INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

D4.1.2 – Modelica extensions for multi-mode DAE systems

WP4.1 Theory, semantics and standardization of multi-mode DAE systems

WP4 Systems with multiple operating modes

MODRIO (11004)

Version M36 (3.0)

Date April 22, 2016

Authors

Hilding Elmqvist, Sven Erik Mattsson	Dassault Systèmes AB, Sweden
Martin Otter	DLR, Germany
Marc Bouissou	EDF, France
Albert Benveniste	INRIA, France

Executive summary

This report summarizes the developments in the MODRIO project to extend Modelica for multi-mode Differential Algebraic Equations (DAEs) including stochastic transitions. The goal is (a) to model physical systems with drastic structural changes including model changes due to failure situations and (b) to model stochastic hybrid systems as needed for reliability and availability calculations. In particular systems shall be handled where the number of equations can change dynamically during simulation. The following new extensions of Modelica and of symbolic transformation algorithms have been developed and evaluated with a Dymola prototype of Dassault Systèmes AB, Sweden (see deliverable D4.2.2):

(1) A new method and a Modelica design is proposed to describe conveniently models with drastic structural changes (*Elmqvist et al. 2014*): The Modelica 3.3 discrete-time synchronous state machines are generalized to continuous-time state machines having continuous-time *models* as “states”. Every model can be a “state” of a state machine, especially also models with *physical/acausal interfaces*. The latter is a completely new concept and has the advantage that the user does not have to cope with the typical “explosion” of variants as it is the case with standard hybrid automata.
(2) Systems of (1) can no longer be handled with the standard symbolic transformation algorithms of Modelica tools. For this, a simple technique was developed in (*Elmqvist et al. 2014*) so that existing algorithms can be extended with reasonable effort. These techniques are applicable as long as the set of equations that needs to be differentiated does not change during simulation.
(3) Furthermore, a generalization of the Pantelides algorithm has been developed in (*Mattsson et al. 2015*) in order to handle also systems with dynamically varying index where in principal in every mode a different set of differentiated equations is needed.
(4) With the continuous-time state machine of (1) and a random number generator, it is in principal possible to formulate a continuous time state machine with stochastic transitions. This basic feature was used in (*Bouissou et al. 2014*) to model stochastic hybrid systems and perform Monte Carlo simulations on them. Dymola was extended so that a large set of simulations (> 100 000 simulations) can be carried out, the simulation results collected together and analyzed to compute the reliability and availability of a system.
(5) The developed methods have been evaluated at hand of small benchmark problems that are available together with this deliverable.

The performed developments are very promising. However, they are still in a prototyping phase and larger and more realistic benchmarks are needed to evaluate and improve the implemented algorithms. Furthermore, there are also still unresolved issues such as defining the initialization of a multi-mode system in a convenient way, or handling Dirac impulses.

In the accompanying deliverable D4.1.1 (Benveniste et al. 2016) a fundamental new structural analysis method for multi-mode DAE systems is developed that is based on a rigorous mathematical derivation using Non-Standard-Analysis. In particular also Dirac impulses can be handled. This approach looks very promising, although there are still missing pieces in order that the theory could be used for larger models.

Contents

Executive summary **2**

1. Overview **4**

2. Synchronous state machines in current Modelica **4**

3. Continuous-time multi-mode state machines **5**

3.1. Overview 5
3.2. Hybrid automata 5
3.3. Acausal automata 6
3.4. Multi-mode systems with varying index 8
3.5. Limitations 9
3.6. Benchmark library 10

4. Stochastic Hybrid Systems **11**

5. Literature **13**

1. Overview

The goal in WP4 is to handle systems with multiple operating modes. Especially:

- Changing a controller from nominal operation to startup, shutdown or manual operation.
- Structural changes of a physical model (e.g. modeling the separation mechanism of a two or three stage rocket).
- Describing failure situations where the model structure is changing (e.g. an electrical line or a mechanical shaft breaks).
- Describing failure situations where the model changes completely (e.g. the normal behavior of a pump is a 0D model. In case of cavitation, a 1D model is needed to describe physics, requiring switching dynamically from a 0D to a 1D model when cavitation occurs).

As a result, the number of continuous-time state variables in the model may dynamically change during simulation. Such models cannot be described with current Modelica version 3.3 (*Modelica Association 2012*) that requires a fixed number of continuous-time states during a simulation.

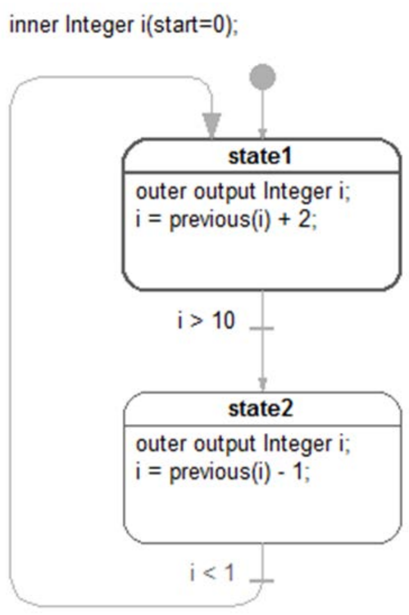
Note, the following sections utilize partially text and figures from (*Elmqvist et al. 2014*), (*Mattsson et al. 2015*), (*Bouissou et al. 2014*).

2. Synchronous state machines in current Modelica

In order to describe multi-mode systems in Modelica, the basic idea is to generalize the synchronous state machines introduced in version 3.3 of the Modelica specification. For the convenience of the reader, this concept of Modelica 3.3 is briefly summarized. From the Modelica Specification 3.3:

> Any Modelica block instance without continuous-time equations or algorithms can potentially be a state of a state machine. A cluster of instances which are coupled by transition statements makes a state machine. All parts of a state machine must have the same clock. All transitions leaving one state must have different priorities. One and only one instance in each state machine must be marked as initial by appearing in an initialState statement.

Example:



The system starts with i=0 and goes into “state1”. At every clock tick, “i” is incremented by 2. Once “i” is larger than 10, a transition from “state1” to “state2” occurs where “i” is decremented by 1 in every clock tick. Once smaller than 1, a transition from “state2” to “state1” occurs.

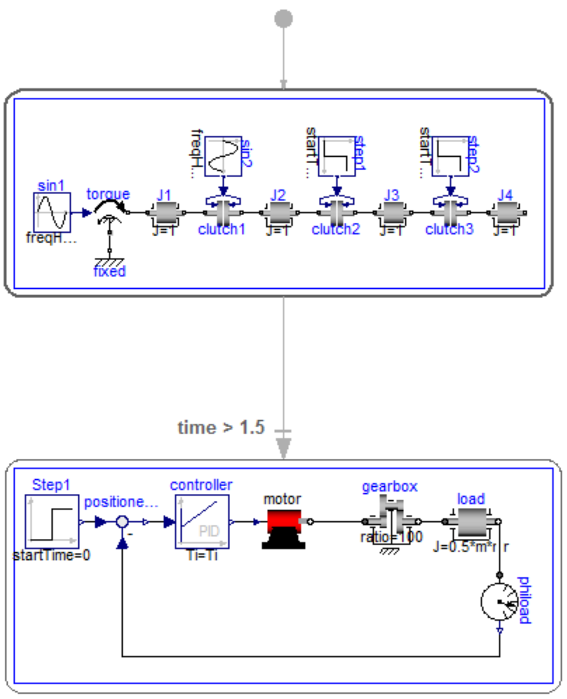
3. Continuous-time multi-mode state machines

3.1. Overview

In order to define multi-mode systems, the basic idea is to generalize the clocked state-machines from Modelica 3.3 to continuous-time. Two cases are distinguished:

(1) All states and transitions of one state machine are clocked and belong to the same clock (= semantics in Modelica 3.3).
(2) All states and transitions of one state machine are continuous-time (= new, additional semantics; discussed below).

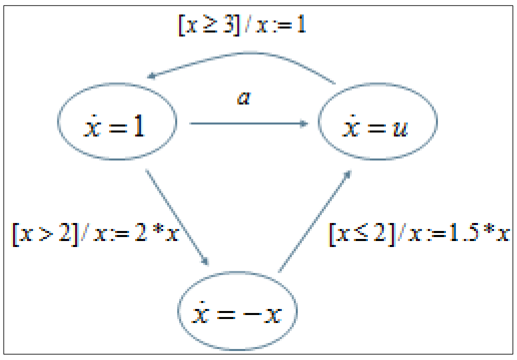
As a simple example consider the following continuous-time state machine with two “states”:

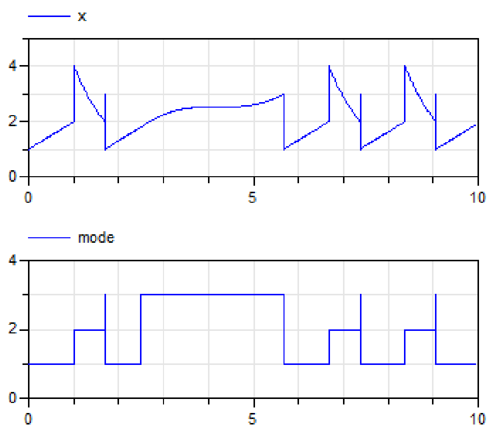


This state machine consists of “states” that consist of completely unrelated blocks (in the upper “state” it is a drive train with clutches, and in the lower “state” it is a controlled electrical motor with load). At the start of the simulation the upper state is active and the drive train is simulated. At time = 1.5 s, the simulation of the drive train is stopped and the controlled electrical motor block starts simulating. As can be seen, the number of continuous-time state variables changes dynamically during simulation and the state machine switches between unrelated models.

3.2. Hybrid automata

The system above is not very useful in practice since signal values are not exchanged between the different “state” blocks. More useful state machines are so called hybrid automata (see for example http://en.wikipedia.org/wiki/Hybrid_automaton). A simple example is shown in the next figure (on the left side a hybrid automata with “a” as input signal and on the right side a solution with “a = time > 2.5”).



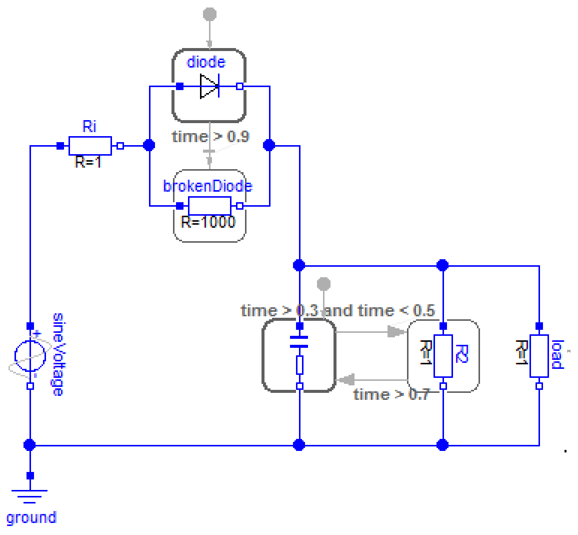


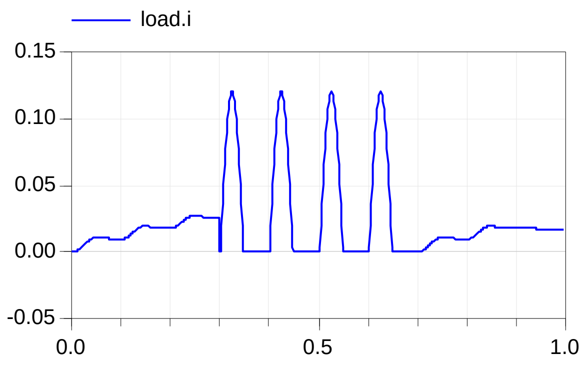
In every “state” an Ordinary Differential Equation (ODE) is present. The transition conditions consist of trigger conditions when to switch the state (e.g. “$[x \geq 3]$”). Furthermore, transition define in which way to reset the state of the ODE (e.g. “$/x := 1$” means that continuous-state x is reset to 1, before entering the target “state”). Hybrid automata can be described by the proposed Modelica extensions below and are supported in the Dymola prototype.

3.3. Acausal automata

Hybrid automata are of limited use for physical system modeling, because the equations have to be first manually transformed into input/output blocks and this is inconvenient and might be non-trivial. Furthermore, the graphical representation in an object diagram might not be “nice” due to input/output connections in a diagram that uses physical (that is acausal) connectors. Therefore, from a user point of view, it is important to support acausal models as “states”. In general this is non-trivial, because different “states” may require different symbolic handling of the equations in the environment since causality and the DAE index might change between “states” of a state machine. However, a quite large class of acausal model “states” that can reasonably be handled has been identified.

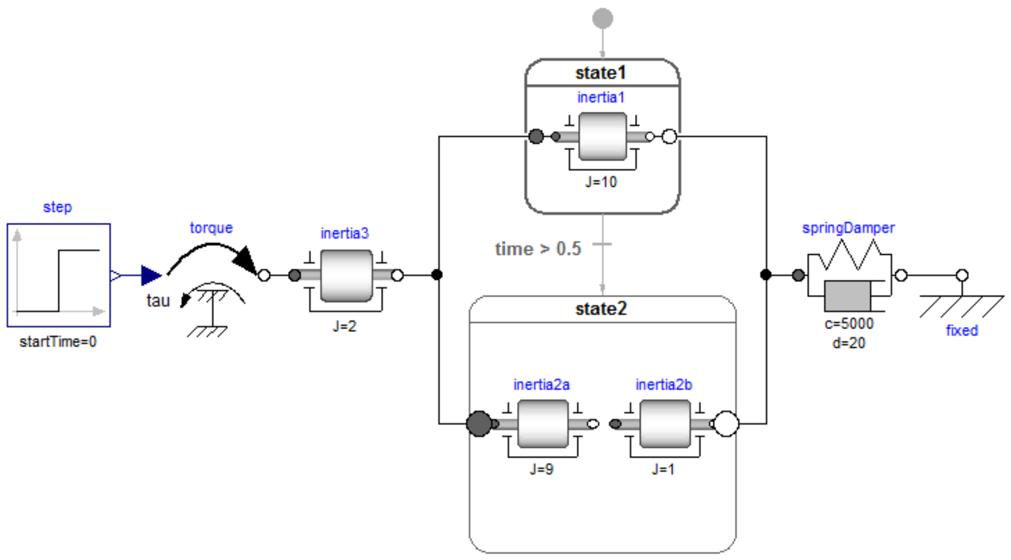
As an example, consider the following electrical circuit with two state machines included (the right figure shows a simulation result of the “load” resistor current):





In the upper part of the electrical circuit a diode model is present as a “state”. At “time > 0.9” this state is left and state “brokenDiode” is activated consisting of a resistor with a large resistance. Note, that the “states” have “electrical pins” that connect them to the rest of the circuit. In the lower right part of the circuit diagram, another state machine is present. It consists of a capacitor “state” (modeled as a resistor in series with an ideal capacitor model). At time 0.3 s, the hardware configuration is changed and the state switches to state “R2” consisting of a small resistance. At time 0.7s, the configuration is switched back. Note, that in this second case the number of continuous-time state variables changes when switching occurs.

Another example is shown in the next figure:



This example models a drive train, where a rotational inertia breaks during operation. In particular, this model consists of

- a state1 with inertia1 and J=10,
- a state2 with inertia2a (J=9) and inertia2b (J=1) that are not connected,
- an inertia3 outside of the state machine that is connected to inertia1 and inertia2a and is driven by a step torque[1], and
- a spring-damper that is connected to inertia1 and inertia2b.

At time = 0.5, the state machine switches from state1 to state2 and therefore inertia1 is replaced by two unconnected inertias that together have the same moment of inertia as inertia1. In other words, the “breaking” of inertia1 is modelled. Note, that (a) the number of continuous-time state variables changes (from two variables in state1 to four in state2), (b) there is an algebraic loop over inertia3, inertia1 and inertia2a, and (c) index reduction over inertia3, inertia1 and inertia2a is needed, because the flange angles of these components are constrained to be identical, but the tool has to deduce that flange angular velocities and angular accelerations are also constrained to be identical.

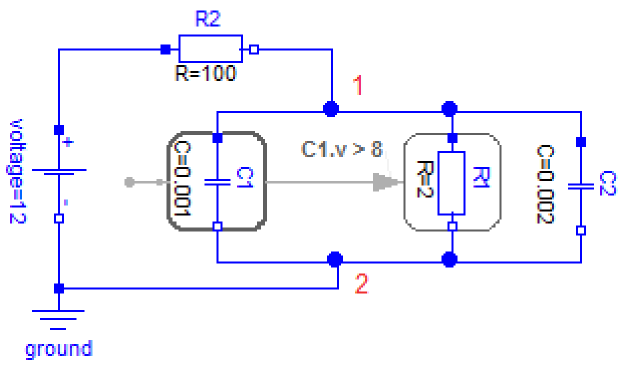
(*Elmqvist et al. 2014*) describes how to symbolically transform state machines of the above kinds into a form that can be efficiently simulated.

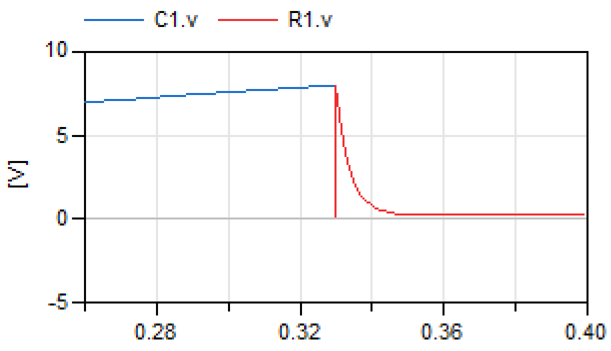
[1] inertia3 is only present to demonstrate index reduction and algebraic loops over a state machine

3.4. Multi-mode systems with varying index

There are models where the algorithms of (*Elmqvist et al. 2014*) fail. To cope with them, an involved generalization of the Pantelides algorithm was developed in (*Mattsson et al., 2015*) and evaluated with a Dymola prototype. This new algorithm allows for example to handle the following type of systems.

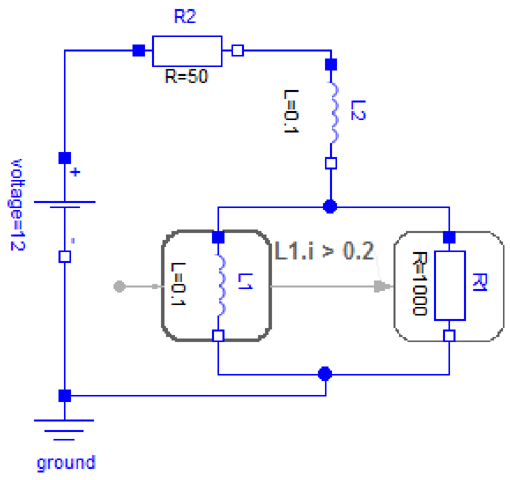
The circuit below describes a capacitor C1 that is destroyed when the voltage becomes too large. The destroyed capacitor is modelled with a small resistor R1. When C1 is active, there is a state constraint between C1.v and C2. When C1 is not active, this state constraint is no longer present.



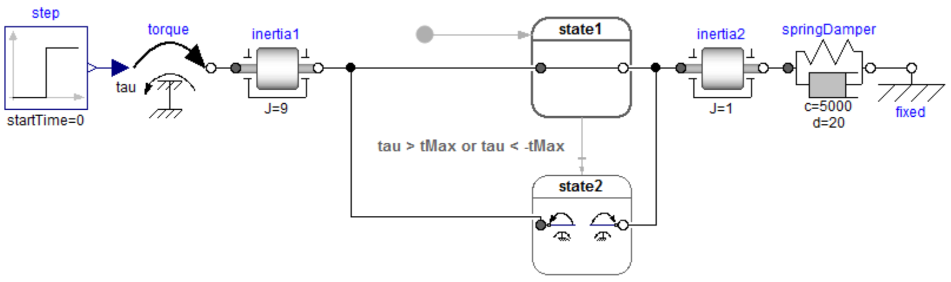


Circuit with different numbers of differentiated equations in the two modes.

Below other examples are shown that can be handled with the “multi-mode” Pantelides algorithm:



Inductors in series, where one of the inductors is destroyed when the current becomes too large.



Shaft that breaks due to an overload toque tau > tMax or tau < -tMax

3.5. Limitations

The algorithms have been tested with several simple examples. However, many more tests especially with large models are needed. It might still be the case that improvements of the algorithm are needed. The following limitations are already known:

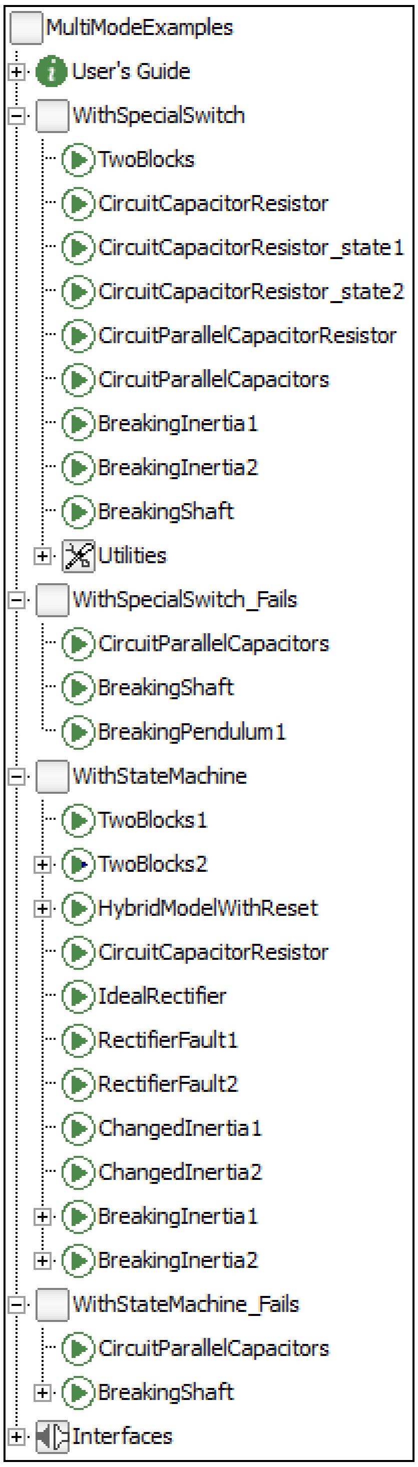
When using continuous-time state machines it is easy to model systems where Dirac impulses occur. For example, when closing an electrical switch the voltage drop is not zero and therefore a Dirac impulse occurs. Simulation is usually successful. However, the “propagation” of impulses is not taken into account and therefore in many cases the simulation results will not be correct.

Another issue arises from the transition conditions: When they are functions of the state connector variables and these variables are differentiated, then the transition conditions might need to be differentiated as well. Again, it is not yet clear how to handle such situations.

Furthermore, initialization of multi-mode systems is currently only performed in a rudimentary, inconvenient way. Especially, it must be known in which mode the system shall start simulation (in practical applications, it might be difficult to figure this out).

3.6. Benchmark library

Sven Erik Mattsson from Dassault Systèmes implemented a Dymola prototype for evaluation of the concepts. Hilding Elmqvist and Martin Otter developed many models for evaluating the ideas, concepts and the Dymola prototype. A cleaned-up version of a subset of the developed evaluation models is provided in the public domain as Modelica library “MultiModeExamples”:



```
MultiModeExamples
+ User's Guide
- WithSpecialSwitch
    TwoBlocks
    CircuitCapacitorResistor
    CircuitCapacitorResistor_state1
    CircuitCapacitorResistor_state2
    CircuitParallelCapacitorResistor
    CircuitParallelCapacitors
    BreakingInertia1
    BreakingInertia2
    BreakingShaft
  + Utilities
- WithSpecialSwitch_Fails
    CircuitParallelCapacitors
    BreakingShaft
    BreakingPendulum1
- WithStateMachine
    TwoBlocks1
  + TwoBlocks2
  + HybridModelWithReset
    CircuitCapacitorResistor
    IdealRectifier
    RectifierFault1
    RectifierFault2
    ChangedInertia1
    ChangedInertia2
  + BreakingInertia1
  + BreakingInertia2
- WithStateMachine_Fails
    CircuitParallelCapacitors
  + BreakingShaft
+ Interfaces
```

The library consists of two main parts:

- Sub-library WithSpecialSwitch contains models that should work with any Modelica 3.2 tool, by using special switch components. These examples allow analyzing the occurring equation structures and the needed symbolic transformation algorithms for multi-mode systems, without actually supporting continuous-time state machines.
  In order to generate efficient code, the non-active equations should not be evaluated. This is not possible when using Modelica 3.2 standard conformant elements and is only possible when using continuous-time state machines.
- Sub-library WithStateMachine contains models based on the sketched Modelica extension of continuous-time state machines.

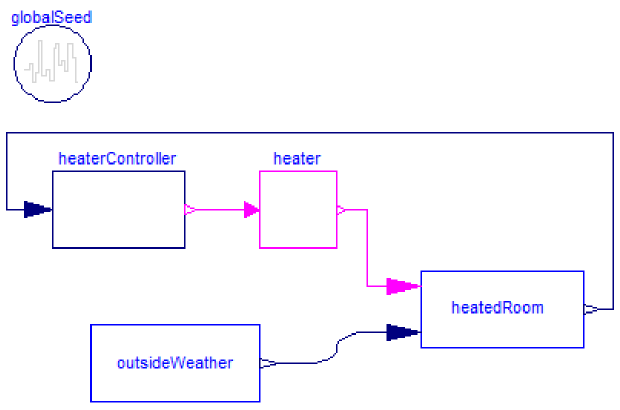
All models in the above sub-libraries simulate with the Dymola prototype. There are additional sub-libraries “WithSpecialSwitch_Fails” and “WithStateMachine_Fails” containing models that do not yet simulate in the Dymola prototype, due to varying DAE index.

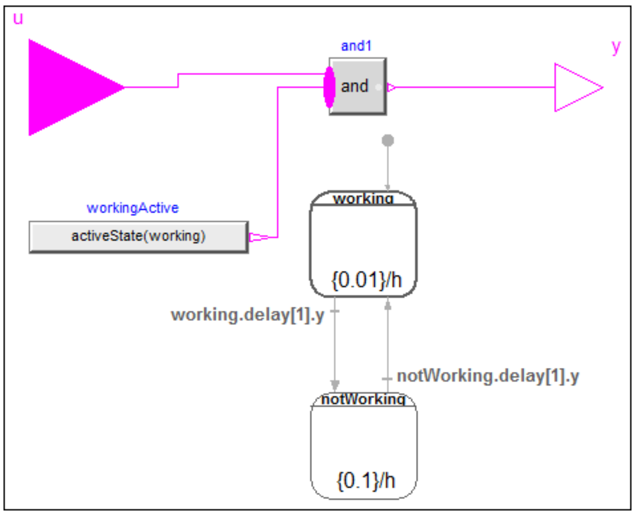
All models discussed in the publication about multi-mode modeling are contained in this library as well.

4. Stochastic Hybrid Systems

The continuous-time state machines from the previous section can be used to model continuous-time state machines with stochastic transitions. In (*Bouissou et al. 2014*) a new method was developed to utilize this feature to compute the reliability and availability of systems with hazard rates that depend on the underlying physical system (e.g. if the temperature of a component is increasing, the probability of a component failure may become larger).

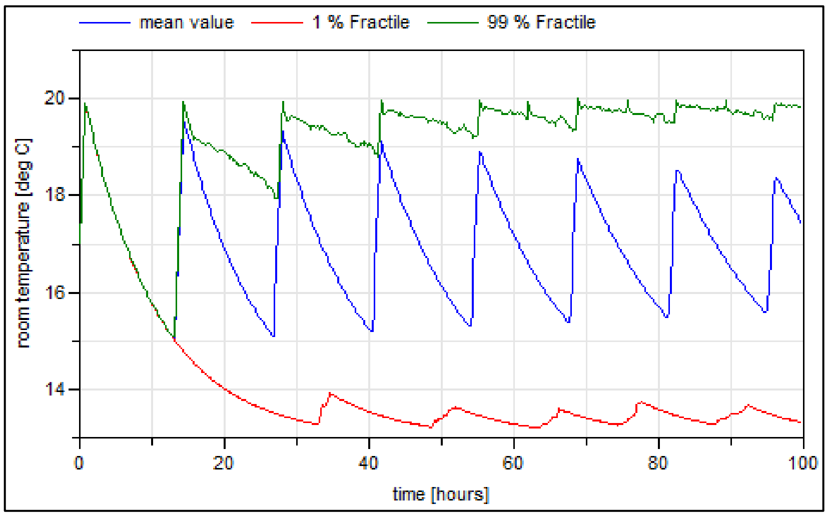
The new approach was evaluated using a small benchmark:





The system above describes a room with a heater. A temperature sensor with a hysteresis (`heaterController`) switches the heater on when the ambient temperature falls below 15°C and switches it off when the temperature reaches 20°C. The outside temperature is constant: 13°C. At time t = 0, the temperature of the room is 17°C, and the heater is on. The flow of energy (power) traversing the walls is proportional to the difference of temperature between the inside and the outside of the room. When the heater is on, it injects a constant power inside the room. If the heater was not subject to failures, the trajectory of temperature as a function of time would be a deterministic succession of portions of exponential functions, alternatively convex and concave, "oscillating" between 15 and 20 °C. But in fact, the heater has a constant failure rate λ = 0.01/h, and a constant repair rate μ= 0.1/h. The question is how does this random behaviour affect the evolution of the temperature?

The above model can directly be used to simulate one realization of the random process corresponding to the life of the system. To perform a Monte Carlo simulation to estimate, for example, the mean temperature as a function of time, it is necessary to generate a large number of such trajectories using different initial seeds for every simulation. This task has been performed in Dymola by using appropriate script functions (that are based on the algorithmic part of the Modelica language). A special Modelica/Dymola script has been implemented for this case in the MORIO project to run the simulations and store the desired fractiles. In the figure below the mean value of the room temperature is shown, as well as the 1% and 99% fractiles at each time point respectively. 10 000 simulations were performed with 500 output points per simulation. On a notebook, these simulations took 25s. Computing the result values for figure 8 took another 45s (the reason is that a very simple algorithm was implemented to compute the fractiles and a better implementation will give a considerable speed-up).



The computation of the 99 % fractile z from 10000 simulation runs means that at every grid point of the result 10000 result points are available and that 99 % of these are smaller than z. In other words, 1% of 10000 values = 100 values are larger than z.

Another example, very well known in the reliability community as the “heated tank test case”, has been solved and compared to previous published results in (*Bouissou & De Bossoreille 2015*). This example is more interesting than the preceding example because it includes two coupled continuous variables: the water level and the temperature in a tank, and, more importantly, the failure rates of components are highly dependent on the temperature. Neglecting this feature would lead to completely false results. The simulation time with Dymola compares well to the best results published in the literature so date.

5. Literature

Benveniste A., Bourke T., Caillaud B., Pouzet M. (2016): **Semantics of Multi-Mode DAE Systems**. MODRIO Deliverable D4.1.1.

Bouissou M., Elmqvist H., Otter M., Benveniste M. (2014): **Efficient Monte Carlo simulation of stochastic hybrid systems**. Proceedings of the 10th International Modelica Conference, Lund, Sweden, March 10-12. http://www.ep.liu.se/ecp/096/075/ecp14096075.pdf

Bouissou M., de Bossoreille X. (2015): **From Modelica models to dependability analysis**. Proceedings of DCDS 2015, Cancun, June 2015. IFAC-PapersOnLine, Volume 48, Issue 7, 2015, Pages 37-43.

Elmqvist H., Mattsson S.E., Otter M. (2014): **Modelica extensions for Multi-Mode DAE Systems**. Proceedings of the 10th International Modelica Conference, Lund, Sweden, March 10-12. http://www.ep.liu.se/ecp/096/019/ecp14096019.pdf

Mattsson S.E., Otter M., Elmqvist H. (2015): **Multi-Mode DAE Systems with Varying Index**. Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23. http://www.ep.liu.se/ecp/118/009/ecp1511889.pdf

Modelica Association (2012): **Modelica, A Unified Object-Oriented Language for Systems Modeling. Language Specification, Version 3.3,** May 9, 2012. https://www.modelica.org/documents/ModelicaSpec33.pdf