Simulation of Variable Structure Models using Rand Model Designer

Yury Senichenkov
Dept. Distributed Computing and Networking
Saint Petersburg State Polytechnic University
Saint Petersburg, Russia
senyb@dcn.ftk.spbstu.ru

Yury Kolesov
Dept. Research and Development
SP-Center, Federal State Company
Moscow, Russia
ybk2@mail.ru

Alfonso Urquia
Carla Martin-Villalba
Dept. Informática y Automática
Universidad Nacional de Educación a Distancia (UNED)
Madrid, Spain
{aurquia,carla}@dia.uned.es

***Abstract*— Rand Model Designer (RMD) is a modeling environment that supports Model Vision Language, an object-oriented modeling language for hybrid-DAE systems. Model Vision Language allows describing the continuous-time part of the model combining the use of equations (i.e., acausal modeling) and sequences of assignment statements. The hybrid behavior is described using behavior charts (B-chart). A B-chart is a diagram consisting of modes, transitions internal to the modes and transitions between modes. The activity of the mode is specified by a local class (the so-called activity class of the mode) that describes the system structure and behavior while in this mode. The system of equations that describes the complete model at certain time is automatically built by RMD at run time. Every time a transition takes place (i.e., the model structure changes), RMD constructs the mathematical description of the actual model, eliminates the redundant variables and trivial equations resulting from component connections, analyzes the model solvability and structure, selects the best-suited numerical method and generates the input to this numerical method. This approach provides high flexibility in the description of variable structure models. This feature is demonstrated using variable structure models arisen in two different applications. The first application is the run-time change in the selection of the model state variables. Support for several selections of the state variables is typically required in interactive simulations. The second application is the description of systems with variable behavior. The model of an industrial boiler is used to illustrate the description of this type of variable structure models.**

Keywords- object oriented modeling; computer simulation; numerical models; numerical simulation; engineering education

I. INTRODUCTION

Rand Model Designer (RMD) is a modeling environment that has been developed by the MvStudium Group. RMD supports Model Vision Language [1,2], an object-oriented modeling language for hybrid systems based on the Unified Modeling Language [3]. RMD is a commercial tool. However, a free version of RMD, licensed for private non-commercial and educational use, can be downloaded from [4]. This free version of RMD, named Rand Model Designer Lite, has been used to develop and simulate the modeling examples discussed in this manuscript.

On the other hand, Modelica [5] is a de-facto standard language for object-oriented modeling of hybrid DAE (differential-algebraic equation) systems. Model Vision Language and Modelica have many common features. Some of them are listed below. This is not intended to be an exhaustive comparison. Some features of Model Vision Language are not discussed and many relevant features of Modelica, that is a more extensive language, are not mentioned. Both languages facilitate:

- Modular and hierarchical modeling. New models can be defined as the connection between the previously defined models.
- Model connection on the basis of the energy-balance principle. The connector class facilitates the description of model interfaces and connections. Connector variables can be either across or through (also known as contact and flow variables). Across variables in a connection point have the same value, while through variables are summed up and the sum is equaled to zero.
- Class inheritance and parametrization.
- Definition of partial classes (also known as abstract classes) that cannot be instantiated.
- Application of object-oriented modeling principles, including abstraction (i.e., capability of using components without knowing their internal details) and information encapsulation (i.e., only the model interface variables are accessible by other models). Abstraction is supported by distinguishing between model interface and internal description.
- Acausal model description using differential and algebraic equations, supporting the use of vector and matrix equations.
- Combined use of equations and algorithms (i.e., sequences of assignment statements).
- Reuse of algorithms as functions.
- Interfacing with C and FORTRAN code.
- Including annotations with additional information in the model (e.g., graphical annotations).
- Declaration of user-defined data types.
- Development and reuse of model libraries.

978-0-7695-5073-2/13 $26.00 © 2013 IEEE
DOI 10.1109/EUROSIM.2013.60

Model Vision Language and Modelica have also some differentiated characteristics. Different approaches can be observed regarding class inheritance and parametrization. These are briefly discussed below.

Modelica supports multiple inheritance, while Model Vision Language supports single inheritance. In both languages, a derived class (subclass) inherits the behavior and structure of the base class (superclass), and may extend the base class adding new behavior and structure (i.e., equations, components, connections, etc.). In addition, Model Vision Language allows removing in the derived class a part of the inherited behavior.

A model class parameter is any property that can be modified in order to adapt the model to its different applications. Model Vision Language and Modelica allow time-independent variables to be declared as class formal parameters, so that they can receive their actual values in each class instance. In addition, Modelica supports redeclaring the class of the objects instantiated in the model.

The description of the hybrid behavior is a major difference between Model Vision Language and Modelica. Different compromises are achieved between flexibility in the model description and computational efficiency. This will be discussed in Section II.

The approach adopted in the model translation is also different. Modelica modeling environments [6] (e.g., Dymola [7] and OpenModelica [8]) perform the symbolic manipulations on the model, which are intended to generate the executable simulation code from the Modelica model, in a step (the so-called model translation) that is previous to the simulation run. On the contrary, RMD obtains the actual description of the complete model, performs the solvability and structural analyses, selects the best-suited numerical method and generates the input to the numerical method at run time, every time the model structure changes. These two different approaches will be discussed in Section III.

Variable structure models developed for different type of applications can be described using Model Vision Language and simulated using RMD. Two typical applications are considered in Sections IV and V. The first application is the run-time change in the selection of the model state variables, which is typically required in interactive simulations. The second application is the description of systems with variable behavior. Other applications could be devised, e.g., run-time change in the detail level of component descriptions.

Therefore, the objective of this paper is threefold: (1) comparing the features provided by Modelica and Model Vision Language to model hybrid behavior; (2) comparing the different approaches for model translation into programming language and simulation adopted by the Dymola and OpenModelica Modelica tools, and RMD; and (3) illustrating Model Vision Language and RMD use in describing and simulating variable structure models.

II. Description of Variable Structure Models

Conceptual differences can be observed in the description of the event trigger conditions and the event actions in Modelica and Model Vision Language. These differences are discussed below.

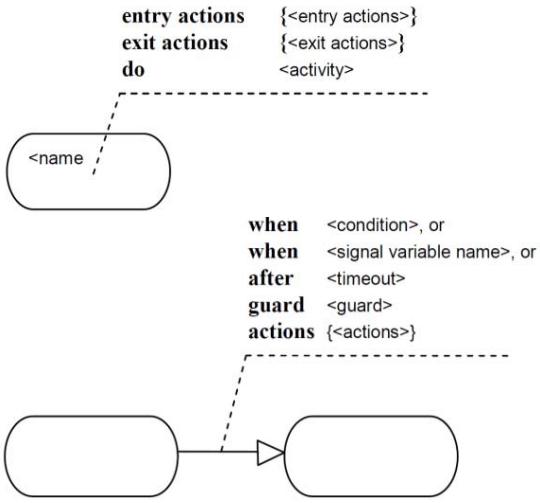

Figure 1. B-chart mode (above) and transition between modes (below).

Modelica provides language constructs to describe the trigger conditions of time and state events, and also the actions associated to the events [9–11]. These actions can be: (1) update the value of discrete-time variables (e.g., using the *when* clause and the *pre* function); (2) reinitialize continuous-time state variables, using *when* clauses and the *reinit* function; and (3) change the mathematical description of equations and assignments, using the *if* statement.

A Modelica model must comply with the single-assignment rule. This means that the number of unknown variables and equations in the model has to be equal and constant, and that the number of equations in each branch of a conditional *if* equation must also be equal and constant. Otherwise, the model is incorrect.

Equations in Modelica follow the synchronous data flow principle. This means that at each time instant the active equations describe relationships among variables that have to be satisfied concurrently [11]. The set of active equations can be composed of only continuous equations (during continuous integration), or mixed continuous and discrete equations, if an event has been triggered and needs to be evaluated. The equation evaluation order is automatically determined by analyzing the model structure, leading to unique computations of the unknown variables [12]. Both continuous and discrete equations have to be considered during the sorting procedure, in order to obtain the correct evaluation order for the possible sets of active equations.

On the other hand, the hybrid behavior is described in Model Vision Language using behavior charts. A behavior chart (B-chart) is a diagram consisting of modes, transitions internal to the modes and transitions between modes. Modes are represented as rectangles with rounded corners and transitions between modes as arrows connecting the source mode to the target mode (see Fig. 1).

Entry and exit actions and an activity can be associated to each mode. The entry and exit actions are sequences of assignment statements that are executed every time the mode is entered and exited, respectively. The activity of the mode is specified by a local class (the so-called activity class of the

mode) that describes the system structure and behavior while in this mode. Activity classes can be as complex as required; including composed models and hybrid models described by B-charts. Quantities that exclusively define the model behavior while in certain mode can be declared as local variables of the activity class for that mode. Quantities common to all modes can be declared as variables of the B-chart.

Trigger and guard conditions, and actions (i.e., sequence of assignment statements) can be associated to the transitions between modes. If the source mode of a transition is the active mode of the B-graph, the transition is triggered and the guard condition is satisfied, then the action is executed and the target mode becomes the active mode of the B-graph. If the guard condition is not satisfied when the trigger occurs, then the transition is not executed and the trigger occurrence is lost.

Models can be composed in Model Vision Language as the connection of components. Hybrid components are described using B-charts. Exactly one mode of each B-chart is active at every time. The system of equations that describes the complete model at certain time is automatically built at simulation time by RMD from all the active modes (one active mode per B-chart) at that time. The following example illustrates this point.

Consider a model library of electrical components that contains models of a voltage generator, resistor, capacitor, diode, etc. Suppose that the diode model of this library is described using a B-chart with two modes [11], named `off` and `on`, as shown in Fig. 2. The interfaces of the library components are described using connectors that represent electrical pins. This electrical connector is composed of an across variable and a through variable: the voltage and electric current, respectively. Electric circuits can be described by instantiating and connecting these library components.

Suppose that the diode model, together with other models of the electrical library, is used to compose the rectifier circuit shown in Fig. 3. At any time, every diode model is in one of the two modes of its B-chart. As the rectifier circuit model contains four diodes, there are $2^4 = 16$ possible combinations of modes. Every time a transition takes place in the model of any of the four diodes, RMD automatically constructs the equation system that represents the new structure of the complete circuit model and resumes the simulation run using this new equation system.

As local variables can be declared in the activity classes, the number and meaning of the model variables can change during the simulation run. In addition, a B-chart variable can be a state variable while in certain modes, an algebraic variable while in other modes and can be undefined (i.e., set to NaN - Not a Number) in the remaining modes.

The flexibility of Model Vision Language in the description of variable structure models is one of its strong points. However, this approach has inherent limitations given that it implies constructing, analyzing and manipulating the actual model in run time, every time the model structure changes. It will be discussed in the next section.

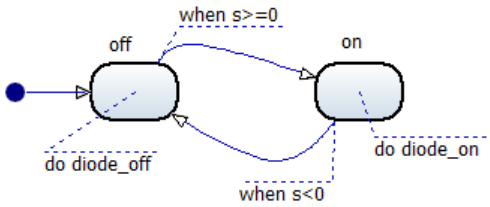

Figure 2. B-chart describing the behavior of a diode.

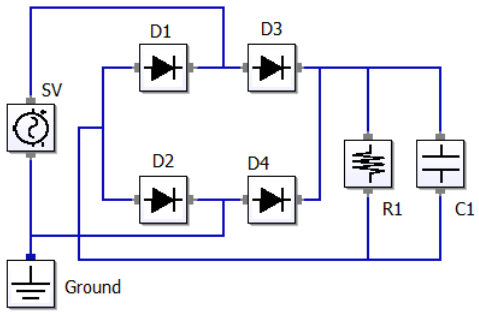

Figure 3. Rectifier circuit model composed using RMD.

III. Model Translation and Simulation

As indicated previously, RMD and the Modelica tools perform the model translation differently. Modelica tools approach will be discussed first. Two state-of-the-art Modelica environments are taken as a reference in this discussion: Dymola [7] and OpenModelica [8]. The symbolic manipulations that these software tools carry out on the model can be classified into two types according to their purpose [13].

The first type of manipulations is intended to translate the object-oriented description of the model into the so-called flat model. The flat model contains the complete set of model equations and functions, with all the object-oriented structure removed.

The second type of manipulations transforms the flat model into an efficiently solvable form. These manipulations include [14]:

- The efficient formulation of the complete-model equations, eliminating the redundant variables and the trivial equations resulting from the component connections.
- The reduction of the system index to zero or one [15–17].
- The analysis of computational causality (partition) and equation sorting.
- The symbolic manipulation of linear systems of simultaneous equations.
- The tearing [18] of nonlinear systems of simultaneous equations.

All these manipulations, which are intended to generate the executable simulation code from the Modelica model, are performed in a step (named model translation) that is previous to the simulation run. This approach allows use of complex algorithms for model structure analysis, model

symbolic manipulation and generation of highly efficient numerical code.

On the contrary, model manipulations are performed by RMD during the simulation run. Every time a transition takes place, RMD constructs the mathematical description of the actual model, eliminates the redundant variables and trivial equations resulting from component connections, analyzes the model solvability and structure, selects the best-suited numerical method and generates the input to the selected numerical method.

RMD approach has strong points. As discussed in Section II, a strong point is high flexibility in the description of variable structure models. Another strong point is that RMD needs to keep in memory the simulation code of only the active modes of the model. When the model structure changes, the simulation code corresponding to the old model structure is deleted and the simulation code of the new model structure is automatically generated.

A limitation of RMD approach is the following. As model analyses and simplifications need to be applied frequently during the simulation time, only fast algorithms can be implemented in RMD. RMD contains a highly efficient algorithm for numerical method selection that is applied every time the model structure changes. Numerical methods for algebraic equations, differential equations and differential-algebraic equations (DAE) are supported. Supported DAE solvers for stiff systems include RADAU, DDASSL and DDASPK. However, symbolic differentiation and index reduction are not currently supported by RMD.

IV. Run-Time Change on the Selection of the Model State Variables

The distinctive characteristic of interactive simulations is that external objects are allowed to change at event instants the value of certain model quantities, named interactive quantities. These events are named interactive events. The time instants when these changes are triggered are determined by the external objects. An arbitrary finite number of interactive events can be triggered during the simulation run. Depending on the application, the external objects can be people (e.g., in virtual laboratories), hardware (e.g., in hardware-in-the-loop simulations) and another model simulations (e.g., in distributed real-time simulation). A bi-directional flow of information between the interactive model and the external objects is established during the simulation. The model sends to the external objects the actual value of selected model quantities. The external objects send to the model the information required to execute the interactive events.

RMD has been specifically conceived for interactive simulation. RMD generates two types of executable simulation models [1].

- The first one is a stand-alone Windows application that facilitates analyzing the mathematical structure of the model, defining and running experiments on the model, and visualizing the simulation results. RMD provides some ready-to-use 2D and 3D animation and control components that can be used to define interactive user-to-model interfaces. These interfaces may contain animated diagrams, plots and controls that allow the user to interact with the model during the simulation run.
- The second one is a Windows dynamic link library that can be used as an embedded interactive application (dll generation is not supported in the lite version of RMD).

Interactive quantities have to be state variables. Therefore, adapting a model for interactive simulation implies modifying the model so that all the interactive quantities are selected as state variables [19]. The original model of the system is named physical model and its reformulation for interactive simulation is named interactive model.

Parameters (i.e., time-independent variables) of the physical model can also be interactive quantities. To this end, they are defined as continuous-time state variables of the interactive model. The derivative of these state variables is set to zero. As a result, the value of these states remains constant between interactive actions.

An interactive model can define several interactive events, each one with its own interactive quantities. The following restrictions apply on the selection of the interactive quantities [20]:

- A time-dependent variable can be an interactive quantity if and only if there is at least one selection of the state variables that includes this variable.
- A set of variables can be interactive quantities modified in the same interactive action if and only if there is at least a selection of the state variables that includes to all the variables in the set.

In general, different choices of the state variables are possible in the physical model. Different interactive events may require of different selections of the state variables. On the other hand, as the state variable values that are not explicitly modified in the interactive event action remain unchanged at the event instant, the result of the interactive actions depends on the state variable selection.

The model shown in Fig. 4 will be used to illustrate the previous discussion. The voltage applied to the pump (v) is an input variable (i.e. its value is not calculated from the model equations). The cross-sections of the tank (A) and the outlet hole (a), the pump parameter (k) and the gravitational acceleration (g) are parameters (i.e. time-independent quantities of the model). The liquid volume (V), the input and output flows (F_{in}, F), and the liquid level (h) are time-dependent variables of the physical model.

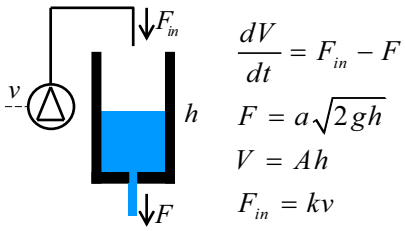

Figure 4. Model used to illustrate run-time selection of state variables.

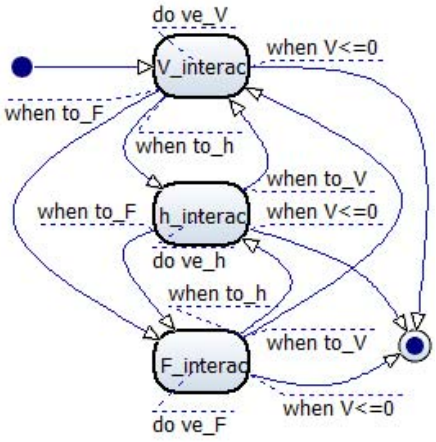

Figure 5. B-chart of the tank system shown in Fig. 4.

Different choices of the state variables are possible in this model. For instance, the liquid volume (V), the liquid level (h) or the output flow (F) could be chosen as state variable. The state selection should be made so that it includes all the interactive quantities. For instance, if the user wants to change interactively the level value, the appropriate choice for the state variable is h. Likewise, if the user wants to change the liquid volume, then the right choice is V; and if they want to change the output flow, then F.

In addition, changes in the value of interactive quantities can have different effects depending on the state variable choice. Consider that the user changes interactively the tank cross-section (A). If the volume (V) is a state variable, then the change in A produces an abrupt change in the value of the liquid level (h) and flow (F), whereas the liquid volume remains constant. On the contrary, if the state variable is the height (h) or the flow (F), these quantity values would not change as the result of an instantaneous change in the cross section (A), but the volume would.

Consider a virtual lab intended to illustrate the tank system behavior. Suppose that the virtual lab is required to support three ways of describing interactive changes in the amount of liquid contained in the tank: changes in the liquid volume (V), changes in the liquid level (h) and changes in the output flow (F). Every time users need to change the amount of liquid, they have to be allowed to choose between describing it in terms of the volume, the level or the output flow. Different choices are possible during a given simulation run.

The interactive model can be implemented in RMD as shown in Fig. 5. The B-chart is composed of as many modes as different state selections are required: `V_interact`, `h_interact` and `F_interact` modes. The `ve_V` activity class, which is associated to the `V_interact` mode, contains the tank system model with the liquid volume as state variable. Analogously, the `ve_h` and `ve_F` activity classes, which are associated to the `h_interact` and `F_interact` modes, contain the model with the liquid level and the output flow as state variables, respectively. The tank area (A), hole area (a) and pump input voltage (v) are defined as interactive quantities in the three activity classes. The adequate mode (i.e., that with the required state selection) is used for executing each interactive action from the user. The transition trigger conditions are defined using three signals (`to_V`, `to_h` and `to_F`) that are emitted by buttons placed in the graphic model-to-user interface. The simulation ends when the liquid volume becomes equal or less than zero.

V. Models with Variable Behavior

Industrial boilers are widely used in the chemical industry. As this process unit constitutes a good example of a multivariable system, it is also often used in control education. A virtual laboratory intended to illustrate the dynamic behavior of an industrial boiler that operates under two different control strategies (manual and decentralized PID) is described in [21,22]. The input of liquid water is placed at the boiler bottom and the vapor output valve is placed at the top. The water contained in the boiler is continuously heated. The water level inside the boiler and the output flow of vapor are the controlled variables. The pump throughput and the heater power are the manipulated variables.

This process unit can also be used to illustrate the description of variable behavior models using Model Vision Language. To this end, the water boiler, heating system, liquid source, vapor output valve and vapor downstream pressure have been modeled using RMD. The model diagram is shown in Fig. 6. The boiler model is based on the mathematical model described in [23]. The constitutive relation of the valve is (1), where $F_0 = 1.1 \cdot 10^{-7}$ Kg·s^{-1}·Pa^{-1}. The downstream vapor pressure at normal operating conditions is $p_0 = 1.14 \cdot 10^6$ Pa.

$$F = F_0 \cdot \sqrt{p \cdot (p - p_0)} \tag{1}$$

The relationship between vapor pressure p [Pa] and boiling temperature T [K] is (2), where $x = 647.27 - T$ K.

$$\log_{10}\left(\frac{220.89 \cdot 10^5}{p}\right) = \frac{x}{499} \cdot \frac{3.346 + 4.14 \cdot 10^{-2} \cdot x}{1 + 1.38 \cdot 10^{-2} \cdot x} \tag{2}$$

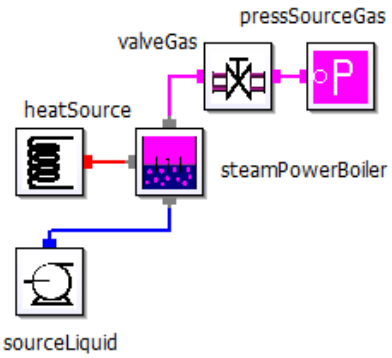

Figure 6. Diagram of the boiler model composed using RMD.

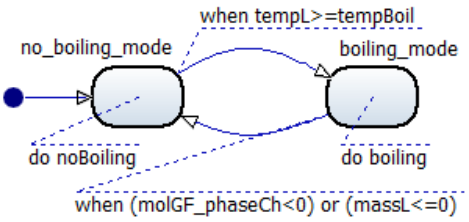

Figure 7. B-chart describing the boling process.

The boiling process has been modeled as follows [24]. While the liquid temperature is below the boiling temperature, no phase change takes place. If under this condition the liquid is heated, its temperature will grow until it reaches the boiling temperature. When the boiling temperature is reached, the boiling process starts. While the liquid is boiling, the liquid temperature equals the boiling temperature at the corresponding vapor pressure and the heat supplied to the liquid is employed in producing steam.

The boiling process is described using the two-mode model shown in Fig. 7. The degree-of-freedom number depends on the mode. While in the no-boiling mode, the mass and temperature of the liquid, and the number of moles and temperature of the vapor can be set independently. While in the boiling mode, the liquid mass, and the mol number and temperature of the gas can be set independently. However, the liquid temperature is a function of the vapor pressure, which is a function of the vapor pressure and moles, and the liquid mass (that determines the gas volume).

VI. Conclusions

Two strong points of Model Vision Language and RMD are high flexibility in the description of variable structure models and support for interactive simulations. However, the adopted approach has inherent limitations, given that it implies constructing, analyzing and manipulating the equations of the complete model in run time, every time the model structure changes. As model analyses and simplifications need to be applied frequently during the simulation time, only fast algorithms can be implemented in RMD. Future research includes developing highly efficient algorithms for symbolic differentiation and index reduction.

Acknowledgment

This work has been supported in part by Banco Santander and UNED, under the grant "Ayudas Banco Santander UNED para estancias en otros Centros de Investigación" (BICI 18-2-2013).

References

[1] Rand Service Ltd, Rand Model Designer User Manual. Rand Service Ltd, 2010. http://rand-service.com/instructions (accessed in 2013).

[2] D. Inihov, Y. Kolesov, and Y. Senichenkov, "Physical Modeling of Hybrid Systems with Rand Model Designer", Proc. 7th Vienna International Conference on Mathematical Modelling, vol 7, part 1, 2012, pp. 49–54.

[3] J. Rumbaugh, I. Jacobson, and G Booch, The Unified Modeling Language Reference Manual. Addison-Wesley, 2005.

[4] MvStudium Group, Download site for Rand Model Designer Lite: http://www.mvstudium.com/download.php (accessed in 2013).

[5] Modelica Association, Modelica® - An Unified Object-Oriented Language for Systems Modeling, Language Specification version 3.3. Modelica Association, 2012. https://www.modelica.org (accessed in 2013).

[6] Modelica Association, Modelica Tools. https://www.modelica.org/tools (accessed in 2013).

[7] Dynasim AB,. Dymola – Dynamic Modeling Laboratory User's Manual. Dynasim AB, Lund, Sweden, 2008.

[8] Open Source Modelica Consortium. http://www.ida.liu.se/labs/pelab/modelica/OpenSourceModelicaConsortium.html (accessed in 2013).

[9] H. Elmqvist, F.E. Cellier, and M. Otter, "Object-oriented modeling of hybrid systems", Proc. European Simulation Symposium, Delft, The Netherlands, 1993.

[10] S.E. Mattsson, M. Otter, and H. Elmqvist, "Modelica hybrid modeling and efficient simulation", Proc. 38th IEEE Conference on Decision and Control, Phoenix, AZ, USA, 1999, pp. 3502–3507.

[11] M. Otter, H. Elmqvist, and S.E. Mattsson, "Hybrid modeling in Modelica based on the synchronous data flow principle", Proc. 10th IEEE International Symposium on Computer Aided Control System Design, Kohala Coast, HI, USA, 1999, pp. 151–157.

[12] F.E. Cellier, and H. Elmqvist, "Automated formula manipulation supports object-oriented continuous-system modeling", IEEE Control Systems 13 (2), pp. 28–38, 1993.

[13] P. Fritzson, et al., "The Open Source Modelica Project". Proc. 2nd Intl. Modelica Conference, Oberpfaffenhofen, Germany, 2002.

[14] F.E. Cellier, and E. Kofman, Continuous System Simulation. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[15] K.E. Brenan, S.L. Campbell, and L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. SIAM, Philadelphia, PA, USA, 1996.

[16] S.E. Mattsson, and G. Söderlind, "A New Technique for Solving High-Index Differential Equations Using Dummy Derivatives". Proc. of the IEEE Symposium on Computer-Aided Control System Design, California, USA, 1992.

[17] C.C. Pantelides, "The Consistent Initialization of Differential-algebraic Systems". SIAM J. Sci. Stat. Comput. 9 (2), 1988, pp. 213–231.

[18] H. Elmqvist, and M. Otter, "Methods for Tearing Systems of Equations in Object-Oriented Modeling", Proc. ESM'94, European Simulation Multiconference, Barcelona, Spain, 1994.

[19] C. Martin-Villalba, A. Urquia, and S. Dormido, "An approach to virtual-lab implementation using Modelica", Mathematical and Computer Modelling of Dynamical Systems, 14(4), 2008, pp. 341–360.

[20] A. Urquia, C. Martin-Villalba, and S. Dormido, "Interactive Simulation in Modelica: a Proposal", Proc. 24th annual European Simulation and Modelling Conference, Hasselt, Belgium, 2010.

[21] C. Martin-Villalba, A. Urquia, and S. Dormido, "Development of an industrial boiler virtual-lab for control education using Modelica", Computer Applications in Engineering Education, In press, DOI 10.1002/cae20449.

[22] C. Martin-Villalba, "Object-Oriented Modeling of Virtual Laboratories for Control Education", PhD Dissertation, Dept. Informática y Automática, UNED, Madrid, Spain, 2007.

[23] W. F. Ramirez, Computational Methods for Process Simulation, Butterworths Publishers, Boston, 1989, pp 209–217.

[24] A. Urquia; and S. Dormido, "Object-oriented design of reusable model libraries of hybrid dynamic systems. Part two: a case study", Mathematical and Computer Modelling of Dynamical Systems, 9(1), 2003, pp. 91–118.