MODELLING TUMOR-IMMUNE SYSTEMS WITH CELL-DEVS

Rhys Goldstein
Gabriel Wainer
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada
Email: {rhys, gwainer} @sce.carleton.ca

KEYWORDS

DEVS formalism, Cell-DEVS, cellular automata, tumor growth, immune cells

ABSTRACT

Cell-DEVS, an extension of the DEVS formalism, is used to model tumor-immune systems that involve growing tumors interacting with immune cells. Tumors can be regarded as a core of necrotic cells, surrounded by dormant cells, surrounded in turn by proliferative cells. The growth of a tumor is effected by the division of its proliferative cells, but inhibited by nearby immune cells. We present a Cell-DEVS model representing a tumor-immune system. The use of Cell-DEVS is advantageous as it facilitates the formal specification and reuse of cellular models. The Cell-DEVS model was implemented and tested using the CD++ toolkit. Simulation results indicate that, in a qualitative sense, the desired behaviour of tumors and immune cells was captured.

INTRODUCTION

Tumor growth is a complex process; more complicated still when the response of an immune system to the tumor is considered as well. The combination of a growing tumor and the immune response will be referred to as a "tumor-immune system". Animal experiments have revealed many interesting aspects of tumor-immune systems. It has been observed, for example, that while inoculation with a certain number of tumor cells results in the rejection of the tumor by the immune system, a smaller number of tumor cells may lead to progressive tumor growth (Takayanagi et al., 2006). The modelling of tumor-immune systems is of interest as a means of investigating tumor growth and immunity.

Deterministic models based on differential equations have been designed for tumor-immune systems, but are regarded as inadequate for processes of considerable complexity. They are often restricted to 1-dimensional or radially symmetric tumor growth. Such limitations have been overcome by the adoption of stochastic cellular automata (Mallet and Pillis, 2006). In one case, a 3-dimensional cellular model was developed to simulate tumors growing over three orders of magnitude in radius (Kansal et al., 2002). Another cellular automaton has modelled the release of proteins by immune cells in response to a tumor (Takayanagi et al., 2006).

Facilitating the formal specification and reuse of cellular models, Cell-DEVS (Wainer and Giambiasi, 2002) is a compelling tool for the modelling and simulation of biological systems (Wainer et al., 2007). This paper demonstrates how the formalism can be applied, using a tumor-immune system model as an example. One such cellular automaton was chosen for this purpose (Huricha and Ruanxiaogang, 2003), which defines a 2-dimensional cell space through which immune cells wander in search of a tumor. Tumors generally form a core of necrotic cells, surrounded by a ring of dormant cells, surrounded in turn by a ring of proliferative cells. The tumor grows as the outermost proliferative cells divide. A growing tumor may overwhelm the immune system, or may be defeated by the immune cells. The Cell-DEVS model was implemented and tested using the CD++ toolkit (Wainer, 2002). Simulation results exhibited the desired qualitative behavior of tumors and immune cells.

After providing a brief overview of Cell-DEVS and a more detailed description of the model we based our work on, we describe the formal specification of a tumor-immune system model. Following this is a description of the CD++ implementation of the model. Several test results that demonstrate the behaviour of the model under simulation, showing how the Cell-DEVS formalism may aid in the modelling and simulation of biological systems such as these.

Proceedings 22nd European Conference on Modelling and Simulation ©ECMS Loucas S. Louca, Yiorgos Chrysanthou, Zuzana Oplatková, Khalid Al-Begain (Editors)
ISBN: 978-0-9553018-5-8 / ISBN: 978-0-9553018-6-5 (CD)

CELL-DEVS AND CD++

The DEVS formalism (Zeigler et al., 2000) provides a framework for the construction of hierarchical modular models, allowing for model reuse, and reducing development and testing times. Basic models, called "atomic models", are specified through transition functions. Multiple DEVS models can be integrated together to form hierarchical structural models, called "coupled models".

The Cell-DEVS formalism was defined as an extension to cellular automata (Wolfram, 2002) combined with DEVS. In Cell-DEVS, each cell in a cellular model is seen as a DEVS atomic model, and a procedure for coupling cells is defined based on the neighborhood relationship. Only the active cells in the cell space are triggered, independently from any activation period. Each cell of a Cell-DEVS model holds state variables and a local computing function, which updates the cell state by using its present state and its neighborhood.

A timed DEVS cell atomic model, associated with each cell in a cellular model, is specified as follows:

$$TDC = \langle X, Y, S, N, delay, d, \delta_{ext}, \delta_{int}, \tau, \lambda, D\rangle$$

The variable X defines the external inputs, Y defines the external outputs, and S is the cell state definition. The variable N represents the set of relative coordinates of each cell in the neighborhood. The $delay$ is the kind of delay used for the cell, and d is its duration. A $transport$ delay can be associated with each cell, which defers the outputs for the cell. A state change will be discarded if it is not steady during an $inertial$ delay. The local computing function τ is used to evaluate the future state of the cell. The remaining functions drive the cell's behavior: δ_{int} for internal transitions, δ_{ext} for external transitions, λ for outputs, and D for the state's duration.

A Cell-DEVS coupled model, representing an entire cell space, is specified as follows:

$$GCC = \langle X_{list}, Y_{list}, X, Y, n, [t_1, \ldots, t_n], N, C, B, Z\rangle$$

Here, X_{list} and Y_{list} are input/output coupling lists, used to define the model's interface. X and Y represent the input/output events. The n value defines the number of dimensions of the cell space, and $[t_1, \ldots, t_n]$ is the number of cells in each dimension. N is the neighborhood set. The cell space is defined by C, together with B, the set of border cells, and Z, the translation function.

CD++ (Wainer, 2002) is a modelling tool that implements the DEVS and Cell-DEVS formalisms. DEVS atomic models are programmed in C++, while both DEVS coupled models and Cell-DEVS models can be defined using a built-in specification language. CD++ makes use of the independence between modelling and simulation provided by DEVS, and different simulation engines have been defined for the platform.

TUMOR-IMMUNE SYSTEM MODEL

A cellular automaton, describing the growth of tumors interacting with an immune system, was previously designed and tested (Huricha and Ruanxiaogang, 2003). This original model, upon which the new Cell-DEVS version is based, consists of a 2-dimensional cell space. Each cell of the model represents a biological cell. The various types of cells are listed below.

- **Normal Cell**: a cell that is neither part of a tumor, nor a part of the immune system.
- **Immune Cell**: a cell that is part of the immune system. No distinction is made between the different types of immune cells that may respond to a tumor.
- **Proliferative Cell**: a tumor cell that divides, facilitating tumor growth.
- **Dormant Cell**: a tumor cell that was once proliferative, but no longer divides.
- **Necrotic Cell**: a tumor cell that was once dormant, but is now considered dead.

In a typical simulation, the initial cell space is composed primarily of normal cells. Dozens of immune cells are scattered throughout the cell space, and a single proliferative cell is situated in the center.

With the simulation underway, the single proliferative cell divides, converting adjacent normal cells into other proliferative cells. These converted cells then divide in turn, effecting the growth of the tumor. The innermost proliferative cells gradually become dormant cells, and the innermost dormant cells later become necrotic. The necrotic, dormant, and proliferative cells are expected to form three roughly concentric circles.

The immune cells wander randomly through the space of normal cells that surround the tumor. When an immune cell encounters a tumor cell, one of the two cells is destroyed. Such interactions may eventually lead to the death of the entire tumor.

CELL-DEVS MODEL SPECIFICATION

The tuple below is a Cell-DEVS coupled model that describes a tumor-immune system as a cell space.

$$\begin{aligned} TIS(t_1, t_2, d, p_{resisting}, p_{moving}, p_{curing}, &\\ p_{dividing}, p_{dying}) &\\ = \langle X_{list}, Y_{list}, X, Y, n, [t_1, t_2], N, C, B, Z \rangle \end{aligned}$$

The parameters t_1 and t_2 are the dimensions of the cell space, and d is the delay for each cell. The parameters $p_{resisting}$, p_{moving}, and p_{curing} represent, respectively, the probabilities with which an immune cell is added to the model, moves from its present location, and cures a proliferative cell. Higher values of these probabilities favour the immune system. The parameter $p_{dividing}$ represents the probability with which a proliferative cell divides, and p_{dying} is the probability that a dormant cell becomes necrotic.

In our case, the model has neither inputs nor outputs, and is always 2-dimensional.

$$X_{list} = Y_{list} = X = Y = \varnothing$$

$$n = 2$$

The model uses a 5-by-5 cell extended Moore neighborhood, defined below.

$$\begin{aligned} N = \{[i, j] \mid i \in \{-2, -1, 0, 1, 2\} &\\ \wedge\ j \in \{-2, -1, 0, 1, 2\}\} \end{aligned}$$

The borders of the model are wrapped. This is convenient in that it ensures that immune cells incident on the border are not removed from the model completely. The disadvantage is that the model losses validity as the growing tumor nears the boundary.

$$B = \varnothing$$

The translation function Z is defined by the Cell-DEVS formalism, while each timed DEVS cell model is defined as follows.

$$C([i, j]) = \langle X, Y, S, N, delay, d, \delta_{ext}, \delta_{int}, \tau, \lambda, D \rangle$$

The input and output event sets X and Y are defined by the Cell-DEVS formalism.

The state associated with each individual cell consists of both a type and a direction. The type indicates whether the represented biological cell is a normal, immune, proliferative, dormant, or necrotic cell. The direction, which is either $[0, 0]$ or the coordinates of an adjacent cell, indicates either the way immune cells are moving or the way proliferative cells are dividing.

$$\begin{aligned} S = \{[\mathit{type}, \mathit{direction}] \mid \mathit{type} \in \mathit{types} &\\ \wedge\ \mathit{direction} \in \mathit{directions}\} \end{aligned}$$

$$\begin{aligned} \mathit{types} = \{normal, immune, prolif, &\\ dormant, necro\} \end{aligned}$$

$$\begin{aligned} \mathit{directions} = \{[i, j] \mid i \in \{-1, 0, 1\} &\\ \wedge\ j \in \{-1, 0, 1\}\} \end{aligned}$$

The model uses transport delays.

$$delay = transport$$

The variables δ_{int}, δ_{ext}, λ, and D are defined by the Cell-DEVS formalism. The following section describes the local computing function τ.

LOCAL COMPUTING FUNCTION

The term "state" refers to the value associated with each cell in the cell space. A state must be a value in the set S. A "state function" is a function that takes a pair of relative coordinates as an input, and results in the state of the cell associated with those coordinates. The state function s therefore maps values in the set N to values in the set S. The example equation below indicates that there is an dormant cell at a relative position of $[-2, 1]$.

$$s([-2, 1]) = [dormant, [0, 0]]$$

In any scope that contains the state function s, the functions type and $\mathit{direction}$ will also be available. These functions provide the types and directions associated with neighboring cells. They are defined implicitly below.

$$s([i, j]) = [\mathit{type}([i, j]), \mathit{direction}([i, j])]$$

The local computing function τ maps the state function s, which carries the present states of neighboring cells, onto the new state of the cell with relative coordinates $[0, 0]$. It is defined below, using a conditional expression in which conditions are placed on the left of arrows that point to the corresponding results.

$$\tau(s) = \begin{pmatrix} \mathit{type}([0,0]) = normal & \rightarrow \tau_{normal}(s) \\ \mathit{type}([0,0]) = immune & \rightarrow \tau_{immune}(s) \\ \mathit{type}([0,0]) = prolif & \rightarrow \tau_{prolif}(s) \\ \mathit{type}([0,0]) = dormant & \rightarrow \tau_{dormant}(s) \\ \mathit{type}([0,0]) = necro & \rightarrow \tau_{necro}(s) \end{pmatrix}$$

The local computing function depends on the functions τ_{normal}, τ_{immune}, τ_{prolif}, $\tau_{dormant}$, and τ_{necro}. Each of these were formally specified, and the definition of τ_{normal} is presented below as an example. Several other sets and functions are described first.

Two random value functions are used. The function $rand$ results in a value randomly selected from the argument set. The expression $rand(\{1, 2, 3, 4\})$, for example, would have a 25% chance of resulting in 2. The function $rand_U$ takes no arguments, and results in a real number randomly chosen between 0 and 1.

The variable adj represents the set of relative coordinates of all adjacent cells. In this specification, "adjacent cells" are the eight cells surrounding the one in question.

$$adj = directions - \{[0, 0]\}$$

The function $count_{near}$ results in the number of cells, with coordinates contained in the argument M, that are of type $type_{near}$. Among other things, it is used to determine whether there is a proliferative cell adjacent to an immune cell.

$$\begin{aligned} &count_{near}(s, type_{near}, M) \\ &\quad = \sum_{[i,j] \in M} \begin{pmatrix} type([i, j]) = type_{near} & \rightarrow 1 \\ type([i, j]) \neq type_{near} & \rightarrow 0 \end{pmatrix} \end{aligned}$$

The function $count_{incoming}$ results in the number of cells, with coordinates adjacent to $[k, l]$, that are of type $type_{incoming}$. In order to be counted, those adjacent cells must also be have a direction that points to $[k, l]$. Among other things, this function is used to prevent collisions between immune cells.

$$\begin{aligned} &count_{incoming}(s, type_{incoming}, [k, l]) \\ &\quad = \sum_{[i,j] \in L} \begin{pmatrix} incoming([i, j]) & \rightarrow 1 \\ \neg incoming([i, j]) & \rightarrow 0 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} &incoming([i, j]) \\ &\quad = (s([i, j]) = [type_{incoming}, [k - i, l - j]]) \end{aligned}$$

$$L = \{[k - i, l - j] \mid [i, j] \in adj\}$$

Demonstrating the $count_{incoming}$ function, Figure 1 shows a possible arrangement of types and directions of each cell in a neighborhood.

normal $[0, 0]$	normal $[0, 0]$	immune $[1, -1]$	normal $[0, 0]$	normal $[0, 0]$
normal $[0, 0]$	normal $[0, 0]$	prolif $[1, 0]$	**normal** **[0, 0]**	immune $[-1, 0]$
normal $[0, 0]$	normal $[0, 0]$	prolif $[0, -1]$	immune $[-1, -1]$	normal $[0, 0]$
normal $[0, 0]$	normal $[0, 0]$	normal $[0, 0]$	normal $[0, 0]$	normal $[0, 0]$
normal $[0, 0]$	normal $[0, 0]$	normal $[0, 0]$	normal $[0, 0]$	normal $[0, 0]$

Figure 1: A hypothetical neighborhood of types and directions. Cell $[0, 0]$, at the center, is a proliferative cell pointing downwards. Cell $[1, 1]$ is indicated by boldface text.

There are three immune cells adjacent to the cell $[1, 1]$, The immune cell on the upper left and the one to the right are both approaching that cell. The one below, however, is pointing elsewhere. The cell $[1, 1]$ therefore has 2 incoming immune cells.

$$count_{incoming}(s, immune, [1, 1]) = 2$$

As specified above, the function τ_{normal} is used to evaluate the local computing function for a cell that is currently normal. A normal cell is considered to be "receiving" an immune cell if there is exactly one incoming immune cell. In this event, the cell becomes an immune cell that is "preparing" its next action. If a normal cell is not "receiving", then it is "praying" that is will not be overtaken by the tumor.

$$\begin{aligned} &\tau_{normal}(s) \\ &\quad = \begin{pmatrix} receiving(s) & \rightarrow \tau_{preparing}(s) \\ \neg receiving(s) & \rightarrow \tau_{praying}(s) \end{pmatrix} \end{aligned}$$

$$\begin{aligned} &receiving(s) \\ &\quad = (count_{incoming}(s, immune, [0, 0]) = 1) \end{aligned}$$

If a "praying" cell has any incoming proliferative cells, the previously normal cell is considered to be "mutating". It then becomes a proliferative cell itself, "plotting" its next division. A "praying" cell that is not "mutating" is "waiting".

$$\begin{aligned} &\tau_{praying}(s) \\ &\quad = \begin{pmatrix} mutating(s) & \rightarrow \tau_{plotting}(s) \\ \neg mutating(s) & \rightarrow \tau_{waiting}(s) \end{pmatrix} \end{aligned}$$

$$\begin{aligned} &mutating(s) \\ &\quad = (count_{incoming}(s, prolif, [0, 0]) > 0) \end{aligned}$$

An option to replenish immune cells was added to the Cell-DEVS version of the model. A "waiting" cell may spontaneously become an immune cell. This happens with a probability of $p_{resisting}$, but only if the cell is surrounded by normal cells. Otherwise, the "waiting" cell remains a normal cell.

$$\tau_{praying}(s) = \left(\begin{array}{ll} resisting(s) & \rightarrow \tau_{preparing}(s) \\ \neg resisting(s) & \rightarrow [normal, [0,0]] \end{array} \right)$$

$$resisting(s) = ((count_{near}(s, normal, adj) = 8) \wedge (rand_U() < p_{resisting}))$$

Note that if $p_{resisting}$ is zero, immune cells will never spontaneously appear. Instead they will die off until there are none left, or until there are no proliferative cells left that can kill them.

The remainder of the local computing function was also specified in this manner. The formulae above, for example, indicate situations in which a normal cell receives an incoming immune cell. Similar logic is used to determine when an immune cell vacates its present location, leaving a normal cell in its place.

In the original cellular automaton, the radius of a tumor was restricted by a constant parameter. This parameter was omitted from the Cell-DEVS version to allow the interaction of the tumor and immune cells to determine the extent to which the tumor grows. In the Cell-DEVS model, immune cells only interact with proliferative cells. If they manage to destroy all proliferative cells, a cluster of necrotic cells may remain. This scenario is interpreted as an immune system victory. The alternative possibility is that the tumor reaches the boundary of the cell space. This result is interpreted as a tumor victory.

CD++ IMPLEMENTATION

The specified model was implemented using the CD++ toolkit, a task that involved re-writing the mathematical formulae in the built-in specification language. The $\tau_{wandering}$ formula, which determines the direction of an immune cell, provides an example of this translation.

$$\tau_{wandering}(s) = \left(\begin{array}{ll} moving(s) & \rightarrow [immune, rand(adj)] \\ \neg moving(s) & \rightarrow [immune, [0,0]] \end{array} \right)$$

$$moving(s) = (rand_U() < p_{moving})$$

Below is a CD++ expression corresponding to the right-hand side of the definition of $\tau_{wandering}$.

```
if(
  ((uniform(0,1)) < #Macro(p_moving)),
  1 + ((trunc(uniform(0,8))+1)*0.0625),
  1)
```

The condition $moving(s)$ is captured by the sub-expression on the second line. If this condition is true, the value of the entire expression is taken from the third line. The integer 1 indicates the $immune$ cell type, while the random multiple of 0.0625 represents the random direction. If $moving(s)$ is false, a stationary immune cell results from the fourth line.

For testing purposes, a parameter was introduced to approximate the initial density of immune cells. Each cell was given a probability $p_{initial}$ of starting as an immune cell instead of a normal cell. This feature was defined in CD++ by the rule below. The `if` expression chooses between the immune cell type with value 1, and the normal cell type with value 0. The next line specifies the duration d of the delay. As indicated by the last line, the transition takes effect only at the beginning of a simulation, when all cells have been initialized with the value -1.

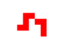
```
rule :
  { if(
      uniform(0, 1) < #Macro(p_initial),
      1,
      0) }
  #Macro(d)
  { (0,0) = -1 }
```

SIMULATION RESULTS

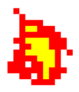
The four simulation results presented below demonstrate the uninhibited growth of a tumor, the movement of immune cells, the victory of a tumor over the immune system, and the victory of the immune system over a tumor. All tests were performed on cell spaces of 41-by-41 cells.

The first test investigates the growth of a tumor in the absence of immune cells. As shown in Figure 2, the simulation exhibited a growing tumor roughly shaped as three concentric circles.

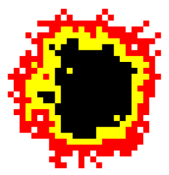
The images in Figure 3 show the results of a simulation of immune cell movement. A p_{curing} value of 1 ensured that immune cells were never killed by proliferative cells, while a $p_{resisting}$ value of 0 ensured that immune were never added to the cell space. As there were 21 immune cells at the beginning and end of the simulation, the test served to partially verify that the issue of collision detection was adequately addressed.



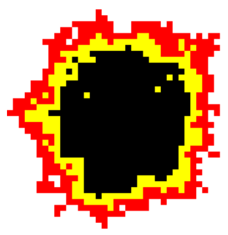
(a) $time = 11$



(b) $time = 23$

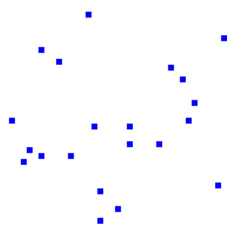


(c) $time = 49$

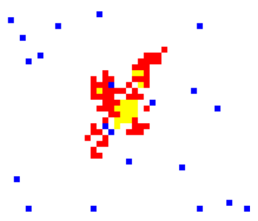


(d) $time = 64$

Figure 2: A simulation of tumor growth uninhibited by the immune system. In (a), the tumor consists of a small number of proliferative cells. In (b), the tumor has developed an inner cluster of dormant cells. These dormant cells, as shown in (c) and (d), later become an inner ring around a core of necrotic cells. The parameters were as follows: $p_{initial} = p_{resisting} = 0$, $p_{dividing} = p_{dying} = 0.5$.



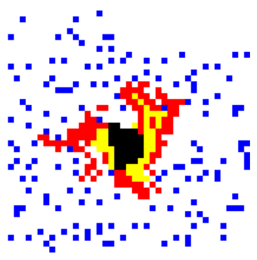
(a) $time = 1$



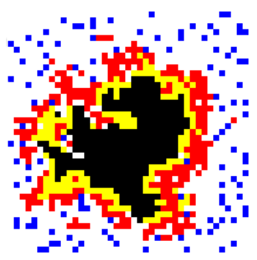
(b) $time = 100$

Figure 3: A simulation of immune cell movement. The image (a) shows the initial distribution of immune cells, and (b) shows a cell space with the same number of immune cells after 100 time units. These results were obtained with the following parameters: $p_{initial} = 0.01$, $p_{resisting} = 0$, $p_{moving} = p_{curing} = 1$, $p_{dividing} = 0.2$, $p_{dying} = 0$.

The final two tests demonstrate the interaction between growing tumors and immune cells. Figure 4 shows the results of a test in which the immune cells were defeated by a growing tumor. In the simulation presented in Figure 5, the parameters were changed such that the tumor was defeated by the immune system.

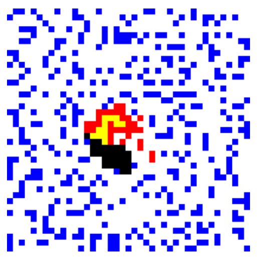


(a) $time = 38$

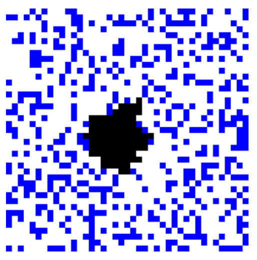


(b) $time = 64$

Figure 4: A simulation in which a tumor overwhelms the immune system. In (a), the immune cells have cleared away a large section of proliferative cells on the upper side of the tumor. Later, in (b) the tumor is shown having regained this region. It thereafter proceeded to expand towards the boundaries of the cell space. The parameters for this test were chosen to favour the tumor: $p_{initial} = 0$, $p_{resisting} = 0.01$, $p_{moving} = 0.5$, $p_{curing} = 0.8$, $p_{dividing} = 0.6$, $p_{dying} = 0.6$.



(a) $time = 65$



(b) $time = 150$

Figure 5: A simulation in which a tumor is defeated by immune cells. In (a), the proliferative cells were restricted to a single side of the otherwise dead tumor. The mass of necrotic cells increased roughly four times thereafter, but as shown in (b), the last proliferative cell was eventually destroyed. The parameters for this test were as follows: $p_{initial} = 0$, $p_{resisting} = 0.03$, $p_{moving} = 0.4$, $p_{curing} = 0.8$, $p_{dividing} = 0.4$, $p_{dying} = 0.8$.

CONCLUSION

A tumor-immune system model was specified using the Cell-DEVS formalism and implemented with CD++. Simulation results indicated that the model captured the intended qualitative aspects of tumor growth and immune system response.

Although similar simulations have previously been developed without Cell-DEVS, the use of this formalism was advantageous in that it facilitated the complete formal specification of the model. It was not necessary to explicitly indicate the advancement of time, nor the application of the local computing function to each cell in the cell space. These rules were part of the simulation, not the model, and were therefore implicit in the formalism itself.

Another advantage of DEVS was not demonstrated in the paper, but is worth noting. Any two DEVS or Cell-DEVS models can be integrated by the specification of a link between the output of one model and the input of the other. Suppose, for example, that the Cell-DEVS model of the tumor-immune system was re-designed with the ability to accept immune cells as input values. Other DEVS models, perhaps representing blood vessels or lymph nodes, could be defined to output immune cells. All the models could be then be integrated, with the lymph nodes and blood vessels supplying immune cells to the site of a tumor.

Though inherently complex, biological systems can often be regarded as sets of interacting subsystems. The DEVS and Cell-DEVS formalisms offer a compelling approach to the modelling and simulation of such systems.

REFERENCES

Huricha, R. and Ruanxiaogang, X. (2003). A simple cellular automaton model for tumor-immunity system. *Proceedings. 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing.*

Kansal, A. R., Torquato, S., Harsh, G. R., Chiocca, E. A., and Deisboeck, T. S. (2002). Simulated Brain Tumor Growth Dynamics Using a Three-Dimensional Cellular Automaton. *Journal of Theoretical Biology*, 203(4):367–382.

Mallet, D. G. and Pillis, L. G. D. (2006). A cellular automaton model of tumor-immune system interactions. *Journal of Theoretical Biology*, 239:334–350.

Takayanagi, T., Kawamura, H., and Ohuchi, A. (2006). Cellular Automaton Model of a Tumor Tissue Consisting of Tumor Cells, Cytotoxic T Lymphocytes (CTLs), and Cytokine Produced by CTLs. *IPSJ Digital Courier*, 2:138–144.

Wainer, G. (2002). CD++: a toolkit to develop DEVS models. *Software, Practice and Experience*, 32(3):1261–1306.

Wainer, G. and Giambiasi, N. (2002). N-dimensional Cell-DEVS models. *Discrete Event Systems: Theory and Applications*, 12(1):135–157.

Wainer, G., Jafer, S., Al-Aubidy, B., Dias, A., Bain, R., Dumontier, M., and Cheetham, J. (2007). Advanced DEVS models with application to biomedicine. *Artificial Intelligence, Simulation and Planning, Buenos Aires, Argentina.*

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.

Zeigler, B., Kim, T., and Praehofer, H. (2000). *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press.

AUTHOR BIOGRAPHIES

Rhys K. Goldstein received a B.A.Sc. (2003) degree from the Engineering Physics Department at the University of British Columbia (Vancouver, BC, Canada). He then worked in the mining industry, developing data processing software and conducting geophysical surveys in Alaska, Tasmania, Cyprus, and other parts of the world. He is now working towards a M.A.Sc. in Biomedical Engineering from the Department of Systems and Computer Engineering at Carleton University (Ottawa, ON, Canada). His email is `rhys@sce.carleton.ca`.

Gabriel A. Wainer received the M.Sc. (1993) and Ph.D. degrees (1998, with highest honors) of the University of Buenos Aires, Argentina, and Université dAix-Marseille III, France. In July 2000, he joined the Department of Systems and Computer Engineering, Carleton University (Ottawa, ON, Canada), where he is now an Associate Professor. He has held positions at the Computer Science Department of the University of Buenos Aires, and visiting positions in numerous places, including the University of Arizona, LSIS (CNRS), University of Nice and INRIA Sophia-Antipolis (France). He is author of three books and over 160 research articles, and helped organizing over 70 conferences. He is Associate Editor of the Transactions of the SCS, and the International Journal of Simulation and Process Modeling. He was a member of the Board of Directors of the SCS, a chairman of the DEVS standardization study group (SISO). He is Director of the Ottawa Center of The McLeod Institute of Simulation Sciences and chair of the Ottawa M&SNet, and one of the investigators in the Carleton University Centre for Advanced Studies in Visualization and Simulation (V-Sim). His current research interests are related with modelling methodologies and tools, parallel/distributed simulation and real-time systems. His e-mail and web addresses are `gwainer@sce.carleton.ca` and `www.sce.carleton.ca/faculty/wainer`.