Aplanado eficiente de grandes modelos Modelica

Mariano Botta

FCEIA, UNR

Agosto 2015

Contenido de la charla

Motivaciones

- Modelado, Simulación y Control en Tiempo Real con Aplicaciones en Electrónica de Potencia.
- Simulación en paralelo utilizando los métodos de cuantificación de estado.
- Modelos grandes.
- Aprovechar las ventajas de Modelica.

Contenido de la charla

Modelica

- Orientado a Objetos.
- Modelado de sistemas complejos, con componentes mecánicos, eléctricos, electrónicos, hidráulicos, térmicos, etc.
- Desarrollado por la asociación sin fines de lucro “Modelica Asociation”.
- Entornos de desarrollo: OpenModelica, MathModelica, Dymola, etc.
- El modelo esta en texto plano

Clases

- Define un objeto.
- Son instanciadas mediante la definición de variables.
- Tienen tres secciónes:
  1. Definiciones.
  2. Ecuaciones.
  3. Sentencias.

```
1  class X
2      // Definiciones de variables y
           clases
3  equation
4      // Ecuaciones
5  statements
6      // Sentencias
7  end X;
```

Prefijos de Clases

Prefijos de clase: *model*, *record*, *block*, *connector*, *function*, *package*

- Mejoran la lectura del código:
- Agregan restricciones a la clase

```
1  class Circuits
2      cclass Pin
3          Real v;
4          flow Real i;
5      end Pin;
6      class Componente
7          Pin n,p;
8      equation
9          n.v = p.v;
10     end Componente;
11 end Circuits;
```

```
1  package Circuits
2      connector Pin
3          Real v;
4          flow Real i;
5      end Pin;
6      model Componente
7          Pin n,p;
8      equation
9          n.v = p.v;
10     end Componente;
11 end Circuits;
```

Herencias de Clases

- Agrega significado semantico al modelo.
- Facilita la reutilización de código.
- Se utiliza la palabra reservada: *extends*.

```
1  model OnePort
2      Pin p;
3      Pin n;
4      Real v;
5      Real i;
6  equation
7      v = p.v - n.v;
8      i = p.i;
9      i = -n.i;
10 end OnePort;
11 model Capacitor
12     extends OnePort;
13     parameter Real C = 1;
14 equation
15     C * der(v) = i;
16 end Capacitor;
```

Declaraciones de tipo

- Tipos básicos: *Real*, *Integer*, *Boolean* y *String*
- Las clases definen un nuevo tipo.
- Sinónimos de tipos:
  type Nombre = [Prefijos] Tipo-Existente [Array] [Modificaciones]

Prefijos de Tipo: *flow*, *constant*, *parameter*, *discrete*, *input* y *output*.

```
1 package Circuits
2     type Current = flow Real;
3     type Voltage = Real;
4     connector Pin
5         Voltage v;
6         Current i;
7     end Pin;
8     type TenPin = Pin[10];
9 end Circuits;
```

Definiciones de variables

1. **Prefijos de tipos**: *flow*, *constant*, *parameter*, *discrete*, *input* y *output*.
2. **Tipo**: Nombre del tipo de la variable. Puede ser un tipo básico, una clase o un sinónimo de tipo. Ejemplo: *Real*, *String*, *Pin*, *TenPin*.
3. **Nombre de la variable**.
4. **Dimensión**: Modelica permite la definición de arreglos.
5. **Modificaciones**.

```
1    type TenPin = Pin[10];
2
3    TenPin pines;
4    Pin pines2 [10];
```

Modificaciones

Aparecen en:

1. Declaraciones de variables.
2. Sinónimo de tipo.
3. Definiciones de herencia.

Se permite:

1. Cambiar el valor inicial de una variable.
2. Anidar modificaciones.
3. Alterar la definición de una variable.
4. Cambiar la definición de un tipo.

```
1  package Circuits
2      model CircuitX
3          Capacitor cap;
4          Resistor res;
5          ...
6      equation
7
8      ...
9
10     end CircuitX;
11     model MainCircuit
12         Capacitor x(C = 2);
13         CircuitOne co1 (cap(C = 10));
14         CircuitOne co2 (cap(C = 15));
15     end MainCircuit;
16 end Circuits;
```

Ecuaciones de igualdad

Las ecuaciones no representan una asignación, sino igualdades. Pueden tener expresiones complejas de ambos lados de la igualdad y expresan una relación entre las variables. Se definen dentro de la sección *equation*.

Igualdades

```
1    p.v - n.v = v;
2    i = p.i;
3    i = -n.i;
```

Ecuacón For

```
1    for i in 1:N loop
2        v[i] = p[i].v - n[i].
             v;
3        i[i] = p[i].i;
4        i[i] = -n[i].i;
5        C[i] * der(v[i]) = i;
6    end for;
```

Ecuaciones Connect

Los conectores:

- Son clases con ciertas restricciones.
- Se definen con el prefijo *connector*.
- No tienen ecuaciones.
- Tienen variables de dos tipos:
  1. Variables de potencial. Ejemplo: presión, voltaje, etc.
  2. Variables de flujo: definidas con el prefijo flow. Ejemplo: corriente, caudal, etc.
- Ejemplo: Clase *Pin*.

Ecuaciones Connect

Las ecuaciones *connect*:

- Conectan dos clases del mismo tipo.
- Genera relaciones entre las variables internas de los conectores:
  - Las variables de potencial dentro de una misma conexión deben ser iguales entre sí.
  - Las variables de flujo siguen las reglas de Kirchhoff: la suma de los flujos es igual a cero. Para mantener esta regla hay que considerar como flujo positivo aquel que tenga dirección hacia dentro del componente. En caso contrario, será considerado negativo.

Ecuaciones Connect

```
1  package Circuits
2  ...
3  model ground
4  Pin p;
5  equation
6  p.v = 0;
7  end ground;
8  model inductor
9  extends OnePort;
10 parameter Real L = 1;
11 equation
12 L * der(i) = v;
13 end inductor;
14 model LC_circuit
15 Capacitor cap(v(start = 1));
16 inductor ind(L = 2);
17 ground gr;
18 equation
19 connect(ind.p,cap.p);
```

```
1 // Variables de Potencial
2 ind.p.v = cap.p.v;
3 ind.n.v  = cap.n.v;
4 cap.n.v = gr.p.v;
5
6 // Variables de flujo
7 ind.p.i + cap.p.i = 0;
8 ind.n.i + cap.n.i + gr.p.i = 0;
```