SOFTWARE NOTES

Modeling Hybrid Systems in MvStudium

D. B. Inichov, Y. B. Kolesov, Yu. B. Senichenkov, Saint Petersburg State Polytechnic Univ., Russia

SNE Simulation Notes Europe SNE 20(1), 2010, 31-34, doi: 10.11128/sne.20.sw.09966

A new version of graphical environment MvStudium 6.0 for modeling and simulation of complex dynamical systems enables to design hierarchical event-driven systems using blocks with oriented (Inputs and Outputs) and non-oriented (Contacts and Flows) connections (oriented and non-oriented blocks), whose internal activity may be described by Behavior Charts (B-Chart is a state machine with continuous Do-activities and without orthogonal states) MvStudium 6.0– is a graphical environment with universal equation-based and UML–based object-oriented modeling language, which graphical form based on B-Charts and hierarchical functional diagrams. The environment supports technology of designing hierarchical models using oriented and nonoriented blocks, that behavior is may be described by hierarchical B-Charts. The environment consists of Model Editor and Virtual Test-bench.

Model Editor has four user's interfaces that sequentially become more complex for different types of models: (1) an isolated classical dynamical system, (2) an isolated hybrid system, (3) a hierarchical model with components from multiple domains, (4) a model with predefined plan of computer experiment. User can build model using his own original blocks or imported blocks from other projects or libraries. A solved system of differential-algebraic equations for each current model mode is formed and analyzed on compilation stage for models with oriented blocks, and it formed on runtime for models with non-oriented blocks. Designed by user model is checked and compiled. A model may run under Virtual Test-bench, may be a standalone executable program, or may be realized as hidden (un-visual model) in the form of DLL for using as a component of more complex model or for parameter optimization with the help of Virtual Test-bench toolbox.

Virtual Test-bench is used for debugging, simulation and computer experiments with visual model.

Introduction

Nowadays tools for modeling and simulation of complex dynamical systems are graphical environments equipped by modeling, planning and doing computer experiment languages used in graphical and textual forms, toolboxes for analysis and synthesis of models and engineering libraries. For short we will call them visual modeling tools (VMT). VMT support different modelling technology that make it possible designing models with components from multiple domains, control and controlled blocks, working in model and real time. It's more than enough to call them universal tools taking in account additionally that they are used for research, designing and leaning. Developers of VMT consider universality also as ability to design models of different types of complexity. A dynamical system is said to be complex if it has:

- *Structural complexity*: a system has hierarchical block structure with blocks from multiple domains.
- *Behavior complexity*: a system in whole and each block may have different modes, in other words they have event-driven behavior.
- *Changeable structure*: structure of a system and number of its blocks depends on time.
- *Large dimension*: number of variables and blocks is so large that it is impossible to realize model without special technologies for large scale systems.

In this article we will speak only about behavior complexity of models and our goal is to demonstrate MvStudium's [3, 4, 5] capability to model hybrid or event-driven systems.

1 'Block-modeling' in MvStudium

Graphical language for MvStudium should to enable users easy designing of:

- hybrid systems (mathematical model – differential or algebraic-differential equations with discontinuous right-hand sides),
- multi-component systems with oriented blocks, that are open hybrid system with Inputs and Outputs,
- multi-component systems with non-oriented blocks, that are open hybrid system with Contacts and Flows,
- multi-component systems with both types blocks.

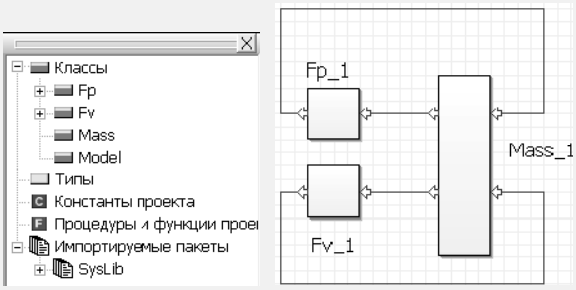

Figure 1a,b. In the project manager (left), there are four classes: Model, Mass, Fp, and Fv. Classes Fp, Fv have two local classes pL, pQube, vL, vQuad thereafter. The class model (right) has type 'Functional diagram'. External variables are of 'Input' and 'Output' types.

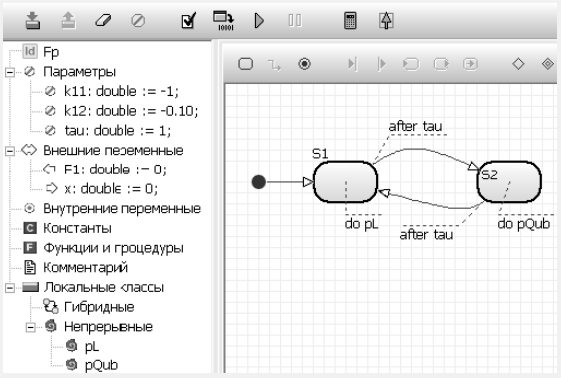

Figure 1c. Fp class anf its B-Chart.

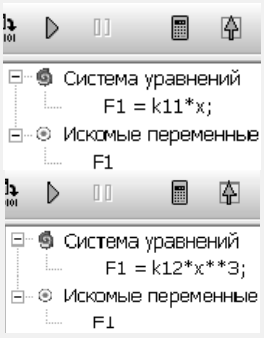

Figure 1d. Equations of local classes pL, pQube.

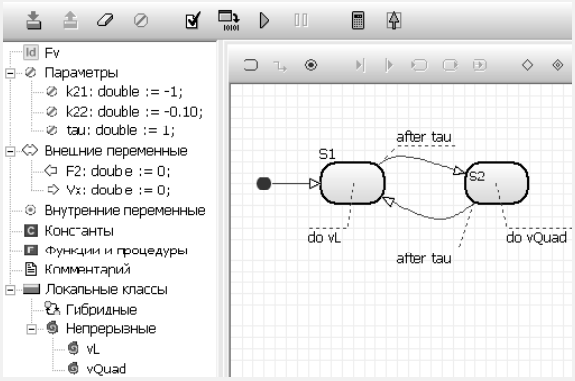

Figure 1e. Fv class and B-Chart.

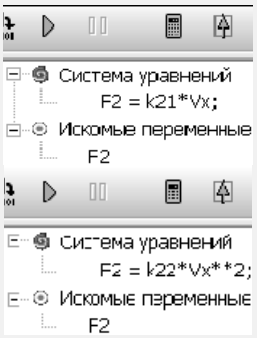

Figure 1f. Equations of local classes vL, vQuad

Hybrid system in MvStudium is tuple HS={**G, Pr, Inv, F**} where

- **G** is state graph (state machine) with usual states and pseudo states ('initial', 'final', 'decision'). Usual state may be simple or composite one. Simple state is a state with behavior defined by system of algebraic-differential equations. Composite state or submachine state is defined by its own state graph. A state may have entry and exit actions. Actions are written in Model Vision Language (MVL) and considered as momentary actions. Orthogonal states are forbidden.
- **Pr** is a set of predicates $\Pr B\colon \mathbb{R}^n \to \{\text{True, False}\} \wedge \Pr R\colon \mathbb{R}^n \to \mathbb{R}^1$, governing conditions of leaving current state and choosing a new current state.
- **Inv** is a set of invariants for states of a graph.
- **F** is a set of algebraic-differential equations governing behavior in usual state. A set **F** includes the element NULL.

Graphical notation for hybrid systems we should call *Behavior Charts*. Thereby, a B-Chart is an UML state machine without orthogonal and historical states and with Do-Activates in form of algebraic-differential equations.

First versions of MvStudium allow designing only hybrid systems and block with oriented blocks (HS_IO). The new one deals with non-oriented blocks (HS_CF).

MVL is object-oriented language. For designing any block system you need to define classes with external variables (input, output, contact, flow) and internal variables – state variables. Variables may be scalar or vectormatrix types. Each class may have a structure and behavior. Behavior is defined by state machine. Singular state machine with one usual state is said to be continuous one and may be present only by system of equation. Systems of algebraic-differential equations from **F** are presented as local classes. You may create your own classes or import them from another projects or libraries. Mechanism of inheritance allows forming trees of classes. Modeling system (class 'Model') may be oriented or non-oriented block or closed block (container block without external vari-

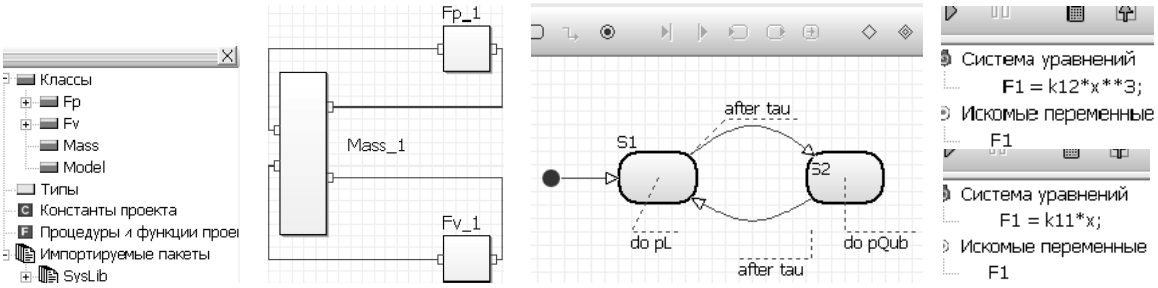

Figure 2. MvStudium project with non-oriented blocks for mechanical system. (a) Project manager with four classes Model, Mass, Fp, and Fv. Classes Fp, Fv have two local classes pL, pQube, vL, vQuad thereafter. (b) Class Model of type 'Functional diagram'. It is used only one type of external variables that is 'contact'. (c) B-Chart of Class Fp. (d) Equations of local classes.

ables). Class 'Model' may have its own B-Chart used for planning computer experiment.

There is no necessity to generate full code for all variants of hybrid behavior for models with oriented blocks. It is enough to have code for all local classes.

Example of a mechanical system. Let us consider mass m, subjects to periodical forces F_1 and F_2 with $2T$ period:

$$m \cdot \frac{d^2x}{dt^2} = F_1 + F_2; \quad x(0) = x_0; \quad x'(0) = x'_0$$

$$F_1 = \begin{cases} k_{11} \cdot x, & [0,\ T] \\ k_{12} \cdot x^3, & [T,2T] \end{cases}; \quad F_2 = \begin{cases} k_{21} \cdot \dfrac{dx}{dt}, & [0,\ T] \\ k_{22} \cdot \left(\dfrac{dx}{dt}\right)^2, & [T,2T] \end{cases}$$

A model with oriented blocks for this case may look like in Figure 1.

2 'Physical Modeling' in MvStudium

MvStudium 6.0 suppots 'physical modeling'. It is possible designing and using non-oriented blocks with Modelica's external variables 'contacts' and 'flows'.

Figure 2 shows fragment of MvStudium project for the examined example of mechanical system realized with the help of non-oriented blocks.

The examined example may be not interesting for users dealing with 'classical' devices of physical modeling such as 'resistor', 'capacity', 'inductance' and other analogous components. Let us consider electrical circuit with an 'ideal' diode that is very simple hybrid system.

In Figure 3 you see Modelica's 'oneport' device that is parent for classes 'resistor', 'capacity', 'ground' and so on. Class 'IdealDiode' is open hybrid system

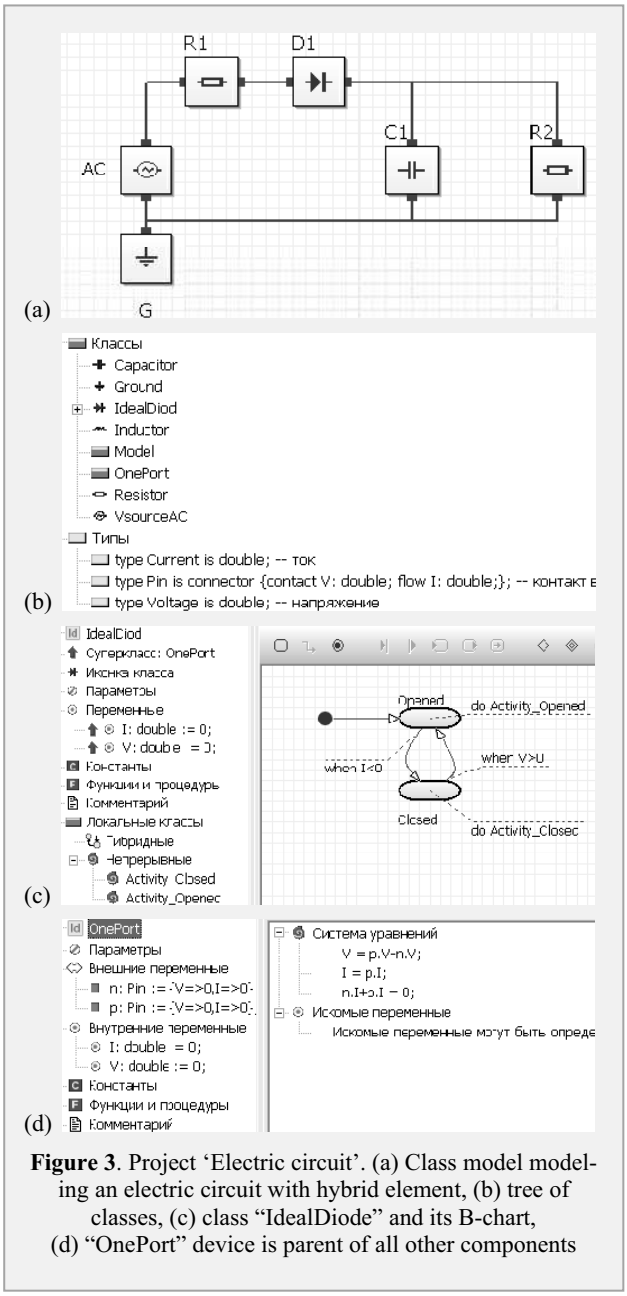

Figure 3. Project 'Electric circuit'. (a) Class model modeling an electric circuit with hybrid element, (b) tree of classes, (c) class "IdealDiode" and its B-chart, (d) "OnePort" device is parent of all other components

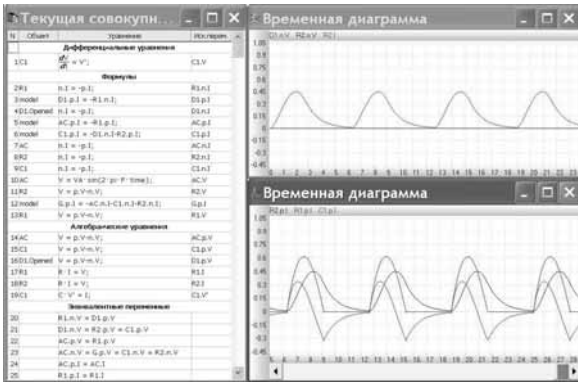

Figure 4. Test-bench windows for project "electric circuit". On the right is the window with solved system of equations divided on formulas, differential and algebraic equations, and equivalent variables.

with external variables 'contact' and 'flow'. Figure 4 demonstrates time diagrams for some components of electric circuit and the special window with final (solved) system of eqution. The solved system depends on time. When current state in B-Chart of "IdealDiode" is changed the solved system is changed too. This window is used while debugging. With the help of debugger it is possible to hold execution in the point of changing current state and check solved system.

3 Computer experiments with models

B-Charts may be used for planning computer experiments. A class Model may have type "functional diagram with its own B-Chart". Imagine that you want to compare two models of mechanical system realized with oriented (InOutModel) and non-oriented (CFModel) blocks (Figure 5). Let us create the class 'Plan' with composite B-Chart for planning experiment. Composite B-Chart has Do-Activity 'Tick-Tack'. 'Tick-Tack' is local class that has B-Chart with two states S1 and S2. Do-Activities of states S1 and S2 are direct instances of classes 'InOutModel' and 'CFModel'.

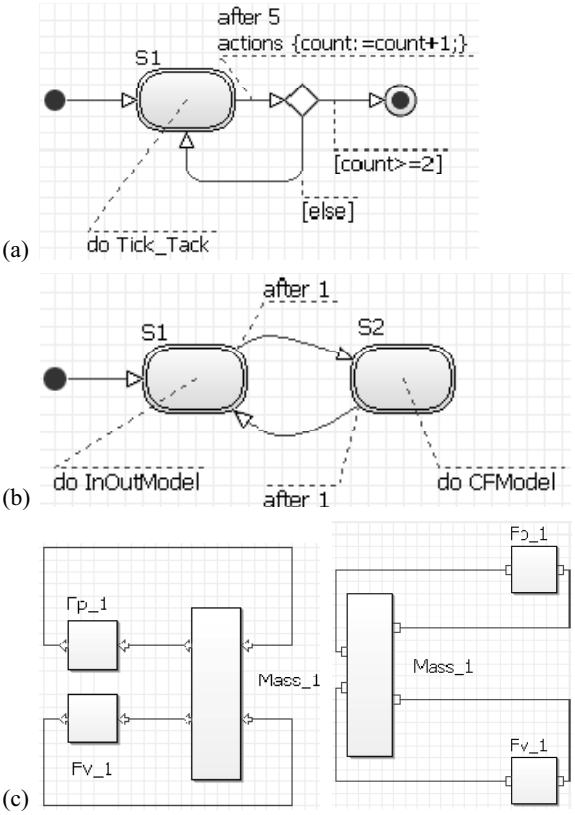

Figure 5. Using B-charts for planning computer experiments. (a) B-chart of highest level which contol number of experiments, (b) B-Chart "Tick-Tack", which periodically switches models under investigation, (c) Do-Activities in form of direct instances of a classes "InOutModel" and "CFModel".

References

[1] Yu.B. Kolesov. *Object-Oriented Modeling of Complex Dynamical Systems*. St. Petersburg: Publishing House of the Polytechnic University, 2004. 239 pp.

[2] Yu.B. Senichenkov. *Numerical Modeling of Hybrid Systems*. St. Petersburg: Publishing House of the Polytechnic University, 2004. 206 pp.

[3] Yu.B. Kolesov, Yu.B. Senichenkov. *Modeling of Systems. Dynamical and Hybrid Systems*. St. Petersburg, BHV, 2006. 224 pp.

[4] Yu.B. Kolesov, Yu.B. Senichenkov. *Modeling of Systems. Object-Oriented Approach*. St. Petersburg, BHV, 2006, 192 pp.

[5] Yu.B. Kolesov, Yu.B. Senichenkov. *Modeling of Systems. Practical Work on Computer Modeling*. St. Petersburg, BHV, 2007, 352 pp.

Corresponding author: Yu. B. Senichenkov,
Polytechnical University Saint Petersburg
Polytechnicheskaj 29, Russia, 195251,
senyb@dcn.infos.ru

Accepted: MATHMOD 2009
Revised: March 20, 2009
Accepted: July 25, 2009