*DIFFERENTIAL EQUATIONS*
*AND*
*CONTROL PROCESSES*
*№ 2, 2011*
*Electronic Journal,*
*reg.Эл № ФС77-39410 at 15.04.2010*
*ISSN 1817-2172*

*http://www.math.spbu.ru/user/diffjournal*
*e-mail: jodiff@mail.ru*

# Physical Modeling in MvStudium

Senichenkov Y.B.

Saint-Petersburg State Polytechnical University,
Polytechnical str. 21, Saint-Petersburg, Russia,
E-mail: senyb@dcn.infos.ru

and

Kolesov Y.B.

Saint-Petersburg State Polytechnical University,
Polytechnical str. 21, Saint-Petersburg, Russia,
E-mail: ybk@mail.ru

## Abstract

In this paper, we describe “physical” modeling technology in MvStudium that is a graphical environment for modeling and simulation of complex dynamic systems. MvStudium.6 modeling language (MVL) is based on open hybrid automata or Behavior-Charts (for short B-Charts), functional diagrams, and supports object-oriented modeling. B-Chart is an extension of UML state machine with do-activities in the form of differential-algebraic equations. External variables of MVL classes may be in the form of “Inputs-Outputs” (I/O) or “Contacts-Flows” (C/F). C/F components in MvStudium are analogous to Modelica components but in MVL there is no limitation on type (NAE, ODE, DAE), dimension, and form (explicit, semi-explicit, implicit) for a current solved system of equations prescribed to a current state of a B-Chart. MvStudium compiler automatically builds an executable model. It may run under the environment or it may be built in the form of DLL and used even in real-time applications.

***Keywords:*** design tools and techniques, Simulation and modeling, simulation languages, Modeling methodologies, object-oriented design methods, model development, UML-based approach, multi-component models, physical modeling.

## 1. General

Graphical environment MvStudium.6 [1,2,3] for modeling (Model Editor) and simulation (Virtual Test Bench) of complex dynamical systems is based on formalism of open hybrid automata or Behavior-Charts (for short B-Charts) (Figure 1,2).

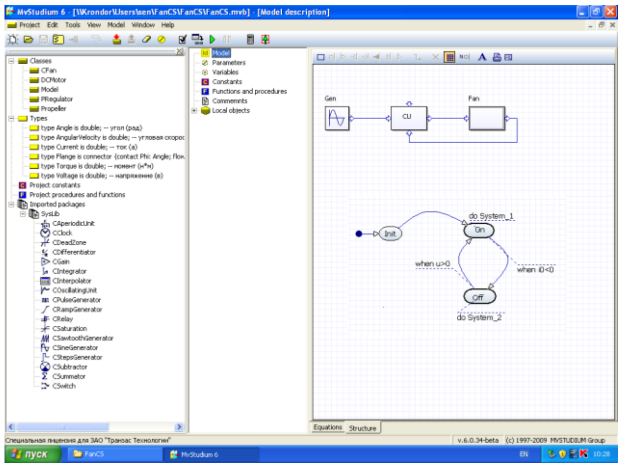


**Figure 1.** Studium model editor: a functional diagram and B-Chart.

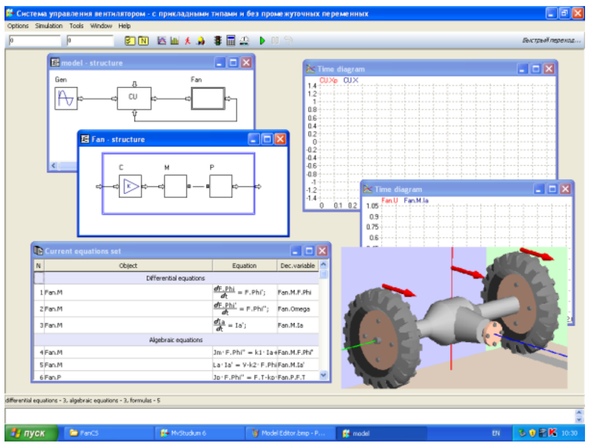


**Figure 2.** MvStudium virtual test bench (ModelViewer).

B-Chart is an extension of UML state machine with do-activities in the form of differential-algebraic equations without History and Orthogonal states. It is used in MVL for describing hybrid behavior and planning computer experiments.

MvStudium.6 supports object-oriented modeling: MVL classes and packages make it possible to reproduce hierarchical structure and complex behavior of real world system (physical modeling). External variables of MVL classes (model components) may be in the form of “Inputs-Outputs”(Figure 3) or “Contacts-Flows” (Figure 4).

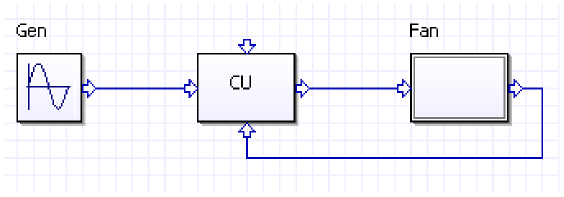


**Figure 3.** I/O components.

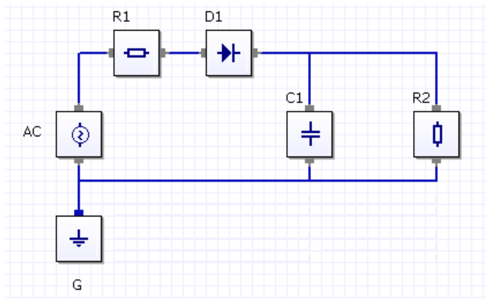


**Figure 4.** C/F blocks.

Physical modeling or using objects with “Contacts-Flows” and B-Charts requires rebuilding of a solved system of equations after any change event resulting in changing of a local behavior of a B-Chart. Rebuilding assumes structural analysis and reducing of a new system on run-time. Building a new final system often leads to underdetermined differential equations especially on early stage of designing a new model. It is not quite clear, should we consider such systems as user’s error or we need to help him and automatically build a determined differential equations (if possible).

It is very important for “physical” modeling to decrease as much as possible dimension of final system of equations. For example we can delete from it contact and flow equations, solve some equations symbolically, or divide algebraic equation on non-linear and liner parts and consider linear equations as substitutions in Newton method resolved by Gauss elimination. For this purpose a special module for symbolic calculations was developed.

## 2. Object-oriented modeling in MvStudium

MvStudium’s classes may be divided into four groups (stereotypes) that are: a) isolated classical dynamical systems, b) isolated hybrid systems, c) multi-component models with “Input/Output“ and/or “Contact/Flow” components, d) hierarchical multi-component models equipped out its own B-Chart for planning virtual experiment. A class [2] may have attributes: external variables (links), structure and behavior, and it may inherit attributes of its parent. A special class MODEL has only one instance and it is compiled in executable program.

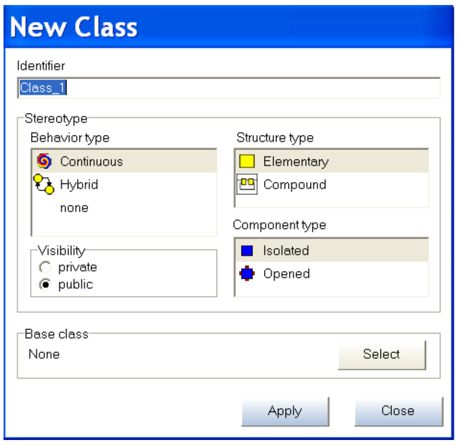
MVL uses B-Chart for describing complex event-driven behavior. Graphical notation of B-Chart uses elements of UML state machines and activity diagrams. It is forbidden to use History and Orthogonal states. However the main difference between UML state machine and B-chart is in interpretation of do-activities. In MvStudium do-activity is a solution of a system of algebraic-differential equations. Do-activity is a local class with stereotypes “continuous” or “hybrid”.

## 3. Model stereotypes

There are four model stereotypes.  A MvStudium’s model stereotype that forms a tree (Figure 5) with the parent  is called “continuous”.

Continuous” [3] stands for an isolated system written in the form of a system of algebraic-differential equations:

$$
\begin{aligned}
&F(t, x, \frac{dx}{dt}, \frac{d^2x}{dt^2}) = 0 \\
&x(t), F \in \Re^n, \qquad\qquad (1) \\
&x(0) = x_0
\end{aligned}
$$



Identifier: Class_1

Visibility: ( ) private [✓ public]

Base class: None

**Figure 5.** “New class” dialog.

Equations may be written in scalar or vector forms. A special Equation Editor allows writing and editing equations in a usual mathematical form.

The second stereotype is called “hybrid”. This stereotype uses UML state machine notation for describing hybrid systems or systems of algebraic-differential equations with discontinuous right sides. A “continuous” class may be automatically transformed into a “hybrid” class. In this case “hybrid” class will have a B-Chart with one node and one local class. For editing a hybrid class, a user needs additionally a B-Chart Editor.

The third stereotype is used to design multi-component models using a common for an engineer “functional diagrams” graphical image. Classes used in a functional diagram may have external variables of the following kinds: “Input/Output” (I/O) and/or “Contact/Flow” (C/F).

Using blocks with I/O and B-Charts does not create any difficulties for building a final system of equations for a current state of a MODEL. A code for a final system for a current Model state may be built automatically from codes of local class systems compiled beforehand. “Correct” local equations lead to “correct” final system.

In the case of classes with C/F and B-Charts we deal with underdetermined local systems of equations and we can speak only about “correctness” of a final system for a current Model state. There is no need of checking “correctness” of all possible final systems beforehand. It is quite reasonable to analyze only “correctness” of realized on run time final systems (particular trajectory of hybrid automaton). For this stereotype it is necessary to use additionally a Functional Diagram Editor.

The forth stereotype called “Functional diagram with control B-Chart” is used for complex computer experiments. A functional diagram is equipped by control B-Chart. While using a B-Chart for planning experiment it is possible to prescribe to a state behavior of a model component.

Model stereotypes make it possible to support “natural” technology of designing “from simple to complex”. As the first step you realize isolated model in the form of differential equations disregarding structure and different modes (stereotype “Continuous”). As the second step you design event-driven model (stereotype “Hybrid”). Then you take into account structure (stereotype “Functional diagram”) and start planning experiments with the whole model (“Functional diagram with control B-Chart”). This simple technology works well for new scientific problems, but for industrial design it is preferable to use the UML design technology.

## 4. Physical modeling in MvStudium

Using C/F components in MvStudium does not differ from using similar components in Modelica (Figure 6, 7, 8, 9).

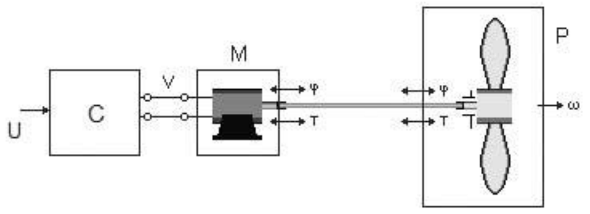
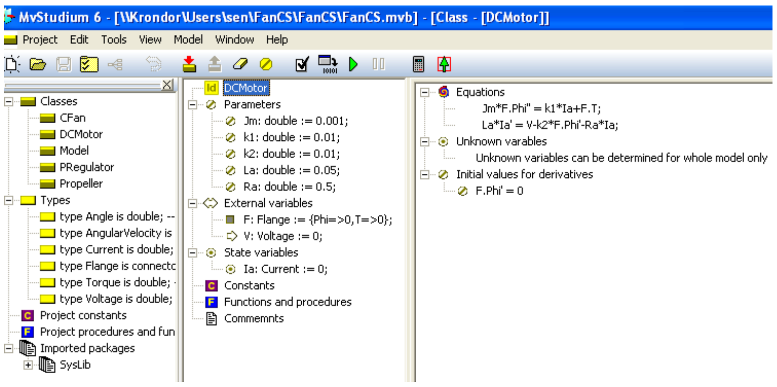


**Figure 6.** Control system for “Motor and Propeller” is well known Modelica’s example of “physical” modeling.

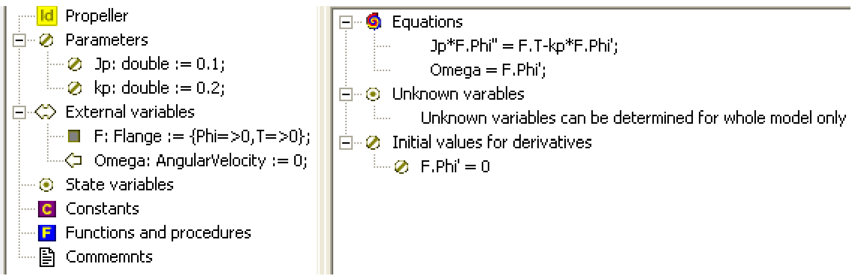

MvStudium 6 - [\\Krondor\Users\sen\FanCS\FanCS\FanCS.mvb] - [Class - [DCMotor]]
Project Edit Tools View Model Window Help

```
Classes
    CFan
    DCMotor
    Model
    PRegulator
    Propeller
Types
    type Angle is double; --
    type AngularVelocity is
    type Current is double;
    type Flange is connectc
    type Torque is double; ·
    type Voltage is double;
Project constants
Project procedures and fun
Imported packages
    SysLib
```

```
DCMotor
Parameters
    Jm: double := 0.001;
    k1: double := 0.01;
    k2: double := 0.01;
    La: double := 0.05;
    Ra: double := 0.5;
External variables
    F: Flange := {Phi=>0,T=>0};
    V: Voltage := 0;
State variables
    Ia: Current := 0;
Constants
Functions and procedures
Commemnts
```

```
Equations
        Jm*F.Phi'' = k1*Ia+F.T;
        La*Ia' = V-k2*F.Phi'-Ra*Ia;
Unknown varables
      Unknown variables can be determined for whole model only
Initial values for derivatives
      F.Phi' = 0
```

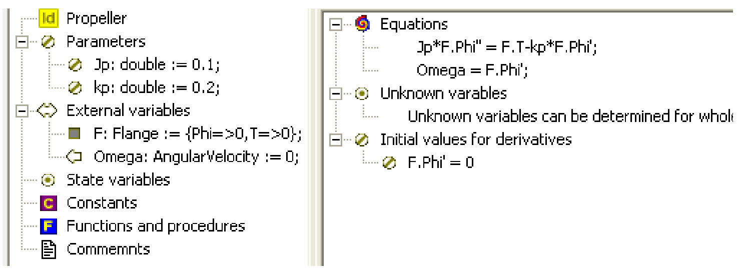
**Figure 7.** Controller.



```
Propeller
Parameters
    Jp: double := 0.1;
    kp: double := 0.2;
External variables
    F: Flange := {Phi=>0,T=>0};
    Omega: AngularVelocity := 0;
State variables
Constants
Functions and procedures
Commemnts
```

```
Equations
        Jp*F.Phi'' = F.T-kp*F.Phi';
        Omega = F.Phi';
Unknown varables
      Unknown variables can be determined for whole model only
Initial values for derivatives
      F.Phi' = 0
```

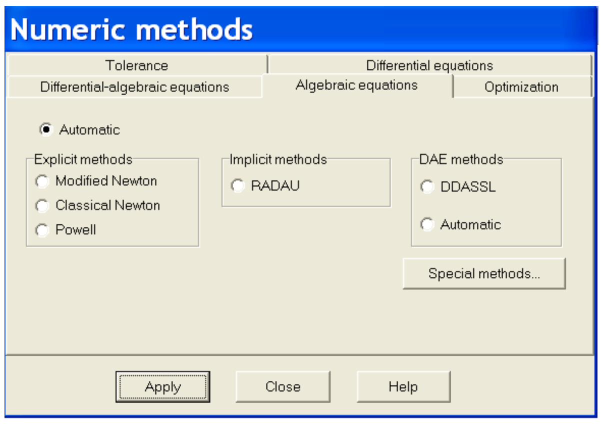
**Figure 8.** Propeller.



```
Propeller
⊟ Parameters
    Jp: double := 0.1;
    kp: double := 0.2;
⊟ External variables
    F: Flange := {Phi=>0,T=>0};
    Omega: AngularVelocity := 0;
  State variables
  Constants
  Functions and procedures
  Commemnts
```

```
⊟ Equations
      Jp*F.Phi'' = F.T-kp*F.Phi';
      Omega = F.Phi';
⊟ Unknown varables
    Unknown variables can be determined for whol
⊟ Initial values for derivatives
    F.Phi' = 0
```

**Figure 9.** Motor.



Numeric methods

Tolerance | Differential equations | Differential-algebraic equations | Algebraic equations | Optimization

[✓ Automatic]

Explicit methods: Modified Newton, Classical Newton, Powell

Implicit methods: RADAU

DAE methods: DDASSL, Automatic

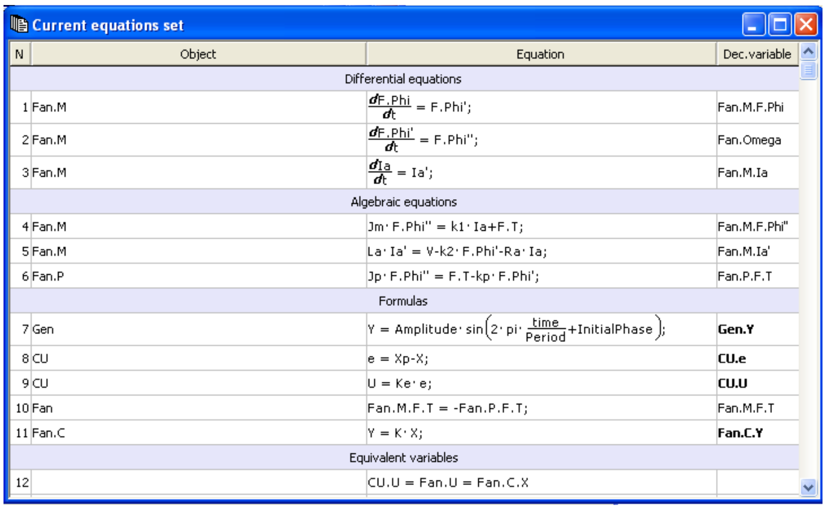
**Figure 10.** Numerical methods.

C/F components in MvStudium are analogous to Modelica components but in MVL there is no limitation on type (NAE, ODE, DAE), dimension, and form (explicit, semi-explicit, implicit) for a current solved system of equations prescribed to a current state of a B-Chart. Using information about an “old” system MvStudium builds, analyzes structure and finds consistent initial conditions for a “new” system. An “old” and a “new” system may have different unknowns, dimensions, types and forms. We do not know conditions guarantying against appearance of underdetermined final systems for all admissible behaviors of a hybrid automaton composition with given local behaviors. Our experience shows that underdetermined final systems more often appear during conceptual stage of new model designing. Automatic building of a determinaed system in this case is a cost-based and dangerous operation resulting sometimes in not physically feasible behavior. What is more it is very difficult to distinguish a very similar behavior from a true one.

In the present version of MvStudium a final system is built using topological equations (structure of functional diagram) and component equations (local behavior of components). MvStudium detects underdetermined and high index final systems (using Pantelides’ algorithm) and warnings for a user. For finding consistent initial conditions different modifications of Newton’s methods are used. Numerical methods are divided into three groups: numerical methods for solving non-linear algebraic equations, ordinary differential equations and algebraic-differential equations. Numerical solution MvStudium for ODE and DAE starts initially with an explicit method and if stiffness is detected an explicit method is changed for an implicit one automatically (Figure 10).

## 5. Visual debugging

Modern graphical environments for modeling and simulation copes well with building and solving final systems. However the main problem is not in automatically building of a model, but in proving that the built model corresponds to a real world system. In the present version of MvStudium a user can trace a process of forming and solving equations and watch a built final system in a usual mathematical form. It is not enough for a large scale system, but better than nothing.


Current equations set

| N | Object | Equation | Dec.variable |
|---|---|---|---|
| | | Differential equations | |
| 1 | Fan.M | $\frac{dF.Phi}{dt}$ = F.Phi'; | Fan.M.F.Phi |
| 2 | Fan.M | $\frac{dF.Phi'}{dt}$ = F.Phi''; | Fan.Omega |
| 3 | Fan.M | $\frac{dIa}{dt}$ = Ia'; | Fan.M.Ia |
| | | Algebraic equations | |
| 4 | Fan.M | Jm·F.Phi'' = k1·Ia+F.T; | Fan.M.F.Phi'' |
| 5 | Fan.M | La·Ia' = V-k2·F.Phi'-Ra·Ia; | Fan.M.Ia' |
| 6 | Fan.P | Jp·F.Phi'' = F.T-kp·F.Phi'; | Fan.P.F.T |
| | | Formulas | |
| 7 | Gen | Y = Amplitude·sin$\left(2\cdot pi\cdot\frac{time}{Period}\right.$+InitialPhase$\left.\right)$; | **Gen.Y** |
| 8 | CU | e = Xp-X; | **CU.e** |
| 9 | CU | U = Ke·e; | **CU.U** |
| 10 | Fan | Fan.M.F.T = -Fan.P.F.T; | Fan.M.F.T |
| 11 | Fan.C | Y = K·X; | **Fan.C.Y** |
| | | Equivalent variables | |
| 12 | | CU.U = Fan.U = Fan.C.X | |

**Figure 11.** Virtual Test bench: a current final system.

## 6. Symbolic calculations

One possible way of increasing effectiveness of simulation is to use symbolic calculations to transform and simplify the final system. Symbolic calculations in MvStudium are used for differentiation of expressions and equations, solving some types of equations (when it is not expensive), simplification of equations, decreasing the dimension of the solved numerically final system.

## References

[1] Breitenecker F., Proper N. Classification and evaluation of features in advanced simulators. Proceedings MATHMOD 09 Vienna, Full papers CD Volume.

[2] Kolesov Y. B. Object-Oriented Modeling of Complex Dynamical Systems. St. Petersburg: Publishing House of the Polytechnic University, 2004. – 239 p.http://www.eurosim.org.

[3] Senichenkov Y. B. Numerical Modeling of Hybrid Systems. St. Petersburg: Publishing House of the Polytechnic University, 2004. – 206 p.

[4] Altunin K.Y., Kolesov Y. B., Senichenkov Y. B. A tool for modeling and simulation of complex dynamical systems. Proceedings of the distributed intelegebt systems and technologies. Workshop DIST, St. Petersburg, Russia, 6 p. 2009.