The use of the UML within the modelling process of Modelica-models

Christoph Nytsch-Geusen
Contact: christoph.nytsch@first.fraunhofer.de
Overview

- **UML\(^H\) and Modelica**
  - Class diagrams
  - Collaboration diagrams
  - Statechart diagrams

- **Example for UML\(^H\)-modelling**
  - Model of a Pool-Billiard game
  - Simulation experiment

- **Simulation tool MOSILAB**
  - IDE for UML\(^H\)-modelling
Motivation

- **UML**: Unified Modeling Language for Hybrid systems
- **Advantages for UML in the Modelica context**
  - UML offers different views on OO-models
    1. Class diagrams
    2. Collaboration diagrams
    3. Statechart diagrams
  - Modelling of complex systems mostly based on complex model structures
  - UML-IDEs can generate the “basic” Modelica-code

Class diagram of a hygrothermal wall model

Statechart diagram of a string pendulum
UML$^H$: Class diagrams

1. **Class types**: Model, Block, Connector, …
2. **Class attributes**: Variables, Parameter
3. **Class relations**: Inheritance, Composition

## Modelica code

```modelica
package UML_H

annotation(UMLH(ClassDiagram="<umlhclass><name>…));
class A(annotation(UMLH(classPos=[31,53])));
end A;
model A1 annotation(Icon(Text(extent=…,string="A1", _));
  annotation(UMLH(classPos=[31,146]));
  extends A;
  event Boolean on;
  event Boolean off;
  Real x;
  input Real z;
  parameter Real y;
  C c;
  ...
end A1;
...
connector C annotation(UMLH(classPos=[192,54]));
  Real u;
  flow Real i;
end C;
...
end UML_H;
```

## UML$^H$-class diagram

![UML$^H$-class diagram](image-url)
UML$^H$: Component diagrams

Different connection types

1. Connector variables
   (thin black line with filled squares at the ends)

2. Scalar variables
   (thin blue line with unfilled squares at the ends)

3. Scalar input/output variables
   (thin blue line with an arrow and an unfilled square)

4. Mixture connection types of 1. to 3. (fat blue line)

Modelica code

```model System
  annotation(CompConnectors(CompConn(label="label2", 
      points=[-81,52; -81,43; -24,43; -24,51]));
  UML_H.A1 c1 annotation(extent=[-87,72; -74,52]);
  UML_H.A1 c2 annotation(extent=[-57,71; -44,51]);
  UML_H.A1 c3 annotation(extent=[-30,71; -18,51]);
  UML_H.B b annotation(extent=[-57,91; -44,77]);
  equation
    // connection type 1:
    connect(c1.c,c2.c) annotation(points=[-74,62;-57,62]);
    // connection type 2:
    c2.y=c3.y annotation(points=[-44,62; -30,62]);
    // connection type 3:
    b.y=c1.z annotation(points=[-57,84; -79,84; -79,72]);
    // connection type 4 (mixture of type 1 and 2):
    connect(c1.c,c3.c) annotation(label="label2");
    c1.x=c3.x annotation(label="label2");
  end System;
```
UML\(^H\): Statechart diagrams

Different state types

1. **Initial states** (black filled circle)
2. **Final states** (point in an unfilled circle)
3. **Atomic states** (flat internal structure)
4. **Normal states** (can contain additional entry or exit actions and can be sub-structured in further statechart diagrams)

![Modelica code](UMLH-Statechart diagram)

```model A1 ...
statechart
  state A1SC extends State annotation(extent=[-88,86; 32,27]);
  state State1 extends State;
  exit action x:=0; end exit;
end State1;
State1 state1 annotation(extent=[-66,62; -41,48]);
State A3 annotation(extent=...);
State I5(isInitial=true)...;
State F7(isFinal=true)...;
  transition I5->state1 end transition
    annotation(points=[-76,73;-64,71; -64,62]);
  transition 11:state1->A3 event on action x:= 2.0;
end transition annotation(points=...);
  transition 12:A3->state1 event off guard y < 5
    action x:=3.0;
end transition annotation ...
  transition state1->F7 end transition annotation...
end A1SC;
end A1;```

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

EOOLT-Workshop, ECOOP 2007, Technische Universität Berlin 30.7.2007
Example for $\text{UML}^H$-modelling: Model of a Pool-Billiard game (1)

**Model assumptions**

1. The Pool-Billiard game knows only a black (bb), a white (bw) and a coloured ball (bc).
2. The table (t) has only one hole instead of 6 holes.
3. The collision-model is strong simplified.
4. The balls are moving between the collisions and reflections only on straight directions in the dimension $x$ and $y$.
5. The reflections on the borders take place ideal without any friction losses.
6. The rolling balls are slowed down with a linear friction coefficient $f_r$:

   \[
   m \cdot \frac{dv_x}{dt} = -v_x \cdot f_r \quad m \cdot \frac{dv_y}{dt} = -v_y \cdot f_r
   \]

UML$^H$-class diagram for the ball model
Example for UML\textsuperscript{H}-modelling: Model of a Pool-Billiard game (2)

Model events on the ball model-level:
1. Reflection on the left border (reflection\_left)
2. Reflection on the top border (reflection\_top)
3. Reflection on the right border (reflection\_right)
4. Reflection on the lower border (reflection\_down)

```plaintext
model Ball
  extends MassPoint(m=0.2);
  parameter SIunits.Length width, length;
  parameter SIunits.Length d = 0.0572 "diameter";
  parameter Real f_r = 0.1 "friction coefficient";
  SIunits.Velocity v_x, v_y;
  event Boolean reflection_left(start = false);
  ...
  equation
    reflection_left = if x < d/2.0;
    m * der(v_x) = - v_x * f_r; der(x) = v_x;
    ...
  statechart
    state BallSC extends State;
      State Rolling;
        State startState(isInitial=true);
        transition startState -> Rolling end transition;
        ...
        transition Rolling->Rolling event reflection_left
          action v_x := -v_x; x := d/2.0;
          end transition;
      end BallSC;
  end Ball;
```

UML\textsuperscript{H}-Statechart diagram for the ball model
Example for UML$^H$-modelling: Model of a Pool-Billiard game (3)

Model events on the system model-level

1. Collision of two balls
   - bb / bc; bb / bw; bw / bc
2. Disappearance of a ball in the hole
   - bb, bw and bc

UML$^H$-Statechart diagram for the system model

```plaintext
model System
  parameter SIunits.Length d_balls = 0.0572;
  parameter SIunits.Length d_holes = 0.15;
  dynamic Ball bw, bb, bc;  //structural dynamic submodels
  Table t(width = 1.27, length = 2.54);
  event Boolean disappear_bw(start = false);
  event Boolean disappear_bb(start = false);
  event Boolean disappear_bc(start = false);
  event Boolean collision_bw_bb(start = false);
  ...
  event Boolean push(start = false);

  equation
    push = if fabs(bw.v_x)<0.005 and fabs(bw.v_y) < 0.005;
    disappear_bw = if((p[1].x-0)^2+(p[1].y-0)^2)^0.5 < d_holes;
    collision_bw_bb = if((p[2].x-p[1].x)^2 + (p[2].y-p[1].y)^2)^0.5 < d_balls;
```

Example for UML$^H$-modelling: Model of a Pool-Billiard game (4)

**Model transition on the system model-level**

1. **Initial transition**
   initialization of the balls and their positions

2. **Playing $\rightarrow$ Playing**
   triggered by collision or disappearance events

3. **Playing $\rightarrow$ GameOver**
   triggered by the disappearance event of bb

**UML$^H$-Statechart diagram for the system model**

```
statechart
state SystemSC extends State;
State Playing, startState(isInitial=true), GameOver; ... 
transition startState -> Playing action
   bw := new Ball(d = d_balls,...); add(bw);
   bb := new Ball(...); add(bb);
   bc := new Ball(...); add(bc);
end transition;
transition Playing->Playing event disappear_bw action
   ... remove(bw);
   bw := new Ball(x(start=1.27/2.9), y(start=0.6));
end transition;
transition Playing->Playing event disappear_bc action
   ... remove(bb);
end transition;
transition Playing -> GameOver event disappear_bb
end transition;
transition Playing->Playing event collision_bw_bb action
   v_x := bw.v_x; v_y := bw.v_y;
   bw.v_x := bb.v_x; bw.v_y := bb.v_y;
   bb.v_x := v_x; bb.v_y := v_y;
end transition;
end SystemSC;
...```
Example for UML$^H$-modelling: Simulation experiment (1)

- Simulation experiment
  - Duration: 4 seconds
  - Event sequence:
    1. bw hits on bb
    2. bb reflects on the left and the lower border
    3. wb disappears in the hole
Example for UML$^H$-modelling: Simulation experiment (2)
Example for $\text{UML}^H$-modelling: Simulation experiment (2)

Collision events:
- white and black ball

Reflection events:
- white ball
- black ball

$x$- and $y$-positions of the white and the black ball
MOSILAB-IDE for model based development (GENSIM-Project)

Graphical Editors (UML\(^{4}\)):
- Class diagrams
- Component diagrams
- Statechart diagrams

Text Editor (Modelica)

Class Browser

Component Browser

Development Workflow
Summary

- UML\textsuperscript{H} offers three model views on OO-Modelica-models
- The modelling example of the Pool-Billiard game demonstrates the advantages of UML\textsuperscript{H}-modelling
- The Modelica-tool MOSILAB supports code generation starting from UML\textsuperscript{H}-models