Cellular Modeling with Cell-DEVS: A Discrete-Event Cellular Automata Formalism

Gabriel A. Wainer

Department of Systems and Computer Engineering
Carleton University, Ottawa, ON, Canada
gwainer@sce.carleton.ca

Abstract. In recent years, grid-shaped cellular models have gained popularity to understand physical systems. Complex cell spaces can require large amounts of compute time, mainly due to its synchronous nature; the use of a discrete time base also constrains the precision of the model. The Cell-DEVS formalism was defined in order to deal with these issues. We give a brief introduction to the main characteristics of Cell-DEVS, and show how to use the method to model complex cell spaces. We present different examples of application, and show how to integrate cellular models with external data collection and visualization.

1 Introduction

In recent years, there has been a trend in studying natural and humanmade systems using advanced modeling and simulation techniques. These problems were traditionally modeled with differential equations, and standard numerical methods. New methods based on Cellular Automata (CA) have provided new ways to solve these problems [1]. CA are represented as a cell space (a regular n-dimensional lattice whose cells can take discrete values). The states in the space are updated according to a local rule in simultaneous and synchronously, in discrete time steps, as dictated by a local transition function using the cell state and a finite set of neighbors. When CA are used to study complex systems, the use of a discrete time base poses restrictions in performance and in the precision of the model. In [2, 3, 4] we showed how the Cell-DEVS formalism solves these problems by using the Discrete Events Systems Specification formalism (DEVS) [5]. The goal is to build discrete event cell spaces, improving their definition by making the timing specification more expressive.

The DEVS and Cell-DEVS formalisms were implemented in the CD++ environment [3, 6, 7] which has been was used successfully to develop different types of systems: biological (ecological models, heart tissue, ant foraging systems, fire spread, etc.), physical (diffusion, binary, solidification, excitable media, surface tension, etc.), artificial (robot trajectories, networking, traffic, etc.), and others [3, 7-11]. We have developed different kinds of simulation engines (centralized, parallel distributed and real-time), which were used to execute the same models [12, 13].

In the following sections we give an introduction to Cell-DEVS, and show how to model cell spaces in an asynchronous environment.

J. Wąs, G.C. Sirakoulis, and S. Bandini (Eds.): ACRI 2014, LNCS 8751, pp. 6–15, 2014.

2 Background

DEVS is a formalism for discrete-event dynamic systems. It defines a way of specifying models whose states change upon the reception of an input event or the expiration of a time delay. It also allows for hierarchical decomposition of the model by defining a way to couple existing models. A coupled model can be regarded, due to the closure property, as another DEVS model. This allows for hierarchical model construction. A model that is not constructed as a coupled model is known as an atomic model.

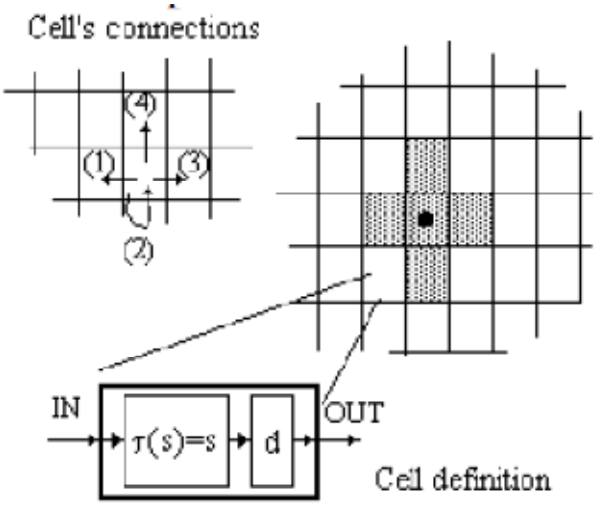

Fig. 1. Informal definition of a Cell-DEVS model [3]

Cell-DEVS is a formalism based on DEVS for cellular models. As in CA, a Cell-DEVS model is defined as a lattice of cells, each of which has a value and a local rule that defines how to obtain a new value based on the current state of the cell and the values its neighbors. Cell-DEVS defines a cell as a DEVS model and a cell space as a coupled model. It introduces a new flexible way of defining the timing for each cell (each cell defines its own update delay asynchronously from the others). A cell uses a set of input values to compute its future state, which is obtained by applying the local computation function **τ**. A delay function **d** is associated with each cell, deferring the output of the new state value. After the basic behavior for a cell is defined, the complete cell space will be constructed by building a coupled Cell-DEVS model.

The CD++ tool has been used to model numerous applications in different fields [3]. In http://cell-devs.sce.carleton.ca the reader will find a list of hundreds of models available for use, in different fields that range from basic chemistry and physics problems, up to advanced environmental and networking applications. Fig. 2 shows a number of different results obtained with the related tools.

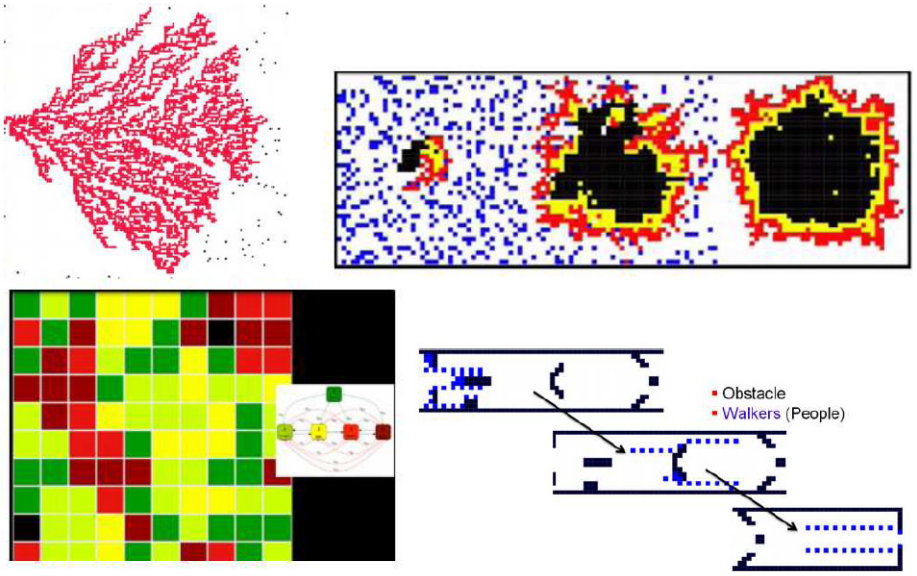

Fig. 2. a) Diffusion Limited Aggregation Model b) Tumor-Immune Model c) HIV Influence Model d) Pedestrian movement

The first example shows the Simulation results of a *Diffusion Limited Aggregation Model* [3], which begins with particles moving at random (in this case, from right to left), and an initial seed (in this case, on the left of the figure). The diffusing particles stick to and progressively enlarge an initial seed, growing in an irregular shape. This figure presents a case with concentration of 40%, showing fractal growth properties. The second example shows three different scenarios used for modeling tumor-immune systems [14]. The model shows how to model a core of necrotic cells, surrounded by a ring of dormant cells, surrounded in turn by a ring of proliferative cells. The immune cells attack the tumor in an attempt to stop it from growing. The next example focuses on the attitudes and influences of neighbors for intravenous drug user. Some people (green cells) affect their neighbors in a positive way (i.e., clinics and aid workers). The light green cells represent individuals that can be influenced by negative neighbors, or to remain drug free. The red cells represent HIV+ people who can be convinced to stop using. Finally the brown cells are users with HIV who will influence their neighbors negatively and will soon turn into a black square (someone who died of HIV). The last example shows a simulation scenario for a two dimensional pedestrian movement model in a corridor with obstacles.

The following sections are devoted to show how to define this kind of models and how to generate varied Simulation results based on the execution of the cellular models in CD++.

3 Basic Model: Human Circulatory System

The human circulation system transports oxygen and minerals through a network of arteries, veins and capillaries. The blood never comes into contact with any of the body's cells: the substances are diffused through the capillaries. In this section we present an example model of how oxygen is transported to the muscle cells using Cell-DEVS.

The first set of rules focus on the movement of blood cells. Since blood flow is a driven activity, a directional movement rule with fixed priority was defined. A blood cell will first attempt to move to the cell in front; if it is occupied it will then attempt to move into NE; if this cell is occupied it will try to move to the SE and if this space is full it will stop.

```
rule:{ if((0,-1)=1, if(((-1,-1)=6 or (1,-1)=6),
1, 2), 2)} 1 {((0,0)=0 and ((-1,-1)=7 or (-1,-
1)=6)) and ((0,-1)=1 or ((-1,-1)=1 and (-1,0)!=0
) or ((1,-1)=1 and (1,0)!=0 and (2,0)!=0) or ((-
  1,1)=2 and (0,1)=2) or ((1,1)>=8 and (0,1)>=8))
```

A blood cell will have two states, oxygenated or deoxygenated. An oxygenated cell will become deoxygenated when it passes by a deoxygenated muscle cell. They will "re-oxygenate" in the lung cells when they pass an oxygenated cell. Muscle cells become oxygenated when an oxygenated blood cell comes into contact with them. An oxygenated muscle cell will become deoxygenated after some time. The code snippet above shows the de-oxygenation of cells. Like the muscle cells, lung cells become oxygenated after spending a period of time deoxygenated, and deoxygenated when a deoxygenated blood cell comes into contact with it, as follows:

```
rule : { if((1,0) = 1, 6, 7) } 1 { (0,0) = 7 }
%Muscle Cell Becoming Oxygenated
rule : { if((-1,0) = 0.1, 7, 6) } 1 { (0,0)=6 }
%Muscle Cell Becoming De-Oxygenated
rule : { if((-1,0) = 0.1 , 4, 5) } 1 { (0,0)=5 }
%Lung Cell Becoming Oxygen Enriched
rule : { if((1,0) = 2, 5, 4) } 1 { (0,0)=4 }
%Lung Cell Becoming Carbon Dioxide Enriched
rule : {(0,0)} 1 {(0,0) != 0 }
```

The counter is responsible for "resetting" the corresponding lung or muscle cell to either oxygenated or deoxygenate.

```
% Rate of Consumption = (0,1)/(0.004) = 25 s
{ if((1,0)=6, ((0,0)+0.005),0)}1{(1,0)} = 6 }
% Lung Cells Replenishing Oxygen Supply
% Rate of Regeneration = (0.1)/0.004 = 25 s
  { if((1,0)=5, ((0,0)+.025 ), 0)} 1 {(1,0)=5 }
```

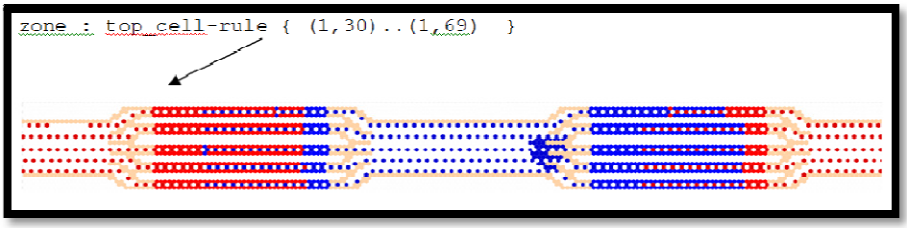

Fig. 3. Lung Model Simulation Results

4 Advanced Models: An Evacuation Cellular Model

The simulation of evacuation processes has been widely used to buildings, ships and the aviation industry. In [3, 5, 17] we presented various models in this area. The model introduced in this section represents people moving through a room trying to leave the building through an exit door.

The model is a 3D Cell-DEVS with two planes: one for the floor plan of the structure and the people, and the other for a Voronoi Diagram representing the orientation to the closest exit. The model characterizes a person's behavior: a normal person goes to the closest exit; a person in panic goes in opposite direction People move at different speeds; if the way is blocked, people can decide to move away and look for another way. The rules in Fig. 5 have two parts: the coupled model definition (size, neighborhood shape, initial conditions, etc.) and the local computing function. The first set of rules serves to define what path a person should follow using the orientation plane. The basic idea is to take the direction that decreases the potential of a cell. We have 8 rules to control the people's movement, one for each direction. A second set of rules governs the panic behavior.

In [15] this model was extended to conduct an integration between 3D visualization software and CD++, using as example the Society for Arts and Technology (SAT) building in downtown Montreal. Fig. 5 shows the results for this model. We can see the initial grid split into two layers. The left represents the walls, exits and initial positions of the people. Red cells represent people who want to escape. The black cells represent walls. On the right we can see the second layer which holds the distances to the exits. Fig. 5 considers a basic model with eight people without panic. In this simple scenario we could observe that they follow the second layer to exit the building without any complications. The building is almost empty (which is a normal condition for SAT). This evacuation is designed to give us a general idea of the exit directions people will follow, which will help us in developing the successive simulations.

```
type : cell dim : (49,27,2) delay : INERTIAL
defaultDelayTime : 1 border : wrapped
neighbors : (-1,-1,0) (-1,0,0) (-1,1,0)
neighbors : (0,-1,0) (0,0,0) (0,1,0)
...
[EvaRule]
% Rules to control the movement of individuals
rule : {#pos1+1} {1000/#pos0} {((0,0,0)>0 AND
#pos0 =0 ...
rule : {#pos1+3} {1000/#pos0} {((0,0,0)>0 AND
#pos0 =0 ...
rule : {#pos1+5} {1000/#pos0} {((0,0,0)>0 AND
#pos0 =0 ...
rule : {#pos1+7} {1000/#pos0} {((0,0,0)>0 AND
#pos0 =0 ...
rule : {#pos1+2} {1000/#pos0} {((0,0,0)>0 AND
  #pos0 =0 ...
```

Fig. 4. Evacuation rules as set in the CD++ Model file

5 Interfacing Cell-DEVS, External Input and Visualization

Several zoonotic diseases have emerged on the Asian landscape; Macaques have been affected by landscape changes caused by humans and these have increased the incidence of human interaction, potentially leading to bi-directional pathogen transmission to macaques.

The model in this section focuses on evaluating how landscape changes might influence pathogen transmission patterns [18]. Macaques can move to surrounding environment randomly, they may or may not carry pathogen, and can be infected by nearby neighbors. This model uses the landscape (the map contains only forest, water, and coastlines), temple (macaques live in their birth temple; females cannot cross the temple borders, while male macaques can), movement (at random into one of the 8 adjacent cells; collision avoidance is implemented), gender, and pathogen (each monkey may carry the pathogen; there are four phases of the transmission cycle: susceptible, latent, symptomatic, and acquired immunity).

The GRASS GIS was used to generate inputs for the model, combining information about the forests in Bali, and the water and coast map. The final map, shown in Fig. 6, is used to get the landscape values to be represented as cells and be used in the model simulation.

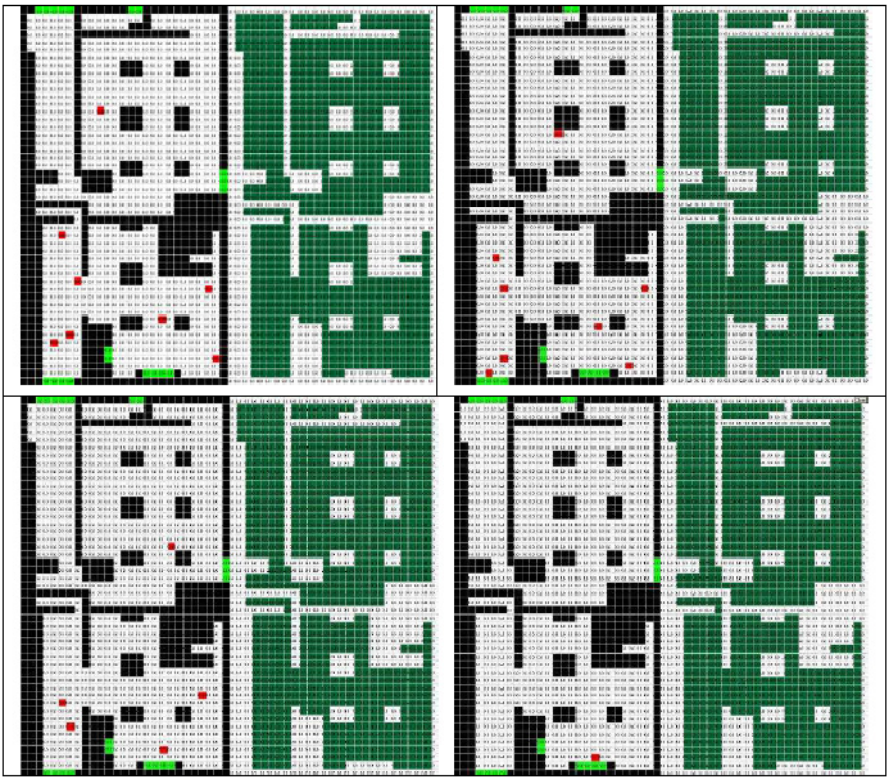

Fig. 5. a) SAT at time: 00:000–Initial placement of people; b) 00:834–First movement of people; c) 02:673–People proceed to the nearest exits; d) 13:015–Last person to leave the building

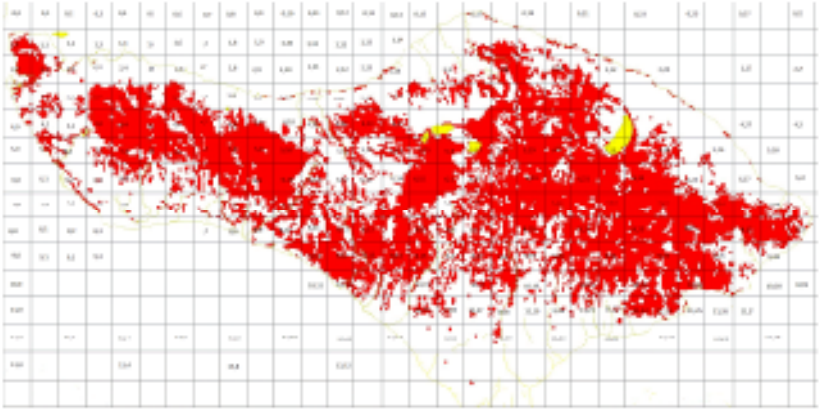

Fig. 6. The map divided into a cell space

Fig. 7 shows the model execution under 3 scenarios. The first one uses an initial monkey occupation of 10%, a river cross probability of 20%, a male ratio of 50%, and an initial pathogen infection ratio of 30%. The second test uses an occupation of 20%, a river cross probability of 20%, a male ratio of 40%, and an infection ratio of 50%. The third test uses a 30% occupation, a river cross probability of 50%, a male ratio of 70%, and an initial pathogen infection ratio of 80%.

Fig. 7. Three test cases comparison

A closer look at what is happening in the pathogen layer can be seen in Fig. 8. The cell marked by a circle is currently in latent infection. On the next step of the simulation, the cell changes stage 3 (symptomatic). The cells in the square show two monkeys with immunity. Since the rules state that when a cell in stage 4 has surrounding cells which are also in stage 4, it will not change phases. The two cells adjacent to the circled cell represent two monkeys that attempted to move to the same cell.

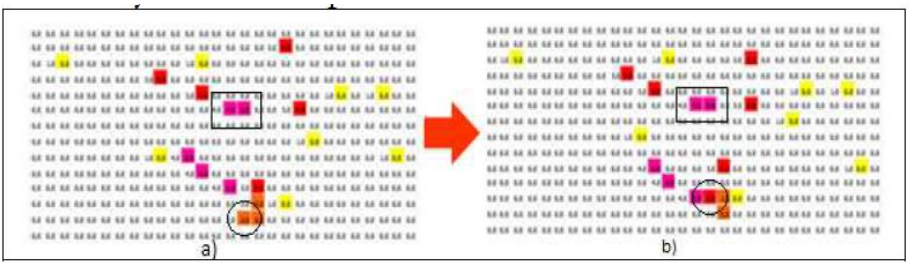

Fig. 8. Phase changes example

Fig. 9 shows the visualization of the simulation results in Google Earth. To do so, a KML file was generated from the CD++ simulation log file and the geographical information for the map generated above. The small white square on the map is the region that was used to test the pathogen transmission. The panel on the left shows visualization with the 5 layers in our model. Here, only the gender layer is shown (pink cells: females; blue cells: males).

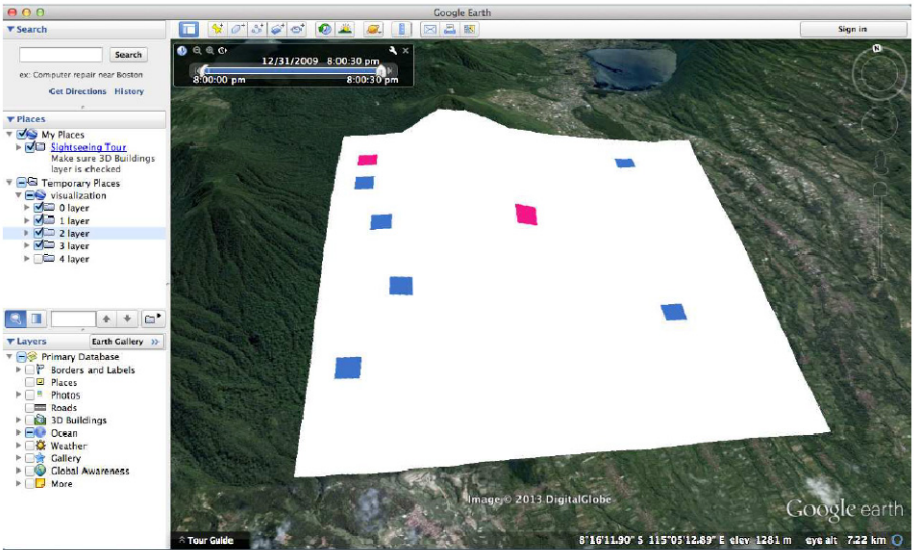

Fig. 9. Gender Layer in Google Earth

6 Conclusions and Future Work

We have presented the Cell-DEVS formalism, and introduced several features CD++, a toolkit for DEVS modeling and simulation. Cell–DEVS allows describing physical and natural systems using an n-dimensional cell-based formalism. Input/output port definitions allow the definition of multiple interconnections between Cell-DEVS and DEVS models. Complex timing behavior for the cells in the space can be defined using very simple constructions. The CD++ tool implements the Cell-DEVS formalism and entitles the definition of complex cell-shaped models. We showed how to develop several Cell-DEVS models using the CD++ toolkit, which provides a general framework to define and simulate complex generic models. Cell-DEVS simplifies the construction of complex simulations, allowing a simple and intuitive model specification.

We showed that different kinds of applications can be easily developed, allowing the study of complex problems through simulation, which, otherwise, could not be attacked. Finally, the use of a formal base improves the development, checking and maintaining phases, facilitating the testing and reuse of their components.

The tools are public domain and can be obtained at http://cell-devs.sce.carleton.ca.

Acknowledgement. This work was partially funded by NSERC. Numerous students participated in the construction of the models presented here, including Eman Al Disi, Rhys Goldstein, Joanna Lostracco, Emil Poliakov, Faezeh Rafsanjani Sadeghi, Michael Van Schyndel, Sixuan Wang and Myriam Younan.

References

1. Burks, A.W.: Von Neumann's self-reproducing automata. In: Burks, A.W. (ed.) Essays on Cellular Automata, pp. 3–64. University of Illinois Press, Champaign (1970)
2. Wainer, G., Giambiasi, N.: Application of the Cell-DEVS paradigm for cell spaces modeling and simulation. SIMULATION 71(1), 22–39 (2001)
3. Wainer, G.: Discrete-Event Modeling and Simulation: a Practitioner's approach. CRC Press, Taylor and Francis (2009)
4. Wainer, G., Giambiasi, N.: N-dimensional Cell-DEVS. Discrete Events Systems: Theory and Applications 12(1), 135–157 (2002)
5. Zeigler, B., Kim, T., Praehofer, H.: Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press (2000)
6. Bonaventura, M., Wainer, G., Castro, R.: A Graphical Modeling and Simulation Environment for DEVS. SIMULATION: Transactions of the SCS 89(1), 4–27 (2013)
7. Wainer, G., Liu, Q., Dalle, O., Zeigler, B.: Introduction to Cellular Automata in Gaming. Simulation and Gaming 41(6), 796–823 (2010)
8. Wainer, G., Castro, R.: A survey on the application of the Cell-DEVS formalism in cellular models. Journal of Cellular Automata 5(6), 509–524 (2010)
9. Saadawi, H., Wainer, G.: Modeling Physical Systems Using Finite Element Cell-DEVS. Simulation Modelling Practice and Theory 15(10), 1268–1291 (2007)
10. Wainer, G., Davidson, A.: Defining a Traffic Modeling language Using Cellular Discrete-Event abstractions. Journal of Cellular Automata 2(4), 291–343 (2007)
11. Wainer, G.: Applying Cell-DEVS Methodology for Modeling the Environment. SIMULATION: Transactions of the SCS 82(10), 635–660 (2006)
12. Liu, Q., Wainer, G.: Parallel Environment for DEVS and Cell-DEVS Models. SIMULATION: Transactions of the SCS 83(6), 449–471 (2007)
13. Al-Zoubi, K., Wainer, G.: RISE: A General Simulation Interoperability Middleware Container. Journal of Parallel and Distributed Computing 73(5), 580–594 (2013)
14. Wainer, G., Goldstein, R.: Modelling Tumor-Immune Systems with Cell-DEVS. In: Proceedings of the European Modeling and Simulation Conference 2008, Nicosia, Cyprus (2008)
15. Poliakov, E., Wainer, G., Hayes, J., Jemtrud, M.: A Busy Day at the SAT Building. In: Proceedings of AIS 2007, Artificial Intelligence, Simulation and Planning. Buenos Aires, Argentina (2007)
16. Castonguay, P., Wainer, G.: Aircraft Evacuation DEVS Implementation & Visualization. In: Proceedings of SCS/ACM Springsim 2009 (DEVS Symposium), San Diego, CA, USA (2009)
17. Wang, S., Van Schyndel, M., Wainer, G., Subashini, V., Woodbury, R.: Interactive DEVS-based Building Information Modeling & Simulation for Emergency Evacuation. In: Proceedings of Winter Simulation Conference, Berlin, Germany (2012)
18. Kennedy, R.C., Lane, K.E., Arifin, S.N., Fuentes, A., Hollocher, H., Madey, G.R.: A GIS aware agent-based model of pathogen transmission. International Journal of Intelligent Control and Systems 14(1), 51–61 (2009)