News from Rand Model Designer for Industry and Education

Y. Kolesov,* D. Inikhov,*
Y. Senichenkov, A. Isakov****

**MvSoft, Moscow, Russia, (e-mail: ybk@mail.ru)*
***Peter the Great Saint-Petersburg Polytechnic University, St. Petersburg, Russia, (e-mail: senyb@dcn.icc.spbstu.ru)*

Corresponding author-Yuri Senichenkov, (e-mail: senyb@dcn.icc.spbstu.ru)

Abstract: Rand Model Designer *(http://www.mvstudium.com)* is a modern tool for modeling and simulation of hierarchical multicomponent event-driven dynamical systems. It was demonstrated in the series of MIM-conferences, including MIM-2013 (Senichenkov Yu. (2013)). It utilizes the UML-based object-oriented Model Vision Language for designing dynamical and hybrid systems, using modification of State Machines - large-scale multicomponent systems, such as: control systems with "inputs-outputs", "physical" systems with "contacts-flows", and novel variable structure component systems, particularly "agent" systems. The recently launched tool "Visual Debugger" includes new modifications of numerical solvers for sparse systems, for instance it is now possible to reorder each solved system to block-triangular form with the help of Tarjan's algorithm in order to accelerate execution speed.

Keywords: object-oriented modeling, dynamical and hybrid systems, components with oriented and non-oriented links with internal hybrid automata, component models with variable structure.

1. INTRODUCTION

Universal visual tools are generally used for designing multicomponent continuous, discreet, and hybrid systems by means of unified graphical representations. Rand Model Designer Graphical Editor manages (Kolesov Yu. (2013)):

- usual mathematical notation for systems of equations for continuous "do-activates" of state machines (algebraic, differential, and differential-algebraic systems);
- behavior-charts (a variant of Unified Modeling Language state machines without parallel activates) for event-driven discreet and continuous dynamical systems;
- hierarchical component diagrams with static and dynamical open components with "input-output" or "contact-flow" connectors.

Unified graphical notations, based on standards are preferred when possible. An environment transforms them to large-scale systems of algebraic, differential, differential-algebraic equations in different canonical forms. However, it is not sufficient for getting unified current (final) system for all tools, because each of those transforms and simplifies the system additionally in its own manner. Subsequently, numerical or symbolic Solver lunches a final system.

It is important to note that the role of symbolic transformations and symbolic Solvers increases in traditional numerical environments for modeling. At the same time, modelling problems become imperative also for designers of mathematical tools such as Maple, Mathematica (for example see http://www.maplesoft.com/products/maplesim/).

At present, there are only a few open-source authoritative packages of numerical solvers of differential and differential-algebraic equations (http://www.netlib.org/), thus those are usually integrated. Reaching unification is a current trend, but it is still a long way to go.

Comparative analysis (Breitenecker F. (2009), Martin-Villalba C. (2014)) of visual environments for modeling and simulations of complex dynamical systems based on criterions: "universality", "commonality" and "object-oriented approach", demonstrates that presently the Rand Model Designer is the only universal and unified object-oriented tool, where "unified" and "object-oriented approach" means following UML that is a "de facto" standard:

- "Simulink+Toolboxes" (http://matlab.ru/products/simulink) is not an object-oriented tool;
- hybrid system usage in Modelica is limited in comparison with UML state machines ((Tiller M. (2001), Fritzson P. (2011));
- it is rather problematic to implement "physical" modeling in AnyLogic (http://www.anylogic.ru/) and Ptolemy (http://ptolemy.eecs.berkeley.edu/ptolemyII/).

UML (Rumbauth J. (2005)) was developed as a standard for discrete systems and further, without thorough discussion, extended to continuous ones. However, it is advised to design similar, yet separate, standard for complex dynamical systems, and for their unified libraries of devices.

Moreover, development of special instruments for debugging and testing, as well as the unified set of test methods is crucial for industrial use.

Similarly, visual environments for modeling and simulation play a substantial role in the field of education. For instance, initial familiarization with computer modeling and choice of profession in schools may rely on lite versions; standard versions may be utilized for training in university-level modeling courses; while professional versions are perfect fit for training in industry.

2. MODELS WITH VARIABLE STRUCTURE

All modern modelling languages support designing event-driven continuous dynamical systems. There is unified standard for hybrid systems in a form of extended version of UML' state machines (http://www.uml.org/). The version covers hybrid time and continuous do-activities in the various forms (http://www.mathworks.com/discovery/finite-state-machine.html; http://www.mvstudium.com) used in line with discrete time and discrete activities (Fig. 3-4). Hybrid systems demonstrate changeableness in a modeling object behavior, while structures of complex dynamical systems may be variable as well.

Changing the number of independent "agents" (open subsystems with constant structure communicated on-stream) is elementary example of systems with variable structure - queuing systems with variable number of servicers and clients.

A more complicated case is component queuing system with behavior described by continuous equations. Changing number of services (equipment failure) or number of clients force rebuilding of the current system of equations on run-time, that is labor-consuming, especially for models with "contacts-flows" connectors of components.

Fig. 1-2 displays an electric circuit with power supply device, load with constant and dynamic components, and rectifier.

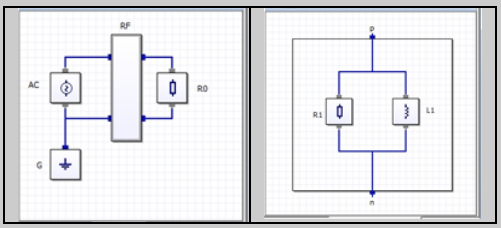

Fig. 1. Constant load – resistor R0 (left) and dynamic load (right).

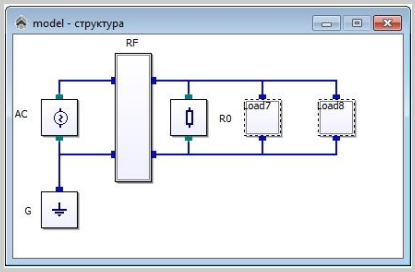

Fig. 2. Circuit with constant and dynamic Load.

A new variant of load is generated by the command `new(Load)` associated with external transition of behavior-chart (Fig. 3) on interval [0, t] with random value t, distributed by exponential law with average T. The new object of class "Load" gets new random values for the following parameters: lifetime (Tend), resistor (RR), and inductance (LL), distributed by normal law. New load is placed on the diagram (Fig.2), whereas a new current system of equations is generated, analyzed, and model is going on until object "Load" will be destroyed (`destroy(Load)`) after Tend.

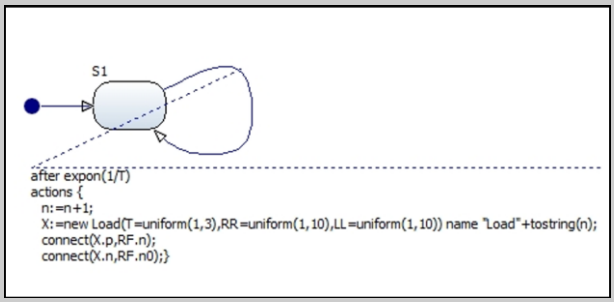

Fig. 3. Generating a dynamic load by behavior-chart

3. VISUAL DEBUGGER

A sequence of events changes behavior of a hybrid system (Fig. 4): continuous do-activates (systems of equations) substitute discrete actions (procedures written on high-level language).

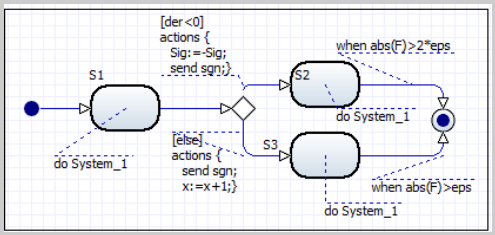

Fig. 4. Behavior of a simple model

By selecting interest events user can form a list of breakpoints for continuous and discrete behavior with breakpoint conditions like: time, conditions, transitions, states, signals (Fig.5).

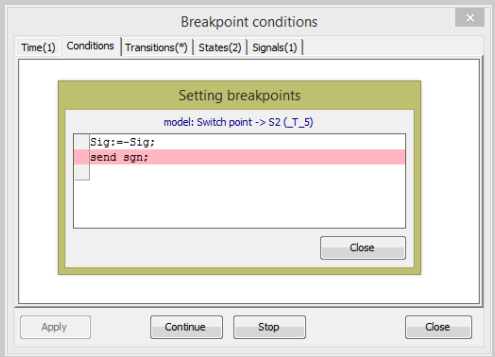

Fig. 5. Breakpoint conditions for hybrid automaton.

Obtaining additional information on current solved system of equations is possible with the help of Services, as presented by Fig. 6 – it is possible to display solved equation in a usual

mathematical form, its structure (structural matrix), Jacobi matrix and its eigenvalues.

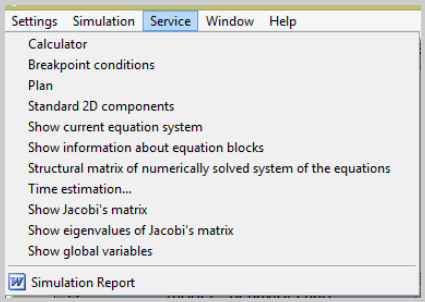

Fig. 6. Additional information about solved system.

Discrete part of hybrid behavior written in modelling language presumes single-step debugging of a sequence of instructions, procedures and functions. On run time, a new dialog window appears before executing designated instruction (Fig.7). It shows the control point and values of all used variables. User can execute instructions one by one, change values of variables, and follow subroutines and functions.

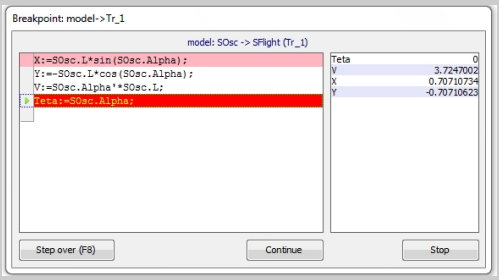

Fig. 7. Single-step debugging.

4. BLOCK TRIANGULAR FORM OF FINAL SYSTEM

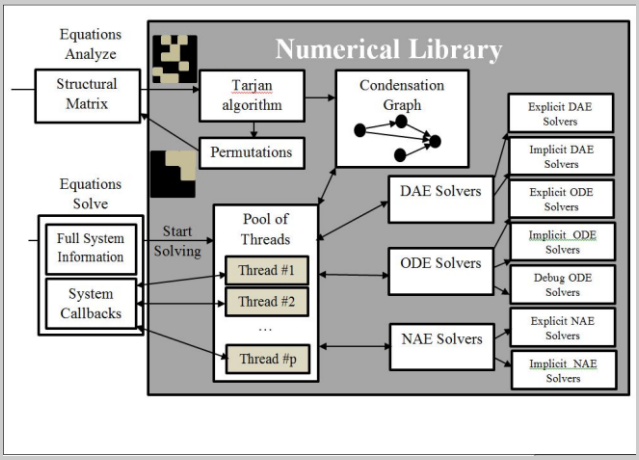

Fig. 8. RMD numerical library

The structure of RMD's Numerical Library, as described by Fig. 8, and the algorithm of interaction between Numerical library and Model Engine (Fig. 9) have been changed for implementing algorithm of building of final systems on run time (see (Kolesov Yu. (2014), Andrey A. Isakov 2015)).

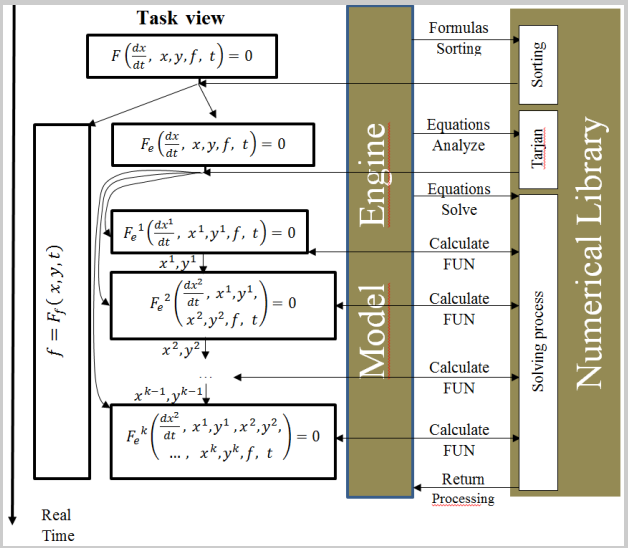

Fig. 9. Interaction between Numerical library and Model Engine.

On run time, in case any new events have occurred, necessity of rebuilding of current solved system is estimated, and the new system is built if deemed necessary. If transversal of new system's structural matrix is full, then structural analysis (for details see (2014, 2015)) of the whole model is starting; otherwise (in case of an underdetermined system), a model is recognized as incorrect. Automatic searching of a subset of unknowns leading to correct system is provided as an option.

40 pt
0.556 in
14.1 mm

For correct systems:

1. Block triangular form of structural matrix is built if possible with the help of Tarjan's algorithm for searching strongly connected components of a graph.
2. Block triangular form may be used for parallel calculations. Strongly connected components (diagonal blocks) are associated with algebraic, differential, and differential-algebraic systems of equations. "Subtask" is a process of solving a system of equations corresponding to a diagonal block with the help of a suitable Solver. Subtasks may be executed sequentially or in parallel. A thread pool for parallel execution is forming automatically using computer hardware information (number of processors, number of kernels for a processor). The subtask readiness for execution is ascertained using condensation of a graph of Tarjan's algorithm (Fig. 11). Initially all nodes (subtasks) of the condensation are marked as "Unresolved". The calculation ends when all nodes become "Solved". If an "unresolved" node has no input edges or its input is "Solved" it becomes "Ready to start", otherwise - "Not ready to start". Running subtask is displayed as "Solving", and after ending becomes "Solved".

Let us consider an example of multithreaded calculations for algebraic system of equations with triangular form shown in

Fig. 10. The matrix diagonal blocks are associated with subtasks.

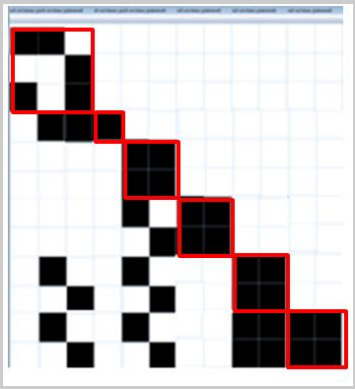

Fig. 10. Structural matrix after ending Tarjan's algorithm

The condensation with current labeling is shown in Fig. 11, and time diagram for threads - in Fig. 12.

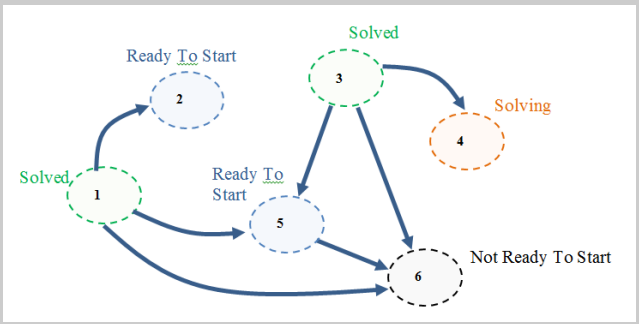

Fig.11. Current labeling of the condensation.

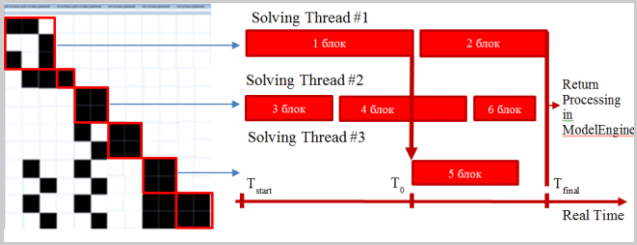

Fig. 12. Time diagram for threads.

A set of models developed by Transas company (Tarasov (2012)) (http://www.transas.com/products) was used for testing a new approach. The results of numerical experiment for two most difficult problems are presented in Table 1 and Table 2. The computer used for calculations had four processors, so it was possible to create maximum four threads, but even if only one thread was used the total time of calculations decreased by a factor of two.

Table 1. Product Tanker, Cargo System, about 2500 equations.

Thread Count	1	2	4
Old Numerical Library	8,53	-	-
New Numerical Library	4,72	3,80	3,67
Time difference	**44,67%**	**55,45%**	**56,98%**
Time on Right Part Calculation	2,44	2,47	2,47
Right Part Calculation Count	28964	28964	28964
Control Time	2,28	1,33	1,20
From one thread	100,00%	**58,33%**	**52,63%**

Table 2. Chemical Tanker, Inert Gas System, about 500 equations + 1000 formulas

Thread Count	1	2	4
Old Numerical Library	8,24	-	-
New Numerical Library	3,90	3,61	3,57
Time difference	**52,67%**	**56,19%**	**56,67%**
Time on Right Part Calculation	2,87	2,96	2,94
Right Part Calculation Count	23329	23329	23329
Control Time	1,03	0,65	0,63
From one thread	100,00%	**63,11%**	**61,17%**

5. USING RAND MODEL DESIGN IN EDUCATION

Rand Model Designer is used for training in schools, universities, and industrial enterprises. (Fig. 13).

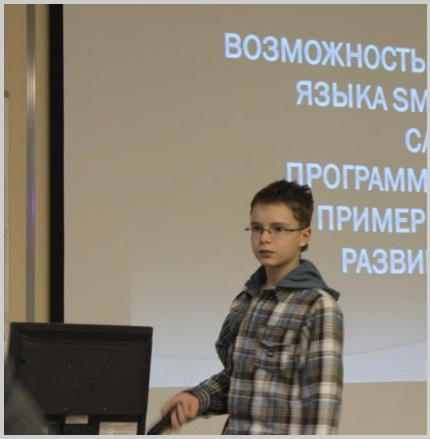

Fig. 13. Presentation by a young programmer.

Students interested in computer modelling may take their first steps with Rand Model Designer. School-level knowledge of physics and mathematics is sufficient for designing simple continuous and hybrid models, such as: dancing ball, parachute jump, and alike, following by more complex tasks of creating 2D- or 3D- animation. With professional electrical and hydraulic libraries, it is possible to design simple virtual experimental facilities and to play with them.

Rand Model Designer is suitable for university training, as its primary function allows the illustration of the main concepts of such university courses as "Basis of mathematical modeling" and "Computer modeling of complex dynamical systems". Besides, students have an opportunity to learn basics of object-oriented modeling, as Rand Model Designer operations follow UML standard. Blocks of the special

library “Syslib” are identical to those of Simulink, thus modeling as in Simulink is possible. Likewise “Physical” libraries “electricity” and “hydraulics” share numerous similarities with the corresponding libraries written in Modelica language.

6. NEWS

Symbolic differentiation of functions had recently become available in the Rand Model Designer.
Let us consider an example of inverse problem for a mechanical system, where solution x(t) of a differential equation is known and we want to find a force F generating it (Fig.14). In this case, it is necessary to differentiate x(t).

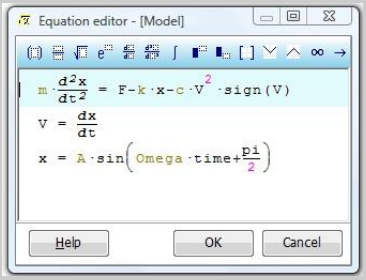

Fig. 14. Inverse problem for pendulum.

Until now, the following method was used: additional differential equations (linear differentiator for example) were added to initial equations (Fig. 15) automatically.

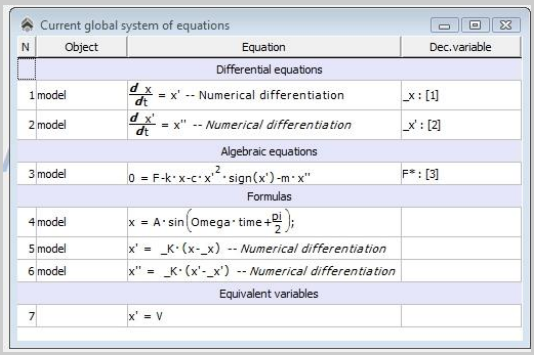

Fig. 15. Initial system with a linear differentiator.

In such a case, accuracy and qualitative conformity of the solution strongly depends on initial conditions of added equations. Now Rand Model Designer builds a solution with the help of symbolic differentiation (Fig. 16).

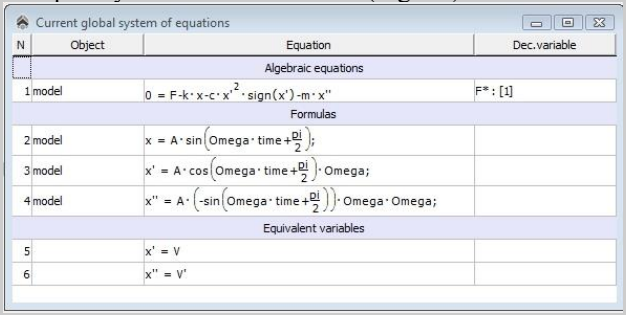

Fig. 16. Using symbolic differentiation for building solution.

REFERENCES

Senichenkov Y., Kolesov Y., Inikhov D. Rand Model Designer in Manufacturing Applications (2013) //, IFAC-PapersOnLine: Manufacturing Modelling, Management, and Control, Vol. 7, Part 1. P. 1572-1577, doi: 10.3182/20130619-3-RU-3018.00300.

Kolesov Yu. B., Senichenkov Yu. B. (2013). *Mathematical modeling. Component technologies.* St. Petersburg, Peter the Great Polytechnic University. 2013, - 223 pp. ISBN 9768-5-7422-3997-0.

Martin-Villalba C., Urquia A., Senichenkov Y., Kolesov Y. (2014). Two approaches to facilitate virtual lab implementation. Computing in Science and Engineering 16 (1) PP. 78 – 86, doi: 10.1109/MCSE.2014.29

Andrey A. Isakov, Yuri B. Kolesov, Yuri B. Senichenkov. (2015). A new tool for visual modelling – Rand Model Designer 7. IFAC-PapersOnLine 48-1(2015) 661-662.

Fritzson P. (2011) Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica, 232 pp, Wiley-IEEE Press

Breitenecker F., Proper N. (2009). Classification and evaluation of features in advanced simulators. Proceedings MATHMOD 09 Vienna, Full papers CD Volume.

Fritzson P. (2006). *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press.

Rumbauth J., Jacobson I., Booch G. (2005). *The unified modeling language. Reference manual.* Second edition. Addison-Wesley.

Tiller M. (2001). *Introduction to physical modeling with Modelica.* The Springer International Series in Engineering and Computer Science.