COMPARING SIMULATION METHODS FOR FIRE SPREADING ACROSS A FUEL BED

Alexandre Muzy

Computer Modeling
University of Corsica
SPE - UMR CNRS 6134
B.P. 52, Campus Grossetti
20250 Corti. France.
E-mail: a.muzy@univ-corse.fr

Gabriel Wainer

Dept. of Systems and Computer Engineering
Carleton University
4456 Mackenzie Building
1125 Colonel By Drive
Ottawa, ON. K1S 5B6. Canada.
E-mail: gwainer@sce.carleton.ca

Eric Innocenti
Antoine Aiello
Jean-François Santucci

Computer Modeling
University of Corsica
SPE - UMR CNRS 6134
B.P. 52, Campus Grossetti
20250 Corti. France.

ABSTRACT

We present a comparison between DEVS, Cell-DEVS and a sequential computer simulation of a semi-physical fire spread model. Forest fire is a complex phenomenon, which requires analyzing model evolution and requires high computing capabilities. Hence, we discuss about different environment language implementations (C, Java and C++) for time improvement looking at trade-off between model evolutivity and simulation time provided by the different approaches. Finally, we show how Cell-DEVS allows developing safe and cost-effective simulations, reducing significantly the development times for cellular models.

1. INTRODUCTION

At present, concerns for environment engender increasing interest in monitoring and predicting ecosystem changes.

The complex behavior of the phenomena studied, and the volume of data that ecological models have to grasp makes computer simulation a good choice to solve the problem.

Diffusion processes (oil spills, fire spread, insect infestation, etc.) are usually represented as partial differential equations (PDEs) that have to be discretized in the form of finite differences or finite elements (FDM or FEM). Starting with a PDE with derivatives in time and space dimension, time and space are discretized. Finally, a localized neighborhood computation is obtained, which can easily be modeled with cellular automata (CA). Therefore, these complex systems can be modeled by a field of cells executing the same instructions synchronously and simultaneously [Wol86].

Nevertheless, cellular automata are limited in terms of spatial and behavioral modularity, as well as temporal and spatial dynamics. The two basic organising principles in ecology are hierarchy and taxonomy. If we add modularity aspect to allow evolution of the models, we infer that an object-oriented approach fit very well to these three principles. Temporal and spatial dynamics have to be provided. Hence, we have chosen to base our modeling and simulation approach on DEVS formalism.

The numerous extensions of this formalism can be very helpful. One extension, the Cell-DEVS formalism [Wai00], is a good choice to solve this problem. This technique considers each cell of a CA as a discrete event model, hierarchical and modular. In this way, complex models can be defined, using a continuous time base.

2. MODELING FIRE SPREAD

Over the last fifty years, several efforts were carried out in the field of modeling forest fire spread. The problem consists of calculating the fire spread rate, flame front position and temperature distribution in a fuel complex.

The main constraint in solving this problem is that we must deal with large databases representing complex phenomena. These sources of data must be used to implement the physical model. The simulations we are developing must be able to describe the spread of a forest fire in order to help fire fighters. To

this end, it must be characterised by a very small calculation time as respected to real propagation in order to predict the fire position more quickly than it propagates. This requires a simple computer and physical model capable of predicting the key features of a fire.

2.1 Physical solution

At present, real-time simulators are based on a stationary model defined by Rothermel [Rot72]. This is a one-dimensional semi-empirical model, in which a second dimension can be obtained using propagation algorithms integrating empirically wind and slope. In the last years, researchers at the University of Corsica have planed adding a physical dimension to the semi-empirical models, in order to increase their precision. They will be integrated in a more robust manner the wind and slope effects. This model is a non-stationary two-dimensional semi-physical model [Bal98].

2.2 Computer solution

The literature presents other approaches based on discrete event simulators. Most of these simulators [Bar98, Vas95, Ame01] are using DEVS formalism and the Rothermel model.

Our semi-physical model was first implemented in C. Despite the good results, the developed code proved to be complex, and showed some problems related to the evolution of the fire spread model (wind influence, non-homogenous vegetation, slope influence). To circumvent those difficulties, we have used DEVS [Zei00], based on an independent and automatic simulator generation of a given model.

This application is a good solution for the model evolution [Muz01] but simulation time is rather long to have an effective real-time simulator. Thus, we decided to use the Cell-DEVS approach [Wai00] to save substantial simulation time and have to make easy the definitions of the model.

The aim of our computer research team is to provide an efficient and easy to use environment for fire spread simulation. We want to help the physical team in charge of the model development to test the model, easily modify it and finally develop a real-time simulator.

3. MODEL DEFINITION

Before modeling fire at a large scale, we must to verify the influence of the mechanisms involved in fire spread. In a first stage, a fire spread has been modeled across a 1 m² pine needles fuel without wind and slope, thanks to experimental data provided by the INRA of Avignon. This study uses elementary cells composed of earth and plant matter. The energy transferred from the cell to the surrounding air is considered proportional to the difference between the temperature of a cell and the ambient temperature. We assume that heat transferred between a cell an its neighboring cells can be represented by a single equivalent diffusion term. To model the combustion reaction, it is assumed that (1) combustion occurs above a threshold temperature T_{ig}, (2) above this threshold, the fuel mass decreases exponentially, and (3) the quantity of heat generated by the combustion reaction per unit fuel mass is constant. This can be represented by the following equations:

$$\frac{\partial T}{\partial t} = -k(T - T_a) + K\Delta T - Q\frac{\partial s_v}{\partial t} \quad \text{in the domain} \qquad (1a)$$

$$\frac{\partial s_v}{\partial t} = 0 \quad \text{for an inert cell} \qquad (1b)$$

$$\frac{\partial s_v}{\partial t} = -a\, s_v \quad \text{for an burning cell} \qquad (1c)$$

$$T(x,y,t) = T_a \quad \text{at the boundary} \qquad (1d)$$

$$T(x,y,0) = T_a \quad \text{for the non burning cells at t=0} \qquad (1e)$$

$$T(x,y,0) = T_{ig} \quad \text{for the burning cells at t=0} \qquad (1f)$$

The model parameters are identified from experimental data of temperature versus time.

Two numerical methods can be used to discretize the model: the FEM and the FDM. The application of these two methods is described in [San97]. If the methods give the same results, the FEM one is more complex to apply, and produces longer execution time. Thus, we have chosen the FDM because of its simplicity and good performance.

The study domain is meshed uniformly with cells of 1 cm². The physical model is solved by the finite differences method, which leads to the following algebraic equation :

$$T_{i,j}^{k+1} = aT_{i-1,j}^{k} + aT_{i+1,j}^{k} + bT_{i,j-1}^{k} + bT_{i,j+1}^{k} + cQ\left(\frac{\partial s_v}{\partial t}\right)_{i,j}^{k+1} + dT_{i,j}^{k} \qquad (2)$$

where T_{ij} is the temperature of a grid node. The coefficients a, b, c and d depend on the considered time step and mesh size [Bal98]. We use a time step of 0.01s.

4. DEVS FORMALISM

A DEVS model can be composed by atomic sub models combined into coupled models. A DEVS atomic model is described as :

$$M = \langle X, S, Y, \ddot{a}_{int}, \ddot{a}_{ext}, \ddot{e}, D \rangle$$

X is the input events set, ***S*** is the state set, and ***Y*** is the output events set. There are also several functions: ***ä$_{int}$*** manages internal transitions, ***ä$_{ext}$*** external transitions, ***ë*** the outputs, and ***D*** the elapsed time. A DEVS coupled model is defined as:

$$CM = \langle X, Y, D, \{M_i\}, \{I_i\}, \{Z_{ij}\} \rangle$$

Here, ***X*** is the set of input events, and ***Y*** is the set of output events. ***D*** is an index of components, and for each i ª ***D***, ***M$_i$*** is a basic DEVS model, where

M_i=< X_i, S_i, Y_i, ä$_{inti}$, ä$_{exti}$, ta_i >. I_i is the set of influences of model *i*. For each *j* ª I_i, Z_{ij} is the *i* to *j* translation function.

The timed Cell-DEVS formalism [Wai00] is a combination of the DEVS and CA formalisms with timing delays. Each cell is defined as an atomic DEVS model, and a procedure to couple cells is depicted. Timing delays allow defining different timing behavior. The atomic models (the cells) can be described as:

TDC = < X, Y, è, N, delay, d, ä$_{int}$, ä$_{ext}$, ô, ë, D >

X defines the external inputs, ***Y*** the external outputs defining the model's interface. ***è*** is the cell state definition, and ***N*** is the set of inputs. ***Delay*** defines the kind of delay for the cell (transportation or inertial), and ***d*** its duration. A **transport** delay can be associated with each cell, allowing to defer the execution of the internal transition functions. An input will be discarded if it is not steady during the **inertial** delay of the cell. Finally, there are several functions: ***ä$_{int}$*** for internal transitions, ***ä$_{ext}$*** for external transitions, ***ô*** for local computations (which uses the state values of the neighborhood to compute the future value of a cell), ***ë*** for outputs and ***D*** for the state's duration. Each cell takes the set of inputs and computes the cell's future state using the ***ô*** functions. The modeler only has to focus in defining the local computing function, the kind of delay and its length. The remaining parameters are defined by the formalism.

A Cell-DEVS coupled model is defined by:

GCC = < X_{list}, Y_{list}, X, Y, N, {t_1, … , t_n}, C, B, Z >

Here, X_{list} and Y_{list} are the input/output coupling lists, used to define the model interface. ***X*** and ***Y*** represent the input/output events. The n value defines the dimension of the cell space, $\{t_1, \dots, t_n\}$ is the number of cells in each dimension and ***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. For coupled models, the modeler only has to focus in the neighborhood shape, the size and dimension of the model, the definition of the border set, and the coupling lists.

Recently, Quantized DEVS theory has been introduced [Zei98]. The value space is quantized in equal quantum steps. Quantizers are associated with each model. They check for boundary crossings. Whenever a crossing occurs the new value is sent to the receiver. The problem consists in a trade-off between reducing message number and error induced by this reduction. This approach will be discussed in the conclusion for future works.

5. THE DEVS APPLICATION

We used the JDEVS environment [Fil01], which implements the DEVS theory in Java. The numerical resolution of the physical problem needs to mesh the spread domain. Atomic models (C elements) are associated to the cells, which constitute the mesh. Each atomic model is linked to its neighbors, in order to allow taking into account the thermal exchanges between the different cells. A coupled model M represents the different interconnections between the atomic models.

An ignition atomic model (A element) is linked to every C element, allowing to initiate the fire spread by specifying the ignition zone:

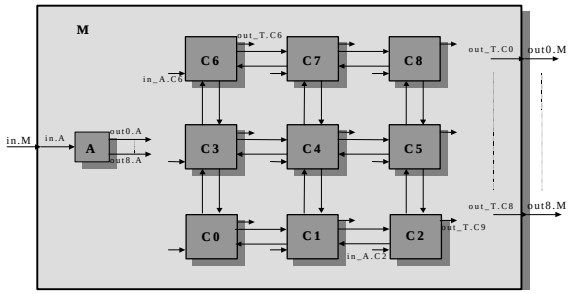

Figure 1: Fire Spread Object Oriented Model

Following, we depict a simplified temperature curve of a cell in the domain and its associated phases:

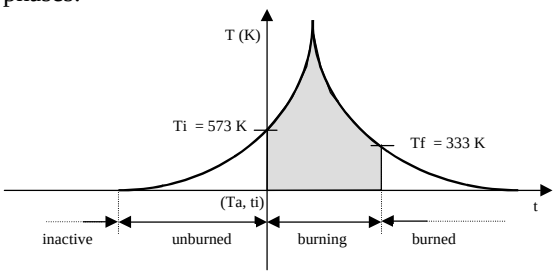

Figure 2: Simplified temperature curve of a cell of the domain

The C element has phases *inactive*, *unburned*, *burning* and *burned*, with a corresponding *ta* of *infinite*, *1*, *1* and *infinite*. We consider that above a threshold temperature T_i, the combustion occurs and under a T_f temperature, the combustion is finished. So we voluntary neglect the end of the real curve to save simulation time. At t=0, we applied a point-ignition on the plate center thanks to a temperature gradient. This is represented by an incoming x-message on the simulator A, with the coordinates and the type of ignition (line or point). Then, the simulator sends y-messages to the coordinator, which transforms them into x-messages and addresses them to the simulators, which are involved by the ignition.

These messages carry out the external transition functions δext of the C elements associated to the simulators. δext stores the cell's temperature and sends a d-message to the coordinator which transform it into a *-message and address it to the output function λ.

The λ function sends four y-messages containing the cell's temperature to the four neighbors (which

by turn will send four y-messages to its own neighbors to active the whole domain).

The internal function transition δint assigns the phase corresponding to the cell's ignition temperature.

The algorithm to pass at time t+1 is depicted in Fig. 3. When the cell receives the fourth temperature (1), λ calculates the temperature of the cell and sends an y-message to the Root (4). The δint function is then carried out. This one assigns the phase corresponding to the cell's temperature calculated and sends a d-message to pass at time *t+1* (6). Above t=1, a cell will be activated if at least one of its cardinal neighbor is greater than T_a.

When a cell is *burned* it sends its temperature (T_a) to the Root and then passes in *burned* phase. When a neighbour sends it an y_message the *burned* cell sends it its temperature.

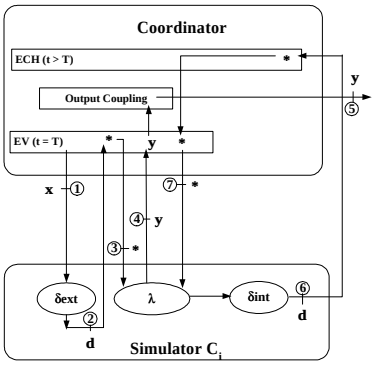

Figure 3: Coordinator and simulator message exchanges

We obtained a good evolution of the model and the discrete event time base allowed to activate only the neighboring cells of the front flame. But the computer model is too complex to develop for a non-computer science specialist and the simulation time is too long to have a real case simulator. Hence, we decided to apply the Cell-DEVS formalism.

6. THE CELL-DEVS APPLICATION

We used the CD++ environment, which implements the DEVS and Cell-DEVS theory, to redefine the C elements using Cell-DEVS specifications. The modeler does not have to manage with message exchanges, port names, couplings and execution of the $\ddot{a}_{int}$ and $\ddot{a}_{ext}$ functions. The cell's behavior is realized defining simple logical rules : {*result*} *delay* {*condition*}. After a *delay* time, the cell takes the value of *result.*

The definition of the fire model is described here:

```
00 #include(rules.inc)

01 [top]
02 components : ForestFire

03 [ForestFire]
04 type : cell
05 dim : (100,100,2)
06 delay : transport
07 defaultDelayTime : 1
08 border : nowrapped
09 neighbors : ForestFire(-1,0,0)
10 ForestFire(0,-1,0) ForestFire(1,0,0)
11 neighbors : ForestFire(0,1,0)
12 ForestFire(0,0,0)  ForestFire(0,0,-1)
13 ForestFire(0,0,1)
14 initialValue : 300.0
15 initialCellsValue : init.val
16 zone : cst { (0,0,0)..(0,99,0) }
17 zone : cst { (1,99,0)..(99,99,0) }
18 zone : cst { (99,0,0)..(99,98,0) }
19 zone : cst { (1,0,0)..(98,0,0) }
20 localTransition : FireBehavior

21 [cst]
22 %Undefined border cells
23 rule : {(0,0,0)} 1 {undefCount >= 1}

24 [FireBehavior]
%Unburned
26 rule : {#macro(unburned)} 1 {cellpos(2)=0
AND (#macro(unburned)>(0,0,0) OR time<=20)
AND (0,0,0)<573 AND (0,0,0) != 209}

%Burning
27 rule : {#macro(burning)} 1 {cellpos(2)=0
AND (((0,0,0)>#macro(burning) AND (0,0,0)>
333) OR (#macro(burning)>(0,0,0) AND
(0,0,0)>= 573)) AND (0,0,0) !=209}

%Burned
28 rule : {209} 1 {cellpos(2)=0 AND
(0,0,0)>#macro(burning) AND (0,0,0)<=333 AND
(0,0,0) !=209}

%ti
29 rule : {time*0.01} 1 {cellpos(2)=1 AND
(0,0,-1)>=573 AND (0,0,0)=1.0 }

%Stay Burned or constant
30 rule : {(0,0,0)} 1 {t}
```

Figure 4: Fire spread model specification: Fire.ma

```
#BeginMacro(unburned)
((1-4*((0.000031*0.01 )/(0.01*0.01))-0.071
*0.01)*(0,0,0)
+ ((0.000031*0.01)/(0.01*0.01))*(0,-1,0)
+ ((0.000031*0.01)/(0.01 * 0.01))*(0,1,0)
+ ((0.000031*0.01)/(0.01*0.01))*(1,0,0)
+ ((0.000031*0.01)/(0.01*0.01))*(-1,0,0)
+ 0.071*300*0.01)
#EndMacro

#BeginMacro(burning)
((1-4*((0.000031*0.01 )/(0.01*0.01))-0.071
*0.01)*(0,0,0)
+ ((0.000031*0.01)/(0.01*0.01))*(0,-1,0)
+ ((0.000031*0.01)/(0.01 * 0.01))*(0,1,0)
+ ((0.000031*0.01)/(0.01*0.01))*(1,0,0)
+ ((0.000031*0.01)/(0.01*0.01))*(-1,0,0)
+ 0.01*274*exp(-0.19*((time+1)*0.01-
(0,0,1)))
+ 0.071*300*0.01)
#EndMacro
```

Figure 5: Definition of the macros: rules.inc

We use two planes to model our fire-spread model. The plane 0 to store the cell temperatures, and plane 1 stores the ignition time needed by the model.

To simplify the code, CD++ allows to define macros in separate files (cf. Fig. 5). The file "rules.inc" contains the rules corresponding to the temperature calculus when the cell is in phase *unburned* or *burning*. The file is included in the "Fire.ma" with the #include directive in line 00.

At the line 01 and 02 we defined the components of the "top" coupled model. Then, we defined the coupled model parameters (neighborhood, dimension, type of delay, etc.).

We define a border zone called "cst" where every cell stays at the ambient temperature (line 21 to 23).

The rules to compute the cell temperature start in line 24. The first corresponds to the phase *unburned*. If the cell belongs to the plane 0, and its temperature at the next time step is greater than its present one, the cell will take the value given by the unburned rule. The same occurs if the simulation time is smaller than twenty (transient period) and it is neither burning nor burned. Based on the same principles, the rules line 27 and 28 correspond to phases *burning* and *burned*.

The line 29 shows how we store the ignition times. The condition is if the cell belongs to the plane 1, and the corresponding cell in plane 0 begins to burn, the cell will take the real time value (simulation time multiplied by the time step).

7. COMPARISON AND ANALYSIS

We simulated a laboratory experiment under these conditions :

- Combustion table of 30 cm long and 60 cm wide;
- Homogenous fuel bed of pine needles;
- Windless and slope less conditions.

For a line ignition, the prediction of spread rate (2.96 mm/s) and the propagation are in agreement with the experimental data for every approach (cf. Fig. 6). The black lines represent the position of the experimental isothermal line of 300 Celsius (ignition interface).

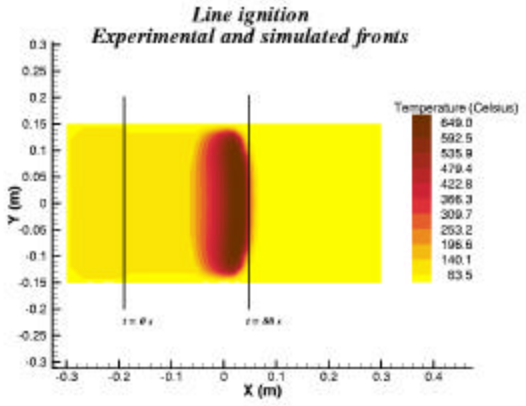

Figure 6: Simulated and experimented temperatures field representation of a line ignition

In Figure 7, we plot the execution time of the Cell-DEVS simulation for different square cell domains and a real propagation of 3 s. Cell-DEVS execution time is definitely better than the JDEVS one (the simulation of 100x100 cells was not possible).

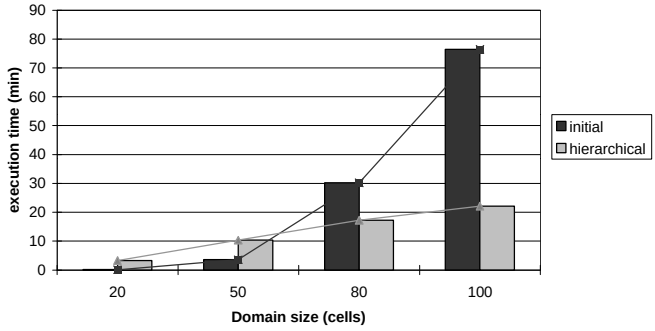

Figure 7: Initial and hierarchical comparison

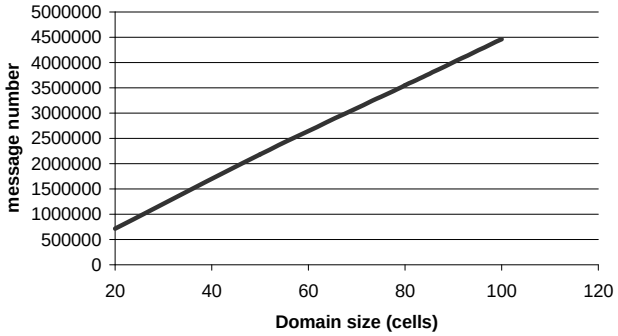

Figure 8: Message exchanges

Cell-DEVS also allowed us to save modeling and simulation time thanks to the automated definition of the model provided by this formal approach.

In the Figure 7, we can see that the initialisation time grows exponentially. In Figure 8, the number of messages exchanged is very high too (up to $4{,}5 \cdot 10^6$ for a grid consisting of 100x100 cells and just 3 s of real propagation).

Moreover, for a real propagation of 150 s of a 100x100 cell space, the execution time of a sequential simulation is of 12 min 20 s and we have 21h20min for the Cell-DEVS simulation.

Cell-DEVS and DEVS formalisms allow to simulate continuous systems by means of events, which produce an high degree of overhead with the message intermodule interactions. Hence, the synchronisation of the active cells can overrule the performance improvements of these asynchronous approaches for synchronous application. The most important number of messages is the cell communications. Moreover, to send its temperature to a neighbor the simulator associated to a cell needs to send an y-message to the coordinator, which send a x-message the destination cell.

8. CONCLUSION AND PERSPECTIVES

We presented a numerical method for diffusion process. Even if the laboratory case studied can be extended to a real-case modifying the cell dimensions, we have seen that this type of processes need continuity of space and time, and are so fine-grained that it is very difficult to simulate them with an event oriented approach in order to make real-time simulations.

This is the problem of an object-oriented approach with independent objects corresponding to the cells. Despite the better evolution and understanding presented before, it produces a high simulation time cost because of the intermodule communications.

Thereby, we think that DEVS and Cell-DEVS approaches need to be modified. We project to formally study and apply the Multicomponent formalism [Zei00], to develop the Flat simulation [Wai00] eliminating the send of messages between the cells and at the initialisation (execution and initialisation time gain) and to append variables to the cells to cancel the second plane.

Once the simulation will be optimized on a single processor, the parallel simulation seems to be the only issue to simulate this type of processes on large scales. Therefore, the use of Quantized DEVS will be interesting to reduce the message number of the distributed simulation. A study will be made on the cost/benefit between reduced traffic and increased error.

REFERENCES

[Ame01] J. Ameghino, A. Tróccoli, G. Wainer, "Models of complex physical systems using Cell-DEVS". *Proceedings of Annual Simulation Symposium. Seattle, WA. U.S.A.*. 2001.

[Bal98] J.H. Balbi, P.A. Santoni, and J.L. Dupuy, "Dynamic modeling of fire spread across a fuel bed", Int. J. Wildland Fire, pp.275-284, 1998.

[Bar98]Barros, F. and G.L. Ball, "Fire modeling using dynamic structure cellular automata", *III International Confer. On Forest Fire Research, 14th Conference on Fire and Forest Meteorology*, VOL I, PP. Luso, 879-888. 16/20 November 1998.

[Fil01] J.B. Filippi, P. Bisgambiglia and M. Delhom, "Neuro-Devs, an hybrid methodology to describe complex systems conference", ESS 2001 – DEVS Workshop, Marseille, France.

[Muz01] A. Muzy, T. Marcelli, A. Aiello, P.A. Santoni, J.F. Santucci and J.H. Balbi, "An object oriented environment applied to a semi-physical model of fire spread across a fuel bed". ESS 2001 – DEVS Workshop, Marseille, France.

[Rot72] Rothermel, R. "A mathematical model for predicting fire spread in wildland fuels". Research Paper INT-115. Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station; 1972. 40 p.

[San97] P.A. Santoni and J.H. Balbi, "Numerical study of a two-dimensional reaction-diffusion model of fire spread across a fuel bed". Int. J. Heat Mass Transfer.1997.

[Vas95] M.J. Vasconcelos, J.M.C. Pereira and B.P. Zeigler, "Simulation of fire growth using discrete event hierarchical modular models". *EARSeL Advances in Remote Sensing*, 4 (3), pp: 54-62, 1995.

[Wai00] G. Wainer and N. Giambiasi, "Application of the Cell-DEVS paradigm for cell spaces modeling and simulation". Simulation. 2000.

[Wol86] S. Wolfram, "Theory and applications of cellular automata". Vol.1, Advances series on complex Systems. World Scientific, Singapore, 1986.

[Zei98] Zeigler B.P. "DEVS theory of quantized systems". Advanced simulation technology thrust DARPA contract, June 1998.

[Zei00] B.P. Zeigler, H. Praehofer and T.G. Kim, "Theory of modeling and simulation". 2nd Edition, Academic Press, 2000.