Object Oriented Modeling

• The aim of this lecture is to illustrate the requirements of a software environment for object-oriented modeling of physical systems and to show how these requirements can be met in practice.
• The lecture offers a first glimpse at features and capabilities of Dymola, a software environment created for the purpose of modeling complex physical systems in an object-oriented fashion. Dymola offers a graphical user interface.
• Some features of the underlying textual model representation, called Modelica, are also introduced.

Table of Contents

• The causality of the model equations
• Graphical modeling
• Model structure in Modelica
• Model topology in Modelica
• Inheritance rules
• Hierarchical modeling
The Causality of the Model Equations

\[ U_0 = f(t) \]
\[ i = \frac{U_0}{R} \]
\[ I_0 = f(t) \]
\[ u = R \cdot I_0 \]

Identical Objects

Different Equations

⇒ The causality of the equations must not be predetermined. It can only be decided upon after the analysis of the system topology.

Basic Requirements of OO Modeling

- Physical objects should be representable by mathematical graphical objects.
- The graphical objects should be \textit{topologically} connectable.
- The mathematical models should be \textit{hierarchically} describable. To this end, it must be possible to represent networks of coupled objects again as graphical objects.
An Example

model Circuit1
  SineVoltage U0(V=10, freqHz=2500);
  Resistor R1(R=100);
  Resistor R2(R=20);
  Capacitor C(C=1E-6);
  Inductor L(L=0.0015);
  Ground Ground;
  equation
    connect(U0.p, R1.p);
    connect(R1.n, C.p);
    connect(R2.p, R1.n);
    connect(U0.n, C.n);
    connect(Ground.p, C.n);
    connect(L.p, R1.p);
    connect(L.l, Ground.p);
    connect(R2.n, L.n);
end Circuit1;

Graphical Information (Annotation)

package CircuitLib
  annotation (Coordsys( extent=[0, 0; 504, 364], grid=[2, 2], component=[20, 20]));
  model Circuit1
    annotation (Coordsys( extent=[-100, -100; 100, 100], grid=[2, 2], component=[20, 20]));
  Modelica.Electrical.Analog.Sources.SineVoltage U0(V=10, freqHz=2500) annotation (extent=[-80, -20; -40, 20], rotation=-90);
  Modelica.Electrical.Analog.Basic.Resistor R1(R=100) annotation (extent=[-40, 20; 0, 60], rotation=-90);
  Modelica.Electrical.Analog.Basic.Capacitor C(C=1E-6) annotation (extent=[-40, -60; 0, -20], rotation=-90);
  Modelica.Electrical.Analog.Basic.Resistor R2(R=20) annotation (extent=[0, -20; 40, 20], rotation=-90);
  Modelica.Electrical.Analog.Basic.Inductor L(L=0.0015) annotation (extent=[40, 20; 80, 60], rotation=-90);
  Modelica.Electrical.Analog.Basic.Ground Ground annotation (extent=[0, -100; 40, -60], rotation=-90);
  equation
    connect(U0.p, R1.p) annotation (points=[-60, 20; -60, 60; -20, 60], style(color=3));
    connect(R1.n, C.p) annotation (points=[-20, 20; -20, -20], style(color=3));
    connect(R2.p, R1.n) annotation (points=[0, 0; 0, -20, 20], style(color=3));
    connect(U0.n, C.n) annotation (points=[-60, -20; -60, -60; -20, -60], style(color=3));
    connect(Ground.p, C.n) annotation (points=[20, -60; 20, -60], style(color=3));
    connect(L.p, R1.p) annotation (points=[60, 60; 60, -60, 20, 20], style(color=3));
    connect(L.l, Ground.p) annotation (points=[60, 20; 60, -60, 20, -60], style(color=3));
    connect(R2.n, L.n) annotation (points=[40, 0; 0, 60, 0, 60, 20], style(color=3));
end Circuit1;
end CircuitLib;
Models in Modelica

- Models in Modelica consist of a description of their model structure as well as a description of their embedding in the model environment:

```model
Model name

Description of the model embedding;

equations

Description of the model structure;

end
Model name;
```

Model Structure in Modelica

- The model structure in Modelica consists either of a set of equations, a description of the model topology, or a combination of the two types of model structure descriptions.
- A topological model description is usually done by dragging and dropping model icons from graphical model libraries into the modeling window. These models are then graphically interconnected among each other.
- The stored textual version of the topological model consists of a declaration of its sub-models (model embedding), a declaration of its connections (model structure), as well as a declaration of the graphical description elements (Annotation).
### Model Topology in Modelica

**Model**
```
model MotorDrive
    class name
    MotorDrive
    class name
    Motor motor;
    Gearbox gearbox(n=100);
    Shaft J1(J=10);
    Tachometer wl;
    equation
        connect(controller.out, motor.inp);
        connect(motor.flange , gearbox.a);
        connect(gearbox.b   , Jl.a);
        connect(Jl.b       , wl.a);
        connect(wl.w       , controller.inp);
    end MotorDrive;
```

**Resistors in Modelica**

**Model**
```
model Resistor "Ideal resistor"
    class name
    Resistor Pin p, n;
    Voltage u;
    parameter Resistance R;
    equation
        u = p.v - n.v;
        p.i + n.i = 0;
        R*p.i = u;
    end Resistor;
```

**Type**
```
ElectricPotential = Real
    final quantity="ElectricPotential",
    final unit="V";
Voltage = ElectricPotential;
```
Similarity Between Different Elements

- **Resistor "Ideal resistor"**
  - Pin `p, n`;
  - Voltage `u`;
  - Parameter `Resistance R`;
  - Equation:
    \[ u = p.v - n.v; \]
    \[ p.i + n.i = 0; \]
    \[ R*p.i = u; \]
  - End `Resistor`;

- **Capacitor "Ideal capacitor"**
  - Pin `p, n`;
  - Voltage `u`;
  - Parameter `Capacitance C`;
  - Equation:
    \[ u = p.v - n.v; \]
    \[ p.i + n.i = 0; \]
    \[ C*der(u) = p.i; \]
  - End `Capacitor`;

Partial Models and Inheritance

- **OnePort**
  - Pin `p, n`;
  - Voltage `u`;
  - Equation:
    \[ u = p.v - n.v; \]
    \[ p.i + n.i = 0; \]
  - End `OnePort`;

- **Resistor "Ideal resistor"**
  - Extends `OnePort`;
  - Parameter `Resistance R`;
  - Equation:
    \[ R*p.i = u; \]
  - End `Resistor`;

- **Capacitor "Ideal capacitor"**
  - Extends `OnePort`;
  - Parameter `Capacitance C`;
  - Equation:
    \[ C*der(u) = p.i; \]
  - End `Capacitor`;

September 8, 2003
Decomposition and Abstraction

Heterogeneous Modeling Formalisms
References
