I. Troch, F. Breitenecker, eds.   ISBN 978-3-901608-35-3

MODELLING STRUCTURAL DYNAMIC SYSTEMS: STANDARD MODELICA VS MOSILAB STATECHART

G. Zauner[1,2], N. Popper[1], F. Breitenecker[2]
[1]"Die Drahtwarenhandlung"- Simulation Services, Vienna, Austria;
[2]Vienna University of Technology, Vienna, Austria;

Corresponding Author: G. Zauner, "Die Drahtwarenhandlung"- Simulation Services, Vienna, Austria
Neustiftgasse 57-59, 1070 Wien, Austria; guenther.zauner@drahtwarenhandlung.at

Abstract. Object-Oriented modelling is a fast-growing area of modelling and simulation that provides a structured, computer-supported way of mathematical and equation-based modelling. Modelica is today the most promising modelling and simulation language in that it effectively unifies and generalizes previous object oriented modelling languages and provides a fundament for the basic concepts.
The Modelica modelling language offers easy to use multi domain physical modelling approaches as well as acausal equation based modelling and simulation. Classical hybrid systems can be implemented using *if-then-els*e or *when* equations, these are conditional equations. But for advanced models dealing with structural dynamic systems the standard does not provide adequate solution methods.
MOSLIAB (***Mo****delling and* ***Si****mulation* ***Lab****oratory*), developed by Fraunhofer-Gesellschaft in a cooperation of six Fraunhofer-Institutes (FIRST, IIS/EAS, ISE, IBP, IWU and IPK) in the research project GENSIM, is a Modelica based multi domain modelling and simulation environment. Besides the classical way of modelling, as defined in the Modelica standard, MOSILAB offers an extension in graphical and textual way for representing structural dynamic systems by using the language extension MOSILA.
The main focus of the presented work lies on the comparison of standard Modelica implementations of structural dynamic systems and the additional possibilities offered by MOSILAB. Using two examples from classical physics, the constrained pendulum and the free pendulum on a string, and one from electronics we show the influence and power of state event modelling using the graphical environment in MOSILAB in combination with the textual description to implement hybrid structures. The graphical hybrid structure layer in MOSILAB is UML (Unified Modelling Language) based, and thereby intuitive handling is supported.

1 Motivation

Modelling and simulation of physical based systems and solution of DAE systems is getting more and more important in modern development and system testing. As the complexity of the systems of interest is growing fast, the importance of new techniques for model coupling and standardization is given.

On the one hand, modern simulation models have to deal with changing conditions and model types during a simulation run because the system of interest includes events. On the other hand the model parts included in a system are often modelled in different depth or a system only has to be modelled in detail while special conditions are fulfilled. This case is somewhat harder to handle, because it includes structural dynamic system problems which cannot be handled as events of type 1 (parameter change) or type 2 (one or more inputs change discontinuously), but have to be implemented as type 3 (one or more states change discontinuously) or type 4 (the dimension of the state vector changes discontinuously) events [3].

Two developments helped to overcome this problem. On modelling level, the idea of physical modelling gave new input and thereby the development of the Modelica [9] standard plays an important role, and on implementation level the object oriented view helped to leave the constraints of input/output relations. Furthermore, UML offers new input for hybrid modelling. In the domain abstract modelling techniques the Unified Modelling Language (UML) [4] is an established standard for development and graphical description of object oriented software systems. It is standardized by ISO (ISO/IEC 19501) and wide spread in the tools for software development and has an intuitive structure. The standard includes a broad basis of graphical diagrams which highlight different aspects of the software in the diverse development phases.

We look at a simulation environment called MOSILAB [7] which is Modelica based and extends this standard with UML H components for modelling and simulation of structural dynamic systems and evaluates the additional features in comparison with pure Modelica.

2 Introduction on physical modelling with Modelica

A typical procedure for physical modelling is to cut a system into subsystems and to account for the behaviour at the interfaces. Each subsystem is modelled by balances of mass, energy and momentum and material equations. The complete model is obtained by combining the descriptions of the subsystems and the interfaces. This approach requires a modelling paradigm different to classical input/output modelling. A model is considered as a constraint between system variables, which leads naturally to DAE descriptions. The approach is very convenient for building reusable model libraries [1].

An international effort was initiated in September 1996 for the purpose of bringing together expertise in object-oriented physical modelling (port-based modelling) and defining a modern uniform modelling language, called Modelica. Modelica is intended for modelling within many application domains and their combinations. It supports several modelling formalisms: ODEs, DAEs, bond graphs, finite state automata, Petri Nets, etc. Modelica is intended to serve as a standard format so that models arising in different domains can be exchanged between tools and users.

Modelica is no simulator, Modelica is a modelling language, supporting and generating mathematical models in physical domains. It covers the broad area of physical modelling as well as features from Petri nets and, furthermore, finite automata are developed in the standard to boost the ability for solutions in application area. Modelica offers a graphical model frame, where the connections are bidirectional physical couplings. The graphical model layout corresponds with a textual model representation. This code can be changed and extended by the user, so that graphical and textual modelling can be combined.

Modelica can handle very different modelling approaches, not only ODEs and DAEs, but also finite state automata. By means of state automata conditions can be described more clear and transparent. The translator from Modelica into the target simulator is not only able to sort equations; it also has to be able to process the implicit equations symbolically and to perform DAE index reduction.

3 The Mosilab simulator

3.1 History and general

In the project GENSIM [8] a cooperation of the six institutes FIRST, IIS/EAS, ISE, IBP, IWU and IPK of Fraunhofer Forschungsgesellschaft the Modelica based simulation environment MOSILAB (MOdelling and SImulation LABoratory) is developed, which offers different modelling approaches, and supports structural dynamic systems
The new in MOSILAB used simulation language MOSILA [6] is close leaned to Modelica simulation standard. From the point of modellers view MOSILA is somewhat expansion of Modelica. That is why already defined models developed in another Modelica based simulator as well as the free accessible Modelica standard library can be used directly or with small adjustments only.

Beyond MOSILA offers additional concepts to support dynamical object structures and model switching in an adequate and practicable way. MOSILAB simulation environment provides modelling and simulation using UML H, an adapted subset of UML. The graphical modelling techniques are the basis for implementing the goal of modelling structural dynamic systems or models with variable modelling depth depending on the global model status.
The models developed in the graphical UML H model layer of the simulation environment, called "State Diagram", are part of the MOSILA language and can thereby be transferred and adjusted in the textual editor of the modelling mode of the MOSILAB environment. The language MOSILA is a result of a combination of modern modelling techniques for complex heterogenic technical systems with the main focus on a combination of graphical, textual and block based modelling techniques.

3.2 Extensions and communication

The MOSILA based models are translated in MOSILAB into C++ class description. Additionally extra functions written in C or C++ can be included in the model. Another way of including external software is by using the data interface for in- and output of model states. There predefined variables and the current value are written in a file or read from there at well defined time points. Afterwards the simulation is started again with the new/actual allocation. Model structure information is not yet communicated for the file, so that the user has to care about the structural equality of the input data himself [6].
An interface to MATLAB, one of the most common numerical computer packages, and to the finite element solver FEMLAB are implemented in the standard version.

3.3 Simulation flow and model initialization

In the current state of the simulation environment only analysis in time domain is supported. Thereby the model passes a sequence of alternating discrete and continuous phases:

- During discrete phases the simulation time is stopped. Thus changes on variables and structures can be done in “no time”. First it has to be proofed if the changed discrete variable results in a structure change. If not, new starting values for the actual parameter values have to be found for the start of the next continuous phase. In this case we are talking about an inner simulation loop. In the other case, when the change of the discrete variables results in a structural switch, the according events are generated and sent to the reactive objects. Afterwards the state machines of these objects are cyclic updated until all state machines are in a steady state. This means that there are no more changes in the configuration of the state machines. The system works hierarchically, this means that the state machines of subsystems are always updated before the state machine of the parent object is updated. When all state machines are in the steady state a new equation system and consistent starting values have to be defined for the next continuous phase. At the end the transition to the next phase is realized.
- During a continuous phase the differential and algebraic state variables and the simulation time are updated. The discrete state variables and the model structure and thereby the equations stay the same during this phase. As soon as a discrete state variable changes the value the continuous phase is breaking and a new discrete phase is starting. Furthermore the calculation is stopped when the simulation end time is reached.

The following figure depicts the upper description.

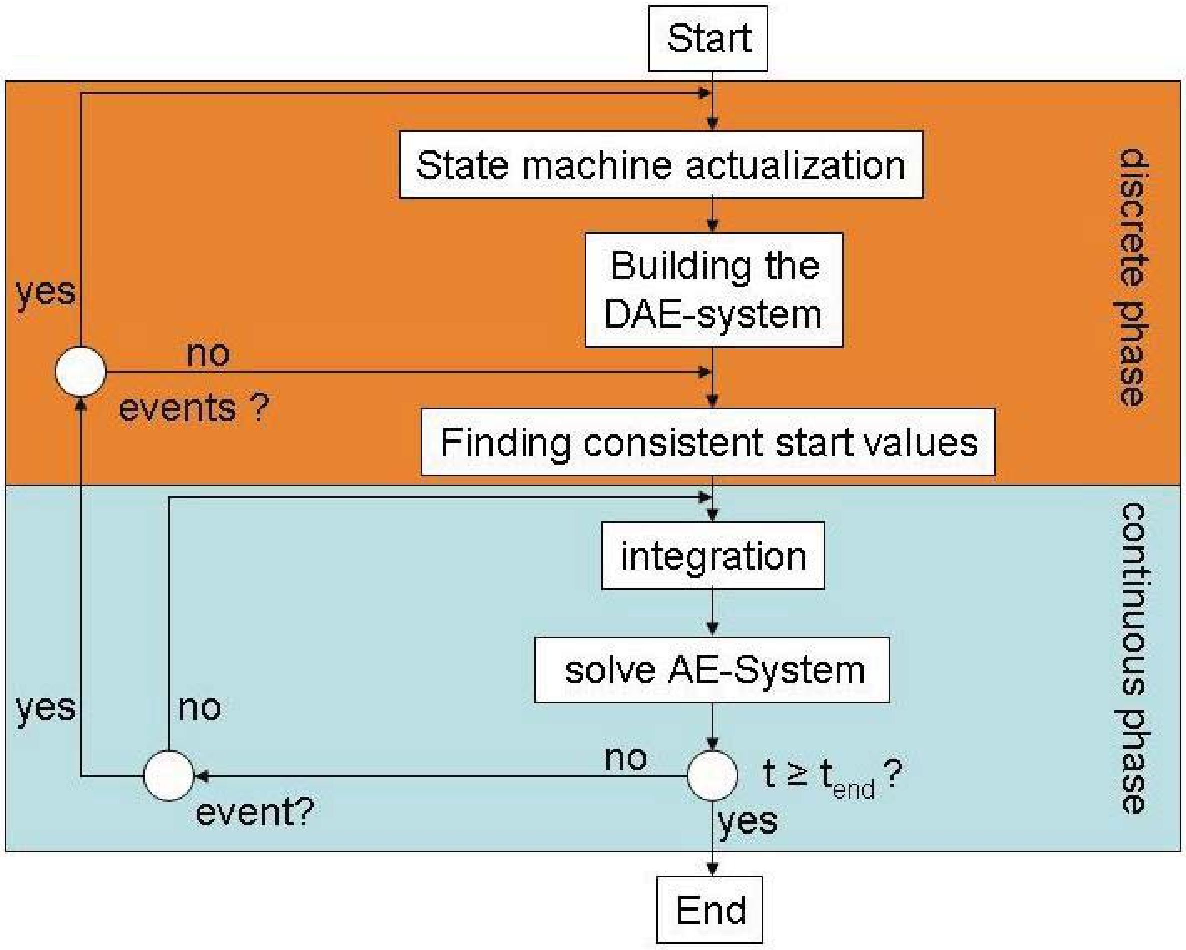

Figure 1 General structure for the system simulation including a discrete and a continuous phase for simulation of structural dynamic systems

4 Extension of the standard connector structure

As defined in the Modelica standard, a special kind of port, the so called connector class is implemented in MOSILA. Accounting the special structure of this Modelica extension the Mosilab connectors have an additional feature: They can be dynamically built and cleared and thereby used for implementation of structural dynamic systems. We can switch off parts from a graphical block based model by simply changing the connectors list.

As shown in the third example this speciality can be used for modelling and simulation of structural dynamic systems modelled with predefined components from the Modelica standard library but also for self defined or commercial ones.

4.1 Additional language elements – Statecharts

This extension to Modelica is made to model reactive subcomponents in a direct very compact form. As defined before the elements can be defined by a graphical editor on the modelling level and then be expanded using the textual layer of the modelling level. Now we look at the most important additional components that are used to implement the hybrid project definitions [2].

`new()` – operator, `add()` and `remove()`

In context with structural dynamic systems it is reasonable to create objects, add objects to the list of numerical active objects or delete objects from this active object list during a simulation run. To implement the creation of new objects (instances of a predefined model) the operator `new()` is developed [2].

To make such objects numerically active from the viewpoint of the simulation environment, it has to be added by using `x.add()`. During this operation it also has to be proofed that the new element is compatible, which means that the connector structure has to be checked for equality. This is done by the MOSILAB simulator. If this is assured variables with the same name in the sub components are connected just like with the `connect()` statement in standard Modelica case.

Certainly this concept is also implemented to abolish the connection in the case of switching to numerically inactive objects for a part of the model. This is realized by using the `remove()` function.

Not yet implemented language elements and properties - index reduction

The implemented solution methods deal with index-0 and index-1 differential algebraic systems in a correct way. Higher order index problems are not supported in the actual version. Another missing part in MOSILAB compared to the Modelica standard notation is: matrices. If necessary matrices have to be split into vectors which are already implemented.

5 Examples

5.1 Constrained Pendulum

The constrained pendulum is a classical nonlinear model in simulation techniques. To make the problem simpler than it is in real life, we assume the mass m is large enough so that, as an approximation, the mass of the rigid shaft of the pendulum is assumed negligible. This basic model definition is taken from the ARGESIM comparison C7 [5, 10]. 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.

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

Hereby φ denotes the angle in radiant measured in counter clockwise direction from the vertical position. The parameters in the model are the mass m and the length of the rope l. The damping is realized with the constant d. In Mosilab it is an important difference, if the modeler is using constant or parameter!

As it is a constrained pendulum a pin is fixed at a certain position. Every time when the rope of the pendulum hits the pin the length of the pendulum has to be shortened. In this case the pendulum swings on with the position of the pin as the point of rotation and the shortened length. The motion of the pendulum is usually defined with φ and `der(φ)` as states. But using the tangential velocity instead of angular velocity has the benefit, that only the length of the pendulum has a discrete change in case of hitting and leaving the pin. A standard Modelica solution can be performed by using an algorithm section with an `if` - or `when` – condition:

```
algorithm
       if (phi<=phipin) then length:=ls; end if;
       if (phi>phipin) then length:=ls1; end if;
equation
       ...  here /*pendulum*/  -equations
end model
```

This way of modelling the constrained pendulum can be implemented in any Modelica compatible simulator. Now we are looking at the speciality of MOSILAB and the possibilities of implementation we can get using the statechart structure. The first way is to implement the pure parameter event within a `statechart` instead of using the algorithm section. This can be done with the following code:

```
event  Boolean  lengthen(start=false), shorten(start  =  false);
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;
```

This MOSILA source code has to produce an equivalent internal code and as can be seen, the numerical results are the same. The computation time of the second implementation lasts longer. This can be explained by the more complex structure of the resulting source code.

The shown mode of using the `statechart` way of modelling is not the only implementation we can get along with. Another way, showing more about the power of the MOSILA language is the way of hybrid model decomposition for solving the constrained pendulum. To explain the structure the pendulum is switched in two models. The first one is the normal pendulum complying with equation above. The second is also a pendulum but now, the length is shortened by the distance between the main centre and the pin and the position of the pin defines the new centre.

Now we can define a new `statechart` for the constrained pendulum. Instead of switching by using parameters we switch between instances of our short and long pendulum (structure shown in the figure below).

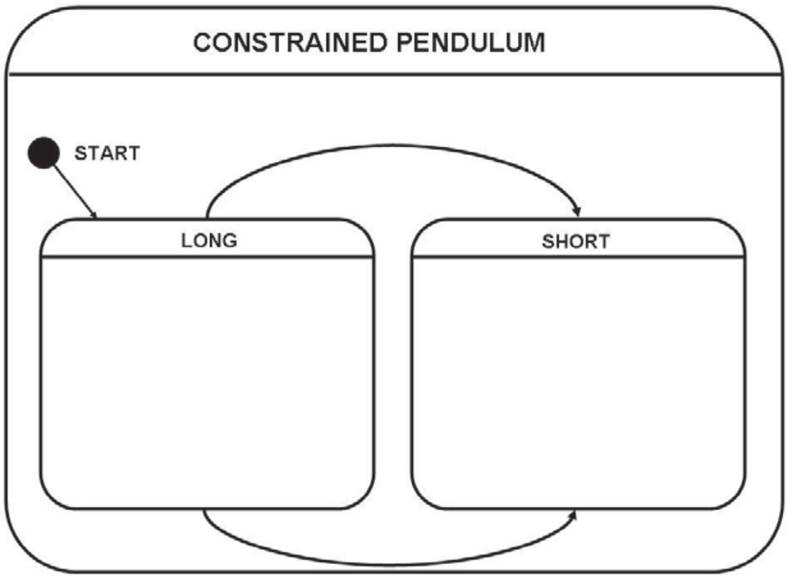

Figure 2 Stucture of the state chart for the MOSILAB implementation of the constrained pendulum

In this model it is also possible to define only one model class and then produce instances with different parameterisation. As can be seen, MOSILAB offers a wide range of model implementations in comparison to standard Modelica. In this example, the standard notation will be favourable because it is the fastest way of implementation and also the solution takes less time then with the state chart approach, but when the system gets more complex this structure cannot handle it.

5.2 Free pendulum on a string

Until now we looked at a system were the state space dimension does not change during simulation. The state events can all be interpreted as simple parameter events. Now a system is given where the state space dimension has to be changed for real.

This example is a little bit more complicated. Let us again consider a pendulum. The massive bob of the pendulum is fixed on a string. In case of a roll-over of the pendulum it can start to fall freely until the constraints of the string apply again. This can happen if the pendulum swings higher than ±π/2 and the centrifugal force is smaller than the gravitational force. Accordingly, the so defined model has two different states which are the normal pendulum movement and the free fall case.

The movement of the pendulum is given in equation (1). We have to define the equations for the free fall case. They are given by

$$\begin{aligned} \dot{v}_y &= -g \\ \dot{v}_x &= 0 \end{aligned}$$

For our model we have an additional constraint, which is based on the fixed length l of the pendulum:

$$x^2 + y^2 \leq l$$

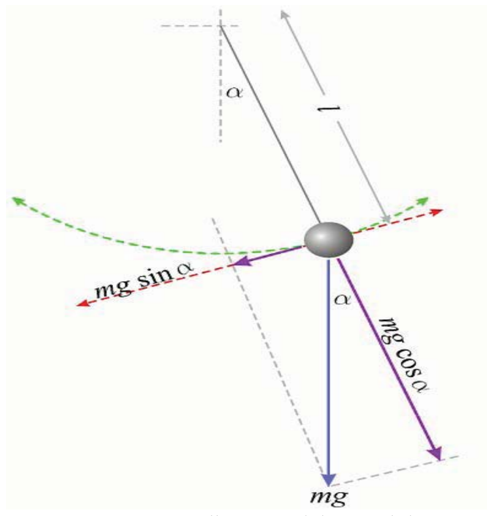

Figure 3 Force diagram of the pendulum

This model cannot be solved using simple parameter state events and is defined here to give an example that problems with hybrid structure can occur in simulation of technical systems as well as in biology, genetics, etc. not only in very sophisticated systems. As can be seen here the need for state space switching in nowadays modelling and simulation techniques is quite common. The number of equations necessary to describe the model is changing during simulation.

The implementation in MOSILAB can be done again by defining two separate models, the normal swinging pendulum and the free fall of the mass. The Boolean condition used as the switching conditions for the state chart approach is the length of the rope. MOSILAB offers a closed physical system for the hybrid problem.

5.3 Class-E amplifier

The definition of a class-E amplifier as defined in ARGESIM Benchmark 3 is used to show another advantage of modelling and simulation with MOSILAB in comparison with pure Modelica implementations.

The basic class-E power amplifier is a switching-mode amplifier that operates with zero voltage and zero slope across the switch at switch turn-off. Kirchhoff laws for voltage and current deliver the following differential equations:

$$\begin{aligned} dx1/dt &= (-x2+VDC)/L1 \\ dx2/dt &= (x1-x2/R(t)-x3)/C2 \\ dx3/dt &= (x2-RL*x3-x4)/L3 \\ dx4/dt &= x3/C4 \end{aligned}$$

The main difficulty of this equations go together with the time dependent resistor *R(t)* which has the form of a trapezoid with very fast slopes (up to *1e-13 sec*) and an *OFF* value of *5MΩ* and an *ON* value of *50mΩ*.

Three different ways have been chosen to model these equations in the MOSILAB simulation environment. The first solution is using textual Modelica notation. Designing the model is relatively easy in Modelica by using the exact equations as given above in the equation section and declaring all variables in the beginning section. The time dependent resistor *R(t)* is modelled using an algorithm section

For the second solution the MOSILA language extension for statechart modelling is used, dividing the system in separated model parts, depending on the state of the time dependent resistor *R(t)*. It switches between the state *OFF(state1)* and the state *ON(state2)*. Before simulation, *state1* is set up as initial state. Thus, the model will change to *state2* when the time dependent resistor *R(t)* reaches value *50mΩ* and the model will again change to *state1* when *R(t)* reaches *5MΩ*. By using the same code in an algorithm section as defined in the previous solution for *R(t)*, the value of the time dependent resistor is implemented.

The third solution deals with another speciality of MOSILA language: namely, to connect statements in the state chart description of a model. Therefore the class-E amplifier is not implemented in the textual modelling layer but in the component diagram. Instead of implementing the input of the time dependent resistor *R(t)* as trapezoid function standard output signal blocks from the Modelica standard library are used. Now it has to be switched back to textual model description because the time dependent connection between the four outputs of the standard blocks and the input of *R(t)* has to be defined. This is again modelled with state charts by using the commands `connect` and `disconnect` and thereby switching parts of a graphically produced model active and inactive by closing and opening the physical coupling.

6 Conclusion

The MOSILAB environment offers a real extension regarding modelling power and applicability in comparison with the Modelica standard notation. The extension with UML based state chart notation leads to a better readable modelling structure and higher flexibility from modeller's point of view. The quality of the results stays the same for all different implementation methods – an indispensable result, because the solution quality must not depend on the way of implementation in one and the same simulation system using the equivalent numerical solution method.
Although some parts of the Modelica standard notation (e.g. matrices) are not yet implemented, MOSILAB can handle a wide range of Modelica based models defined in other simulators and, thereby, provide an interesting alternative.

7 References

[1] Fritzson P.:
"*Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.*", Wiley-IEEE Press, 939 pages, ISBN 0-471-471631

[2] Nytsch-Geusen, C. et. al.:
"MOSILAB: Development of a Modelica based generic simulation tool supporting model structural dynamics." Proceedings of the 4th International Modelica Conference, TU Hamburg-Harburg, 2005.

[3] Breitenecker F., Judex F., Popper N., Troch I., Funovits J.:
"*Structure of Simulators for Hybrid Systems - General Development and Introduction of a Concept of External and Internal State Events*"; Proc. EUROSIM 2007 - 6th EUROSIM Congress on Modeling and Simulation; B. Zupancic, R. Karba, S. Blazic (Hrg.); ARGESIM / ASIM, Vienna (2007), ISBN: 978-3-901608-32-2; 14

[4] Meyer Urs B., Creux Simone E., Weber Marin Andrea K.:
„*Grafische Methoden der Prozessanalyse*", Hanser Verlag, 2005

[5] Breitenecker F., Ecker H. and Bausch-Gall I.:
„Simulation mit ACSL: eine Einführung in die Modellbildung, numerischen Methoden und Simulation", Braunschweig, Vieweg, 1993. -XI, 399 S

[6] Ernst T., Nordwig A., Nytsch-Geusen C., Claus C., Schneider A.:
„MOSILA Modellbeschreibungssprache, Spezifikation, Version 2.0", from webpage: www.mosilab.de/downloads/dokumentation

[7] http://www.mosilab.de/

[8] http://mosilab.de/forschungsprojekt-gensim

[9] http://www.modelica.org/

[10] http://www.argesim.org/