Modeling Structural - Dynamics Systems in MODELICA/Dymola, MODELICA/Mosilab and AnyLogic

Günther Zauner[1,2], Daniel Leitner[3], Felix Breitenecker[1]

[1]Vienna University of Technology, Wiedner Hauptstr. 8-10, 1040 Vienna, Austria
[2]Drahtwarenhandlung – Simulation Services, Neustiftgasse 57-59, 1070 Vienna
[3]Austrian Research Centers GmbH – ARC, Biomedical Engineering,
Donau-City Straße 1/8, 1220 Vienna
guenther.zauner@drahtwarenhandlung.at

Abstract. With the progress in modeling dynamic systems new extensions in model coupling are needed. The models in classical engineering are described by differential equations. Depending on the general condition of the system the description of the model and thereby the state space is altered. This change of system behavior can be implemented in different ways. In this work we focus on three state-of-the-art DAE simulation environments, Dymola, Mosilab and AnyLogic, and compare the possibilities of coupling of different state spaces. This can be done either using a parallel model setup, a serial model setup, or a combined model setup. The analogies and discrepancies are figured out on the basis of the classical constrained pendulum as defined in ARGESIM comparison C7.

Keywords: Structural dynamics, state charts, Dymola, Mosilab, AnyLogic

1 Introduction

In the last decade the increase of computer power and the apace growth of model complexity leads to a new generation of simulation environments. Concurrently ambitions pointed towards establishing standardization. Especially the Modelica organization develops a wide range of syntax description and standard libraries.

This paper will compare the solutions of the constrained pendulum as an easy to model example, implemented in the most common Modelica simulator Dymola, Mosilab, a product from six Fraunhofer Institutes which uses Modelica syntax with extensions for state charts, and the simulator AnyLogic from Xjtek in St.Petersburg. This simulator also has object oriented structure and is fully implemented in JAVA.

We will focus on how the model can be implemented and we will have a look in which time slot the state events are and if there is a significant difference referring to the implementation method.

2 Model

The constrained pendulum is a classical nonlinear model in simulation techniques. This model has been presented in the definition of ARGESIM comparison C7 [1]. There is no exact analytical solution to this problem. Therefore, the results have to be obtained by numerical methods. In this section a description of the model will be given.

The motion of the pendulum is given by

$$ml\ddot{\varphi} = -mg\sin(\varphi) - dl\dot{\varphi} \tag{1}$$

Where φ denotes the angle measured in counter clockwise direction from the vertical position. The parameter *m* is the mass and *l* is the length of the pendulum. The damping is realized with the constant *d*.

In the case of a constrained pendulum a pin is fixed at a certain position given by the angle φ_p and the length l_p. If the pendulum is swinging it may hit the pin. In this case the pendulum swings on with the position of the pin as the point of rotation and the shortened length $l_s = l - l_p$.
Two experiments have been defined. The first one is starting in the long pendulum modus and is swinging towards the pin. The second experiment is a model where the starting conditions are set in a way that the pendulum is shortened in the beginning of the simulation run.

3 Simulation Environments

In this section the focus is on three simulation environments. Two simulators, namely Dymola and Mosilab, are based on the model description standard Modelica [2]. Modelica is a freely available, object-oriented language for modeling of large, complex, and heterogeneous physical systems.

One of its most important features is non-causal modeling. In this modeling paradigm, users do not specify the relationship between input and output signals directly (in terms of a function), but they rather define variables and the equations that must be satisfied.

It is suited for multi-domain modeling, for example, mechatronic models in robotics, automotive and aerospace applications involving mechanical, electrical, hydraulic and control subsystems, process oriented applications. Modelica is designed that it can be utilized in a similar way as an engineer builds a real system: first trying to find standard components like motors, pumps and valves from manufacturers catalogues with appropriate specifications and interfaces and only if there does not exist a particular subsystem, a component model would be newly constructed based on standardized interfaces.

The actual version of the Modelica Standard Library is 2.2.1, which has been released in April 2006.

3.1 Dymola

Dymola, DYnamic MOdeling LAboratory, is an environment for modeling and simulation of integrated and complex systems. It has unique multi-engineering capabilities which mean that models can consist of components from many engineering domains.

The basic structure of the simulator is divided into two separate parts: the Modeling layer and the Simulation layer. Thereby the modeling layer is separated in three parts. One part, the so called ICON layer, is used to define the shape of the new defined blocks. The DIAGRAM layer is the interface for graphical modeling. The third plane is the MODELICA TEXT part where the Modelica source code can be implemented directly.
Dymola has a strong focus on using symbolic methods for mass-matrix inversion and equation sorting.

Integration algorithms for non-real-time simulation typically handle discontinuities by detecting when certain variables cross a boundary. They then calculate the time of the event by iteration and then change the step size to advance the time exactly to the time of the event (crossing) [3].

The default integration method is the Dassl code as defined by Petzold. The method can also be freely chosen out of 15 standard solvers, including algorithms for stiff systems. There is until now no possibility implemented to make graphical model switching for subsystems with different state space dimension.

3.2 Mosilab

The simulator Mosilab (MOdeling and SImulation LABoratory) is an environment developed from the Fraunhofer-Institutes FIRST, IIS/EAS, ISE, IBP, IWU and IPK in the research project GENSIM.

It has been developed for time-continuous and time-discrete analysis of heterogeneous technical systems. The main innovation from point of simulation techniques view in this simulator is the illustration of condition-based changes in the model structure (model structure dynamics). With this mechanism it is possible develop and simulate models with different modeling depth.

The model description in general is done in the Modelica standard. Additional features to assure high flexibility during modeling and the concept of structural dynamics is implemented. This is done by extending the Modelica standard with state charts, controlling dynamic models. The extended object-oriented model description language resulting is called MOSILA [1,4] Moreover simulator coupling with standard tools (e.g. MATLAB / Simulink, FEMLAB) is realized.

Code generation is done in a quite similar way as in Dymola/Modelica. This makes sense, because this relatively new simulator will also be able to simulate problems defined in the standard Modelica notation with other tools, which use the same syntax. The main difference is the extension for graphical representation of state charts. This is solved with an interface where the user can define UML state charts.

The analysis part of the model is split into two layers: the simulation and the post processing layer. The defined code is translated into C++. The default integration method is the so called idadassl. Other implemented methods are for example the implicit trapeze method and the explicit or implicit euler.

3.3 AnyLogic

AnyLogic is a multiparadigm simulator supporting Agent Based modeling as well as Discrete Event modeling, which is flowchart-based, and System Dynamics, which is a stock-and-flow kind of description. Due to its very high flexibility AnyLogic is capable of capturing arbitrary complex logic, intelligent behavior, spatial awareness and dynamically changing structures. It is possible to combine different modeling approaches making AnyLogic a hybrid simulator. AnyLogic is highly object oriented and based on the Java programming language. To a certain degree this ensures a compatibility and reusability of the resulting models.

The development of AnyLogic in the last years has been towards business simulation. In version 6 of AnyLogic it is possible to calculate problems from engineering, but there are certain restrictions. For example the integration method cannot be chosen freely and there is no state event finder.

When a model starts, the equations are assembled into the main differential equation system. During the simulation, this DES is solved by one of the numerical methods built in AnyLogic. AnyLogic provides a set of numerical methods for solving ordinal differential equations (ODE), algebraic-differential equations (DAE), or algebraic equations (NAE).

AnyLogic chooses the numerical solver automatically at runtime in accordance to the behavior of the system. When solving ordinal differential equations, it starts integration with forth-order Runge-Kutta method with fixed step. Otherwise, AnyLogic plugs in another solver – Newton method. This method changes the integration step to achieve the given accuracy.

4 Solution methods

New advantages in computer numerics and the fast increase of computer capacity lead to necessity of new modeling and simulation techniques. In many cases of modern simulation problems state events have to be handled.
There exit more or less different categories of structural dynamic systems which should be focused on and solved.

The first class of hybrid systems are the one, where the state space dimension does not change during the whole simulation time and also the system equations stay the same. Only so called parameter events occur at discrete time points. These are the more or less simplest form of state events. Modern simulators offer different solution methods. A Part of them have a *discrete section* or as implemented in Dymola and Mosilab a so called *algorithm section*. In this part the user can define the parameter value change using the commands *when*, *if*, etc.. In this section the use of acausal modeling has to be switched of. This means that we have to make assignments for the parameter values at time point the event occurs.

Furthermore many software environments support the usage of UML state charts. This is a very intuitive and convenient way to describe a system which contains multiple discrete states. In the combination with dynamical equations this approach enables a simple implementation of structural dynamics. The dynamic equations or parameters are dependent of the discrete state of the model. On the other hand the states can be altered in dependence of the dynamic variables.

In case of the constrained pendulum the states are normally swinging (state 'long') or swinging with shortened length around the pin (state 'short'). The discrete state of the model depends on the angle φ and the pins angle φ_p. The state alters the model parameters or the models set of equations, see figure 1.

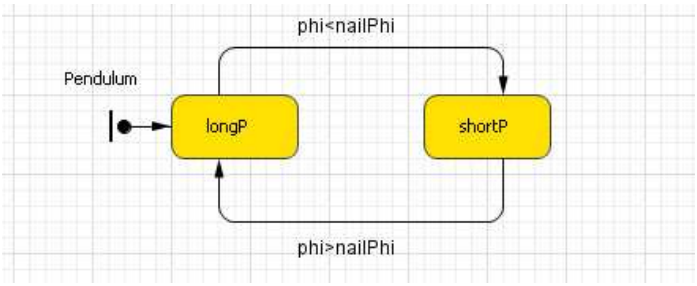

Figure 1: *UML state diagram controlling the pendulum*

4.1 Switching states

When the state of a system changes, often the state space of the model stays unchanged, thus the same set of differential equation can be used for different states. In this situation only certain parameters must be changed when a state is entered.

In case of the constrained pendulum the differential equation for movement stays the same for both states ‘long’ and ‘short’. If the state changes the parameter length and angular velocity are updated before the calculation can continue, see figure 2.

Figure 2: *The parameters of the model are changed by an UML state diagram.*

4.2 Switching models

Often the previous approach is not possible. Sometimes situation occur where the state space of the model changes, thus a simple change of parameters is not possible. Normally the whole set of differential equations, thus the complete model, must be changed. In many simulation environments this approach can lead to complication.

In case of the constrained pendulum two differential equations are set up describing the movement of the pendulum. One describes the normal pendulum the other one the shortened pendulum. Which equation is set to be active is determined by the state diagram. When the states are switched the initial values must be passed on the equation must be activated and the other one must be frozen, see figure 3.

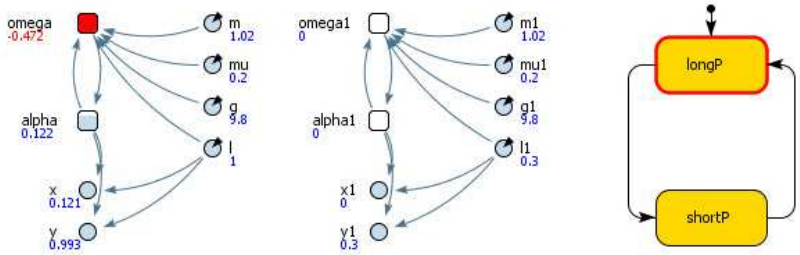

Figure 3: *The differential equations of the system are switched in dependence of the UML state diagram.*

5 Dymola

The implementation of the constrained pendulum has been done in two more or less different ways. As Dymola does not support the UML – notation for state charts and there is in the moment no method implemented to switch between two or more independent models during one simulation run, the solution methods described in section 4.1 and 4.2 can not be used.
In our example the state event, which appears every time when the rope of the pendulum hits the pin or looses the connection to it, is modeled in an *algorithm* section. This can be done with the following code digest:

```
algorithm
if (phi<=phipin) then
    length:=ls;
end if;
if (phi>phipin) then
    length:=l1;
end if;
```

Another method for implementing the constrained pendulum in Dymola is the use of standard blocks in combination with a predefined model which includes the equations or using only the *Modelica.Blocks* components.

In this example the solution is made by using standard blocks with little extension. Figure 4 shows a screenshot of the Diagram layer of this model.

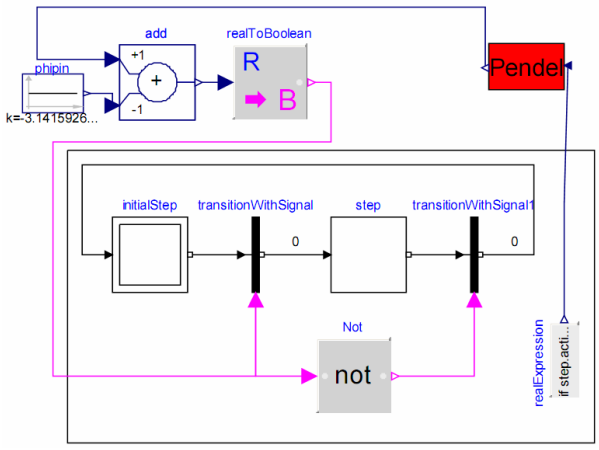

Figure 4: *The screenshot of the Diagram layer in Dymola/Modelica*

The main difference is that no algorithm section is used in the model. The lower part of the system shown in figure 4 is solved with *Modelica.StateGraph* library.

The simulations are done for both tasks and the solutions are compared. This is done by plotting all the results in one picture. The time of the last event in task a (figure 5) is in both cases the same, namely 6.72198 seconds. There is no easy possibility to plot the difference of special variables from different simulation runs.

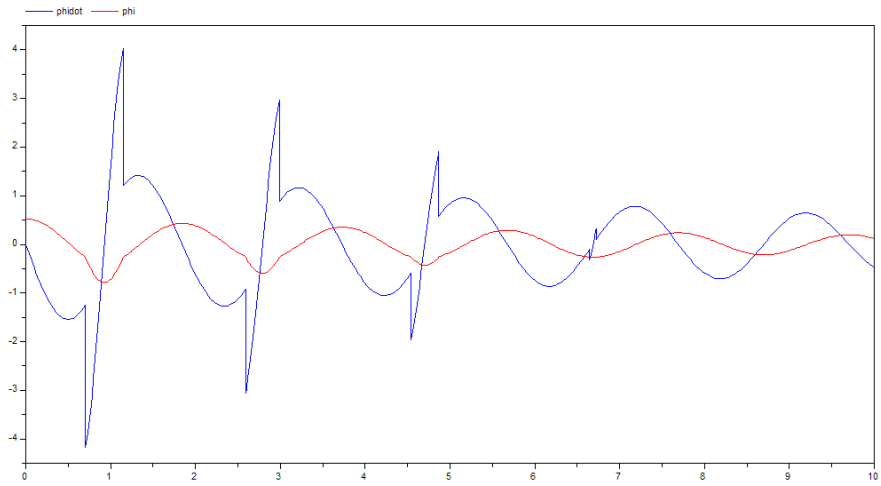

Figure 5: *angle (red) and angular velocity (blue) as a function of time [s] as described in chapter 2*

The same model has to be checked with other starting values. This is done in next step. The figure 6 shows the plot for starting angle $\varphi = -\pi/6$ instead of $\pi/6$.

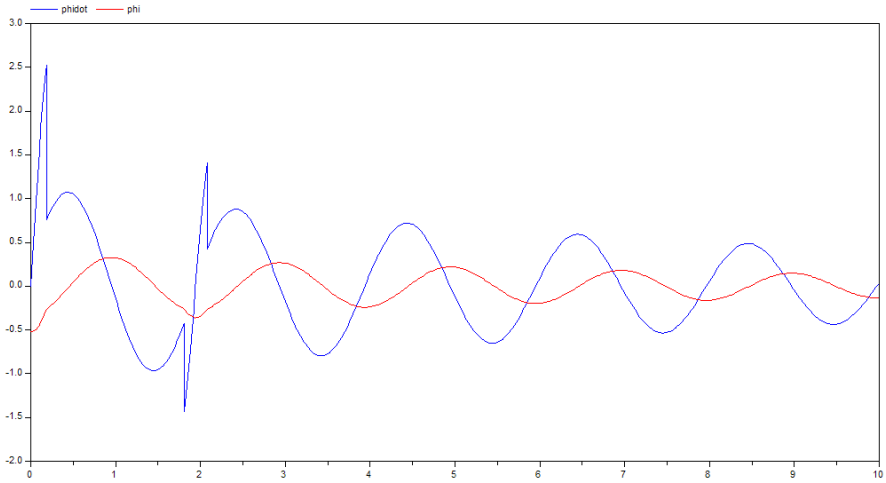

Figure 6: *angle (red) and angular velocity (blue) as described in chapter 2*

6 Mosilab

Similar to the way the solutions in Dymola were calculated, the system can be solved with Mosilab. But as mentioned before, this structure can not handle changes in the state space dimension. The implemented Modelica extension enables the handling of discrete elements as well as structure changes in the general description.

We focus on two different solution methods for the constrained pendulum.
First approach: State charts may be used instead of if- or when- clauses (similar to 4.1 Switching states), with much higher flexibility and readability in case of complex conditions. Boolean variables define the status of the system and are managed by the state chart. The most important part of the source code is as follows:

```
equation
lengthen=(phi>phipin);  shorten=(phi<=phipin);
..  here /*pendulum*/  -equations
statechart
state  LengthSwitch  extends  State;
         State  Short,Long,Initial(isInitial=true);
transition  Initial  ->  Long  end  transition;
transitionLong->Shortevent  shorten  action
         length  :=  ls;
end  transition;
transitionShort->Longeventlengthen  action
         length  :=  l1;
end  transition;
end  LengthSwitch;
```

From the modeling and mathematical point of view, this description is equivalent to the description with if-clauses. The question is, how the Mosilab translator generates the implementation of the equations in both cases. The Mosilab/Modelica simulator performs simulation by handling the state event within the integration over the simulation horizon.

Second approach: These models are the conversion of concepts from chapter 4.2, which is switching models into Mosilab notation. For the constrained pendulum, we decompose the system into two different models, a *short* and a *long* pendulum model, controlled by a state chart. This can again be done with graphical aid in the form of UML-diagrams.
In the development status at the end of 2006, there still occurred several problems with the graphical interface of the state chart layer. The functionality of the system is not restricted.The results are similar to the solutions done with Dymola/Modelica.

7 AnyLogic

The implementation of the constrained pendulum has been done in two different ways. In the first approach only the parameter states have been switched corresponding to section 4.1, in the second approach the whole differential equation is switched corresponding to section 4.2. Both examples from chapter 2 have been calculated with both approaches. The results in AnyLogic are identical in both methods because the times of the state transitions are the same.

In the first approach the model consists of two ordinary differential equations describing the movement of the pendulum. In these equations four parameters are used length *l*, mass *m*, damping *d*, and gravity *g*. Further a state diagram with states 'long' and 'short' and two transitions are used to update the equations. When the state changes length *l* and angular velocity ω are updated. The results calculated by AnyLogic 6 are plotted in figure 7 and figure 8.

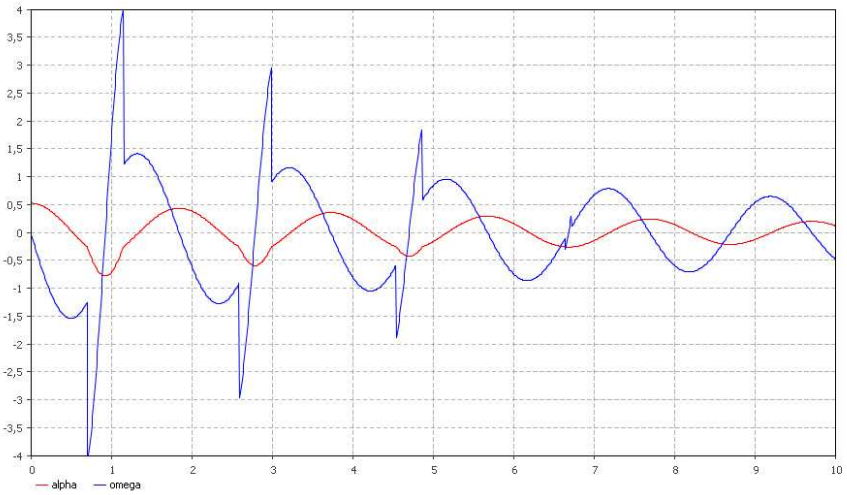

Figure 7: *Results for example 1: angle (red), angular velocity (blue)*

The second approach uses two separate models. The implemented model consists of two times two ordinary differential equations. Both equations have four parameters separately: length *l*, mass *m*, damping *d*, and gravity *g*. A state diagram is implemented analog to the first approach. If the state changes the right differential equations are activated and their initial values are set, while the other differential equation is frozen.

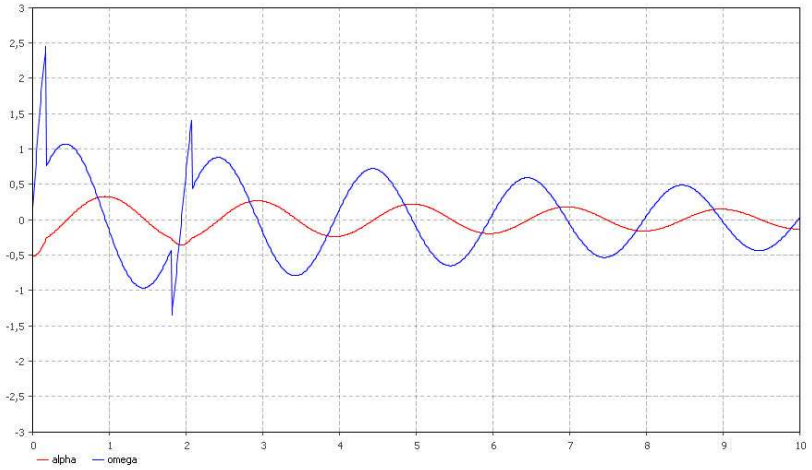

Figure 8: *Results for example 2: angle (red), angular velocity (blue)*

8 Discussion

For this nonlinear model, there exists no exact solution. For this reason we can only calculate the numerical solutions and compare, for example, the time points where the last state event appears. This is the moment when the rope of the pendulum looses the connection to the pin the last time. In the first model under investigation, this happens after the fourth time shortening the pendulum, which means after eight state events all together. In the second simulation run, this occurs earlier, namely already after two times lengthening the rope, which means after three state events, because of the special initial condition (pendulum is in short modus at starting time).

The solutions are calculated with the default simulation method, if possible. With this approach we try to test the simulation environments from the user's point of view. Many programmers and modelers do not care that much about the implemented integration methods. For this reason the standard method has to produce reliable results in an appropriate calculation time.

The solution in the Mosilab simulator with standard Modelica components cannot be calculated with the standard method (*Dassl* code), because during simulation of this task a numerical error occurs and therefore the calculation is interrupted. The integration method pins at the time point of the first state event. Because of this reason the *Implicit Trapez* method was chosen. The other results are all done with the standard integration method and the given step sizes/number of intervals.

Table 1. End time of the last shortening of the pendulum for example 1

Simulator	Simulation	Method
Dymola/Modelica	6.72198	Dassl 500 intervals
Mosilab/Modelica Switch models	6.7204	IDA Dassl Min. step 1e-6 Max. step 0.08
Mosilab/Modelica Pure Modelica	6.7199	Impl. Trapez Min. step 1e-6 Max. step 1e-4
Mosilab/Modelica Parameter switching	6.7224	IDA Dassl Min. step 1e-6 Max. step 0.08
AnyLogic	6.725	No influence Step size 0.001

Table 1 shows that the solutions with Dymola and Mosilab are equivalent, if the solution is rounded towards two digits after the comma. By contrast, the solution in AnyLogic differs. We can try to explain this difference by taking a look on state event finding. This is not implemented in AnyLogic and is missing as an important standard feature of modern simulation environments. The lack of influence on the numerical methods can be explained by the main field of application of AnyLogic. Its main focus is on production and logistics, not on simulation of DAE systems.

In table 1 we see that there is only one row for Dymola/Modelica. This is because of equivalent results in all three implementations. Also AnyLogic delivers the same result for both methods. As we see, in this case Dymola outperforms Mosilab, because the result does not depend on the way of implementation. On the other hand we cannot implement real structural dynamics without blowing up the state space and problems in starting variable definition.

The graphical user interface for UML diagrams is a big advantage of Mosilab and AnyLogic compared to the possibilities of Dymola. But we have to keep in mind, that this feature is not Modelica standard, which complicates model exchange between different simulators based on Modelica.

References

1. Nytsch-Geusen, C. et. al.: Advanced modeling and simulation techniques in MOSILAB: A system development case study. Proc. of the 5th International Modelica Conference, 2006.
2. Fritzson , P. 2004. *Principles of Object Oriented Modeling and Simulation with Modelica 2.1*., IEEE Press, John Wiley&Sons, Inc., Publication, USA.
3. Hilding Elmquist et. al.: Real-time Simulation of Detailed Automotive Models, Proceedings of the 3rd International Modelica Conference, Linköping, Sweden
4. Thilo Ernst, André Nordwig, Christoph Nytsch-Geusen, Christoph Claus, André Schneider: MOSILA Modellbeschreibungssprache, Spezifikation, Version 2.0, from the homepage: www.mosilab.de/downloads/dokumentation