Modelica

Traditionella simuleringsverktyg

- Generella program
  - blockdiagram med in- och utsignaler
  - ACSL, Simulink, Systembuild
  - Baserade på CSSL (1967)
  - Beräkningseffektivitet, ej användarvänlighet
- Specialiserade program
  - Elektriska system (SPICE, Saber)
  - Mekaniska system (ADAMS, DADS)
  - Svårt med blandade system

Ny generell fysikalisk modellering

- Nya program under 80- och 90-talet
  - ASCEND, Dymola, gPROMS, NMF, ObejtMath, Smile mfl
  - Icke-kausal modellering med ekvationer
  - Objektorientering för strukturering
- Standardisering av modellieringsspråk
  - Förening av olika koncept
  - Utbyte av modeller och modellbibliotek
  - Modelica 1.0 klart September 1997.

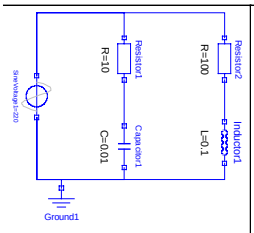
Objektorienterad matematisk modellering

- *Objekt* är en samling av variabler, ekvationer och funktioner.
- *Klasser* är mallar som man kan skapa nya objekt av.
- *Ärvning* tillåter att ekvationer och funktioner som definierats i en klass kan återanvändas när man skapar ett nytt objekt.

Modellering

- Standard komponenter hämtas från modellbibliotek (instansiering av objekt)
- Ekvationer på naturlig form
- Ikoner sammanbinds grafiskt

Exempel



Figur 1. En enkel elektrisk krets.

```
model circuit
  Ground Ground1;
  Resistor Resistor1(R=10);
  Resistor Resistor2(R=100);
  Capacitor Capacitor1(C=0.01);
  Inductor Inductor1(L=0.1);
  SineVoltage SineVoltage1(V=220,
freqHz=50 );
Equation
  connect(SineVoltage1.p, Resistor2.p);
  connect(Resistor1.p, SineVoltage1.p);
  connect(Capacitor1.p, Resistor1.n);
  connect(Inductor1.p, Resistor2.n);
  connect(Inductor1.n, SineVoltage1.n);
  connect(Ground1.p, SineVoltage1.n);
  connect(Capacitor1.n, SineVoltage1.n);
end circuit
```

Figur 2. Modelica kod för kretsen i figur 1.

Variabler och anslutningar

- Anslutningar och förbindelser motsvaras av komponentens fysikaliska förbindelser.

```
connector Pin "Pin of an electrical component"
  SIunits.Voltage v "Potential at the pin";
  flow SIunits.Current i "Current flowing into the pin";
end Pin;
```

- Vid förbindelse genereras automatiskt:

```
Pin1.v = Pin2.v
Pin1.i + Pin2.i = 0
```


Delmodeller och ärvning

- Gemensamma egenskaper i delmodeller

```
partial model TwoPin "Component with two electrical pins p and n "
  SIunits.Voltage v "Voltage drop between the two pins (= p.v - n.v)";
  Pin p, n;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
end TwoPin;
```


Ekvationer och icke-kausal modellering

- Ekvationer används istället för tilldelningar.
- In- och utsignaler definieras inte.
- Kausaliteten bestäms då ekvationssystemet löses. -> Icke-kausal modellering.

Icke-kausalt	**Kausalt**
R * I = V	I: = V / R
	V:= R * I

Ärvning och parametrar

```
model Resistor "Ideal resistor"
  extends TwoPin;
  parameter SIunits.Resistance R;
equation
  R * p.i = v;
end Resistor;
```

- Extends: variabler, ekvationer och gränssnitt ärvs.
- Parameter: konstant under simuleringen, ändras lätt mellan simuleringar.

Lösning och simulering

- DAE-systemet sorteras i beräkningsordning
- Ekvationerna förenklas om möjligt.
- Tilldelningar skapas genom att obekanta löses ut algebraiskt.
- Endast algebraiska loopar löses numeriskt.
- C-kod genereras och kompileras.

Exemplets ekvationssystem

```
Ground1.p.v = 0;
Resistor1.v = Resistor1.p.v-Resistor1.n.v;
0 = Resistor1.p.i+Resistor1.n.i;
Resistor1.i = Resistor1.p.i;
Resistor1.R*Resistor1.i = Resistor1.v;
Resistor2.v = Resistor2.p.v-Resistor2.n.v;
0 = Resistor2.p.i+Resistor2.n.i;
Resistor2.i = Resistor2.p.i;
Resistor2.R*Resistor2.i = Resistor2.v;
Capacitor1.v = Capacitor1.p.v-Capacitor1.n.v;
0 = Capacitor1.p.i+Capacitor1.n.i;
Capacitor1.i = Capacitor1.p.i;
Capacitor1.i = Capacitor1.C*der(Capacitor1.v);
Inductor1.v = Inductor1.p.v-Inductor1.n.v;
0 = Inductor1.p.i+Inductor1.n.i;
Inductor1.i = Inductor1.p.i;
Inductor1.L*der(Inductor1.i) = Inductor1.v;
SineVoltage1.v = SineVoltage1.p.v-SineVoltage1.n.v;
0 = SineVoltage1.p.i+SineVoltage1.n.i;
SineVoltage1.i = SineVoltage1.p.i;
```

```
for i in (1:SineVoltage1.signalSource.nout) loop
  SineVoltage1.signalSource.outPort.signal[i] =
SineVoltage1.signalSource.p_offset
  [i]+(if time <
SineVoltage1.signalSource.p_startTime[i] then 0 else
SineVoltage1.signalSource.p_amplitude[i]*sin(2*Sine
Voltage1.signalSource.pi*
  SineVoltage1.signalSource.p_freqHz[ i]*(time-
SineVoltage1.signalSource.p_startTime
  [i])+SineVoltage1.signalSource.p_phase[i]));
end for;
SineVoltage1.v =
SineVoltage1.signalSource.outPort.signal[1];
Capacitor1.n.i+Ground1.p.i+Inductor1.n.i+SineVoltag
e1.n.i = 0;
Ground1.p.v = Capacitor1.n.v;
Inductor1.n.v = Capacitor1.n.v;
SineVoltage1.n.v = Capacitor1.n.v;
Capacitor1.p.i+Resistor1.n.i = 0;
Resistor1.n.v = Capacitor1.p.v;
Inductor1.p.i+Resistor2.n.i = 0;
Resistor2.n.v = Inductor1.p.v;
Resistor1.p.i+Resistor2.p.i+SineVoltage1.p.i = 0;
Resistor2.p.v = Resistor1.p.v;
SineVoltage1.p.v = Resistor1.p.v;
```

Standard bibliotek

- Matematik funktioner
- SI-enheter
- Komponenter
  - Block (input / output)
  - El
  - Mekanik
  - Div kommersiella

Verktyg med Modelica

- Dymola
- MathModelica

Referenser

[1] Modelica home page http://www.modelica.org

[2] H. Elmqvist, S-E. Mattson. Modelica - the next generation modeling language an international design effort. In *Proceedings of the 1st World Congress on System Simulation (WCSS '97)*, Singapore, 1-3 Sept, 1997.

[3] S-E. Mattson, H. Elmqvist. An overview of the modeling language Modelica. *Eurosim '98 Simulation Congress*, Helsinki, Finland, 14-15 April, 1998.

[4] P. Fritzson, V. Engelson. Modelica - a unified object-oriented language for system modeling and simulation. *12th European Conference on Object-Oriented Programming (ECOOP'98)*, Brussels, Belgium, 20-24 July, 1998.