УНИВЕРСИТЕТСКИЙ
КОНСОРЦИУМ

Университетский научный журнал

Информационные технологии

№ 8 (2014)

Hybrid Systems. Preliminary Comparative Analysis of Modelica and Model Vision Language

Yu. B. Kolesov
The Federal center of information of Accounting chamber of Russian Federation
Yu. B. Senichenkov
St. Petersburg State Polytechnical University
A. Urquia, C. Martin-Villalba
Universidad Nacional de Educación a Distancia

Гибридные системы. Сравнительный анализ языков моделирования Modelica и Model Vision Language

Ю. Б. Колесов
Федеральный центр информатизации Счетной палаты Российской Федерации
Ю. Б. Сениченков
Санкт-Петербургский государственный политехнический университет
А. Уркия, К. Мартин-Виллалба
вуз русск

Keywords: *object oriented modeling; hybrid systems; algebraic-differential equations; computer simulation; numerical models; numerical simulation; engineering education*

Object-Oriented Modeling (OOM) and hybrid systems are basic concepts of modern tools to model and simulate complex dynamical systems. However, object-oriented modeling technologies and state machines may be implemented in different ways in various tools. Implementations of OOM and hybrid systems in tools that support Modelica and Model Vision Language are discussed. The paper may be interesting for researches and instructors who use Dymola, OpenModelica, and Rand Model Designer in research and education.

Ключевые слова: *объектно-ориентированное моделирование; компьютерное моделирование; численное моделирование; моделирование в инженерном образовании.*

Объектно-Ориентированное Моделирование (ООМ) и гибридные системы являются основными понятиями, используемыми для моделирования сложных динамических систем. Однако объектно-ориентированные технологии и машины состояний могут быть по-разному использованы в различных средах визуального моделирования. В статье проводится предварительный сравнительный анализ языков Modelica и Model Vision Language. Статья может быть интересна пользователям пакетов Dymola, OpenModelica, and Rand Model Designer.

I. Introduction

Object-Oriented Modeling (OOM) of complex dynamical systems is facilitated by such modeling languages as Modelica [5] and Model Vision Language (MVL) [1, 2, 3]. Modelica and MVL are used in several tools [1, 4, 7, 8]. Both modeling languages have many features in common, but there are also essential differences between them in terms of hybrid system modeling.

Prof. Alfonso Urquia visited St. Petersburg Polytechnical University and familiarized himself with Rand Model Designer (RMD), a MVL-based tool for modeling and simulating complex dynamical systems. During his work at Prof. Yu. B. Senichenkov's lab, he made a preliminary comparative analysis of Modelica and MVL, taking into consideration the capabilities these languages bring for hybrid model description and their implementations in tools based on Modelica and MVL. He gave his own interpretation of differences and illustrated them by examples that are well-known to Modelica users.

The differences under consideration concern inheritance, hybrid systems, and ways of their implementations in Dymola and RMD:

1. Inheritance	
Modelica	multiple inheritance
MVL	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 through new behavior and structure (i.e., equations, components, connections, etc.). In addition, MVL allows removing part of the inherited behavior in the derived class. MVL and Modelica allow declaring time-independent variables as a class of formal parameters so that they can receive their actual values in each class instance. Additionally, Modelica supports re-declaring classes of the objects instantiated in the model.	

2. Hybrid Systems	
Modelica	An «if» statement is used to specify hybrid systems. Current equations for composition of hybrid automata are built during compilation.
MVL	«State Machine» is used to specify hybrid systems. Current equations for composition of hybrid automata are built at run time.
Description of the hybrid behavior is a major difference between MVL and Modelica (see Section II.) Modelica [6] environments (e.g., Dymola [7] and OpenModelica [8]) build and reduce all systems of equations needed for hybrid automata composition during the compilation stage. RMD_Transas, RMD, and OpenMVLShell (www.rand-service.com, www.mvstudium.com, http://dcn.ics.spbstu.ru) build and reduce current systems at run time.	

The differences under discussion are illustrated by examples (Sections IV and V).

II. Hybrid Systems in Modelica and MVL

Hybrid systems («event-driven» systems) exhibit mixed discrete and continuous behavior. Specification of a hybrid system includes specification of modes' behavior, conditions for transitions between modes, and entry and exit actions for each mode. Hybrid model specification is different in Modelica and MVL.

Modelica
1.1. Equations in Modelica follow the «synchronous data flow» principle [11]. The set of active equations can be composed of continuous equations only (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]. 1.2. Modelica offers special constructions to describe hybrid systems [9–11]. Users can: (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. 1.3. Modelica allows describing special (limited) types of hybrid systems. A Modelica model must comply with the single-assignment rule. This means that the number of unknown variables and equations for hybrid automation states 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.

MVL
2.1. MVL uses Behavior Charts (B-Charts for short) to specify hybrid systems (see Fig. 1). B-Chart is a special version of State Machine similar to StateFlow (see Simulink).
2.2. Construction of the current system of equations for composed models depends on the type of links (bonds) between components. If «input-output» connections are used, everything that is needed to compose current systems may be built during compilation. If «contact-flow» connections are used, building at run time is preferred.

The allowed specification of hybrid systems does not affect translating isolated hybrid systems and component models with «input-output» links between components into the simulation code. The number of possible modes for composed hybrid automata may be huge in any case, but for «physical» component models mode switching leads to the necessity of re-assigning computational causality, which increases the computational effort. Decreasing computational effort for huge automata is possible by limiting the supported

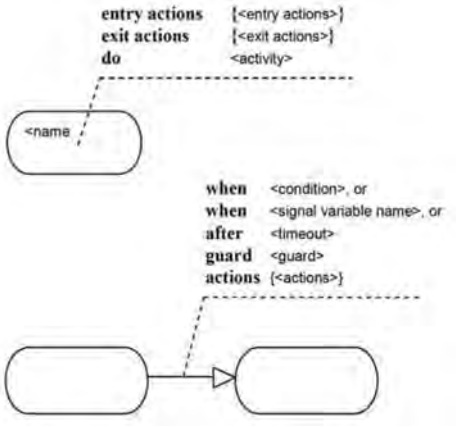

Fig. 1. B-chart mode (above) and transition between modes (below)

hybrid behavior (Modelica) or handling events at run time (Rand Model Designer).

Let us consider a library of electrical components that contains models of a voltage generator, resistor, capacitor, diode, etc. Let us suppose that the diode is described with a B-chart with two modes [11], named off and on, as shown in Fig. 2.

Interfaces of the library components are described with 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.

Let us consider the rectifier circuit shown in Fig. 3.

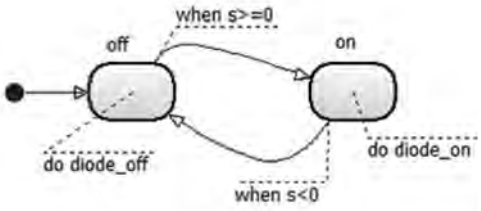

Fig. 2. B-chart describing the behavior of a diode

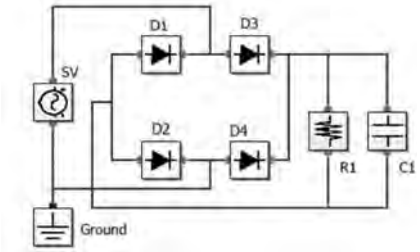

Fig. 3. Rectifier circuit model composed using RMD

The rectifier circuit model contains four diodes, so there are $2^4 = 16$ possible combinations of modes. For every «switching» of any of the four diodes, RMD automatically constructs the current system or equations that represents the new structure of the complete model.

Modelica
Two state-of-the-art Modelica environments are taken as a reference in this discussion: Dymola [7] and OpenModelica [8]. Symbolic manipulations that these tools carry out on may be divided into two types according to their purpose [13]. The first type is intended for translating object-oriented description of the model into the so-called «flat model». The flat model contains a 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]: Efficient formulation of the complete-model equations, eliminating the redundant variables and the trivial equations resulting from the component connections. Reduction of the system index to zero or one [15–17]. Analysis of computational causality (partition) and equation sorting. Symbolic manipulation of linear systems of simultaneous equations. Tearing [18] of nonlinear systems of simultaneous equations.

All these manipulations, which are intended to generate an 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 using complex algorithms for model structure analysis, model symbolic manipulation and generation of a highly efficient numerical code.

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

Two strong points of RMD are the following. One is the expressiveness of its language in the specification of hybrid systems. The other one is that the system of equations that describes the complete model at a certain time is automatically built at simulation time by RMD from the active modes (one active mode per B-chart) at that time.

RMD approach requires using only very fast algorithms to analyze and build equations. However, symbolic differentiation and index reduction are not currently supported by RMD.

III. Example 1. Interactive Model. Selection of State Variables

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

RMD generates two types of an executable code [3]:

- Stand-alone Windows application. It is used to run experiments on the model under RMD in the interactive mode. RMD provides some ready-to-use 2D and 3D animation components and tools to carry out typical computational experiments.
- Application in the form of Window's dynamic link library (dll). It is used as an embedded interactive application (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 called a physical model and its reformulation for interactive simulation is called an 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 to the selection of interactive quantities [20]:

• A time-dependent variable can be an interactive quantity if and only if there is at least one selection of state variables that includes this variable.

• A set of variables can represent interactive quantities modified in the same interactive action if and only if there is at least a selection of the state variables that includes all the variables in the set.

In general, different choices of the state variables are possible in a physical model. Various interactive events may require different selections of 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 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.

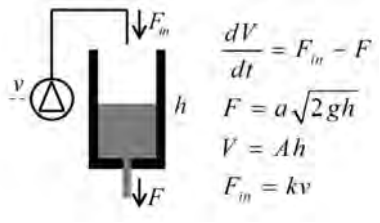

Fig. 4. Model used to illustrate run-time selection of state variables

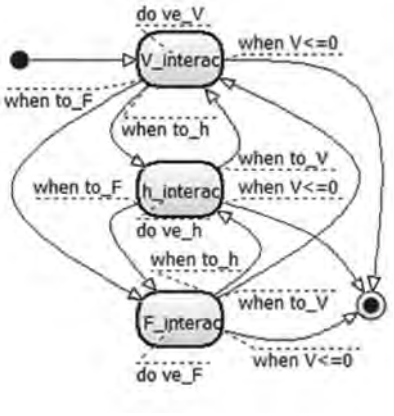

Fig. 5. B-chart of the tank system shown in Fig. 4

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 a state variable. The state should be selected 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 it is F.

In addition, changes in the value of interactive quantities can have different effects depending on the state variable choice. Let us 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 vol-

ume remains constant. On the contrary, if the state variable is the height (*h*) or the flow (*F*), these quantity values will not change as a result of an instantaneous change in the cross section (*A*), but the volume will.

Let us consider a virtual lab intended to illustrate the tank system behavior. Let us suppose that the virtual lab is required to support three ways to describe 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, level or output flow. Different choices are possible within a given simulation run.

An 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 with the V_interact mode, contains the tank system model with the liquid volume as a state variable. Analogously, the ve_h and ve_F activity classes, which are associated with 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., the one with the required state selection) is used to execute 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.

IV. Example 2. Models with Variable Behavior

Industrial boilers are widely used in chemical industry and education to illustrate control laws. A virtual laboratory for an industrial boiler with 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. Water in the boiler is heated permanently. Water level inside the boiler and output flow of vapor are controlled variables. Pump throughput and heater power are manipulated variables.

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.

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

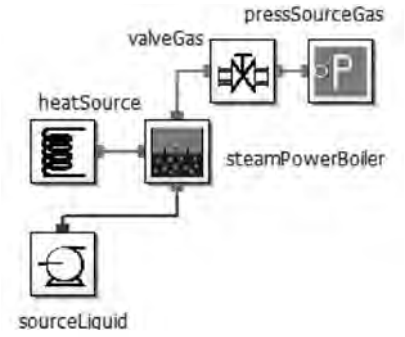

Fig. 6. Diagram of the boiler model

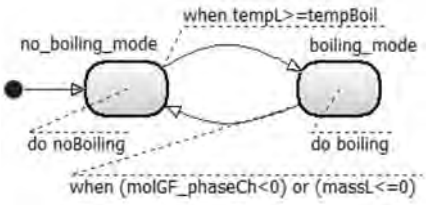

Fig. 7. B-chart for the boling process

The boiling process is described in [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 to the boiling temperature at the corresponding vapor pressure and the heat supplied to the liquid is employed in producing steam.

B-Chart for the boiling process is shown in Fig. 7. The degree-of-freedom number depends on the mode. While being 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 being 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, which determines the gas volume.

V. Conclusions

Two strong points of MVL used in RMD are high flexibility in the description of variable structure models and support for interactive simulations. However, the adopted approach has limitations, because building equations for «physical» component models with general hybrid behavior for large scale systems is only possible at run time. Future research will provide development of highly efficient algorithms for symbolic differentiation and index reduction.

Acknowledgment

This work has been partly supported by Banco Santander and UNED, under the grant called “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.

Колесов Юрий Борисович
Доктор технических наук.
Научная отрасль:
Федеральный центр информатизации Счетной палаты Российской Федерации,
ведущий научный сотрудник.
E-mail: ybk2@mail.ru
Сениченков Юрий Борисович
Доктор технических наук.
Научная отрасль:
Санкт-Петербургский государственный политехнический университет,
профессор кафедры распределенных вычислений и компьютерных сетей,
Институт информационных технологий и управления
E-mail: senyb@dcn.icc.spbstu.ru
Уркия Альфонсо
Научная степень
Научная отрасль
Название вуза,
доцент кафедры информатики и управления.
E-mail: aurquia@dia.uned.es
Мартин-Виллалба Карла
Научная степень
Научная отрасль
Название вуза,
доцент кафедры ??
E-mail: carla@dia.uned.es