Advanced DEVS models with application to biomedicine

Gabriel Wainer, Shafagh Jafer, Banan Al-Aubidy, Alex Dias, Roderick Bain, Michel Dumontier, James Cheetham

Abstract—**Simulation is becoming increasingly important in the analysis and design of complex systems such as those involving biological processes. The complexity of these systems makes computer simulation an adequate tool to study them under particular experimental conditions. We presented a simulation model for the metabolic pathways in the cells (namely, the Glycolysis and Krebs cycle), using the DEVS formalism and the CD++ tool. Here, we extend this experience, and we show a model on synapsin and vesicles interaction in nerve cells, and another one on the liver functions using similar techniques. Our long term goal is to provide realistic simulations of different biological processes when specific internal or external parameters are applied.**

Index Terms— **Cellular Automata, DEVS, liver, synapsin, vesicle**

I. INTRODUCTION

SIMULATION is becoming increasingly important in the analysis of biological processes. The complexity of these systems makes computer simulation an adequate tool to study them under particular experimental conditions. Our long term goal is to provide realistic simulations of different biological processes when specific internal or external parameters are applied. We want to provide environments that the researchers in biology and medicine could use to understand and control the dynamics of these biological processes through computer simulation in a whole organelle scale.

In [1], we showed a first proof of concept, by defining a detailed model of the behavior of the mitochondrion, which fulfills different important roles in cellular metabolism. In this paper, we introduce two new models. One of them is devoted to modeling the reserve pool of synaptic vesicles in a presynaptic nerve terminal, and predicting the number of synaptic vesicles released from the reserve pool as a function of time under the influence of action potentials at differing frequencies. We also introduce a model that could be used by a liver specialist in identifying possible harmful outcomes to the liver by simulating certain conditions.

Our approach is based on the use of the DEVS formalism [2]. DEVS provides a framework for the construction of hierarchical models in a modular fashion, which makes DEVS ideal for describing naturally hierarchical systems as those presented here.

G. Wainer, S. Jafer, B. Al-Aubidy, A. Dias are with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6 Canada (phone: 613-520-2600 x 1957; e-mail:, gwainer@sce.carleton.ca).

R. S. Bain, M. Dumontier and J. J. Cheetham are with the Department of Biology, Carleton University, Ottawa, ON K1S 5B6 Canada (e-mail: james_cheetham@carleton.ca).

Likewise, DEVS discrete-event nature improves the execution performance of the models, due to the asynchronous nature of the events occurring in the cell. DEVS also uses explicit timing information; hence, we can adequately represent timing of the reactions occurring in the organism.

II. BACKGROUND

A real system modeled with DEVS is described as a composite of submodels, each of them being behavioral (atomic) or structural (coupled). A DEVS atomic model is can be informally described as in Fig. 1. DEVS coupled model is composed by several atomic or coupled submodels. Coupled models are defined as a set of basic components (atomic or coupled), which are interconnected through the model's interfaces. The model’s coupling defines how to convert the outputs of a model into inputs for the others, and to inputs/outputs to the exterior of the model.

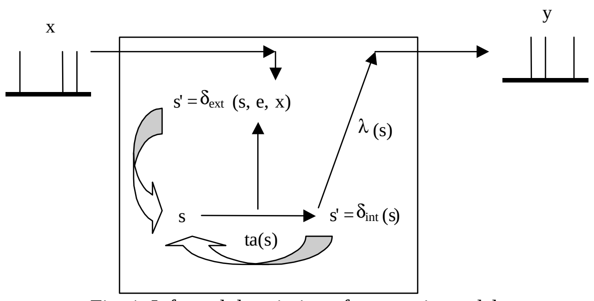


Fig. 1. Informal description of an atomic model.

Cell-DEVS [3] has extended DEVS, allowing the implementation of cellular models with timing delays. Each cell is defined as a DEVS atomic model, and it can be later integrated to a coupled model representing the cell space. Once the cell behavior is defined, a coupled Cell-DEVS can be created by putting together a number of cells interconnected by a neighborhood relationship. A Cell-DEVS coupled model is informally presented in Fig. 2.

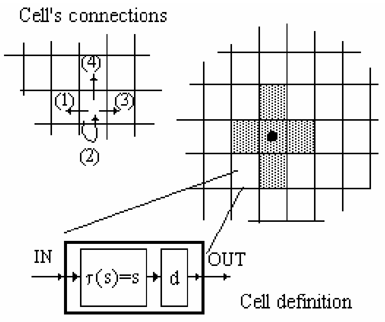
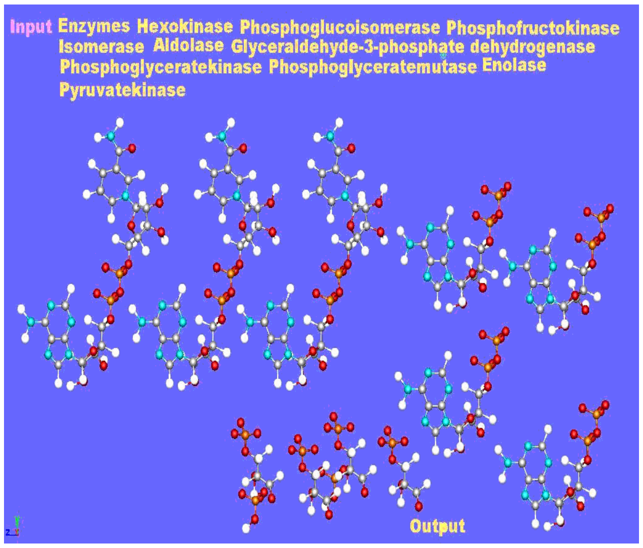


Fig. 2. Description of a Cell-DEVS atomic model.

CD++ [4] is a modeling tool that was defined using the DEVS and Cell-DEVS specifications. The toolkit includes facilities to build DEVS and Cell-DEVS models. DEVS Atomic models can be programmed and incorporated onto a class hierarchy programmed in C++. Coupled models can be defined using a built-in specification language. Cell-DEVS models are built following the formal specifications for DEVS models (informally presented in the previous section), and a built-in language is provided to describe them. CD++ makes use of the independence between modeling and simulation provided by DEVS, and different simulation engines have been defined for the platform.

At present, we have created different models at the in a whole organelle scale. The first one, presented in [1], was devoted to create a precise model of the reactions in the mitochondrion. The chief function of the mitochondria is to create energy for cellular activity by the process of aerobic respiration. In this process, glucose is broken down in the cell's cytoplasm via the **glycolysis** process, to form pyruvic acid. We used CD++ to model and simulate biological pathways, thus providing a systematic method for creating models consisting of sets of lower-level interactions. shows a snapshot of some of the reactions in the Krebs Cycle Animation done in CD++/Maya [5], which shows the formation of Acetyl CoA,.


Input
Enzymes Hexokinase Phosphoglucoisomerase Phosphofructokinase
Isomerase Aldolase Glyceraldehyde-3-phosphate dehydrogenase
Phosphoglyceratekinase Phosphoglyceratemutase Enolase
Pyruvatekinase
Output
Fig. 3. Krebs cycle coupled model 3D visualization.

We are also interested in the interaction between synapsin and vesicles in nerve cells. Synapsin is a neuron-specific phosphoprotein that binds to small synaptic vesicles and actin filaments in a phosphorylation-dependent pattern. Microscopic models have demonstrated that synapsin inhibits neurotransmitter release either by forming a cage around synaptic vesicles or by anchoring them to the F-actin cytoskeleton of the nerve terminal [6], [7].

Finally, we are interested in creating models on the liver cells. There are many functions the liver is responsible for, ranging from the regulation of blood aminoacids, sugar and lipids; up to storing vitamins (A and B12), and forming plasma proteins [8]. Our model is based on the one defined in [9], which defines the various reactions that the liver maintains. Our design demonstrates the process of substance transformations occurring with in the liver's lobule. These transformations occur when substances travel through various zones inside the lobule. Since the lobule is the building block of the liver, we are able to simulate a realistic structure of the human liver by connecting thousands of them.

III. A MODEL OF THE HUMAN LIVER IN CD++

Lobules make up the main functional and structural component of the liver, which is comprised of thousands of them. The lobule can be seen as a tube, where blood flows from the outside to the inside of the tube. The outside of the lobule is surrounded by the portal vein (PV), which brings blood into the liver. The inner vein is the central vein (CV), carries blood out of the lobule and eventually out of the liver.

When blood flows through the lobule, it undergoes several chemical reactions in multiple stages. In [9], a solution was suggested as to how to model the inner workings of a lobule in a simulation model. The lobule is modeled like a hexagonal cylinder. There are only 3 stages (zones), and several nodes are placed inside the lobule, where each node is connected to at least one other node in the lobule. The number of nodes in each zone is proportional to the approximated lobule volume of that particular zone. Each node is responsible for receiving a substance, and transforming it, and each node works interdependently of each other [9].

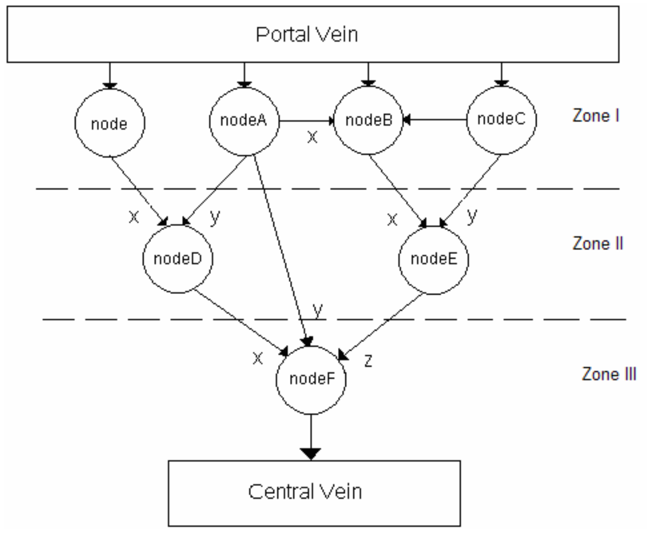


Fig. 4. Zones and Nodes [9].

We built a DEVS model based on these assumptions (using the CD++ toolkit), which represents the chemical composition of blood entering the liver lobule. A substance would enter the portal vein (PV), and it is then fed to all the nodes that are in zone I. After the nodes of zone I are finished transforming the substance, their output is fed to the nodes of zone II and then zone III. After this, the output is supplied to the central vein (CV). Each node has its own set of parameters to determine the output when given a certain input. Each node is given a delay to represent the time it takes for a substance reaction to reach completion. Following, we show code excerpts showing the definition of the node model in CD++.

```
Private:
  const Port &WinFX, &WinFY, &WinFZ; // output ports
  Port &WoutF;         // input ports
  Time reactionTime, time_zero;
  int Value, ValueX, ValueY, ValueZ;

Model &livernodeF::externalFunction (const
            ExternalMessage &msg) {
   if (msg.port == WinFX) ValueX=msg.value();
   if (valueX != 0 && ValueY != 0 && ValueZ != 0)
      holdIn(active, reactionTime);
    return *this;
}

Model &livernodeF::outputFunction (const
           InternalMessage &msg) {
  if (ValueX==1 & ValueY ==1)
      switch (ValueZ) {
        case 1: Value=7; break;
        case 2: Value=5; break;
        case 3: Value=3; break;
        case 4: Value=1; break;
        case 5: Value=2;
      }
     sendOutput(msg.time(), WoutF, Value);
    return *this;
    }

    Model &livernodeF::internalFunction (const
                 InternalMessage &msg) {
     passivate();
    }
```

Fig. 5. CD++ definition of a node

As we can see in Fig. 5, we initially define a number of I/O ports and state variables for the model. Then, we show the external transition function, which takes a message received on the "X" port, and then assigns that value to the variable ValueX. For node F which has three input ports, if it has received values from each of the ports, then it can go ahead and figure out the output value. The variable "reactionTime" defines the delay for the reaction. After this time has been elapsed, the output function is called. This function, based on the value of ValueX, ValueY and ValueZ, it will output a value representing the right chemical reaction on the variable "Value".

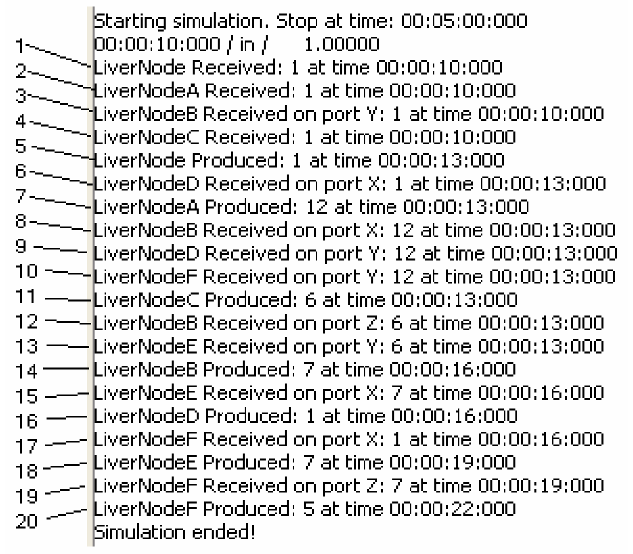
We checked each node individually, and then all the nodes were connected together and the entire lobule was check for accuracy. A coupled model file was used to couple all the atomic models together. Following is an excerpt of the code used in the lobule model that was responsible for coupling all the nodes together.

```
components: noder@livernode nodeA@liverNodeA
nodeB@liverNodeB nodeF@liverNodeF
in : in
out : out
Link : in win@node       Link : in winA@nodeA
Link : in winBY@nodeB    Link : in winC@nodeC
Link : wout@node winDX@nodeD
Link : woutA@nodeA windDY@nodeD
Link : woutA@nodeA windBX@nodeB
Link : woutC@nodeC windBZ@nodeB
...
```

Fig. 6. Structure of the lobule coupled model

The *components* line defines all the nodes were included in the lobule. The *link* lines are used to define which port is connected to which other port of another atomic model (e.g., the line *Link: WoutA@nodeA WinDY@nodeD* connects the output port of node *A* to the *Y* input port of node *D*). The *in* port represents the portal vein and the *out* port represents the central vein. The coupled model above is based on the model presented in Fig. 4 [9].

Based on this model, we studied different reactions in the lobule. One of the functions of the liver is the maintenance of a steady concentration of glucose in the blood. This is done through three types of reactions: gluconeogenesis, glycogen synthesis and degradation. Most substance reactions in the liver need energy and the source for this energy is ATP and ADP. In most cases, ATP is broken down into ADP and energy is released. This energy is used to drive the substance reactions. Oxaloacetato is used in the Matriz Mitochondrial. Oxaloacetato cannot cross the mitochondrial membrane until it is converted to Malato first. Once Malato passes through the membrane, it can then be converted back to Oxaloacetato to be used. This reaction shows that Oxaloacetato is produced by Pyruvate Carboxylase, and is then converted to Malato. These reactions were tested, and we show the results following.



```
    Starting simulation. Stop at time: 00:05:00:000
    00:00:10:000 / in /      1.00000
1   LiverNode Received: 1 at time 00:00:10:000
2   LiverNodeA Received: 1 at time 00:00:10:000
3   LiverNodeB Received on port Y: 1 at time 00:00:10:000
4   LiverNodeC Received: 1 at time 00:00:10:000
5   LiverNode Produced: 1 at time 00:00:13:000
6   LiverNodeD Received on port X: 1 at time 00:00:13:000
7   LiverNodeA Produced: 12 at time 00:00:13:000
8   LiverNodeB Received on port X: 12 at time 00:00:13:000
9   LiverNodeD Received on port Y: 12 at time 00:00:13:000
10  LiverNodeF Received on port Y: 12 at time 00:00:13:000
11  LiverNodeC Produced: 6 at time 00:00:13:000
12  LiverNodeB Received on port Z: 6 at time 00:00:13:000
13  LiverNodeE Received on port Y: 6 at time 00:00:13:000
14  LiverNodeB Produced: 7 at time 00:00:16:000
15  LiverNodeE Received on port X: 7 at time 00:00:16:000
16  LiverNodeD Produced: 1 at time 00:00:16:000
17  LiverNodeF Received on port X: 1 at time 00:00:16:000
18  LiverNodeE Produced: 7 at time 00:00:19:000
19  LiverNodeF Received on port Z: 7 at time 00:00:19:000
20  LiverNodeF Produced: 5 at time 00:00:22:000
    Simulation ended!
```

1) LiverNode received (NADH + H^+)
2) LiverNodeA received Piruvato
3) LiverNodeB received CO_2 on node Y
4) LiverNodeC received ATP
5) LiverNode produced (NADH + H^+)
6) LiverNodeD received (NADH + H^+) on port X
7) LiverNodeA produced Piruvato
8) LiverNodeB received Piruvato on port X
9) LiverNodeD received substance on port Y, ignore
10) LiverNodeF received substance on port Y, ignore
11) LiverNodeC produced ATP
12) LiverNodeB received ATP on port Z
13) LiverNodeE received substance on port Y, ignore
14) LiverNodeB produced Oxaloacetato
15) LiverNodeE received Oxaloacetato on port X
16) LiverNodeD produced (NADH + H^+)
17) LiverNodeF received (NADH + H^+) on port X
18) LiverNodeE produced Oxaloacetato
19) LiverNodeF received Oxaloacetato on port Z
20) LiverNodeF produced Glucose-1-P

Fig. 7. Forming Glucose-1-P

The next reaction we tested checked the formation of UDP-glucose. UDP-glucose can get attached to glucose chains, which can be acted upon by glycogen synthesis. Glucose enters the cells by facilitated diffusion, and then the cell modifies glucose by phosphorylation.

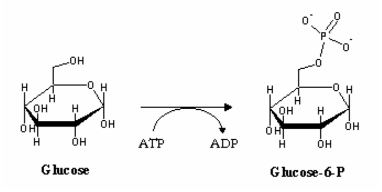


Fig. 8. Diagram for phosphorylation

Glucose-6-phosphate is used in the synthesis of glycogen. To do so, glucose-6- phosphate is first isomerized to glucose-1-fosfato by the enzyme fosfoglucomutase.

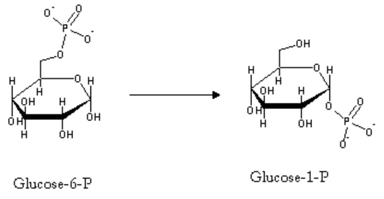


Fig. 9. Isomerization

UDP-glucose has the ability to attach the glucose part of itself to glucose chains. This new chain can be acted upon during glycogen synthesis.

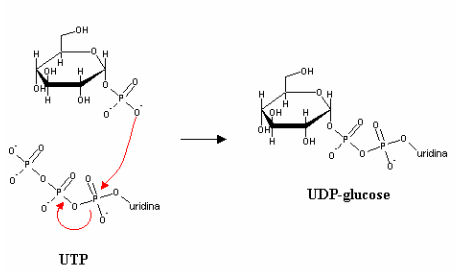


Fig. 10. Forming of UDP-Glucose

IV. Synapsin and Vesicles Interaction in a Neuron

Synapsin is a neuron-specific phosphoprotein that binds to small synaptic vesicles and actin filaments in a phosphorylation-dependent pattern. Microscopic models have demonstrated that synapsin inhibits neurotransmitter release either by forming a cage around synaptic vesicles (cage model) or by anchoring them to the F-actin cytoskeleton of the nerve terminal (crosslinking model) [6], [7].

We have modeled the reserve pool of synaptic vesicles in a presynaptic nerve terminal, predicting the number of synaptic vesicles released from the reserve pool as a function of time under the influence of action potentials at differing frequencies. Time series amounts for the components were obtained using different methods: deterministic (using ordinary differential equations), stochastic using Gillespie's stochastic simulation algorithm and stochastic using rule-based methods (the rules defined by Cell-DEVS) [7].

We modeled the molecular interactions of *synapsin* (**S**) with *vesicles* (**V**) which occur inside a nerve cell [6]. The model describes the behavior of synapsin movements until reaching a vesicle and binding to it. Once binding has occurred, depending on *offrate* V and S can break their bindings and separate. The *onrate* and *offrate* describe how often bindings occur or break then after. The following formula describes the nature of the reaction:

S + V ? SV

From the above formula, the left hand side of the equation demonstrates the binding scenario where *synapsin* and *vesicles* bind at a rate specified by *onrate*, while the right hand side of the equation illustrates the bind-break scenario where a *synapsin-vesicle* at an *offrate* which is always smaller than *onrate*, breaks apart and again *synapsin* and *vesicles* get released. Then, *synapsin* and *vesicles* can again perform binding and break apart. This equation shows an on-going process of "binding" and "breaking apart" which depends on *offrate/onrate*. The larger the *offrate* is, the more bindings get broken apart. Similarly, the larger the *onrate* is, the more V-S binds are produced.

Three different scenarios are modeled: 1) V is stationary (with a fixed position on cell space), and S is mobile, 2) V is mobile and S is stationary, and 3) V and S are both mobile (leads to maximum number of total movements and therefore bindings). Binding patterns are in such a way that each S can bind to more than one V, and V can bind to more than one S. Examples of such binding would be:

```
  S-V-S-V
  |   |
V-S-V-S-V-S-V
```

Each cell space in Cell_DEVS is used to represent one S or V. The neighboing pattern of V and S is in such a way that they can be adjacent cells or diagonal cells, as shown in the following Figure (Gray – Red Cell=S, Black - BlueCell=V).

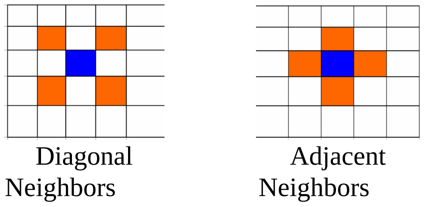


Fig. 11. Neighborhood definition

The model inclues 100 S molecules and 100 V in a 26x22 cell space. Mobile S or V change position to the right, to the left, up or down at random. According to the specifications, the *onrate* was set to 10, and the *offrate* to 0.1 (these are variable parameters that can take positive values). Therefore, in the model more the formation of S-V is favored compared to the splitting of S-V complexes. This is the reason behind not having all the red cells (S) being mobile and changing their direction. The *offrate* can be modified, so the larger it is the more mobile S will be observed. However, the *onrate* is kept constant which therefore results in having same number of S-V bonds at the end of any execution. The model presented in this report demonstrates the scenario of mobile S and stationary V. In the following code, we show an extract of the model definition in CD++.

```
rule : {round(uniform(11,14))} 100 { (0,0)=1  }
rule : {round(uniform(21,24))} 100 { (0,0)=2  }

rule : {round(uniform(31,34))} 100 {((0,0)=21 or
(0,0)=22 or (0,0)=23 or (0,0)=24) and ( ((-1,0)-10=1
or (-1,0)-10=2 or (-1,0)-10=3 or (-1,0)-10 =4 )
or ((1,0)-10=1 or (1,0)-10=2 or (1,0)-10=3
or (1,0)-10=4) or ((0,-1)-10=1 or (0,-1)-10=2
or (0,-1)-10=3 or (0,-1)-10=4) or ((0,1)-10=1
or (0,1)-10=2 or (0,1)-10=3 or (0,1)-10=4 )
or ((-1,1)-10=1 or (-1,1)-10=2 or (-1,1)-10=3
or (-1,1)-10=4) or ((1,-1)-10=1 or (1,-1)-10=2
or (1,-1)-10=3 or (1,-1)-10=4) or ((1,1)-10=1
or (1,1)-10=2 or (1,1)-10=3 or (1,1)-10=4)
or ((-1,-1)-10=1 or (-1,-1)-10=2 or (-1,-1)-10=3
or (-1,-1)-10=4)) and random > 0.10}
...
%moving up
rule : 91 100 {(0,0)=21 and (-1,0)=0 }
rule : {round(uniform(21,24))} 0 {(0,0)=0 and
1,0)=91 }
rule : 00 0 {(0,0)=91}
...
```

```
%release 0.1 of the S (the offrate is 0.1)
rule : {round(uniform(21,24))} 100 {((0,0)=33 or
(0,0)=32 or (0,0)=31 or (0,0)=34) and random < 0.10}
```

Fig. 12 (a) shows the grid at the initial point, case where S and V have not yet interacted to form any bonds. Then Fig. 12 (b) shows how bonds are formed and the corresponding cells change their values to represent the binding. As illustrated on Fig. 12, the bold boxes show bindings between synapsin (31-34) and vesicle (41-44). The first illustration (Fig. 12.a) represents the initial scenario where synapsins (21-24) and vesicles (11-14) are free and have not yet bonded. Once synapsins walk toward vesicles, the values of the corresponding cells change to 31-34 (bonded synapsins) and 41-44 (bonded vesicles). It is shown that vesicles can be surrounded by more than one synapsin, and some synapsins have can bind to only one vesicle at any time.

```
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
    +----------------------------------------------------------------------------------------+
   0|          13      24      14  24      23  23      11      13  21      22      12        |
   1|              12      23      13  22      14      12      22      12  23      13        |
   2|                                                                                        |
   3|          12      12  11  23      14      13  21      22  23      22  23      11        |
   4|              23      11              14  11  22      22  23      13  14  14  23        |
   5|                                                                                        |
   6|          12  22      12  23      23      13  22      23      13  13  11  22            |
   7|              13  21      24  23      12      12  22      14  24      13  22            |
   8|          12      22      13  22      23  23      11  13      14  22      13  24        |
   9|                                                                                        |
  10|              12      13      12      13  23      23      11  23      24      11        |
  11|              12  22      24      22      12  13  13  22      24  22      13            |
  12|          13      22      13  24      12  14  24      24      12      22      12        |
  13|          24      13      24      12  13      22      12  24      12  21      14        |
  14|                                                                                        |
  15|          13  22      21  22      12      14      12      13  24      23      13        |
  16|          12  23  11  23  21      22      13  21      14      21      12      13        |
  17|              12      13      12      12  22      21      12  21      22      12        |
  18|                                                                                        |
  19|          14      22      13      22      14      11  23      13  24      13            |
  20|                                                                                        |
  21|          14  23      23  24      11  23      22      23      13      12      13        |
  22|                                  21  12                              24                |
  23|                                                      22  13                            |
  24|          12      24      13  22      21  21      12  14      12  22      12  23        |
  25|              12  21      22      23      12  14  12  21      22  23      14            |
    +----------------------------------------------------------------------------------------+
```

```
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
    +----------------------------------------------------------------------------------------+
   0|          13  32          41  22  32  34  31      44      42  34      31      12        |
   1|              12      32      44  21      42      12      23      42  32      13        |
   2|                                                                                        |
   3|          41      42  41  32      14      42  32      22          33  33      44        |
   4|              32      43              14  41  34          23      13  43  14  32        |
   5|                                                          33                            |
   6|          42  33      44  32      31      43  32      21      44  42  42  34            |
   7|              42  31      34  32      41      42  23      42  32      44  31            |
   8|          12      31      41  24      34  33      41  13      43      34  42  31        |
   9|                                                                                        |
  10|              42      44      42      41  23      31      43  32      33      11        |
  11|              41  32      32      33      44  44  13  33      33  33      43            |
  12|          42      31      41  32      42  43  33      32      12      22      12        |
  13|          22      41      32      42  13      34      42  32      44  31      14        |
  14|                                                                                        |
  15|          41  34      32      33  44      42      42      42  34      32      13        |
  16|          44  31  42  31  31      31      42  31      44      33      41      13        |
  17|              12      43      43      42  32      31      44  33              12        |
  18|                                                                          24            |
  19|          14          22  13          32  14      42  31      13  33      13            |
  20|                                                                                        |
  21|          43  33      22          44  31              22      13      44      13        |
  22|                          23      32  42              32              33                |
  23|                                                      33  41                            |
  24|          12      33      43  31      33  33      42  14      42  23      44  34        |
  25|              42  32      34              41  42  42  31      34  21      14            |
    +----------------------------------------------------------------------------------------+
```

Fig. 12. (a) V and S before binding at Time::00:100 (bold boxes represent examples of binding structures (b)V and S after binding at Time: 00:300

Several initial parameters were tested in order to see the running process of the neuron with different *offrate*. The case presented in the following figure shows an *offrate* of 0.1.

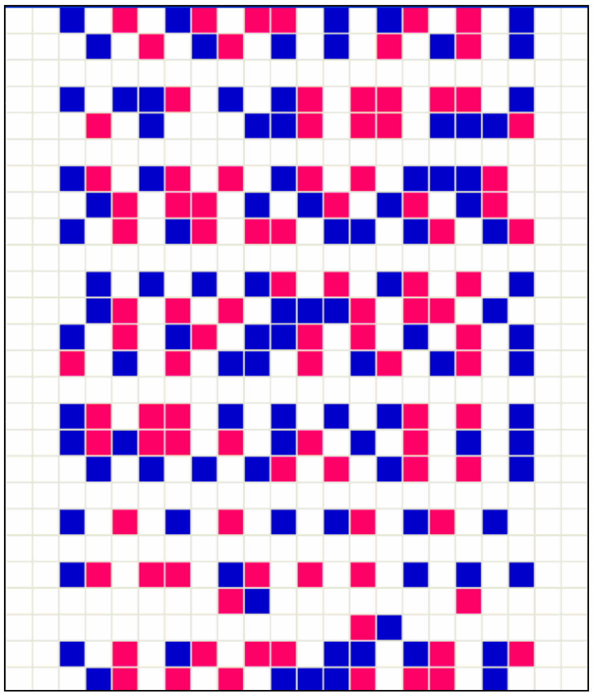
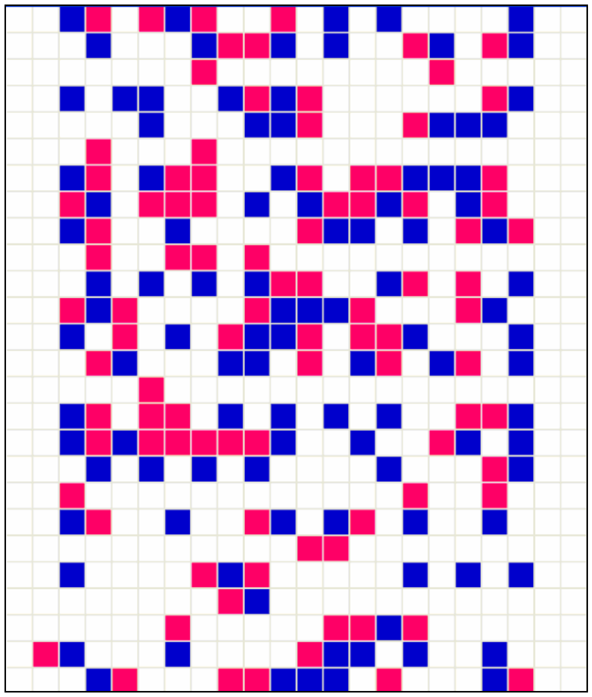



Fig. 13. Model Execution Results: initial values; final execution

The final execution results on Fig. 13 present a stable image of synapsin-vesicles bindings where single/ double/multiple bindings had occurred within the neuron.

V. CONCLUSION

In the last few decades, computer simulation has become an integral part in both the basic and applied fields of biological research. The hierarchical and discrete-event capabilities of DEVS makes it an ideal method for modeling biological events. As illustrated herein, CD++ can be used to model and simulate biological pathways using a systematic method with models that consist of sets of lower-level interactions. The approach also enables reuse of simulation components and allows seamless integration of these components into more complex simulation models. This capability would allow for sharing and inter-operability of model components thus aiding the development of a wide variety of model implementations for the broader medical research field.

We defined three types of models: chemical reaction in mitochondria, a model of liver lobules, and synapsin-vesicles interactions of a nerve cell, showing how to define models of cells in different organs. This is an initial step on the creation of a model base that could contribute in helping specialists in the biomedical field make a better assessment of diseases by studying detailed reactions at the level of the cell. Currently, we are working on:

1. Integrating a graphical interface to the model to make it much more visual. This would provide doctors with a better understanding of the development of specific liver conditions in their patients.
2. Integrate more detailed substance transformations in the model of the liver lobule, to make the model more biologically accurate. This will increase the validity of the model.
3. Make extensions to the design so it would be capable to interact with other similar organs simulations.

REFERENCES

[1] Djafarzadeh, R.; Mussivand, T.; Wainer, G. “Modeling energy pathways in cells”. In *Proceedings of the 2005 DEVS Integrative M&S Symposium, Spring Simulation Conference.* San Diego, CA. U.S.A. 2005.

[2] Zeigler, B.; Kim, T.; Praehofer, H. "Theory of Modeling and Simulation". Academic Press. 2000.

[3] Wainer, G.; Giambiasi, N. "Timed Cell-DEVS: modeling and simulation of cell spaces ". In "Discrete Event Modeling & Simulation: Enabling Future Technologies", Springer-Verlag. 2001.

[4] WainerG.. “CD++: a toolkit to define discrete-event models”. Software, Practice and Experience. Vol. 32, No.3. pp. 1261-1306. November 2002.

[5] Khan, A.; Venhola, W.; Wainer, G.; Jemtrud, M. “Advanced DEVS model visualization”. Proceedings of IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation 2005. Paris, France. 2005.

[6] Benfenati, F.; Valtorta, F.; Greengard, P. "Computer Modelling of Synapsin 1 Binding to Synaptic Vesicles and F-actin: Implications for Regulation of Neurotransmitter Release", *Proc Natl Acad Sci U S A* 88, no. 2:575-579. October, 1990

[7] Bain, R.; Jafer, S.; Dumontier, M. ; Wainer, G.; Cheetham., J. "Vesicle, Synapsin And Actin Concentration Time Series Modelling At The Presynaptic Nerve Terminal". presented at the Ottawa Institute of Systems Biology Conference, Ottawa, Canada. 2006.

[8] Canadian liver foundation, website available at: http://ww.liver.ca/english/yourliver/your_liver.html, 2005.

[9] C. Anthony Hunt, Glen E.P. Ropella, Michael S. Roberts, and Li Yan, Biomimetic In Silico Devices, in V. Danos and V. Schachter (Eds.): Computational Methods in Systems Biology 2004, Lecture Notes in Bioinformatics 3082: 35 - 43, 2005.