STUDIA UNIVERSITATIS

Revistă științifică a Universității de Stat din Moldova, 2012, nr.2(52)

A NEW APPROACH TO MODELING OF REAL WORLD SYSTEMS

Roman DAMASCHIN, Aurelia PROFIR

Department of Programming Technologies

În prezenta lucrare se face o încercare de a elucida importanța elaborării instrumentelor software pentru modelarea adecvată a sistemelor reale, cum ar fi, spre exemplu, sistemele biologice, computaționale, economice, fizice etc. şi modul în care astfel de sisteme pot fi modelate. De asemenea, este prezentat un instrument software nou pentru modelarea şi simularea sistemelor membranale, numit 3D VMPN. Această aplicație a fost realizată în perioada elaborării tezei de licență a lui Roman Damaschin.

Introduction

Imagine a situation: a young scientist is occupied by a research of the cell life-cycle of some living organism. The birth and life of cells depend on both external factors, such as: temperature, humidity, chemicals/elements in the air, etc.; and internal factors: genotype, viruses, bacteria, state of the immune system, etc. The main purpose of the research - to learn how the cellular birth, life and death depend on those factors above. It is a little bit problematic to perform such research on real cells. Further, it's need to wait n-quantity of time until the cells pass all stages of the life-cycle. And what if such process would be costly in material term? What if the researches are not bind with some cells, but with launching of a satellite to the space?

All these researches are widespread in the world and it is not important what is the object of the research - the cells of some living organism or launching of the satellite. The most important is what is the best way to do these researches? One of these ways is modeling, which suggests creating an abstract model for simulating the behavior of a real system. The question is how exactly to model? For this purpose we elaborated a special instrument, called 3D Visual Membrane Petri Nets (3D VMPN) [1], although this is not actually an instrument, but a framework based on the concept of P systems [2,3] and the theory of Petri nets [4]. P systems, also called membrane systems, allow to create formal models not only of simple systems, but models of systems with membrane structure. A classic Petri net is a particular kind of bipartite directed graphs and contains such objects as locations, transitions and some kinds of directed arcs.

The 3D VMPN application represents a new extension of Petri nets - membrane Petri Nets, and it has additional basic components - membranes. All these objects have their own special properties and rules of behavior and can interact with each other. The interaction can be described by special mathematical expressions using standard mathematical functions, logical operators and object property values. Some of objects can be connected with other ones to be able to transfer some data, but here is one interesting detail: these data can be transferred according to a math expression, calculation time, a priority value or a guard function. So, if it can be build such a model and properly create communications between the objects of the model, then an adequate model can be realized to simulate real world systems.

Now we know what our young scientist can choose – continue researching under real conditions or use computer simulation of relevant processes. There is just one thing - how to build such model?

Modeling Tool

The theory of the Petri nets, membrane calculations and P-systems are always developed, it has been already created a lot of applications allowing to build and simulate some models, but there are two questions here: are such models convenient at work and is the simulation fast? All previous versions of such applications allowed to create models, using 2D geometry primitives and algorithms of simulation were lineal. Today, when the multi-core processors are widespread, it is not clever to create lineal algorithms, which use only one processor. As for models, surely, it would be much more convenient if they could work as real systems, looking at them as if they are on the your workplace. For this purpose it's necessary to use not only 3D geometry primitives, but some other 3D objects having any appearance you want. The new application called MPN 3D Builder suggests all these features, including powerful instrument of analyzing data (including tables of values and chart builders) gained from the simulation, improved formula builder, possibility to create models declaratively using special language, and an instrument of importing .3ds files which contains 3D objects created by some 3D editors.

For developing a graphic part of the 3D VMPN application the newest and the most effective technology such as Windows Presentation Foundation (WPF) [5] has been used. It is a graphical subsystem for rendering user interfaces which utilizes DirectX. The Parallel Extensions and Task Parallel Library (PE & TPL) [6] is used to synchronize and co-ordinate the execution of concurrent tasks improving performance of the algorithms for simulation. But all of them were created not only for improving the models appearance, because all objects of model "become alive" during the simulation.

This means that at the real-time it can be observed the form transformation of objects; data transferring between the objects, motion of the objects at some different points of space and all changes are happened according speed of animation of each object.

Also during the simulation process the user can interact with camera and move between objects or membranes to watch what's happening inside of them. All of these create an impression of working with a real system, but at the same time don't forget, that the main purpose of these models is receiving data after the simulation for researching, but not only to create a model which apparently would copy a real system.

So let's take a look at the application closer. The main window contains the main toolbar with all available objects, menu, model constructor, mini-toolbar for some manipulations with model and status bar. The object properties panel is situated at the right side of the screen.

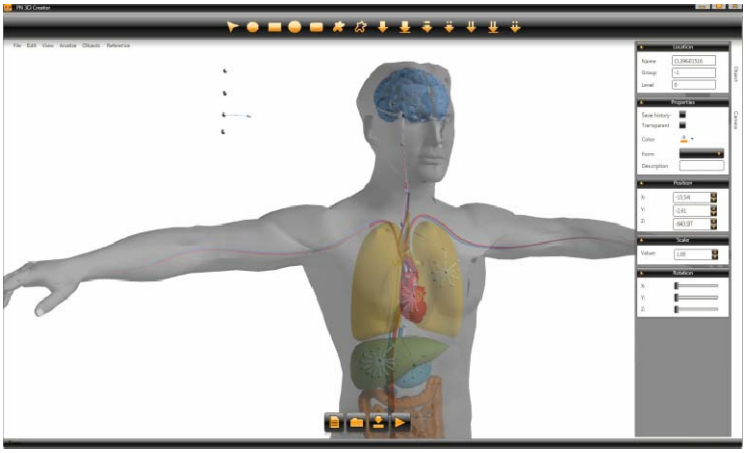

Fig.1. The main window of the 3D VMPN application.

The constructor allows to create models by mouse-click adding objects at any place you point, but also there is a possibility to create models declaratively utilizing so-called ODL (Object Description Language) which is similar with XML syntax, but with some improvements.

The constructor and ODL Editor are connected reflectively and any object added via the constructor will be pasted at the code of ODL and conversely. Each object, of course, can be moved, scaled, rotated. Also there is a possibility to change appearance of all objects (color, form, etc.).

When the model was built some initial parameters for the objects, for example, default values, math expressions, formulas, etc. can be set.

```
<Model Name="model1" >
    <Membrane Name="m1">
        <ContinuousLocation Name="c1" Level="1220" />
        <ContinuousLocation name="CL41961994" Level="0" />

        <ContinuousTransition Name="CT20085125" Expectance="12" ExpectanceFormula="c1+6*(a3+1)" />
        <ContinuousTransition Name="CT25041904" Expectance="1" />

        <ContinuousArc Name="cArc46711052" Weight="1" StartObject="c1" EndObject="CT20085125" />
        <ContinuousArc Name="cArc26848244" Weight="10.99" WeightFormula="9+2*Sin(Tan(a4))"
                                           StartObject="Bag" EndObject="CT30876502" />

        <ContinuousLocation Name="c4" Level="0" />
        <ContinuousLocation Name="c5" Level="0" />
        <ContinuousLocation name="CL41961321" Level="1" />
        <ContinuousLocation Name="c1" Level="1220" />
        <ContinuousLocation name="CL43961976" Level="100" />
        <ContinuousLocation name="CL48895212" Level="210" />
```

Fig.2. ODL Editor.

For this option it can be used the Object Properties Panel, but if the model is large it's not so convenient and it's better to use Model Configurator. It contains all properties for all kinds of objects and a table with objects which can be found by name or some other parameters.

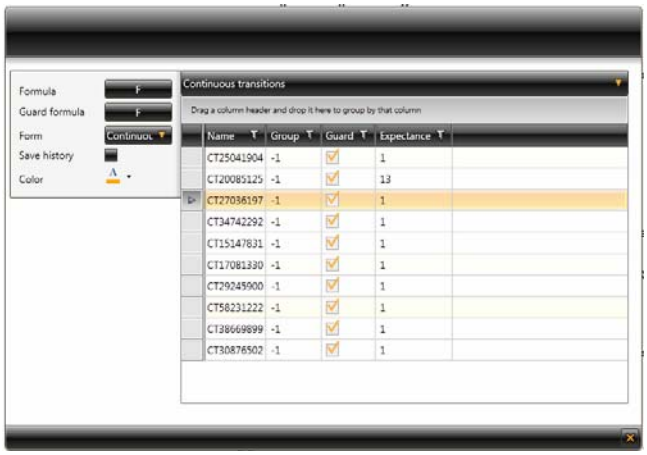

Fig.3. Model Confugurator.

As it has been already mentioned, the math expressions for the certain values of objects that can be set.

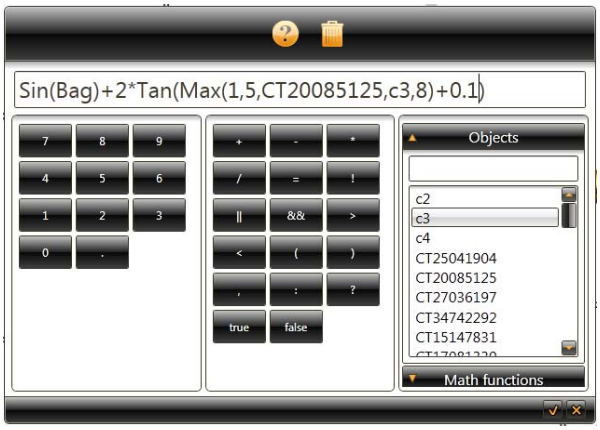

Fig.4. Formula builder.

The Formula Builder is a very powerful and very important tool here because with its help values can be changed dynamically. It contains a lot of standard math functions including some additional ones (Min, Max, etc.). Its main function is to compile the expression and return calculated value. An expression can use property values of objects (for example, for a discrete locations such value is tokens quantity, for a discrete transition - delay time, etc.). So if you attach a formula for the chosen object, for example, let it be a discrete transition, the delay time of this transition will be changed at each step of the simulation according this formula. When the simulation is run, the formula accepts values from all objects at each iteration which are contained in it (if they are there of course) and according to these values it is calculated and returned a new value.

So when the initial data are set up the model can be simulated. There are two kinds of simulations: with animation (it means that users can watch simulation process step by step, including changing behavior, size and form of each object) and without animation (it means that users cannot watch simulation process and they can receive only some results of simulation).

On Figure 5 it can be seen the simulation with animation where users can change the animation speed. Also here is a possibility to observe a live-chart which shows value changing of the chosen object.

After the simulation the powerful chart builder tool which is very useful for data analyzing can be used. There some kinds of charts here and just few of them will be described. The first chart (Figure 6) shows how each object of model was changed at each passed simulations.

Fig.5. Simulation window.

Fig.6. Chart builder (in brief).

The second chart (Figure 7) shows how selected object was changed at each step of certain simulation.

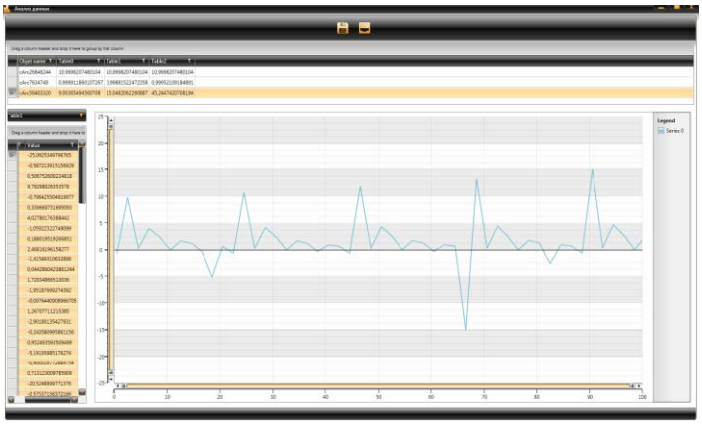

Fig.7. Chart builder (detailed information).

Here it is a data table on the left side of the screen and it contains just two columns: "Time" and "Value" telling about: "at the instant X the value is Y".

All selected rows of the table can be reflected on the chart, which is situated on the right side of the screen. The chart is scalable, so if there are a lot of selected rows the user can zoom in the chart and observe the data.

The charts and data table can be exported on some kinds of documents such as MS Word, MS Excel, HTML (for data table only) and image files.

The application has some other functional ability, such as: global properties for interface and for models and other features that can help to do some manipulations with models and their objects, but there is no sense to describe them here.

Conclusions

At the present time with help of this application a patient-specific model has been created [1]. This model allows to simulate compensatory mechanisms in type I diabetes. The model gave some results, which could help to choose an effective treatment. But the progress is not standing still and the development of the theory of Petri nets is being continued creating new extensions, which could be useful for creating models with complex structure and/or behavior.

Also it concerns to new technologies which could help to create more flexible and effective applications. Therefore even if the application allows to create real world models there a lot of improvements and new features which must be added at the close future to make such applications more comfortable for work with models and more faster for simulation.

References:

1. Profir A., Damaschin R., Opinca C., Prepeliță L., Prepeliță A., Yang B. Patient-specific computer modeling using 2D and 3D visual membrane Petri Nets. - ISSN 1857-2073 // Studia Universitatis. Seria „Ştiinţe exacte şi economice", (Chişinău), 2011, nr.2(42), p.57-64.
2. Paun Gh. Membrane Computing. An Introduction. – Natural computing Series / Ed. G. Rozenberg, Th.Back, A.E. Eiben, J.N. Kok, H.P. Spaink. Leiden Center for Natural Computing. - Springer – Verlag, Berlin Heidelberg New York, 2002, p.420.
3. Paun Gh., Rozenberg G. A Guide to Membrane Computing // Theoretical Computer Science, 2002, vol.287, p.73-100.
4. Rene D., Hassane A. Discrete, continuous, and hybrid Petri Nets. - Springer, 2005, p.516.
5. MacDonald M. WPF for professionals. - Wilson, 2008, p.25-44.
6. Nagel C., Evjen B., Glynn J. Professional C# 4.0 and .NET 4.0. - Wrox, 2010, p.730.

Prezentat la 02.03.2012