SIMULATORS FOR PHYSICAL MODELLING – CLASSIFICATION AND COMPARISON OF FEATURES / REVISION 2010

Felix Breitenecker[1], Niki Popper[2], Günther Zauner[2], Michael Landsiedl[2], Matthias Rößler[1], Bernhard Heinzl[1], Andreas Körner[1]

[1] Vienna University of Technology, Institute for Analysis and Scientific Computing
1040 Vienna, Wiedner Hauptstraße 8, Austria
[2]DWH Simulation Services
1070 Vienna, Neustiftgasse 57-59

matthias.roessler@tuwien.ac.at (Matthias Rößler)

Abstract

These series of contributions elaborate, classify and compare features of modern simulation systems, with special emphasis on physical modelling. First each contribution shortly reviews *classical features* of simulators: model sorting, event description, time event handling, state event handling, and DAE support with or without index reduction, and introduces a feature table with 'yes' – available, '(yes)' – available but difficult to use, 'no' – not available, and '(no)' – not available, but implementation possible or way around – shown with classic simulators. Next, *structural features* are introduced, which reflect the developments in the last decade: object-oriented approach, a-causal modelling, physical modelling, structural dynamic systems, modelling standardisations as Modelica and VHDL-AMS, impacts from computer engineering (e.g. state charts), co-simulation, and environments. The feature list of 2009 included textual and/or graphical physical modelling, simulation-driven visualisation, Modelica modelling standard, textual and/or graphical state chart modelling, modelling of structural-dynamic systems, frequency analysis, real-time capabilities, solver splitting, and access to derived state equations. Mainly based on solutions to the ARGESIM Benchmarks - Benchmarks for Modelling and Simulation Techniques, published in ASIM's scientific journal SNE – *Simulation News Europe*, the a comparison list could be set up, which in 2009 included 21 simulators (combination of different modules of simulators). This Revision 2010 extends the list of *structural features* by three new items: functional hybrid modelling, co-simulation, and multibody notations. Functional hybrid modelling is an essential extension of hybrid decoupling for structural-dynamic systems, co-simulation is an alternative approach in multidomain modelling, and the entry of multibody notations remember that Modelica is not the only standard. And consequently, the list of simulators is extended by two systems supporting functional hybrid modelling.

Keywords: Simulators, Classification, Comparison, Features, Structures

Presenting Author's biography

Matthias Rößler. He is on his way to finish his diplomastudy in technical mathematics. He currently writes on his diploma thesis in the course of the INFO-project, a collaboration between Vienna University of Technology and industrial partners.
His field of activity is physical modeling and simulation.

1 Classical features in Simulation Systems

Model sorting (MS), event description (ED), time event handling (TEH), state event handling (SEH), and DAE support (DAE) with or without index reduction (IR) became desirable *Classical Features* of simulators, supported directly or indirectly in simulation systems. Evaluation of the ARGESIM Benchmark solutions allows evaluating the availability of these features in simulation software. Tab. 1 compares the availability of these features in the MATLAB / Simulink System, in ACSL and in Dymola, the first representative of a physical simulator (simulation system with physical a-causal modelling level).

ACSL is a classical simulator with sophisticated state event handling, and since version 10 (2001) DAEs can be modelled directly by the residuum construct, and they are solved by the DASSL algorithm (a well-known direct DAE solver, based on the simultaneous approach), or by modified ODE solvers (nested approach) – so 'Y' for ED, SHE, and DAE. In case of DAE index $n = 1$, the DASSL algorithm guarantees convergence, in case of higher index integration may fail. ACSL does not perform index reduction (IR 'N'). ACSL comes with a sophisticated state event handling, so that all kind of events can be modelled and handled in a comfortable manner.

Dymola is a modern simulator, implemented in C, and based on physical modelling. Model description may be given by implicit laws, symbolic manipulations extract a proper ODE or DAE state space system, with index reduction for high index DAE systems – all classical features are available. Dymola started a new area in modelling and simulation of continuous and hybrid systems. Dymola is based on F. Cellier's works on object-oriented a-causal multidomain modelling, e.g. [11, 12].

Tab. 1: Availability of classical features in selected simulators

	Model Sorting MS	Event Description ED	Time Event Handling TEH	State Event Handling SEH	DAE Solver DAE	Index Reduction IR
MATLAB	N	N	N	(Y)	(Y)	N
Simulink	Y	(Y)	Y	(Y)	(Y)	N
MATLAB / Simulink	Y	Y	Y	Y	(Y)	N
ACSL	Y	Y	Y	Y	Y	N
Dymola	Y	Y	Y	Y	Y	Y

In Tab. 1, the availability of features is indicated by 'Y' ('Y') and 'N' ('N'); a 'Y' in parenthesis '(Y)' means, that the feature is available but complex to use; an 'N' in parenthesis '(N)' means, that the feature is yet not available, but foreseen or way around. MS - 'Model Sorting', is a standard feature of a simulator – but missing in MATLAB (in principle, MATLAB cannot be called a simulator). On the other hand, MATLAB's ODE solvers offer limited features for DAEs (systems with mass matrix) and an integration stop on event condition, so that SHE and DAE get a ('Y'). In Simulink, event descriptions are possible by means of triggered subsystems, so that ED gets a '(Y)' because of complexity. A combination of MATLAB and Simulink suggest putting the event description and handling at MATLAB level, so that ED and SHE get both a 'Y'. DAE solving is based on modified ODE solvers, using the nested approach (see before), so DE gets only a '(Y)' for all MATLAB/Simulink combinations. Time events are not supported in MATLAB, but they are basic feature in Simulink.

2 Extended Structures and Structural Features in Simulators

There are three basic developments to extend the structure of simulators. First, the extension from ODEs to DAE stimulated the evolvement of *Physical Modelling* – modelling based on laws and physical 'modules', textually und graphically – Dymola started the development. Second, influences from computer engineering suggest use of UML – *Unified Modelling Language*, especially the use of UML state charts for discrete events. And third, as consequence of the hybrid decomposition of models by state charts, and influenced

by experiences from co-simulation, handling of structural dynamic systems became more important over the years.

2.1 Physical Modelling

In the 1990s, many attempts have been made to improve and to extend structures, especially for the task of mathematical modelling. The basic problem was the state space description, which limited the construction of modular and flexible modelling libraries. Two developments helped to overcome this problem. On modelling level, the idea of physical modelling gave new input, and on implementation level, the object-oriented view helped to leave the constraints of input/output relations. In physical modelling, a typical procedure for modelling is to cut a system into subsystems and to account for the behaviour at the interfaces. Balances of mass, energy and momentum and material equations model each subsystem. 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.

In 1996, the situation was thus similar to the mid 1960s when CSSL was defined as a unification of the techniques and ideas of many different simulation programs. An international effort was initiated in September 1996 for bringing together expertise in object-oriented physical modelling (port based modelling) and defining a modern uniform modelling language – mainly driven by the developers of Dymola. The new modelling language is called *Modelica*, [9]. Modelica is intended for modelling within many application domains such as electrical circuits, multibody systems, drive trains, hydraulics, thermo-dynamical systems, and chemical processes etc. It supports several modelling formalisms: ordinary differential equations, differential-algebraic equations, bond graphs, finite state automata, and Petri nets etc. For details and applications see e.g. [12, 10].

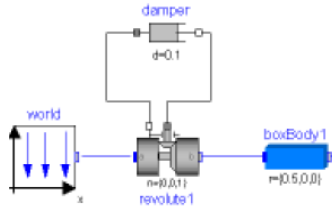
When the development of Modelica started, also a competitive development, the extension of VHDL towards VHDL-AMS was initiated. Both modelling frames aimed for general-purpose use, but VHDL-AMS mainly addresses circuit design, and Modelica covers the broader area of physical modelling; and modelling constructs such as Petri nets and finite automata could broaden the application area.

While also VHDL-AMS was aiming for multidomain modelling, there still very efficient simulators for special domains are used, like simulators for multibody systems. In principle, these systems are application-based modelling environments, which support a-causal modelling in their domain. An interesting theoretical questions occurs – is it possible to translate from Modelica to such specific model notation – and vice versa? – For a subset modelling modules the answer must be 'yes'. As multibody dynamics and Modelica multidomain modelling are closer with respect to a 'common' application area than VHDL-AMS, application-oriented a-causal modelling entered in this comparison in Rev.2010, while the entry of VHDL-AMS is still under discussion.

The translator from Modelica into the target simulator must not only be able to sort equations, it must be able to process the implicit equations symbolically and to perform DAE index reduction (or a way around). Up to now – similar to VHDL-AMS – some simulation systems understand Modelica (2009; generic – new simulator with Modelica modelling, extension - Modelica modelling interface for existing simulator):

- Dymola from Dynasim (generic),
- MathModelica from MathCore Engineering (generic)
- SimulationX from ISI (generic/extension)
- Scilab/Scicos (extension)
- MapleSim (a Maple Toolbox)
- Open Modelica - (since 2004 the University of Lyngby develops and provides an open Modelica simulation environment (generic),
- Mosilab - Fraunhofer Gesellschaft develops a generic Modelica simulator (generic)
- Simscape – the physical modelling language in MATLAB/Simulink – Modelica-like

Recently (2009 - 2010), Modelica-based ODE-solving interpreters and model re-compilers with ODE suites enrich the variety by a new feature, functional hybrid




```
algorithm
if (revolute1.phi
   <= phipin then
   revolute1.length:=ls;
end if;
if (revolute1.phi
   < phipin then
   revolute1.length:=ll;
end if;
```

```
   equation /*pendulum*/
     v = length*der(phi);
     vdot = der(v);
     mass*vdot/length     +     mass*g*
          sin(phi) +damping*v = 0;
   algorithm
    if  (phi<=phipin)  then  length:=ls;
end if;
    if  (phi>phipin)  then  length:=ll;
end if;
```

Fig. 1: Modelica model for *Constrained Pendulum* (mixed model / textual model)

modelling (FHM). Examples for this type of new systems, which allow for causality change (and subsequent state change) during runtime, are

- Sol (ETH Zürich; D. Zimmer, F. Cellier)
- Hydra / SUNDIALS suite (Univ. Nottingham; H. Nilsson)

As Modelica also incorporates graphical model elements, the user may choose between textual modelling, graphical modelling, and modelling using elements from an application library. Furthermore, graphical and textual modelling may be mixed in various kinds. The minimal modelling environment is a text editor; a comfortable modelling environment offers a graphical modelling editor. For the *Constrained Pendulum* example, the basic physical dynamics could be modelled fully textually (Fig. 1, at right), or graphically with joint and mass elements, and the event of length change is described in an algorithm section, with variables interfacing to the predefined variables in the graphical model part (Fig. 1, at left).

2.2 UML State Chart Modelling

In late 1990s, computer science initiated a new development for modelling discontinuous changes. The *Unified Modelling Language* (UML) is one of the most important standards for specification and design of object oriented systems. This standard was tuned for real time applications in the form of a new proposal, *UML Real-Time* (UML-RT). By means of UML-RT, objects can hold the dynamic behaviour of an ODE.
In 1999, a simulation research group at the Technical University of St. Petersburg used this approach in combination with a hybrid state machine for the development of a hybrid simulator (*MVS*), from 2000 on available commercially as simulator *AnyLogic*. The modelling language of AnyLogic is an extension of UML-RT; the main building block is the *Active Object*. Active objects have internal structure and behaviour, and allow encapsulating of other objects to any desired depth. Active objects interact with their surroundings through boundary objects: ports for discrete communication, and variables for continuous communication. While discrete model parts are described state charts, events, timers and messages, the continuous models are described by ODEs and DAEs in CSSL-type notation and with state charts within an object.

2.3 Hybrid Modelling and Structural-dynamic Modelling

Continuous simulation and discrete simulation have different roots, but they are using the same method, the analysis in the time domain. During the last decades a broad variety of model frames (model descriptions) has been developed. In continuous and hybrid simulation, the explicit or implicit state space description is used as common denominator. This state space may be described textually, or by signal-oriented graphic blocks (e.g. SIMULINK), or by physically based block descriptions (Modelica, VHDL-AMS).
Hybrid systems – systems with state events of essential types, often come together with a change of the dimension of the state space, then called *Structural-dynamic Systems*. In principle, structural-dynamic systems can be seen from two extreme viewpoints. The one says, in a maximal state space, state events switch on and off algebraic conditions, which freeze certain states for certain periods. The other one says that a global discrete state space controls local models with fixed state spaces, whereby the local models may be also discrete or static. These viewpoints derive two different approaches for structural dynamic systems modelling, the maximal state space, and the *hybrid* decomposition (Breitenecker et al., [4, 5]).

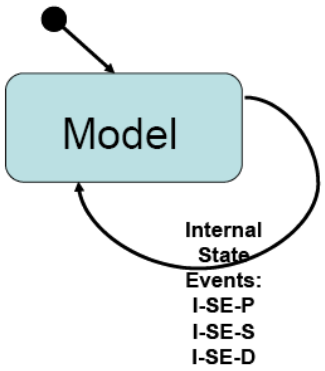


Fig. 2 State chart control of Internal Events for switching in one singular model (one model instance)

Maximal state space for structural-**dynamic systems – internal events**
Most implementations of physically based model descriptions support a big monolithic model description, derived from laws, ODEs, DAEs, state event functions and internal events. The state space is maximal and static, index reduction in combination with constraints keep a consistent state space. For instance, Dymola, OpenModelica, and VHDL-AMS follow this approach. This approach can be classified with respect to event implementation. The approach handles all events of any kind (SE-P – state event causing parameter change, SE-S – state event causing state change, and SE-D – state event causing change of dimension / of degree of freedom) within the ODE solver frame, also events which change the state space dimension (change of degree of freedoms) – consequently called *internal events*.

Using the classical state chart notation, *internal state events I-SE* caused by the model schedule the model itself, with usually different re-initialisations (depending on the event type I-SE-P, I-SES, I-SE-D; Fig. 2-4). For instance, VHDL-AMS and Dymola follow this approach, handling also DAE models with index higher than 1; discrete model parts are only supported at event level. ACSL and MATLAB / Simulink generate a maximal state space.

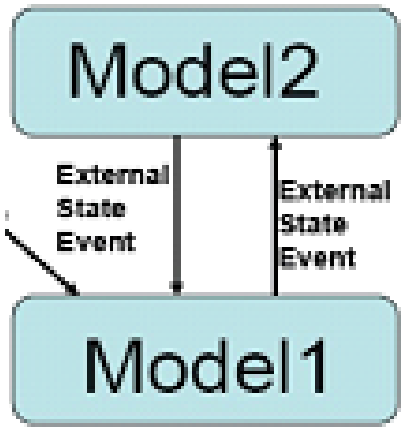


Fig. 3 State chart control of External Events for switching between two different models

Hybrid decomposition **for structural-dynamic systems – external events**

The hybrid decomposition approach makes use of *external events* (E-SE), which controls the sequence and the serial coupling of one model or of more models. A convenient tool for switching between models is a state chart, driven by the *external events* – which itself are generated by the models. Following e.g. the UML-RT notation, control for continuous models and for discrete actions can by modelled by state charts. Fig. 3 shows the hybrid coupling of two models, which may be extended to an arbitrary number of models, with possible events E-SE-P, E-SE-S, and ESE-D. As special case, this technique may be also used for serial conditional 'execution' of one model – Fig. 4 (only for SE-P and SE-S).

This approach additionally allows not only dynamically changing state spaces, but also different model types, like ODEs, linear ODEs (to be analysed by linear theory), PDEs, co-simulation, etc. to be processed in serial or also in parallel, so that also co-simulation can be formulated based on external events. The approach allows handling all events also outside the ODE solver frame. After an event, a very new model can be started. This procedure may make sense especially in case of events of type SE-D and SE-S. As consequence, consecutive models of different state spaces may be used.

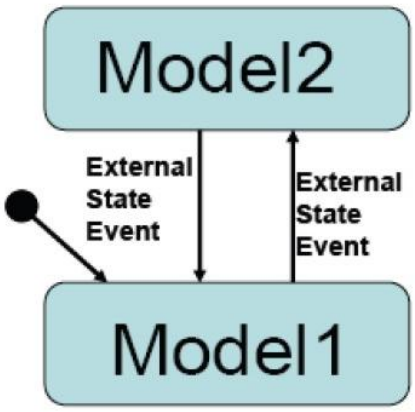


Fig. 4 State chart control of External Events for switching two different instances of one model

2.4 Functional Hybrid Modelling

Functional Hybrid Modelling – (FHM) is a generalisation of the previously discussed hybrids decomposition for structural-dynamic systems: it provides dynamic change of models with different causalities during the simulation, [19, 20].

Modelica – like simulators are designed and implemented on the assumption that the causality of the model does not change during simulation, which simplifies the language design and allows for generation of efficient simulation code. But for genuine hybrid models, the assumption of fixed causality is very limiting and prevents often efficient simulation of hybrid models. Functional Hybrid Modelling is a new approach to non-causal modelling that promotes models as first class entities and supports structurally dynamics by allowing arbitrary switching among model configurations during simulation. Fixed causality is thus *not* assumed in simulators supporting FHM. In such simulators, as Sol [17, 18] and Hydra [20, 21], and partly in Mosilab [13, 14], continuous simulation remains efficient because of dynamic generation of simulation code at each change of the configuration, and the use of a standard DAE solver. Structural change during runtime of course incurs an overhead, but this overhead is modest in the majority of problems – except contact problems. FHM may also be seen as dynamic version of the fore-mentioned hybrid decomposition of structural-dynamic systems with external events. In consistency with the previous definitions of external events, in FHM external events switch not only to a new model, they generate first the new model.

From another viewpoint, hybrid decomposition and functional hybrid modelling go beyond the present Modelica standard (spring 2010), because language elements and translation proposals are still under consideration in Modelica and cause severe problems. FHM feature is new in Revision 2010 of the classification.

2.5 Real-time Solution and Solver Splitting

Ziegler proposes a separation of model frame and experimental frame. But this separation becomes difficult, if the simulation task needs specific algorithmic methods. In case of a real-time simulation task, proper algorithm choice and specific model translations are important and – as real-time simulation (RTS) and DAE iterative solving are almost contradictions – a challenge for simulation systems with physical modelling level.
Another idea, which comes along with the development of Simscape, is 'solver splitting' (SSP). Modelling physical components very often results in DAEs, while control systems around the physical model parts are causal, governed by ODEs, and very often coupled via sampled data. It could make sense, to 'split' the ODE solver into a DAE solver for the physical model, and into an ODE solver for the causal model parts. In principle, this solver splitting is a first step towards co-simulation.

2.6 Co-Simulation

The feature *Co-Simulation* (CO-S) is new in Revision 2010 of the classification. In multidomain modelling, co-simulation is a very renowned tool for coupling two simulators at runtime. At the level of modelling, the advantages of the simulators for specific modelling approaches can be used, and at simulation level, specific solvers calculate in their domain –co-simulation couples both simulators either via a backbone or via one of the simulators at runtime. Clearly, the simulated systems are now hybrid discrete systems, because of the discrete coupling at communication time instants. Co-simulation, although very earlier developed, may be seen as generalisation of the feature 'solver splitting'

3 Availability of Structural Features in Simulation Systems

While the *classical features* discussed before address the CSSL-standard, *structural features* characterise features for physical modelling and for structural dynamic systems. This section sketches the availability or of structural features in some simulators, and summarises the results in Tab. 2.

3.1 Structural Features in Classification and Evaluation

Based on the *ARGESIM Comparisons on Simulation Software and Simulation Techniques* ([1, 2]) at Vienna University of Technology a classification of classical, structural and advanced features of simulators is run since 2006, with evaluation of commercial and experimental simulators [6. 7, 8]. This contribution extends the list of features from [8] – Classification Rev. 2009 – by three items and adds two experimental simulators. The structural features for evaluation of simulation systems may be classified anew follows:

- Support of a-causal physical modelling (sometimes called port-based modelling) at textual (PM-T) or graphical level (PM-G),
- Modelica standard (MOD) for a-causal physical modelling, or application-based a-causal modelling features like multibody notations (AMOD; new in Rev. 2010), and in this context the open access to the derived state equations (OAE)
- Decomposition of structural dynamic systems with dynamic features (SD), and as generalisation functional hybrid modelling (FHM) – new in Rev. 2010
- Support of state chart modelling or a of a similar construct, by means of textual (SC-T) or graphical (SC-G) constructs,
- Features of the simulation environment, like simulation-driven visualisation (VIS), frequency analysis (FA), and open extendable environment (ENV)
- Algorithmic features, like real-time capabilities (RT), solver – splitting (SSP), and co-simulation (CO-S; new in Revision 2010)

In principle, each combination of the above features is possible. By means of the maximal state space approach, each classic simulator can handle structural dynamic systems, but a-causal modelling may be supported or not, and state chart modelling may be available or not. Simulators with a-causal modelling may support hybrid decomposition or not, and state chart modelling may be available or not. Simulators with features for state chart modelling may support hybrid decomposition or not, and a-causal modelling may be offered or not. In general, interpreter-oriented simulators offer more structural flexibility, but modern software structures would allow also flexibility with precompiled models or with models compiled 'on the fly'. With functional hybrid modelling, the difference between model interpretation, model translation and model compilation becomes more or less obsolete, because the 'model transformation' to an implicit state space description (to be used by a DAE solver) is performed also during runtime. In this context, open access to generated equations (OEA) becomes more and more important – for efficiency inspection and re-use in other systems.
In addition, of interest are also structural features of the simulator environment, as simulation-driven visualisation (with visualisation objects defined with the model objects; VIS), frequency domain analysis and linearization for steady state analysis (FA), and extended and extendable environment for complex experiments and data pre- and post-processing (ENV) – discussed in this Revision 2010 of the classification.
For the future the following new classification items are planned: optimisation and identification (OPID), System Dynamics modelling (SYS-D), spatial

dynamics (SPAT), parallel simulation (PAR), and – still under discussion - VHDL-AMS standard (V-AMS).

3.2 A Review of structural Features in Selected Simulation Systems

The mainly interpretative systems **MATLAB / Simulink** offer different approaches. First, MATLAB itself allows any kind of static and dynamic decomposition (SD 'Y'), but MATLAB is not a simulator, because the model equations have to be provided in a sorted manner, to be called from an ODE solver (MS 'N'). Second, MATLAB allows hybrid decomposition at MATLAB level with Simulink models. There, from MATLAB different Simulink models are called conditionally, and in Simulink, a state event is determined by the hit-crossing block (terminating the simulation). Simulink is MATLAB's simulation module for block-oriented dynamic models (directed signal graphs), which can be combined with Stateflow, MATLAB's module for event-driven state changes described by state charts (SC-T and SC-G 'Y'). At Simulink level, Stateflow, Simulink's state chart modelling tool, may control different submodels. But Simulink can only work with a maximal state space and does not allow hybrid decomposition (SD 'N). Neither basic MATLAB nor basic Simulink support a-causal modelling. First MATLAB/Simulink modules for physical modelling (e.g. *Hydraulic Blockset* and others, 2004 - 2008) were precompiled to a classical state space (PM-T and PM-G 'N'); Modelica modelling is not supported (MOD 'N'). But with the basic Simscape model language, MATLAB introduces physical modelling – with stronger relations to application areas than Modelica – so AMOD – 'Y'.

For DAEs, MATLAB and Simulink offer modified LSODE solvers (implicit solvers) for the nested DAE solving approach. In MATLAB any kind of simulation – driven visualisation can be programmed and used in MATLAB or Simulink or in both, but not based on the model definition blocks (VIS '(Y)'). From the beginning on, MATLAB and Simulink offered frequency analysis (FA 'Y'), and clearly, MATLAB is a very powerful environment for Simulink, Stateflow, for all other Toolboxes, and for MATLAB itself (ENV 'Y'). In general, MATLAB and Simulink offer real time features (RT 'Y'), solver splitting can be done manually (SSP '(Y)'). All derived equations are accessible, in m-files, mdl-files and C-files (OEA 'Y').

All MATLAB modules allow access to derived equations and partly generation of C models or Java model, so OAE – 'Y'. Simulink works with a maximal state space, so FHM and also SD -'N', but in MATLAB the new symbolic toolbox may be used to generate new models after stooping a previous model at an event – consequently FHM – '(Y)'.

ACSL – Advanced Continuous Simulation Language – has been developed since more than 25 years. Last extensions were a change to C as basic language (instead of FORTRAN), and DAE features using the nested approach with classical solvers, or direct implicit DAE solving with DASSL Code, DAE - 'Y', IR - 'N'). From the beginning on, steady state calculation, linearization and frequency analysis was a standard feature of ACSL's simulator kernel (FA 'Y'). Since 2000, the environment has been enriched by modules for modelling and environment modules. The first module was a graphical modeller, which seems to make use of physical modelling, but in behind a classical state space is used – PM-T and PM-G 'N'. Furthermore, a simulation-driven visualisation system (third party) is offered – VIS '(Y)'. There exist real time interfaces - RT '(Y)', solver splitting may be done manually (SSP '(Y)'), system equations are open in FORTRAN or C files – OEA – 'Y'. ACSL works with a model compiler which generates a maximal state space, so SD – 'N'. FHM – 'N'.

Dymola, introduced by F. E. Cellier as a-causal modelling language, and developed to a simulator by H. Elmquist, can be called the mother of Modelica. Dymola is based on a-causal physical modelling and initiated Modelica; consequently, it fully supports Modelica these structural features (PM-T, PM-G, and MOD 'Y'). Together with the model objects, also graphical objects may be defined, so that simulation based pseudo-3D visualisation is available (VIS 'Y'). A key feature of Dymola is the very sophisticated index reduction by the modified Pantelides algorithm, so Dymola handles any DAE system, also with higher index, with bravura (DAE and IR 'Y'). For DAE solving, modified DASSL algorithms are used. In software structure, Dymola is similar to ACSL, using an extended CSSL structure as given in Fig. 1 – with the modification that all discrete actions are put into one event module, where CASE - constructs distinguish between the different events (this structure is based on the first simulator engine Dymola used, the DS-Block System of DLR Oberpfaffenhofen). There exist real time interfaces, - RT '(Y)', solver splitting cannot be provided, a way around may be inline coding – SSP - '(N)'. Up to now no access to derived equations is possible -OEA 'N'. As Dymola generates a maximal state space, neither structural decomposition nor functional hybrid modelling is available, SD – 'N', FHM – 'N'.

The goal of the ***Open Modelica*** project is to create a complete Modelica modelling, compilation and simulation environment based on free software distributed in binary and source code form. The whole OpenModelica environment consists of open software: *OMC* – the Open Modelica Compiler translates Modelica models (with index reduction); OMShell as interactive session handler is a minimal experiment frame; Modelica models may be set up by a simple text editor or by a graphical model editor.

Open Modelica is a generic Modelica simulator, so all basic features are met (ED, SEH, DAE, PM-T, PM-G, IR and MOD 'Y'; for DAE solving, variants of DASSL solver are used). P. Fritzson, the initiator of Open Modelica puts emphasis on discrete events and hybrid

modelling, so documentation comes with clear advice for use of IF – and WHEN – clauses in Modelica, and with state chart modules in DrModelica – so SC-T 'Y'. For graphical state chart modelling the experimental Modelica state chart library can be used – SC-G '(Y)'. Real-time is up to now no subject of Open Modelica (RT 'N'), but derived equations are accessible, OEA - 'Y'.

SimulationX is a new Modelica simulator developed by ITI simulation, Dresden. This almost generic Modelica simulator is based on ITI's simulation system ITI-SIM, where the generic IT-SIM modelling frame has been replaced by Modelica modelling. From the very beginning on, ITI-SIM concentrated on physical modelling, with a theoretical background from power graphs and bond graphs. The simulation engine from ITI-SIM drives also SimulationX, using a sophisticated implicit integration scheme, with state event handling.

Consequently, all features related to physical modelling are available: (ED, SEH, DAE, PM-T, PM-G, and MOD 'Y'; index reduction is not really implemented – IR '(N)'. State chart constructs are not directly supported (SC-T 'N'), but due to Modelica compatibility the Modelica state chart library can be used (SC-G – '(Y)'. Frequency analysis is directly supported in the simulation environment (FA – 'Y'), as well as interfaces to other systems (ENV – '(Y)') and pseudo-3D visualisation (VIS – 'Y'), RT interfaces are available (RT '(Y)'). AS SimulationX is based on a previous physical modelling simulator, application-oriented balance network equations can be used, so AMOD – 'Y'.

AnyLogic is based on hybrid automata (SC-T and SC-G - 'Y'). Consequently, hybrid decomposition and control by external events is possible, so ED, SD - 'Y'. But implementation of functional hybrid modelling would cause a big effort, so FHM – '(N)'. AnyLogic can deal partly with implicit systems (only nested approach, DAE '(Y)'), but does not support a-causal modelling (PM-T, PM-G - 'N') and does not support Modelica, MOD - 'N', and also AMOD – 'N'. Furthermore, new versions of AnyLogic concentrate more on discrete modelling and modelling with System Dynamics, but without event detection (SEH '(N)'. AnyLogic offers many other modelling paradigms, as System Dynamics, Agent-based Simulation, and DEVS. AnyLogic is Java-based and provides simulation-driven visualisation and animation of model objects (VIS 'Y') and can also generate Java web applets. From software engineering view, AnyLogic is a programming environment for Java – so ENV – '(Y)'. Real-time is up to now no subject of AnyLogic (RT 'N'), as well as solver complexity (SSP 'N'), but all equation open in Java, OEA - 'Y'. **ModelVision** is related to AnyLogic (similar features).

Maple – developed by Maplesoft, Canada, has developed a toolbox *MapleSim*, which partly understands Modelica models (PM-T, PM-G, and MOD – 'Y'). MapleSim comes with a big library of physical components, basically application-oriented physical notation, AMOD – 'Y'.

MapleSim's modeling libraries are based on physical components with defined connectors (ports). The dynamics of a component may be defined by Maple ODE/DAE notation, Maple notation for any auxiliaries and conditions, or by basic Modelica blocks. MapleSim's modeling concept follows P. Breedveld's ideas on port-based modeling [24]. Furthermore, multibody components can be linked with construction drawings for animation purposes.

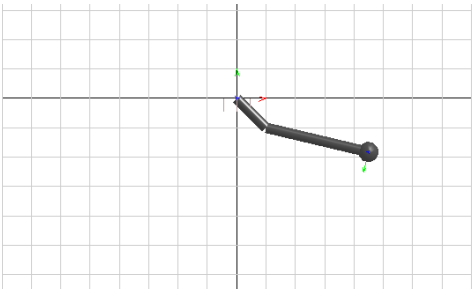


Fig. 5 Visualization of a constrained pendulum in MapleSim using Multibody-Systems

Maple also acts as environment and provides sophisticated DAE solvers, as well as symbolic algorithms for index reduction (DAE, IR, ENV, VIS, and FA – 'Y'). New releases offer event handling (ED, TEH, SEH - 'Y'), there exist limited real time interfaces, RT - '(Y)', and derived equations are open for access and use in other systems, OEA - 'Y'. Graphical state chart notation may come from the experimental Modelica state chart library, SC-T – 'N', SC-G – '(Y)'. Structural dynamic systems can be controlled from Maple loops, SD – '(Y)', but functional hybrid modelling would cause user-defined symbolic manipulations, so FHM – '(N)'.

Scilab/Scicos is an open source alternative to MATLAB / Simulink, developed in France. *Scicos* is a graphical dynamical system modeller and simulator toolbox included in Scilab. So it has nearly the same features as MATLAB: no equation sorting– MS – 'N'; DE, IR, PM-T, PM-G, MOD, SC-T, and SC-G – 'N'; SEH, DAE, and VIS – '(Y)', remarkably – SD, FA and ENV – 'Y'. Similarly, Scicos has classical features ED, SEH, and DAE – 'Y'. The developers of Scicos started early with a kind of physical modelling (PM-T, PM-G – 'Y'). At present extensions are prepared: Modelica models (text/graphic) – so MOD and IR '(Y)', and refining the IF-THEN-ELSE and WHEN clause introducing different classes of associated events, resulting 'state chart clauses' - so SC-T – 'Y'. Real-time features are available, so RT - 'Y', solver splitting could be done – SSP '(Y)', derived equations are open for access – OEA 'Y'. Structural dynamic modelling and functional hybrid modelling would be possible at basic Scilab level, with more or less effort.

Tab. 2: Availability *structural features* in selected simulators

	MS – Model Sorting	ED -Event Description	SEH -State Event Handling	DAE - DAE Solver	IR - Index Reduction	PM-T – Physical Modelling -Text	PM-G - Physical Modelling -Graphics	VIS – 'Online' - Visualisation	MOD – Modelica Modelling	SC-T – State Chart – Modelling - Text	SC-G – State Chart Modelling – Graphics	SD – Structural Dynamic Systems	FA – Frequency Analysis	RT – Real-time Features	SSP – Solver Splitting	OEA – Acess Derived Equations	ENV – Extended Environment
MATLAB	N	N	(Y)	(Y)	N	N	N	(Y)	N	N	N	Y	Y	Y	(Y)	Y	Y
Simulink	Y	(Y)	(Y)	(Y)	N	N	(N)	(Y)	N	N	N	N	Y	Y	(N)	Y	(Y)
MATLAB/ Simulink	Y	Y	Y	(Y)	N	N	(N)	(Y)	N	N	N	Y	Y	Y	(Y)	Y	Y
Simulink / Stateflow	Y	Y	Y	Y	N	N	(N)	(Y)	N	(Y)	Y	N	Y	Y	(N)	Y	(Y)
Simulink / Simscape	Y	Y	Y	Y	(Y)	Y	Y	(Y)	(N)	N	N	N	Y	(Y)	(Y)	Y	(Y)
SL Simscape Stateflow	Y	Y	Y	Y	(Y)	Y	Y	(Y)	(N)	(Y)	Y	N	Y	(Y)	(N)	Y	(Y)
ML/SL Simscape Stateflow	Y	Y	Y	Y	(Y)	Y	Y	(Y)	(N)	(Y)	Y	Y	Y	(Y)	(N)	Y	Y
ACSL	Y	Y	Y	Y	N	N	(N)	(Y)	N	N	N	N	Y	(Y)	(Y)	Y	Y
Dymola	Y	Y	Y	Y	Y	Y	Y	Y	Y	(Y)	(Y)	N	(N)	(Y)	(N)	N	(Y)
Math Modelica	Y	Y	Y	Y	Y	Y	Y	(Y)	Y	(N)	(Y)	N	(N)	(N)	N	(Y)	(N)
MathMod./ Mathematica	Y	Y	Y	Y	Y	Y	Y	Y	Y	(N)	(Y)	Y	Y	(N)	N	(Y)	Y
Mosilab	Y	Y	Y	Y	(N)	Y	Y	(N)	(Y)	Y	Y	Y	N	(N)	(N)	Y	(Y)
Open Modelica	Y	Y	Y	Y	Y	Y	(N)	(N)	Y	(N)	(Y)	N	N	(N)	(N)	Y	N
Simulation X	Y	Y	Y	Y	Y	Y	Y	Y	Y	(N)	(Y)	N	Y	(Y)	(N)		(Y)
AnyLogic	Y	Y	(Y)	(Y)	N	N	N	Y	N	Y	Y	Y	N	(N)	N	Y	N
Model Vision	Y	Y	Y	Y	Y	Y	N	Y	N	Y	Y	Y	Y	N	N	Y	N
Scilab	N	N	(Y)	(Y)	N	N	N	(Y)	N	N	N	Y	Y	Y	(Y)	Y	Y
Scicos	Y	(Y)	Y	Y	(Y)	Y	Y	(Y)	(Y)	Y	(Y)	N	N	Y	(N)	Y	N
Scilab / Scicos	Y	Y	Y	Y	(Y)	Y	Y	(Y)	(Y)	Y	(Y)	Y	Y	Y	(N)	Y	Y
Maple	N	N	(Y)	Y	(N)	N	N	(Y)	N	N	N	Y	Y	N	(Y)	Y	Y
MapleSim	Y	(Y)	(Y)	Y	Y	Y	Y	Y	Y	N	N	(Y)	(Y)	Y	(N)	Y	Y

Mosilab / Sol / Hydra. Mosilab has been developed by Fraunhofer in the years 2006 – 2008, designed as simulator for structural dynamic systems with Modelica modelling and Dymola-like features. So

Mosilab shows features similar to Dymola, but is designed for structural dynamic systems which can be coupled via state charts, so SD – 'Y', SC-T – 'Y', SC-G – '(Y)', and comes with open C and C++ models. OEA – 'Y'. But functional hybrid modelling would cause a major extension of the translator, so FHM – '(N)'. Unfortunately further developments have been postponed. Mosilab's hybrid modelling components are one of the suggestions for hybrid extensions in Modelica, for comparison see [14, 15, 16].

Sol (D. Zimmer, F. Cellier, ETH Zürich, [17, 18]) is a derivative language of Modelica-type and has been especially designed for the convenient expression and simulation of variable-structure systems within an object-oriented, equation-based modelling framework, very close to Modelica and Dymola – so basic features coincide with Dymola, especially Modelica modelling, MOD – 'Y'. It partly follows the ideas of Mosilab – hybrid decomposition into conditional subsequent models – so 'SD' – 'Y'. But as main extension, Sol offers dynamics model change and interpretative model generation at runtime, so it offers functional hybrid modelling in an interpretative manner, FHM – 'Y'.

Sol's intention is to be as close as possible to Modelica, and Sol's hybrid modelling components are one of the suggestions for hybrid extensions in Modelica. The development of Functional Hybrid Modelling is in principle independent on Modelica, but Sol implements some FHM structures.

Hydra / SUNDIALS Hydra is a declarative language for modelling and simulation of physical systems using implicitly formulated (undirected) differential algebraic equations (DAEs) – [20, 21]. The language provides constructs for definition and composition of model fragments that enable modelling and simulation of large, complex, structurally dynamic and hybrid systems. Currently, Hydra is implemented as a domain specific embedded language in Haskell. Thus, full power of a modern functional language is available (e.g. for meta-modelling) in addition to core modelling capabilities. Hydra implements concepts of Functional Hybrid Modelling (FHM, [19]) and is the first publicly available implementation of an FHM language.

Hydra allows generating DAE models at runtime, in any complexity, taking into account changing causalities, state space changes, and changes in degrees of freedom. For simulation, the SUNDIALS DAE solver suite [3] may be used.

It should be noted, that in context with Mosilab, Sol and Hydra (and MKL - Modelling Kernel Language, Broman [22, 23]) different developments are enriching each other - Modelica Extension and Functional Hybrid Modelling, at present both 'converging' at implementation level. FHM may be seen as correct formalisation of the more intuitive concept of structural dynamic systems in Mosilab [13, 14].

3.3 Summary Table for Structural Features in Selected Simulation Systems

Tab. 2 tries to summarise the availability of the discussed structural features. As in Tab. 1, the availability of features is indicated by 'Y' ('Y') and 'N' ('N'); a 'Y' in parenthesis '(Y)' means that the feature is available but complex to use; an 'N' in parenthesis '(N)' means, that the feature is yet not available, but foreseen or way around exists.

4 References

[1] F. Breitenecker, N. Popper, S. Wassertheurer: "Evaluation and Benchmarking of Tools for Modelling and Simulation in Engineering - The ARGESIM Comparisons"; in: "2006 International Conference on Engineering and Mathematics", J. Bilbao, A. Olozaga, Y. Ruiz, C. Cabredo (Hrg.); Publicaciones - Escuela Tecnica Superior de Ingeneria, Bilbao, Spain (2006), ISBN: 84-95809-26-5; S. 217 - 223.

[2] S. Pawlik, N. Popper, F. Breitenecker: "A Classification of Modelling and Simulation Approaches based on the ARGESIM Benchmarks"; in: "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; 8 S.

[3] SUNDIALS suite; http://www.llnl.gov/casc/sundials/

[4] F. Breitenecker, N. Popper: "Structure of Simulation Systems for Structural-Dynamic Systems" in: "Proc. First Asia International Conference on Modelling and Simulation", D. Al Dabass, R. Zobel, A. Abraham, S. Turner (Hrg.); IEEE Computer Society, (2007), ISBN: 978-07695-2845-8; S. 574 - 579.

[5] F. Breitenecker, F. Judex, N. Popper, I. Troch, J. Funovits: "Structure of Simulators for Hybrid Systems - General Development and Introduction of a Concept of External and Internal State Events"; in: "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 S.

[6] F. Breitenecker, N. Popper, G. Zauner: "Structural Features in Simulation Systems - Evaluation and Comparison"; in: "Proc. 20th European Modeling and Simulation Symposium EMSS2008", A. Bruzzone, F. Longo, M. Piera, R. Aguilar, C. Frydman (Hrg.); Univ. Calabria, (2008), ISBN: 978-88-903724-0-7; S. 399 - 407.

[7] F. Breitenecker, N. Popper: "Extended and Structural Features of Simulators - A Comparative Study"; Simulation News Europe SNE, Vol. 18 (2009), no. 3-4; S. 27 - 38.

[8] F. Breitenecker, N. Popper: "Classification and Evaluation of Simulator Features"; in: "Proceedings MATHMOD 09 Vienna - Full Papers CD Volume", I. Troch, F. Breitenecker (Hrg.); ARGESIM / ASIM - Vienna, ARGESIM Report No. 35 (2009), ISBN: 978-3-901608-35-3; 13 S.

[9] The Modelica Association. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling: Language Specification Version3.2, March 2010. http://www.modelica.org/documents/ModelicaSpec32_withRevisionMarks.pdf

[10] Principles of Object-Oriented Modeling and Simulation with Modelica 2.1; P. Fritzson Publisher: Wiley-IEEE Press (January 22, 2004) ISBN-10: 0471471631 ISBN-13: 978-0471471639

[11] F. E. Cellier. Continuous System Modeling, Springer, 1991; ISBN 9780387975023

[12] F. E. Cellier and Ernesto Kofman. Continuous System Simulation. Springer-Verlag, 2006.

[13] C. Nytsch-Geusen, T. Ernst, A. Nordwig, P. Schwarz, P. Schneider, M. Vetter, C. Wittwer, T. Nouidui, A. Holm, J. Leopold, G. Schmidt, A. Mattes, and U. Doll. MOSILAB: Development of a Modelica-based generic simulation tool supporting model structural dynamics. In Proceedings of the 4th International Modelica Conference, pages 527–535, Hamburg, Germany, 2005.

[14] G. Zauner, D. Leitner and F. Breitenecker. Modelling structural-dynamics systems in Modelica/Dymola, Modelica/MOSILAB, and AnyLogic. In Peter Fritzson, Francois Cellier, and Christoph Nytsch-Geusen, editors, Proceedings of the 1st International Workshop on Equation-Based Object-Oriented Languages and Tools (EOOLT), number 24 in Linköping Electronic Conference Proceedings, pages 99–110, Berlin, Germany, 2007. Linköping University Electronic Press.

[15] G. Zauner, F. Breitenecker: "Three Structural Different Modelling Approaches to ARGESIM Comparison C7 `Constrained Pendulum´ using the Modelica-Simulator MOSILAB"; Simulation News Europe SNE, 16 (2007), 3; S. 61 - 62.

[16] G. Zauner, N. Popper, F. Breitenecker: "Modelling Structural Dynamic Systems: Standard Modelica vs. Mosilab Statecharts"; Vortrag: MATHMOD 2009 Vienna, Vienna; 11.02.2009 - 13.02.2009; in: "Proceedings MATHMOD 09 Vienna - Full Papers CD Volume", I. Troch, F. Breitenecker (Hrg.); ARGESIM / ASIM - Vienna, ARGESIM Report No. 35 (2009), ISBN: 978-3-901608-35-3; 8 S.

[17] D. Zimmer. Introducing Sol: A general methodology for equation-based modeling of variable structure systems. In Proceedings of the 6th International Modelica Conference, pages 47–56, Bielefeld, Germany, 2008.

[18] D. Zimmer: Enhancing Modelica towards variable structure systems. In: Proc. of the 1st International

Workshop on Equation-Based Object-Oriented Languages and Tools, Berlin, Germany (2007) 61-70.

[19] H. Nilsson, J. Peterson, and P. Hudak. Functional hybrid modeling. In Proceedings of PADL'03: 5th International Workshop on Practical Aspects of Declarative Languages, volume 2562 of Lecture Notes in Computer Science, pages 376–390, New Orleans, Louisiana, USA, January 2003. Springer-Verlag.

[20] H. Nilsson, J. Peterson, and P. Hudak. Functional hybrid modeling from an object-oriented perspective. Simulation News Europe, 17(2):29–38, September 2007.

[21] Exploiting structural dynamism for straightforward simulation of ideal diodes. G. Giorgidze, H. Nilsson; THESE PROCEEDINGS

[22] D. Broman. Flow Lambda Calculus for declarative physical connection semantics. Technical Reports in Computer and Information Science 1, Linköping University. Electronic Press, 2007.

[23] D. Broman and P. Fritzson. Higher-order acausal models. In P. Fritzson, F. Cellier, and D. Broman, editors, Proceedings of the 2nd International Workshop on Equation-Based Object-Oriented Languages and Tools (EOOLT), number 29 in Linköping Electronic Conference Proceedings, pages 59–69, Paphos, Cyprus, 2008. Linköping University Electronic Press.

[24] P. C. Breedveld Port-based modeling of mechatronic systems. Mathematics and Computers in Simulation (MCS), vol.66/2-3, 2004, pages 99 - 127