Journal of Cellular Automata, Vol. 0, pp. 91–106
Reprints available directly from the publisher
Photocopying permitted by license only

A Survey on the Application of the Cell-DEVS Formalism

GABRIEL WAINER[1,*] AND RODRIGO CASTRO[2,†]

[1]*Department of Systems and Computer Engineering - V-Sim. Carleton University. 1125 Colonel By Drive. ON. K1S 5B6. Ottawa, Canada.*

[2]*Dynamic Systems and Signal Processing Laboratory. Universidad de Rosario. Riobamba 245 Bis. 2000. Rosario, Argentina.*

Received: October 30, 2007. Accepted: August 05, 2008

The Cell-DEVS formalism is based on DEVS, a continuous time technique, which provides a framework for multiformalism/multimodeling simulation. We present a survey of different applications showing how to model complex cell spaces as Cell-DEVS models in the CD++ toolkit.

Keywords: DEVS, Cell-DEVS, CD++, Applications of cellular models, Applications of cellular automata, Discrete-event cellular models

1 INTRODUCTION

The Cell-DEVS formalism allows defining a cell space [11] seen as a lattice of cells holding a computing apparatus in charge of updating the cell state according to a local rule. This is done using the present cell state and the cell's neighborhood information using a discrete-event approach and explicit timing delays. Cell-DEVS is based on DEVS (Discrete EVent systems Specifications), a technique that allows the modular description of discrete-event systems that can be integrated hierarchically [14]. Different existing techniques (Bond Graphs, Cellular Automata, State Charts, Partial Differential Equations, Petri Nets, Timed Automata, etc.) have been mapped to DEVS, and the independence of the simulation engines and the models allowed to

*E-mail: gwainer@sce.carleton.ca
†E-mail: rodrigocastro@ieee.org

execute simulations in standalone, parallel, or distributed fashion. In this article, we survey different Cell-DEVS models in varied application domains. The aim is to provide the means to understand current features, limitations and potentials of the solutions.

2 THE CELL-DEVS FORMALISM AND CD++

A real system modeled using DEVS can be described as a hierarchy of submodels, each being behavioral (atomic) or structural (coupled). An *atomic model* is described as $M = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, D)$. The model's interface is composed of input and output ports X and Y respectively. Input external events are received into input ports, and activate the external transition function (δ_{ext}). Each state has an associated duration (D) that, when consumed, triggers the output (λ) and internal transition function (δ_{int}). S represents all the possible model states. A *coupled model* is defined as: $CM = \langle I, X, Y, D, \{M_i\}, \{I_i\}, \{Z_{ij}\}\rangle$. Here X and Y define the model's interface. D is an index of components, and for each $i \in D$, M_i is a basic (atomic or coupled) model. I_i is the set of influencees of model i, and determines which models it will send outputs to. For each $j \in I_i$, Z_{ij} is the i to j translation function, in charge of translating outputs of a model to inputs for the others.

Cell-DEVS allows defining cell spaces as DEVS models with explicit delays. Cell-DEVS *atomic* models can be specified as:

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

Here, X and Y represent the model's interface, S is the cell state, and N is a set of input events. These inputs activate the local computing (τ) and delay (d) functions associated to each cell. The state changes can be transmitted to other models after this delay. A coupled Cell-DEVS is created by putting together a number of cells interconnected by the neighborhood relationship. The coupled Cell-DEVS is an array of atomic cells of a given size and dimensions. Each cell is connected to its neighborhood and other DEVS models through I/O ports.

CD++ [10] is a simulation tool that provides a specification language that allows describing model coupling and formal specifications of Cell-DEVS. The model specification includes the definition of the size and dimension of the cell space, the shape of the neighborhood and borders. The cell's local computing function is defined specifying a set of rules. We created a large number of applications in a variety of fields, including artificial systems, biology, defense, emergency planning, construction, environmental sciences, physics, urban traffic, etc. [8]. In the following sections, we will introduce a few examples in different fields to show some recent results. All the models and tools are available for public use at http://cell-devs.sce.carleton.ca.

3 EVACUATION AND CROWD MODELING

Sophisticated evacuation models have been developed to assist rescue and emergency response crews with proper situation analysis and reaction procedures. Various efforts focused on the cellular models [6, 7, 5] to understand bottlenecks and to present solutions to congestion. The rules in Figure 1 characterize a person's behavior [1]: A person goes to the closest exit, a person in panic goes in the opposite direction, people move at different speeds. We have created such a model for the Society for Arts and Technology (SAT), a building in downtown Montreal [9].

The model uses two layers. Layer 0 is responsible for the boundaries (walls, exits, etc) and people. Layer 1 is responsible for the objects (i.e. internal walls, chairs, etc.) and the distance to the closest exit. The first set of rules serves to define what path a person should follow using the orientation plane and taking the direction that decreases the potential of a cell. A different set of rules governs the panic behavior (the cell's potential is increased) [7]. To do so, we store *direction* of movement of individuals (1:W; 2:SW; 3:S; 4:SE; 5:E; 6:NE; 7:N; 8:NW), *speed* (expressed as 1-5 cells per second), *last movement direction*, *emotional state* (the higher this value is, the lower the probability that a person panics), *moving potential* (people try to reduce their distance to the exit) and *panic level* (the number of cells a person will move increasing the cell potential).

Figure 2 a) considers eight people without panic placed at random inside the building at a high level of patience. The building is evacuated in 13:015s. In Figure 2 b) we included panic into one of the persons (the person moves away from the exit).

In order to observe the effect of panic, we introduced panic in each person. Figure 2 c) shows that evacuation time is tripled. Further studies, presented in [9], increased the number of people, and added people close to the other exits. The increased amount of people did not result in increased evacuation time: as long as the panic level is low, the evacuation was properly controlled. A higher panic factor produces bottlenecks in front of the two exits, and the chaotic behavior of people affect the total evacuation time.

```
type : cell   dim : (49,27,2)   delay : INERTIAL
border : wrapped   localtransition : EvaRule
neighbors : (-1,-1,0) (-1,0,0) (-1,1,0) (0,-1,0) (0,0,0) (0,1,0)
... (1,0,1) (1,1,1)

[EvaRule] % Rules to control movement of each person
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 ...
```

FIGURE 1
Evacuation rules in CD++.

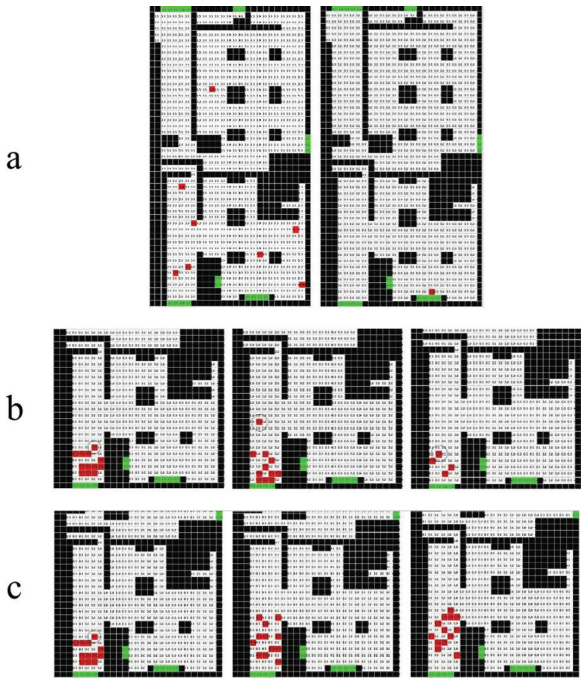

FIGURE 2
Evacuation sequences (left to right) for different scenarios. a) Evacuation without panic and high level of patience: 13:015s. b) Evacuation with one person in panic: 05:004s. c) Evacuation with panic: 15:519s.

4 ROBOT PATH PLANNING

This problem deals with the design of positions and orientations of robots in the presence of obstacles. We need a spatial planner to find a path free of obstacles, which, in general, consists of two phases: *route planning* (a sequence of sub-goals that must be reached before the final goal) and *path generation* (use the plan, apply heuristics to reach the goal).

Our first example presents a model of robots in an industrial plant carrying a load in one-way routes from the source to a destination. The robots can move N/S/E/W, following predefined routes at different speeds. Figure 3 shows the plant represented by a 20x20 Cell-DEVS, linked to four models that generate a load at the source points. A cell containing a route 1 robot can have the values 1,10,11 (moving horizontally) or 2,20,21 (vertically). For example, a route 1 robot at the source is indicated by a 1 in cell (12,19). The next cell on the route will get ready to receive the robot by acquiring a value of 10 or 11 after a delay of 1000 ms. 10 will be used if the robot continues horizontally, and 11 if the robot turns. Then, the original cell will change to 0 (and a the

```
components : Floor Source1@Generator Source2@Generator
             Source3@Generator Source4@Generator
link : out@Source1 in1@Floor  link : out@Source2 in2@Floor
link : out@Source3 in3@Floor  link : out@Source4 in4@Floor

[Floor]
type : cell dim : (20,20)  delay : inertial  border : nowrapped
neighbors : (-1,0) (0,-1) (0,0) (0,1) (1,0)
in : in1 in2 in3 in4
link : in1 in@Floor(12,19)  link : in2 in@Floor(0,10)
link : in3 in@Floor(9,0)    link : in4 in@Floor(19,6)
localtransition : RobotsMov

[RobotsMov]
rule : 10 1000 {(0,1)=1 and (0,0)=0 and cellpos(1)!=4}
rule : 11 1000 {(0,1)=1 and (0,0)=0 and cellpos(1)=4}
rule : 0  0 {(0,-1)=10 and (0,0)=1}
rule : 0  0 {(0,-1)=11 and (0,0)=1}
rule : 2  0 {(0,0)=11}
rule : 1  0 {(0,0)=10}
rule : 20 2000 {(-1,0)=2 and (0,0)=0 and cellpos(0)!=17}
...
```

FIGURE 3
Model definition for robot routes.

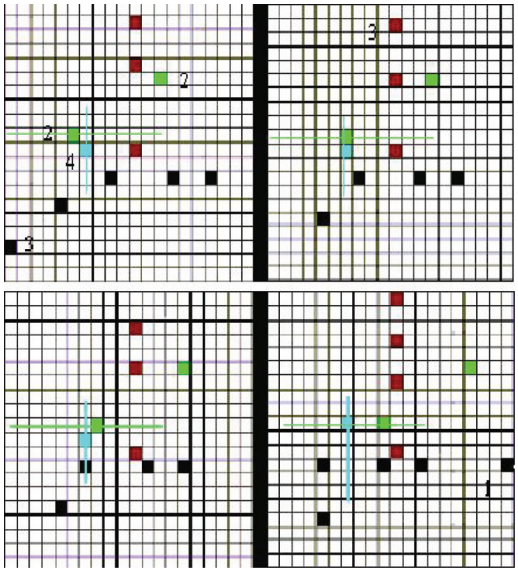

FIGURE 4
Two robots reaching an intersection.

new cell will change to 1 or 2, respectively). Collisions are avoided by only allowing step 1 to take place if the destination cell is empty.

Figure 4 shows different robots at different speeds. This figure also shows the collision avoidance between a robot in route 2 and other in route 4.

More advanced route planners use Voronoi diagrams. We defined a model based on [12] that produces a Voronoi diagram that can be used to determine a path equidistant from obstacles. Paths are calculated by marking the intersections of expanding wavefronts. The model executes in two stages. *Stage 1-Object boundary detection*: the cells and their neighbors are examined and compared to a set of 12 edge code templates. Each cell matching a configuration uses the corresponding code (1-12) for the second stage. *Stage 2-Cells with edge codes are expanded*: cells where expansions intersect are given a timestamp and considered part of the final Voronoi diagram. The 3D cell model is divided into four 2D models by using separate zones for each plane.

Figure 6 shows the execution of the model using a partial boundary and two obstacles. The inputs in the first plane remain unchanged, and edge codes are generated after one iteration. The third plane is initially populated with edge codes > 4, and these values are successively propagated across their neighborhoods. The final plane is the Voronoi diagram which can give a number of possible paths. We built a Cell-DEVS model to generate the shortest

```
[Path-Finding]
dim : (10, 10, 4)   delay : transport
neighbors : (-1,0,0)(0,-1,0)(0,0,0)(0,1,0)...(0,1,-1)
zones : bound-rule  { (0,0,1)..(9,9,1) }
        plane2-rule { (0,0,2)..(9,9,2) }
        plane3-rule { (0,0,3)..(9,9,3) }
[bound-rule]
rule: 1  10 { (0,0,-1) =1 and (0,-1,-1)=1 and (-1,0,-1)=1 and
(0,1,-1) =1 and (1,0,-1) =1 }
...
rule: 12 10 { (0,0,-1) =1 and (0,-1,-1)=1 and (-1,0,-1)=0 and
(0,1,-1) =0 and (1,0,-1) =1 }
[plane2-rule]
rule: {(0,0,-1)+.1} 10 { (0,0,-1) >4 and (0,0,-1)<13}
...
[plane3-rule]
rule: {(time)} 10 { (0,0,0)=0 and (-1,0,-1) != (0,1,-1) }
...
```

FIGURE 5
Cell-DEVS model definition in CD++.

```
 +----------+  +----------+  +----------+
0|1111111111| 0|          | 0|          |
1|111       | 1| 57       | 1| 57777    |
2|          | 2|   222222 | 2| 57  92   |
3|          | 3| 22222222 | 3| 57 1922  |
4|          | 4| 222   22 | 4| 5  192 2 |
5|    111   | 5| 22 192 2 | 5| 11119222 |
6|    111   | 6| 22 857 2 | 6| 88885777 |
7|          | 7| 222   22 | 7|  8 857 7 |
8|          | 8|          | 8|   88577  |
9|1111111111| 9|          | 9|          |
 +----------+  +----------+  +----------+
```

FIGURE 6
Partial boundary and two obstacles.

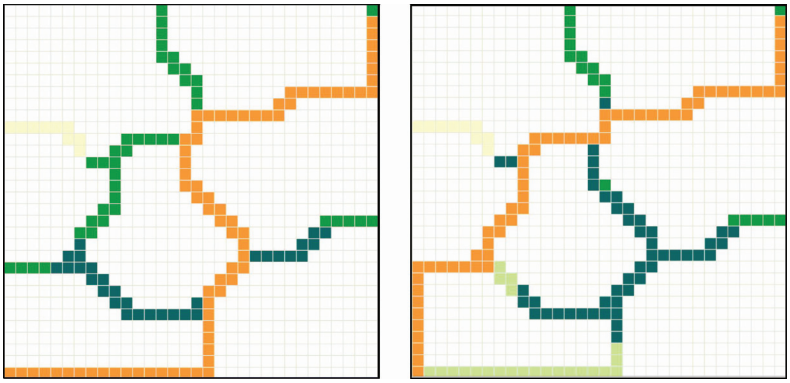

FIGURE 7
Shortest path for two Voronoi diagrams.

path based on a flooding technique introduced in [3]. The algorithm consists of two phases: flooding and selection. The flooding algorithm explores all possible paths starting on the output node in parallel, choosing only valid cells. When a node is found, the path is divided. If two paths are crossed, only the one with the best value continues. Selection starts when we get to the end node; we backtrack, looking for the minimum cost according to the chosen criteria, as seen in Figure 7.

5 MODELS IN PHYSICS AND CHEMISTRY

This section introduces some of our models in Physics and Chemistry [4], which include particle collisions, finite element approximation of heat, flow injection analysis, binary solidification, plastic deformation, etc.

5.1 Diffusion Limited Aggregation (DLA)

DLA occurs when diffusing particles stick to and progressively enlarge an initial seed, growing in an irregular shape (resembling frost on a window). A mobile particle has same probability of walking toward each direction. When a mobile particle finds a seed, it sticks to it, forming aggregates. We defined a Cell-DEVS implementation [4], where a percentage of the cells are occupied by mobile particles. A particle can move in four directions (N/S/E/W), an empty cell will be occupied if there is at least one mobile particle trying to move in (and there is no seed adjacent), a mobile particle that cannot move will select a new direction at random, and it will disappear if it strays too far from the center.

Figure 8 presents a case with a concentration of 40% (grid: 71x71), showing fractal growth properties based on the initial configuration.

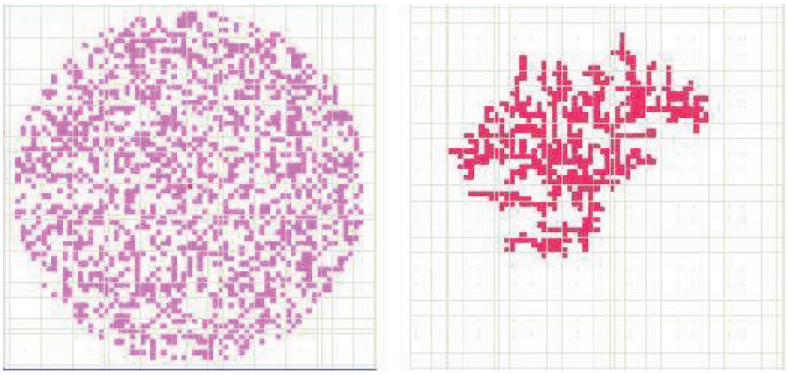

FIGURE 8
One seed and 40% concentration

5.2 Snowflake Growth Simulation

This model is based on the one described in [13], adding some parameters to influence the desired diversity. We assume the value of a cell measures the amount of humidity in the location. Values of one or higher correspond to ice, while lower values represent water. Each cell is classified as either receptive or non-receptive. The first stage is to determine the receptive sites: those that are ice or have an immediate ice neighbor. At the next stage, the values of the cells are given by the value:

$$V_u = 0.5 \cdot V_o + 0.5 \cdot \sum V_n/8 \tag{1}$$

where V_u is the update value of cell, V_o the original value of cell, and V_n the value of the 8 neighbors.

The model was tested using different models for the three different group variable vectors (γ, β), as shown in Figure 9.

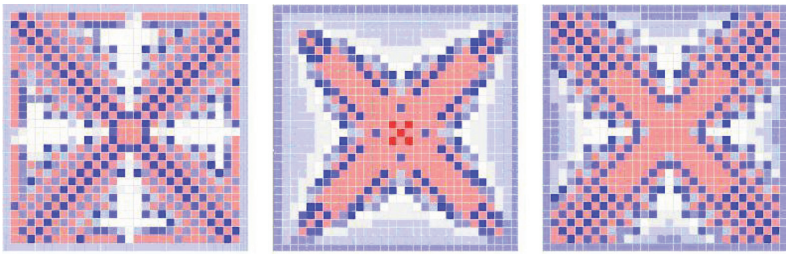

FIGURE 9
Snowflake formation. (a) $\beta = 0.3$, $\gamma = 0.001$, (b) $\beta = 0.4$, $\gamma = 0.01$, (c) $\beta = 0.05$, $\gamma = 0.0035$.

5.3 A 3D model of Reaction-Diffusion

Reaction diffusion systems can be described by a set of partial differential equations as follows [12]:

$$x_i' = D_i \nabla^2 x_i + f_i(x_1, x_2, \ldots, x_n), i = 1 \ldots n \tag{2}$$

where the first term represents the diffusion equation and the second term is the reaction equation. To simulate reaction diffusion systems we apply diffusion first, and then reaction, as follows:

$$\hat{x} = \frac{1}{\#N} \sum_{x_i \in N} x_i \tag{3}$$

$$\dot{x} = f(x) = \frac{x(t) - x(t - \Delta t)}{\Delta t} \tag{4}$$

In this case, we need to know the previous state of the cell, so we used a second 3D model as memory of the previous state and a 3D Von Neumann neighborhood. Thus, we implemented the model as a 4D Cell-DEVS, in which a hyperplane is used to save the previous value from the cell, and a different one is used to calculate the diffusion as in Equation 4. Figure 10 represents the different reaction of two substances. The first group of 5 squares (each consisting of 5x5 cells) from the first row represents the first step of our simulation; the next group of 5 squares (5x5 cells) from the same row represents the initial state.

The next rows represent the following step in our simulation, until the fifth row (34th step of simulation) where the system reached equilibrium.

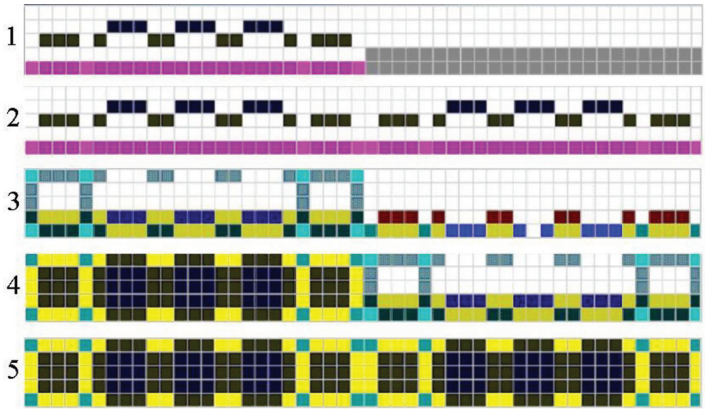

FIGURE 10
Execution results.

5.4 Driven Duffusion

Driven diffusion describes the random motion of two types of particles under the influence of an external field. The field may drive one species of particles along the field direction and the other species against it [3]. This kind of model can simulate the behavior for certain materials such as superionic conductors and solid electrolytes, which are differentiated by their attribute of positive/negative charge. The particle space contains approximately the same amount of positive and negative particles so that the total charge of the system is zero. Initially, the space is occupied by the two randomly distributed particles A and B, and each particle has a randomly chosen direction (N/E/S/W). In the case of an external electrical field appearance (assuming the field points to the NE), the preferable moving direction of particle A is N or E while the preferable moving direction of particle B is S or W. The probability of A and B hops along that preferable direction is a, and the probability against the direction is $(1 - a)$. In our first test case, about 10% of the cell space is occupied by randomly distributed particles, as seen in Figure 11 a).

After 100 time steps, the particles are still randomly distributed. The cell space remains disordered over the simulated time steps, while the distributions of particles A and B are homogeneous (high current in the system). Similar results were obtained with a density of 20%. Then, we show in Figure 11 b) a case in which density of the whole space is higher (40%). We can see that the distribution of the two particles now exhibits a striped structure. Within each strip, there are two sub-strips, each with approximately the same amount of particles. This indicates the non-homogeneities of the distribution of two particles and thus results in reduced current in the system. Similar results were obtained with higher densities.

6 BIOLOGY AND THE ENVIRONMENT

Cell-DEVS has been used to define a variety of models in environmental sciences, including pollution, watershed formation, vegetation growth, etc.

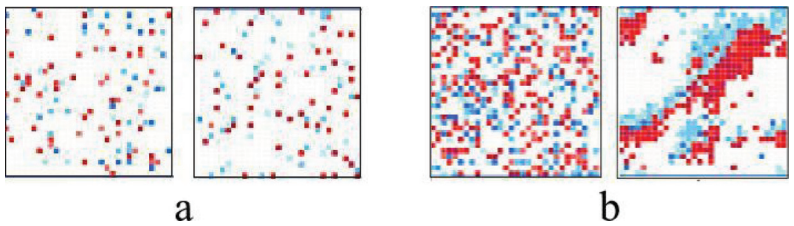

FIGURE 11
Particle diffusion for initial state (left) and after 100 steps (right). a) Density 10%. b) Density 40%.

6.1 Pesticide Percolation in the Soil

This model, based on [2], shows how water (e.g. from rain) gets contaminated with pesticides in the soil, and how the rain washes away the pesticides. Water particles have two states (contaminated or clean), while soil particles use four states for varying degrees of contamination used as the water washes away the contaminants in the soil. We use two different rules for our model:

- *Reaction:* clean cells become contaminated if any of the neighbors is contaminated. Contaminated water particles are assumed to be saturated, and do not cause contamination to clean soil particles. Contaminated soil will pollute neighboring clean water cells.
- *Diffusion:* water cells move their contents to an empty neighbor. Clean water mixed with polluted water remains clean.

The cell values assignment is as follows:
Empty Cells will get:

I. A value of -1 (clean water) if cells W/E and above are sources of clean water (i.e. value=-1 and does not have a direct neighbor with a value$>$ 2).

II. A value of 1 if:

   i. E and up neighbors are not sources of clean water AND either a) W cell=-1, does not have an empty cell below, and it has at least one neighbor with value$>$ 2, or b) W cell=-1 and does not have an empty cell below.

   ii. W and N neighbors are not sources of clean water AND either a) E cell=-1, does not have an empty cell below, and has at least one neighbor with a value$>$ 2, or b) E cell=-1 and does not have an empty cell below.

   iii. W/E neighbors are not sources of clean water AND either a) the N cell=-1 and at least one neighbor with value$>$ 2, or N cell = -1.

Water Cells will get: a value of 0 (empty space) if any of the E/W or S neighbors are empty.

Simulations start with a percolation bed composed of contaminated soil cells, and empty cells in between. Empty cells at the top of the bed are continually filled with water (simulating flooding), which can leave the percolation bed through the bottom row. Figure 12 shows how water is blocked by soil particles and flows around them. It also shows how pollutants get transferred from the soil to the clean water particles. In the end, water has washed away all contaminants. Note how soil particles at the bottom right corner are not cleaned because clean water never reaches them.

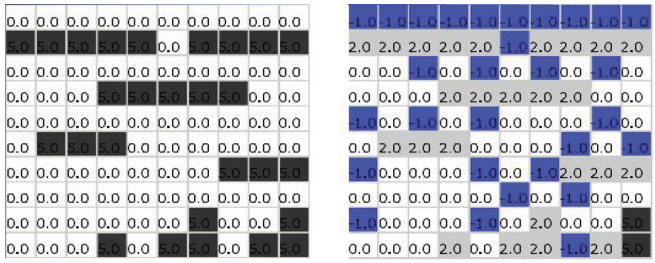

FIGURE 12
a) Initial Environment. b) Final state.

The following scenario represents an environment where water flows, carrying away contaminants as it peculates to the bottom of the soil. Eventually all the soil particles in contact with water have become clean.

6.2 Ants colony

We simulated an ant colony in which ants look for food to pick and carry to their shared anthill. Ants carrying food leave a pheromone trace, allowing other ants of the same colony to find the way to places with food. Thus, a kind of group intelligence is achieved. Ants have no memory, so food searching is carried at random. The field is partitioned in a lattice of cells representing an empty parcel of soil, leaves, an ant or an obstacle. An empty cell may have traces of pheromones. A cell can have an integer number of leaves, and each ant can load only one at a time. There are two types of ants: searchers (B), or carriers (C). When an ant has no food, they look for pheromones. If no pheromones are found, they will move at random. Collision avoidance rules are implemented: if two or more ants want to move to a single cell, they will change direction. When an ant grabs the food on a cell, it starts moving

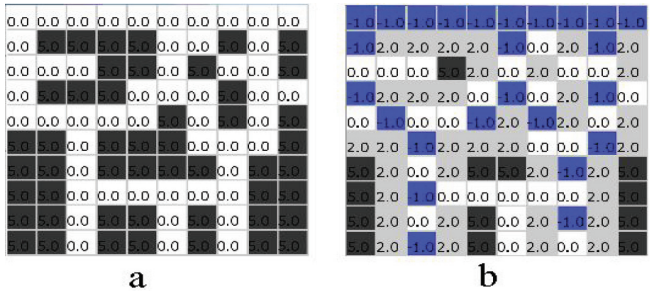

FIGURE 13
a) Initial Environment. b) Final state.

```
[rules] % Check for collision if direction is 0 (North)
rule : {(0,0)+2} 1000 {#(isAnt00) and #(dir00)=0 and ((#(isAnt19)
and #(dir19)=3) or (#(isAnt99) and #(dir99)=1) or (#(isAnt08)
and #(dir08)=2))}
% If my direction is 1 (West)
rule : {(0,0)+2} 1000 {#(isAnt00) and #(dir00)=1 and ((#(isAnt19)
and #(dir19)=2) or (#(isAnt20) and #(dir20)=3) or (#(isAnt11)
and #(dir11)=0))}
...
% Food cell
rule : {if(#(food00)-(if(#(isAntB01)  and #(dir01)=0,1,0)+if(#(isAntB90)
and #(dir90)=1,1,0)+if(#(isAntB09) and #(dir09)=2,1,0)+if(#(isAntB10)
and #(dir10)=3,1,0))>0,(#(food00)-(if(#(isAntB01)
...
```

FIGURE 14
Ants colony model.

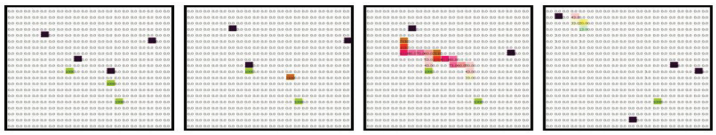

FIGURE 15
4 ants and 4 leaves (2 leaves sharing the leftmost food cell)

back to the mound, leaving a trail of pheromones. The necessary information in each cell is coded with an integer number representing the configurations 0FFD, 00CC, Q00D with the following coding: F (number of pheromones in the cell (1-99)), D (direction (0: N, 1: E, 2: S, 3: W)), C (quantity of food in the cell (1-99)) and Q (load indicator (1: ant of type C; 0: type B). Figure 14 shows the rules used distinguishing the actions to be taken depending on the type of cell. We can see the first leaves found 2 s. after the starting, and then two more ants find a leave (two pheromone trails can be seen). Finally, the 3 carrying ants left their load at the anthill, and they look for new leaves (and no other ant have found food so far).

Figure 15 shows the simulations results for the model.

6.3 Virus Survival

This model shows the competition between population and viruses. Population is represented by individuals residing in a cell, including their age, or a virus. Population and the viruses follow specific active rules, as showed in the following figure.

Initially, there is a group of people (values 1-6: 1 represent children; 2 to 5 represent adults; 6 represent elder people). People in cells will follow the following rules: *Age increment* (each cell increments age periodically by 1 and it dies when is greater than 6 and resets to 0), *People reproduction* (on each empty cell with at least two adult neighbors, the value is 1

```
[survival_rules]
rule : {(0,0)+1} 100 {((0,0)>=1) and ((0,0)<=5) and
(not(((0,-1)=8 and (0,1)=8) or ((-1,0)=8 and(1,0)=8)))}
rule : 0 100 {(0,0)=6}
rule : 1 100 {(0,0)= 0 and (stateCount(2)+
stateCount(3) +stateCount(4)+stateCount(5)>=2)}
rule : 8 100 {(0,0) = 0 and stateCount(8)>=1 and
(stateCount(2)+stateCount(3)+stateCount(4)+stateCount(5)<=1)}
...
```

FIGURE 16
Execution results for the ants colony model.

(newborn)), *Virus state change* (active viruses reproduce themselves and kill people; inactive viruses die after a given time), *Virus reproduction* (active viruses can self-reproduce and share the same living space with persons). Competition between viruses and persons follows the rules: *Virus killing people* (individuals will die in a cell surrounded by at least two viruses vertically or horizontally distributed), *People killing virus* (viruses will die in a cell surrounded by at least two people vertically or horizontally arranged), *People movement* (mature people move at random).

In Figure 17, congregation prevents the individuals from being killed, however, it also restricts reproduction (population size grows, while viruses disappear). In the second case, since individuals are sparse, there is more space to reproduce. However, the number of individuals decreases due to the viruses. Reproduction leads to congregation, and congregation is helpful to avoid being killed. However, movement has an opposite effect.

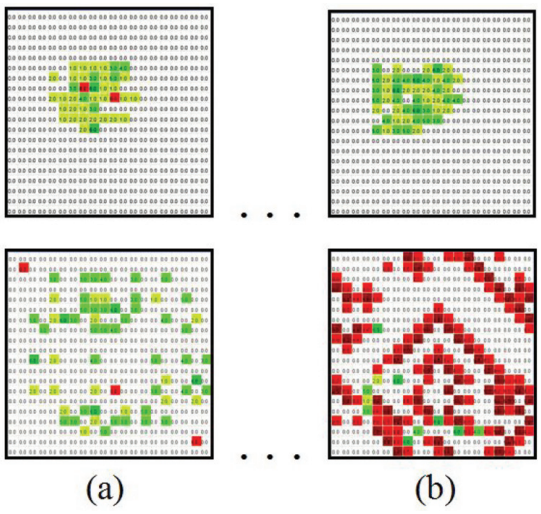

FIGURE 17
(a) Congregated scenario (b) Moving people.

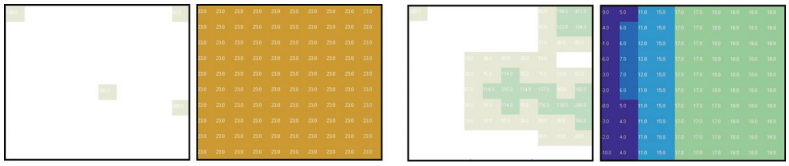

FIGURE 18
Bacteria concentration/surface temperature.

6.4 Bacteria

We simulated the proliferation of the Vibrio parahaemolyticus bacteria growing in a sample of infected food stored in a freezer. In an adequate medium and with enough time, bacteria can reproduce through binary fission. In a rich medium, they proliferate in only ten minutes (we could find around 1 million bacteria in 5 hs.). During their growing state, bacteria secrete different toxins that can modify organoleptic food's properties. Pathogen bacteria have the ability to grow at acidic medium (4,5 pH) but their proliferation is improved in a neutral environment (7 pH). Vibrio parahaemolyticus is a marine bacterium that easily develops at 15°C so it can be found in flake, intestine or directly over fish, crustaceous and mollusk. Optimal conditions for proliferation are 3% salt concentration, 7,5-8,5 pH and 37°C, but it grows in a wide temperature window (15°C-43°C). We considered optimal reproductive values for all variables and studied the impact in bacterial growth during temperature changes. The bacterial development inside the food is modeled in 2D: concentration and temperatures. We used temperatures of (-10°C-0°C), which propagate as the average temperature in neighboring cells. Proliferation depends on temperature: *cannot proliferate* with temperatures < 8°C during 10 s., *proliferate* with temperatures 8°C-60°C during 30 s. and *die* with temperatures over 60°C during 10 s.

We established 100 bacteria/cell, delivering 10% of the excess to the neighboring cells progressively contaminating the food. Figure 18 shows an initial temperature of 23°C for the whole cluster and different contaminated zones.

The results allow us to observe how temperature can propagate and how bacteria progress while they have favorable conditions to proliferate.

7 CONCLUSION

Cell-DEVS allows describing complex systems using cellular models based on DEVS models. Complex timing behavior for the cells in the space can be defined using very simple constructions. The CD++ tool, based on the cell-DEVS formalism, permits defining complex cell-shaped models using a high-level specification language. The discrete event nature of the formalism

provides better precision and performance, due to the independent timing for each cell. The use of a formal base improves the development, checking and maintaining phases, facilitating the testing and reuse of the components.

REFERENCES

[1] J. Ameghino and G. Wainer. (2004). Application of the cell-devs formalism for modeling cell spaces. In *AIS2004*.

[2] S. Bandini, G. Mauri, G. Pavesi, and C. Simone. (1999). A Parallel Model Based on Cellular Automata for the Simulation of Pesticide Percolation in the Soil. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 383–394.

[3] C. Behring, M. Bracho, M. Castro, and JA Moreno. (2000). An Algorithm for Robot Path Planning with Cellular Automata. In *Proceedings of the Fourth International Conference on Cellular Automata for Research and Industry, Springer-Verlag*, volume 19.

[4] W. Ding, X. Wu, L. Checiu, C. Lin, and G. Wainer. Definition of Cell-DEVS Models for Complex Diffusion Systems. In *SUMMER COMPUTER SIMULATION CONFERENCE*, page 57. Society for Computer Simulation International; 1998.

[5] H. Kim, J.H. Park, D. Lee, and Y. Yang. (2004). Establishing the methodologies for human evacuation simulation in marine accidents. *Computers & Industrial Engineering*, 46(4):725–740.

[6] H. Klupfel, T. Meyer-Konig, J. Wahle, and M. Schreckenberg. (2001). Microscopic simulation of evacuation processes on passenger ships. In *Proceedings of the Fourth International Conference on Cellular Automata for Research and Industry*, pages 63–71. Springer-Verlag.

[7] T. Meyer-König, H. Klüpfel, and M. Schreckenberg. (2001). A microscopic model for simulating mustering and evacuation processes onboard passenger ships. In *Proceedings of the International Emergency Management Society Conference, Oslo*.

[8] G. Wainer. (To Appear 2009). *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. CRC.

[9] G. Wainer, J. Hayes, E. Poliakov, and M. Jemtrud. (2007). A Busy day at the SAT building. In *Artificial Intelligence, Simulation and Planning*.

[10] G.A. Wainer. (2002). CD++: a toolkit to develop DEVS models. *Software - Practice and Experience*, 32(13):1261–1306.

[11] G.A. Wainer and N. Giambiasi. (2002). N-dimensional Cell-DEVS Models. *Discrete Event Dynamic Systems*, 12(2):135–157.

[12] J.R. Weimar. (2002). Three-dimensional Cellular Automata for Reaction-Diffusion Systems. *Fundamenta Informaticae*, 52(1-3):277–284.

[13] S. Wolfram. A New Kind of Science. 2002. *Champaign, IL: Wolfram Media*.

[14] B.P. Zeigler, T.G. Kim, and H. Praehofer. (2000). *Theory of Modeling and Simulation*. Academic Press, Inc. Orlando, FL, USA.