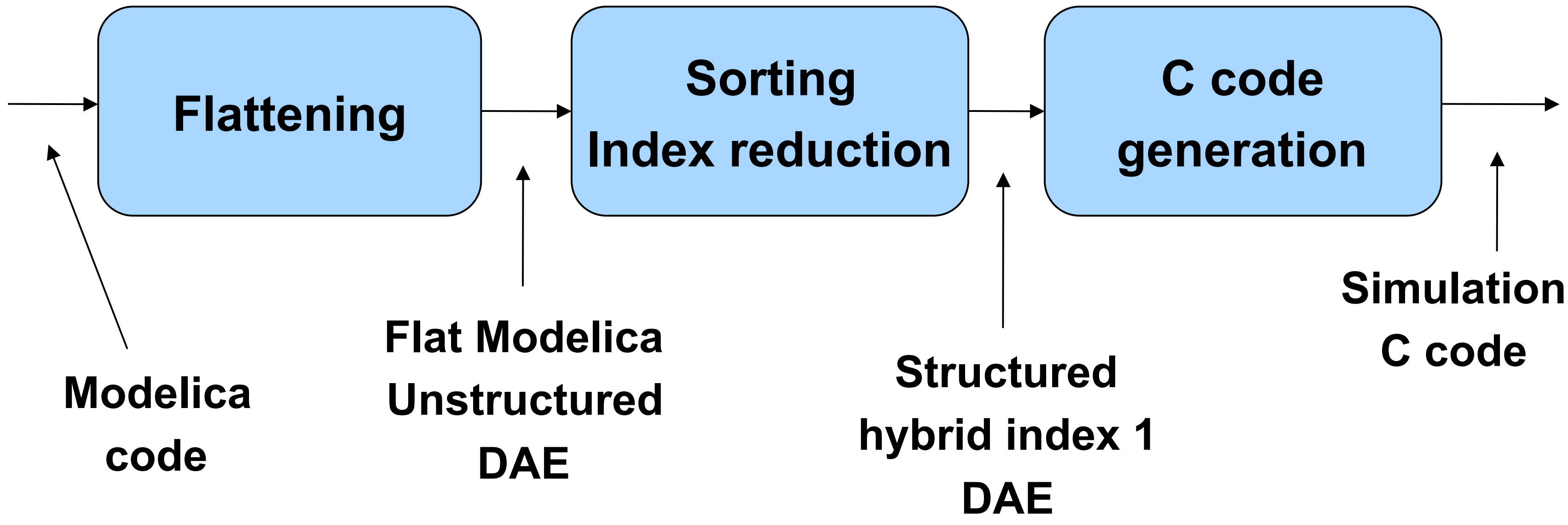
# Graphical Modeling

```
model MotorControl
  Modelica.Mechanics.Rotational.Inertia inertia;
  Modelica.Mechanics.Rotational.Sensors.SpeedSensor speedSensor;
  Modelica.Electrical.Machines.BasicMachines.DCMachines.DC_PermanentMagnet DCPM;
  Modelica.Electrical.Analog.Basic.Ground ground;
  Modelica.Electrical.Analog.Sources.SignalVoltage signalVoltage;
  Modelica.Blocks.Math.Feedback feedback;
  Modelica.Blocks.Sources.Ramp ramp(height=100, startTime=1);
  Modelica.Blocks.Continuous.PI PI(k=-2);
equation
  connect(inertia.flange_b, speedSensor.flange_a);
  connect(DCPM.flange_a, inertia.flange_a);
  connect(speedSensor.w, feedback.u2);
  connect(ramp.y, feedback.u1);
  connect(signalVoltage.n, DCPM.pin_ap);
  connect(signalVoltage.p, ground.p);
  connect(ground.p, DCPM.pin_an);
  connect(feedback.y, PI.u);
  connect(PI.y, signalVoltage.v);
end MotorControl;
```

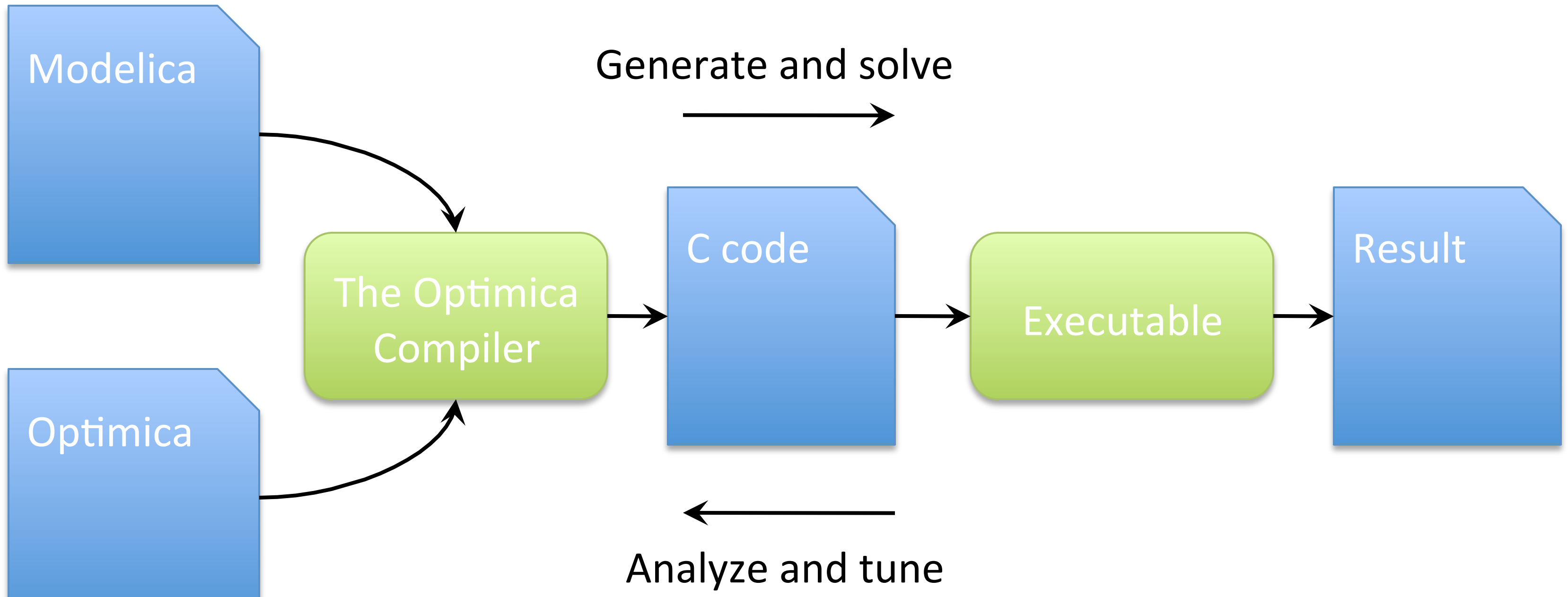


# The Modelica translation process (simulation case)

- Generation of a mathematical model description from Modelica code



# Typical workcycle



## Optimization with Modelica

- Strong support for modeling of dynamic systems
- Missing elements
  - Cost function
  - Constraints
  - What to optimize
  - Initial guesses
- Optimica
  - Small extension of Modelica
  - Enable high-level formulation of optimization problems

## An example

$$\min_{u(t)} \int_{t_0}^{t_f} 1\, dt$$

subject to the dynamic constraint

$$\dot{x}_1(t) = (1 - x_2(t)^2)\, x_1(t) - x_2(t) + u(t), \quad x_1(0) = 0$$

$$\dot{x}_2(t) = x_1(t), \quad x_2(0) = 1$$

and

$$x_1(t_f) = 0$$

$$x_2(t_f) = 0$$

$$-1 \leqslant u(t) \leqslant 1$$

## A Modelica model

```
model VDP
  Real x1(start=0);
  Real x2(start=1);
  input Real u;
equation
  der(x1) = (1-x2^2)*x1 - x2 + u;
  der(x2) = x1;
end VDP;
```

# An Optimica model

```
optimization VDP_Opt(objective=cost(finalTime),
        startTime=0,
        finalTime(free=true, initialGuess=1))
  VDP vdp(u(free=true,initialGuess=0.0));
    Real cost (start=0);
equation
    der(cost) = 1;
constraint
    vdp.x1(finalTime) = 0;
    vdp.x2(finalTime) = 0;
    vdp.u >= -1; vdp.u <= 1;
end VDP_Opt;
```