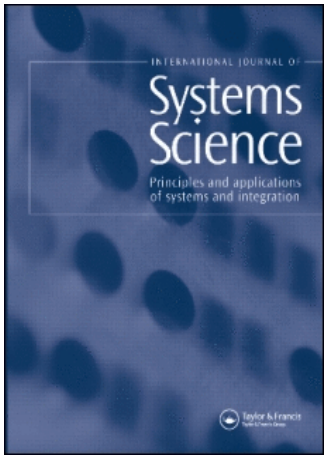

International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713697751

Discrete-event modelling of fire spreading

Alexandre Muzy [a]; Eric Innocenti [a]; Antoine Aiello [a]; Jean-François Santucci [a]; Thierry Marcelli [b]; Paul Antoine Santoni [b]
[a] Fire Modelling & Simulation, University of Corsica, 20250 Corti, France
[b] Fire Physical Modelling, University of Corsica, 20250 Corti, France

Online Publication Date: 01 February 2008
To cite this Article: Muzy, Alexandre, Innocenti, Eric, Aiello, Antoine, Santucci, Jean-François, Marcelli, Thierry and Santoni, Paul Antoine (2008) 'Discrete-event modelling of fire spreading', International Journal of Systems Science, 39:2, 193 - 206
To link to this article: DOI: 10.1080/00207720701755344
URL: http://dx.doi.org/10.1080/00207720701755344

International Journal of Systems Science
Vol. 39, No. 2, February 2008, 193–206

Taylor & Francis
Taylor & Francis Group

Discrete-event modelling of fire spreading

ALEXANDRE MUZY*†, ERIC INNOCENTI†, ANTOINE AIELLO†,
JEAN-FRANÇOIS SANTUCCI†, THIERRY MARCELLI‡
and PAUL ANTOINE SANTONI‡

†Fire Modelling & Simulation, University of Corsica, SPE - UMR CNRS 6134,
B.P. 52, Campus Grossetti, 20250 Corti, France
‡Fire Physical Modelling, University of Corsica, SPE - UMR CNRS 6134,
B.P. 52, Campus Grossetti, 20250 Corti, France

(*Received Spring 2003*)

We deal here with the application of discrete-event System Specification (DEVS) formalism to implement a semi-physical fire spread model. Currently, models from physics finely representing forest fires are not efficient and still under development. If current softwares are devoted to the simulation of simple models of fire spread, nowadays there is no environment allowing us to model and simulate complex physical models of fire spread. Simulation models of such a type of models require being easily designed, modified and efficient in terms of execution time. DEVS formalism can be used to deal with these problems. This formalism enables the association of object-oriented hierarchical modelling with discrete-event techniques. Object-oriented hierarchical programming facilitates construction, maintenance and reusability of the simulation model. Discrete-events reduce the calculation domain to the active cells of the propagation domain (the heated ones).

Keywords: Discrete-event modelling and simulation; DEVS; Cellular models; Fire spread.

1. Introduction

At present, risks and costs of ecological catastrophes make the scientific community increasingly involved in modelling and simulating natural phenomena. The aim is to understand the behaviour of these catastrophes to predict them.

In environmental problems, cellular propagations (oil spills, fire spread, floods, ...) are numerous. The volume of data and the complexity of these phenomena oblige scientists to dispose simple and efficient computer tools, which take program evolutions easily into account.

By their complexity, forest fire propagations are among the most difficult phenomena to model and simulate. Despite the recent availability of very accurate and detailed physical models for laboratory cases (Margerit and Séro-Guillaume 2002, Simeoni *et al.* 2002), there is no effective model capable of correctly predicting actual forest fires. Computer simulation can help in developing such models by comparing simulation model outputs and experimentation. Hence, as long as the simulation results do not match the experimentation, the model has to be modified. However, the multitude of parameters in interrelation in a fire spread requires high computer capabilities and decreases productivity.

Our final aim is to develop an efficient and easy to use computer tool to assist physicists in modelling and simulating fire spread (Muzy *et al.* 2001). Current environments are devoted to the simulation of semi-empirical models (Rothermel 1972) in actual forest fire spreading applications (Lopes *et al.* 2002). Nevertheless, nowadays there is no environment allowing us to model and simulate complex physical models of fire spread.

The physical model we used to represent fire spread is based on a reaction-diffusion equation (Balbi *et al.* 1998). The temperature of each cell is represented as a partial differential equation (PDE) that has to be discretised in the form of finite differences or finite elements.

*Corresponding author. Email: a.muzy@univ-corse.fr

International Journal of Systems Science
ISSN 0020–7721 print/ISSN 1464–5319 online © 2008 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207720701755344

In ecological problems, both hierarchy and taxonomy are the two basic organising principles. The use of an object-oriented approach fits well with these two principles. In fire spreading, another difficulty consists in the fact that only heated and burning cells need to be examined. Discrete-event simulation allows us to easily design this behaviour. Concentrating on processing events rather than cells, cells send messages only when heated or burned. In the first case, they are automatically adjoined to the calculation area; in the second one they are removed from it.

Discrete-event system specification (DEVS) (Zeigler *et al*. 2000) formalism enables the association of object-oriented hierarchical modelling with discrete-event techniques. Using this formal definition, the correctness of the simulation engine can be soundly proved, improving the security of simulation, reducing testing time and increasing productivity. Moreover, models simulation can be performed using the abstract simulator principles introduced in Zeigler (1984). An abstract simulator is an algorithmic description of how to carry out the instructions implicit in DEVS models to generate their behaviour. Simulators corresponding to the formal description of the DEVS models are automatically generated allowing us to save modelling time.

Section 2 sets out the modelling and simulation concepts of DEVS formalism. The formulation of the mathematical model we used is presented in section 3. Section 4 is devoted to presenting the DEVS application to fire spread. In the Section 5, the results of the approach are depicted and discussed. Finally, we conclude and make some prospects.

2. DEVS concepts

The use of a modelling formalism facilitates the development of simulations. A formal paradigm should allow one to model the behaviour of a system and securely translate it into an executable model.

2.1 *DEVS modelling*

A heuristic rule to attack model complexity is to decompose them in less complex ones. DEVS uses different models and sub-models coupled in a network model to specify the behaviour and structure of a system.

Atomic sub-models combined into *coupled* models can compose a DEVS model. The atomic models give a local description of the dynamic behaviour of the studied problem. The coupled models represent the different interconnections between a set of model elements (atomic models or other coupled models).

In DEVS, models behaviour is described in a modular way. With modular specification in general, we must view a model as possessing input and output ports through which all interaction with the environment is mediated. In the discrete-event case, events determine values appearing on such ports. External events are received on the input ports and sent on the output ones.

A DEVS atomic model is described as:

$$M = \langle X, S, Y, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, D \rangle$$

X is the input events set,
S is the state set,
Y is the output events set,
δ_{int} manages internal transitions,
δ_{ext} external transitions,
λ the outputs,
D the elapsed time.

A DEVS coupled model is defined as:

$$CM = \langle X, Y, D, \{M_i\}, \{I_i\}, \{Z_i, j\} \rangle$$

X is the set of input events,
Y is the set of output events
D is an index of components

For each $i \in D$:
$M_i =$ $\langle X_i, S_i, Y_i, \delta_{\text{inti}}, \delta_{\text{exti}}, \text{ta}_i \rangle$ is a basic DEVS model,
ta_i is the time advance function,
I_i is the set of influences of model i,

For each $j \in I_i$:
$Z_{i,j}$ is the i to j translation function:
$Z_{\text{self},j}$: $X \to X_j$,
$Z_{j,\text{self}}$: $Y_i \to Y$,
$Z_{i,j}$: $Y_i \to X_j$.

Here 'self' refers to the coupled model itself and is a device for specifying input and external output couplings and internal couplings.

2.2 *DEVS simulation*

In DEVS, simulators are automatically generated after defining models. Each model is piloted by an automatic generated simulator. This allows us to describe models independently from actual simulation procedure improving model reuse.

Two kinds of simulation entities, called *processors*, correspond to the DEVS models: *coordinators* and *simulators*. Every atomic model is connected to its

simulator and each coupled model to its coordinator. The coordinator manages the simulation process between its inferior simulators. The coordinator of the top-most coupled model in the hierarchy is linked to a *root* processor. The root initialises the simulation and activates its subordinates until some termination condition is met thus signifying the end of the simulation.

To achieve the simulation, messages are exchanged between the simulation entities. These messages contain information describing events to process during the simulation. Kinds of messages represent different activities to be carried out by the models. Four types of messages are used:

- the '$*$-messages' associated to the internal transitions,
- the 'x-messages' associated to the external transitions,
- the 'y-messages' associated to the output functions,
- the 'd-messages' associated to the simulation process synchronisation.

Message exchanges between a coordinator and a simulator are schematised in figure 1.

A coordinator has two schedulers:

- ECH stores messages with a carried time t greater than the simulation time T,
- EV stores messages with a carried time t equal to the simulation time T.

When a simulator receives an x-message, it carries out its external function transition δ_{ext}, which returns a d-message. This d-message is transformed into a $*$-message, which leads to the output function λ execution, which returns a y-message. The internal transition function δ_{int} is then automatically executed. This one can return a d-message.

3. Mathematical model of fire spreading

Over the last 50 years, several efforts have been carried out in modelling forest fire spread. The problem consists in calculating fire spread rate, flame front position and temperature distribution in complex fuel. To make real time simulation, complexity of fire spread and data volume require having simple mathematical models capable of predicting the main behavioural features of fire.

Based on Weber's classification (Weber 1990), three kinds of mathematical models for fire propagation can be identified in accordance with the methods used in their construction. The first type includes *statistical* models (McArthur 1966), which do not consider physical information. The second one incorporates *semi-empirical* models (Rothermel 1972), based on the principle of energy conservation, but which do not distinguish between the different mechanisms of heat transfer. Finally, *physical* models (Albini 1985) describe the various mechanisms of heat transfer and take the finest mechanisms involved in fire spread into account. However, solving such models requires very long

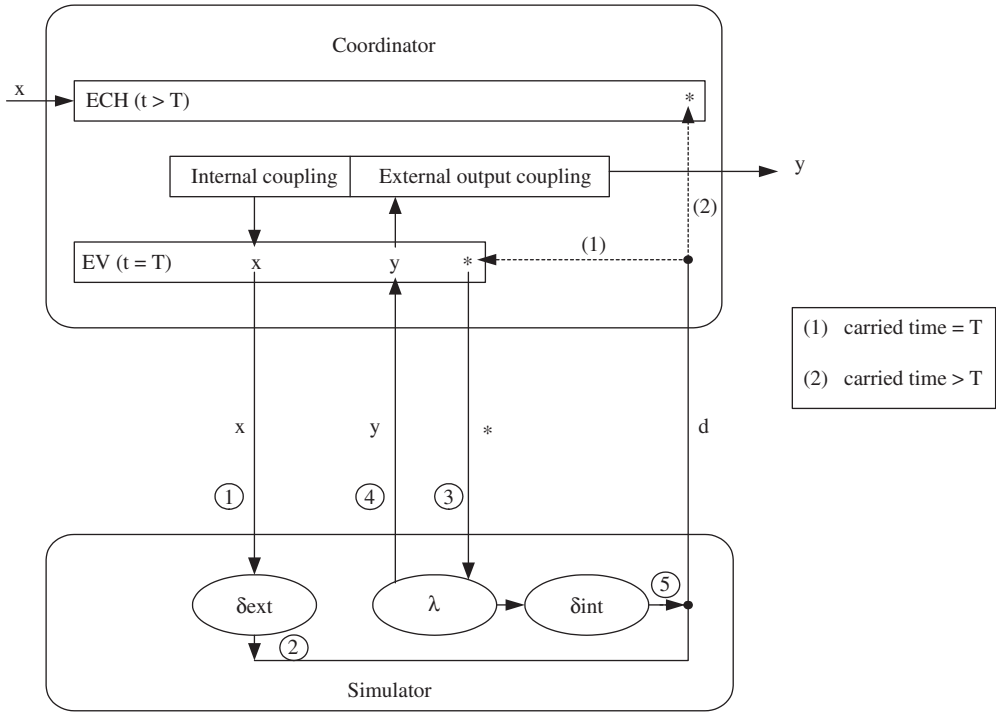

Figure 1. Message exchanges between a coordinator and a simulator.

calculation times and thus they are difficult to integrate into functional fire-fighting tools.

Currently, most real-time simulators are based on Rothermel's stationary model. 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 few years, we planned to add a physical dimension to the semi-empirical models to increase their precision and integrate in a more robust manner the wind and slope effects. This model is a non-stationary two-dimensional *semi-physical* model (Balbi *et al.* 1998).

Different physical assumptions were made to reduce the state variables needed to model a fire-spread:

3.1 *An equivalent medium*

The present study uses elementary cells composed of earth and plant matter. These cells are considered to represent a thin, isotropic and homogeneous equivalent medium.

3.2 *Exchanges with environment*

The energy transferred from the cell to the air is considered to be proportional to the difference between the temperature of a cell and the ambient temperature.

3.3 *Equivalent diffusion*

Heat transferred between a cell and its neighbouring cells is due to three mechanisms: radiation, convection, and conduction. We assume that these exchanges can be represented by a single equivalent diffusion term.

3.4 *Equivalent kinetics*

To model the combustion reaction, it is assumed that:

- combustion occurs above a threshold temperature T_{ig},
- above this threshold, the fuel mass decreases exponentially,
- the quantity of heat generated by the combustion reaction per unit fuel mass is constant.

The above assumptions give rise to the following model:

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

$$\sigma_v = \sigma_{v0} \quad \text{if } T < T_{ig} \tag{1b}$$

$$\sigma_v = \sigma_{v0}.e^{-\alpha(t-tig)} \quad \text{if } T \geq T_{ig} \tag{1c}$$

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

$$T(x, y, t) \geq T_{ig} \text{ for the burning cells} \tag{1e}$$

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

Where, considering a cell:

T_a (27°C) is the ambient temperature,
T_{ig} (300°C) is the ignition temperature,
t_{ig} (s) is the ignition time,
T (°C) is the temperature,
K ($m^2\,s^{-1}$) is the thermal diffusivity,
α (s^{-1}) combustion time constant,
σ_v ($kg\,m^{-2}$) is the vegetable surface mass,
σ_{v0} ($kg\,m^{-2}$) is the initial vegetable surface mass (before the cell combustion).

The model parameters are identified from experimental data of temperature versus time. The heat transfer of the model is schematised in figure 2.

Two numerical methods can be used to discretize the model: the finite element method (FEM) and the finite difference method (FDM). In a previous study, we applied these two methods (Santoni 1997). Although they provided the same results, the FEM appeared more complex to implement, and produced longer execution time. Thus, the FDM was chosen because of its simplicity and good performance.

The study domain is meshed uniformly with cells of $1\,cm^2$ and a time step of 0.01 s. The physical model solved by the FDM leads to the following algebraic equation:

$$T^{k+1}_{i,j} = a\left(T^k_{i-1,j} + T^k_{i+1,j}\right) + b\left(T^k_{i,j-1} + T^k_{i,j+1}\right) + cQ\left(\frac{\partial \sigma_v}{\partial t}\right)^{k+1}_{i,j} + dT^k i, j \tag{2}$$

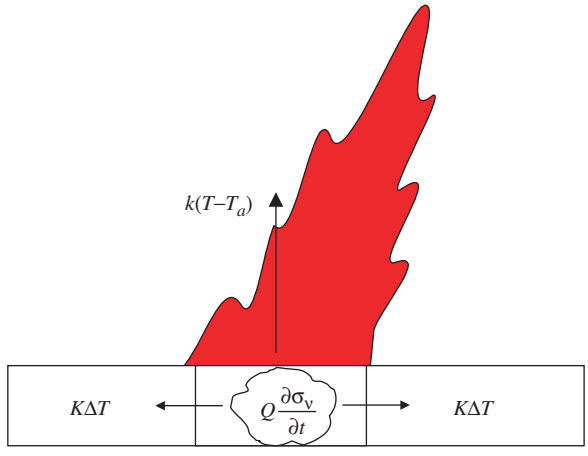

Figure 2. Heat transfer of the semi-physical model.

where T_{ij} is the grid node temperature. The coefficients a, b, c and d depend on the time step and mesh size considered.

4. Fire spread simulation using DEVS

We saw above that the main constraint in real-time simulation of fire spread is that we must deal with large databases representing complex phenomena while simulation time must be very small about the actual spread.

The literature presents other approaches based on discrete-event simulators. These simulators (Barros and Ball 1998, Vasconcelos *et al*. 1995, Ameghino *et al*. 2001) use DEVS formalism and Rothermel's model. This model is a predictable model (we know how long it takes a cell to burn). Our semi-physical model is more complex to simulate and unpredictable. We don't know the time it takes a cell to burn because it depends on the energy flow of the neighbouring cells. However, this model is always in a testing phase and needs to be modified according to simulation results.

Our semi-physical model was firstly implemented with a non-object-oriented simulation model (Santoni 1998). 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 and slope influences, non-homogenous vegetation, etc.). To circumvent those difficulties, we chose to use DEVS to model and simulate fire spread.

4.1 *DEVS fire spread model*

The numerical resolution of the physical problem needs the meshing of the spread domain. As illustrated in figure 3, a first level of hierarchy is used to split up the problem. In this simplest case (we reduced the number of cells for understand ability), atomic models (C elements) corresponding to the cells are interconnected by means of a coupled model M.

A single atomic model (the A element) is linked to every C element. It allows to initiate the fire spread by specifying ignition zones. An input port (in.A) is used to determine ignition location and type of ignition (punctual or linear). Output ports (out0.A to out8.A) are linked to every C element to send ignition temperatures.

As detailed in figure 4, the C elements own several ports to interact with neighbours and external models:

- Four input ports and four output ports for information exchange between the element and its neighbours:

  - in_N.C
  - in_S.C
  - in_E.C
  - in_W.C
  - out_N.C
  - out_S.C
  - out_E.C
  - out_W.C

- an input port for the element ignition: in_A.Ci,
- an output port brings out the temperature value of the element: out_T.Ci.

We depict in figure 5 a simplified temperature curve of a cell in the domain.

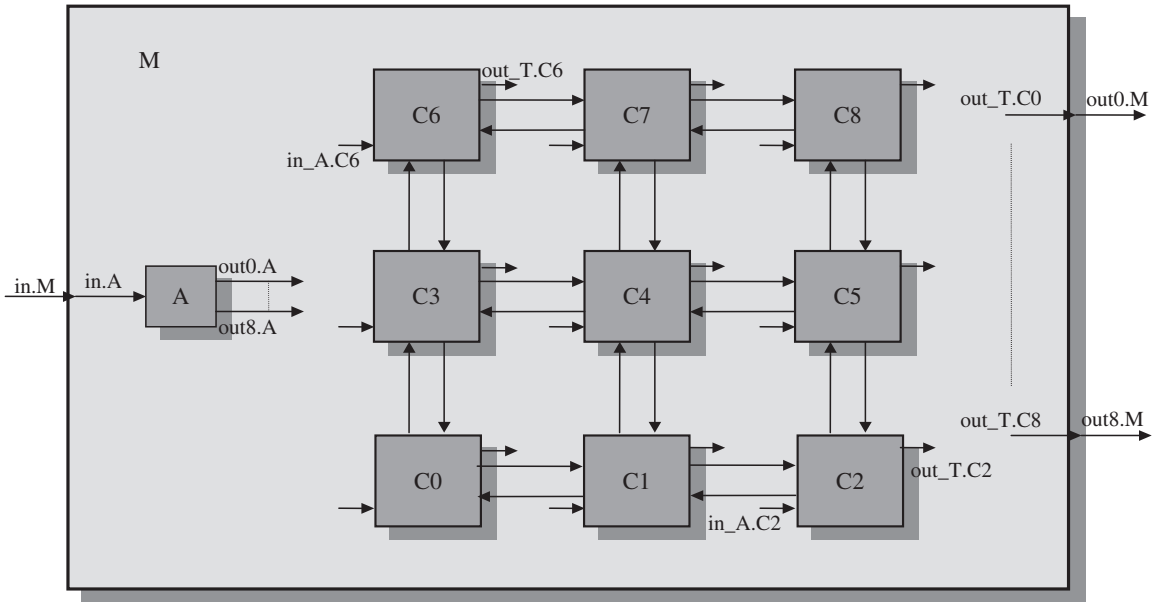

Figure 3. DEVS fire spread model.

Five phases corresponding to time advances and cells' behaviour are defined from this curve. The C elements have phases *inactive, active*, *unburned*, *burning* and *burned*, with corresponding advance time functions t_a of *infinite*, 0, 1, 1 and *infinite*.

We have considered that above a threshold temperature T_{ig} the combustion occurs and under a T_f temperature the combustion is finished. Hence, we deliberately neglect the end of the real curve to save simulation time. An intermediate phase (*active*) allows us to activate the cell when receiving the temperature of a neighbor. As long as a cell does not receive a temperature different from T_a, it does not propagate its temperature to restrict the calculation domain around the flame front.

Actually, we implement the C element algorithm described in figure 6.

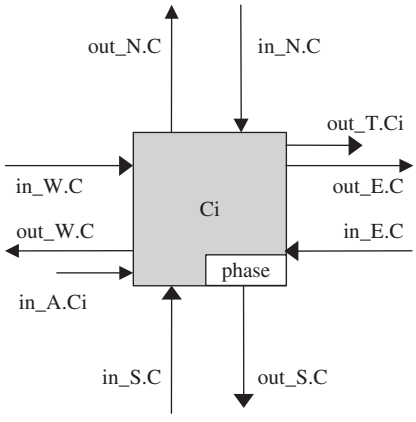

Figure 4. Detail of the C elements.

Lines 00 and 02 define the input/output ports. Line 01 sets out the state variables used. In lines 03 to 16, the δ_{ext} transition function is depicted. This function allows us to store the neighbouring temperatures received (lines 07 to 16) or the initialisation temperature of the cell (lines 04 to 06). This is made by testing the input ports (lines 04 and 07).

The output function λ is activated either by the δ_{ext} or δ_{int} transition functions (lines 19 and 26). The choice is made testing the value carried by the $*$-message received by the function. The λ function allows us: to restart the simulation at each time step (δ_{int} call); to calculate the temperature of the cell; to send the temperature of the cell to the neighbours (δ_{ext} call), according to the phase of the cell and to the neighbouring temperatures received.

The δ_{int} transition function is devoted to affect the phase of the cell after the output function execution. This is made by checking if the temperature of the cell has been calculated or not (lines 46 and 54). If the temperature of the cell has been calculated and the cell has received an initialisation temperature (lines 46 to 53), the phase corresponding to the temperature received will be affected. Otherwise (lines 54 to 78), depending on the temperature of the cell: a new phase is affected (lines 59 and 68); the temperatures are initialised (lines 61 to 63 and 72 to 74); a *d*-message is sent to pass in the burned phase (line 70).

4.2 *DEVS fire spread simulation*

The behavioural model previously introduced and illustrated in figure 3 gives rise to the automatic generation of the simulation tree represented in figure 7.

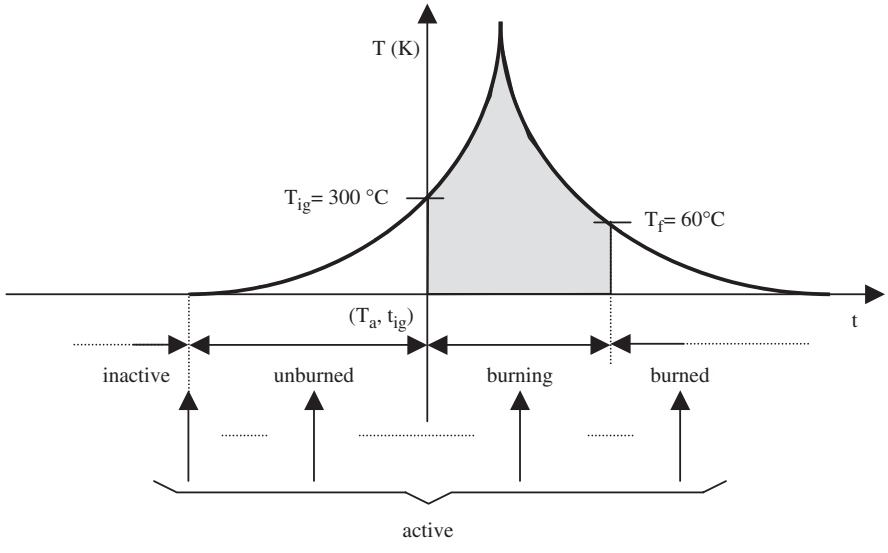

Figure 5. Simplified temperature curve of a cell of the domain.

The root writes and reads data received from the simulators and activates the coordinator. The coordinator Co synchronises the messages exchange process between the simulators SA, S0, S2, . . . , S8 that handle A, C0, C1, . . . , C8 atomic models.

We now describe the simulation process step by step. The evolution of the simulation can be analysed regarding both the algorithm of the C components in figure 6 and the coordinator and simulator message exchange schematised in figures 9 and 10.

As depicted in figure 8, a point-ignition is applied at $t = 0$ on the plate thanks to a temperature gradient. This gradient allows not to create a thermal shock for the semi-physical model. As described in figure 12, this is represented by an incoming *x*-message on the simulator SA (2), with the coordinates and the type of ignition (linear or punctual). Then, SA sends immediately *n* *y*-messages containing the ignition temperatures to the coordinator (5). The coordinator then transforms the *y*-messages into *x*-messages. The *x*-messages are finally addressed to the simulators involved by the ignition (6).

These messages carry out the external transition functions δ_{ext} of the C elements associated to the simulators. The δ_{ext} function stores the temperature of the cell and sends a *d*-message to the coordinator (7),

```
1.  X: in_N.C, in_S.C, in_E.C, in_C_W, in_A.C

2.  S: phase, old_phase, T(t-1),T(t), T(t+1), t_ig, T_ig, T_f, T(t)_N, T(t)_E, T(t)_S, T(t)_W

3.  Y: out_N.C, out_S.C, out_E.C, out_W.C, out_T.C


4.  δext(Ci):

5.  If (port==in_A.Ci)
6.     T(t)←in_A.Ci.valeur
7.     send a d-message containing T(t) at t
8.  Else
9.     old_phase←phase
10.    phase←active
11.    If (old_phase!=burned)
12.       store neighboring cell temperature
13.       send a d-message at t containing the port received
14.    Else
15.       send a d-message at t containing the port received
16.    Endif
17. Endif


18. λ(Ci):

19. If (*-message empty)          // δint call
20.    If (phase==active)         // the cell is burned
21.       send a y-message containing T_a to the root
22.    Else
23.       If (T(t)!=T(t-1))
24.          send a y-message containing T(t) to the neighboring cells
25.       Endif

26. Else                  // δext call
27.    phase←old_phase

28.    Switch(phase)

29.       Case 'inactive':
30.          send a y-message, at t=0,containing T(0) to the neighboring cells and to the root
31.       EndCase

32.       Case 'unburned','burning':
33.          If (the cell received a neighboring cell temperature different from T_a)
34.             send a y-message containing T(t) to the neighboring cells
35.             If (all neighboring cells temperature are known)
36.                calculate the cell temperature
37.                send a y-message, containing T(t+1) to the root if t=100
38.             Endif
39.          Endif
40.       EndCase

41.       Case 'burned':
42.          send a y-message containing T_a to the neighboring cell, which has sent its
                                                                              temperature
43.       EndCase

44.    End Switch

45. δint(C):

46. If (the temperature of the cell has not been calculated)   // initialisation phase
47.    If (phase==inactive and T(t)!=T_a)
48.       If (T(t)<T_ig)
49.          phase←unburned
50.       Else
51.          phase←burning
52.       Endif
53.    Endif
```

Figure 6. Algorithm of the C components.

which transforms it in a $*$-message. The $*$-messages finally induce the output function λ (8) execution.

The λ functions send four y-messages (9) containing the temperature of the cell to the four neighbours. Every neighbour will send by turn four y-messages to its own neighbours until they propagate a temperature equal to T_a. In that case, the requested cells will not respond. The internal function transition δ_{int} finally assigns the phase corresponding to the cell's ignition temperature.

The algorithm to pass at time $t+1$ is depicted in figure 10. When a cell receives the fourth temperature (1), λ calculates the temperature of the cell and sends a y-message to the root (4). The δ_{int} function is then carried out. This one assigns the phase corresponding to the temperature of the cell 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 neighbours has a temperature greater than T_a.

```
54. Else

55.    Switch(phase):

56.        Case 'unburned':
57.           If (T(t+1)≥T_ig)
58.              t_ig←t
59.              phase←burning
60.           Endif
61.           T(t-1)←T(t)
62.           T(t)←T(t+1)
63.           T(t+1)←0
64.           send an empty d-message at t+1
65.        EndCase

66.        Case 'burning':
67.           If (T(t+1)≤T_f)
68.              phase←active
69.              T(t)←T_a
70.              send an empty d-message at t
71.           Else
72.              T(t-1)←T(t)
73.              T(t)←T(t+1)
74.              T(t+1)←0
75.              send an empty d-message at t+1
76.           Endif
77.        EndCase

78. End Switch


79. Ta(C):

80. Switch(phase) :

81.    Case 'inactive':
82.       ta←∞
83.    EndCase

84.    Case 'active':
85.       ta←0
86.    EndCase

87.    Case 'unburned':
88.       ta←1
89.    EndCase

90.    Case 'burning':
91.       ta←1
92.    EndCase

93.    Case 'burned':
94.       ta←∞
95.    EndCase

96. End Switch
```

Figure 6. Continued.

Finally, when a cell is *burned*, it sends its temperature (T_a) to the root and then passes in *burned* phase. If a neighbour sends it a *y*-message the *burned* cell sends it the T_a temperature.

We obtain a good evolution capacity of the simulation model and the discrete-events only allow us to activate the neighbouring cells of the flame front. To modify the physical fire-spread model to include wind and slope (Marcelli *et al.* 1999), a simple modification of the equation contained in the λ output function can be done. Nevertheless, the computer model is too complex to be developed and understood by a non-computer science specialist.

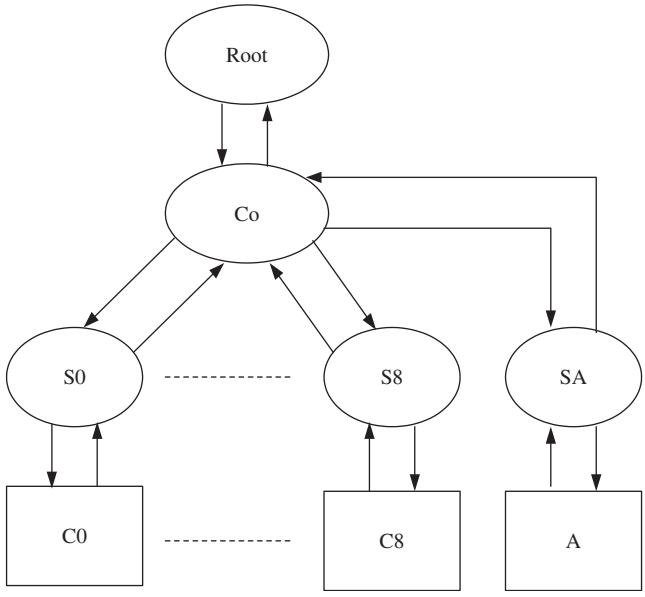

Figure 7. DEVS simulator associated to the fire-spread model.

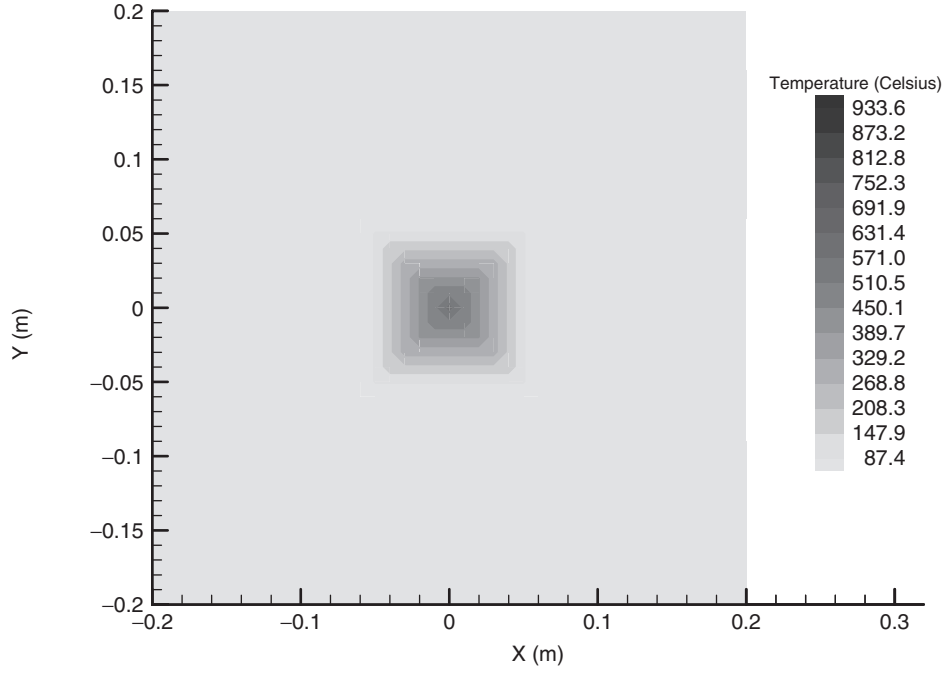

Figure 8. Initial temperature gradient.

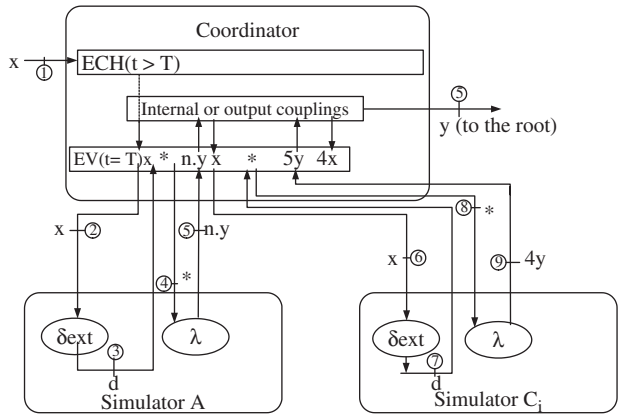

Figure 9. Message exchanges for the ignition of the C elements.

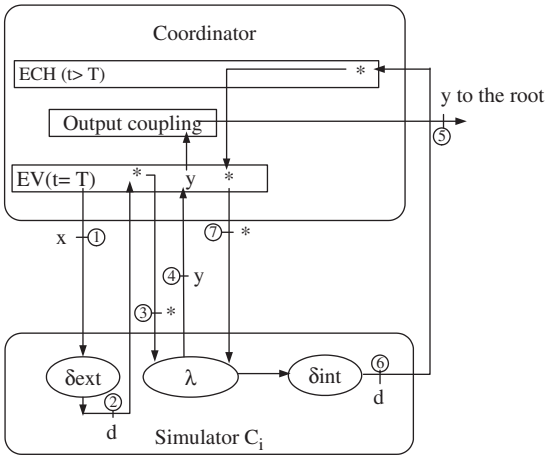

Figure 10. Message exchanges for next time incrementation.

5. Simulation study

We used the JDEVS environment (Filippi *et al*. 2002) to implement the DEVS fire-spread model in Java language.

5.1 *Experimental apparatus*

Our research started by doing a set of experiments to develop and validate a model. We used experimental fires conducted on *Pinus Pinaster* litter, in a closed room without any air motion, at the INRA (Institut National de la Recherche Agronomique) laboratory near Avignon, France (Balbi *et al*. 1998). These experiments were performed to observe fire spread for point-ignition fires under no slope and no wind conditions. The experimental apparatus was composed of a one square meter aluminium plate protected by sand. A porous fuel bed was used, made up of pure oven dried pine needles spread as evenly as possible on the total area of the combustion table to obtain a homogeneous structure with a fuel load of $0.4\,\mathrm{kg\,m^{-2}}$. The fuel depth was approximately 2.3 cm. The needles were conditioned to moisture content between 1% and 3%. The experiment consisted of igniting a point using alcohol. The resulting spread of the flame across the needles was closely observed with a camera and thermocouples located 3 cm above the needle bed to record the temperature. The gauge wire diameter of the thermocouples is 1 mm and the sampling frequency used for these thermocouples is 1 Hz.

5.2 *Simulation results*

Different results have been observed: rate of spread of the fire front, temperature growth of a cell and fire front positions. Those different aspects allow us to make an analysis of the simulation and semi-physical models results.

DEVS simulation model results obtained for a point ignition are depicted in figure 11. In comparison, the white squares represent the front obtained with a non-object-oriented code already validated against experimental data in Balbi *et al*. (1998) and Santoni (1997, 1998). A first observation at time $t = 30$ s of the circular wave fronts gives the same results. Nevertheless, a difference appears between the simulated fire front widths at $t = 50$ s. Indeed, the fire front cools more quickly. This can be explained by the end combustion assumption ($T_f = 60^{\circ}\mathrm{C}$) effect on the predicted temperature curve versus time for a point of the calculation domain as shown in figure 12.

Rothermel's stationary model considers a fire rate of spread as constant even in the ignition phase. This is not a correct representation of the real propagation. As depicted in figure 13, our semi-physical model takes this acceleration phase into account. The fire acceleration in its early stage reveals the unsteady nature of our model.

Based on a more fundamental mathematical model, the simulation model we developed precisely describes fire spread in each phase of the propagation (from ignition to propagation). Nevertheless, simulation time results did not meet real-time deadlines.

The final aim is to dispose of an effective simulator capable of predicting the flame front position. Therefore, execution time is crucial. If the results are qualitatively good, we can now make a quantitative discussion to improve simulation time.

DEVS formalism allows us to simulate continuous systems by means of events, which produce a high degree of overhead with the message inter-module 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 due

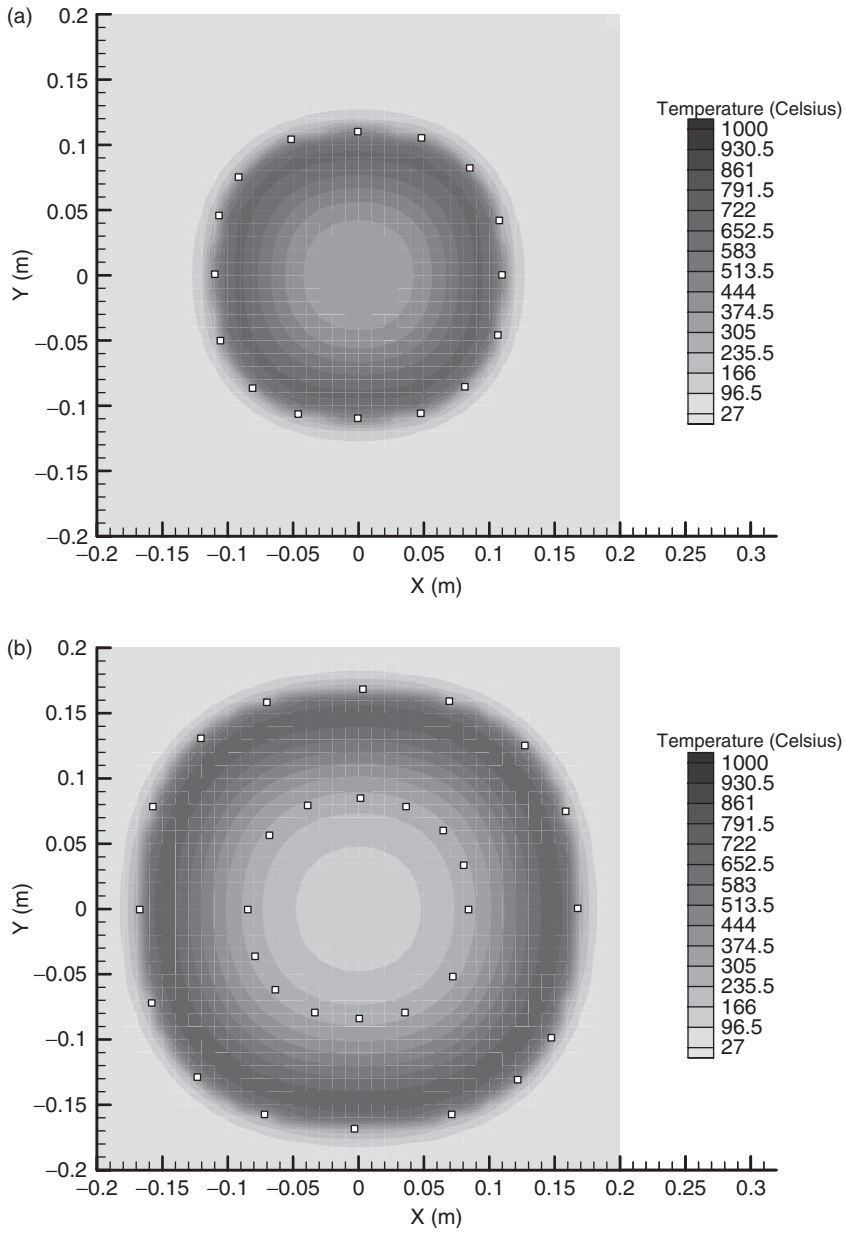

Figure 11. Circular front comparison between both results of non-object-oriented and DEVS simulation models: (a) at $t = 30$ s, (b) at $t = 50$ s.

to cell communications. As schematised in figure 14, to send its temperature to a neighbour, the simulator associated to a cell needs to send a y-message to the coordinator, which sends an x-message to the destination cell. Communication between cells proved too complex and produced simulation time overhead.

6. Conclusion

An application of DEVS formalism for modelling and simulating a laboratory fire spread has been conducted. If the simulation results are qualitatively good and in accordance with the experiment, they can be quantitatively improved.

DEVS hierarchy and modularity ease maintenance and reusability of the simulation model of fire spread. Models modifications can be easily integrated and discrete-event simulation allows us to calculate only active cells of a propagation domain thus optimising the simulation.

Although fire spread needs continuity of time, conversely it necessitates so fine-grained discretisation for simulation that it is very difficult to solve under

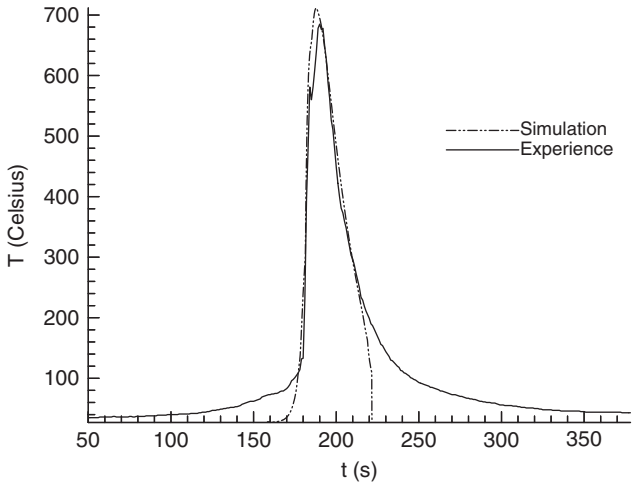

Figure 12. Predicted and observed temperatures curves of a point of the domain.

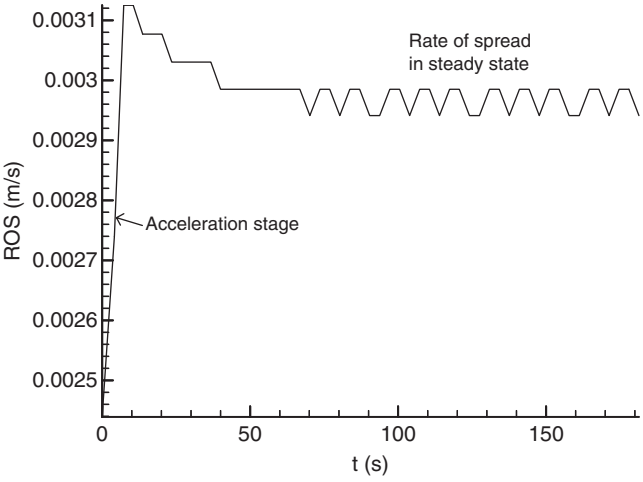

Figure 13. Predicted radial rate of spread over the time.

real-time with discrete-event simulation. DEVS modularity produces simulation time overhead because of its inter-module message communications. Furthermore, DEVS semantic leads to elevated algorithmic complexity of code decreasing understandability and complicating construction of the simulation model.

To optimise the simulation performance, we project to study and apply the multicomponent DEVS formalism (Zeigler *et al.* 2000) to eliminate message exchange overhead between components (or cells) and hierarchy levels. Cells of multicomponent systems directly influence each other through their state transition function reducing message interaction (between cells and levels) to system synchronization improving simulation performances.

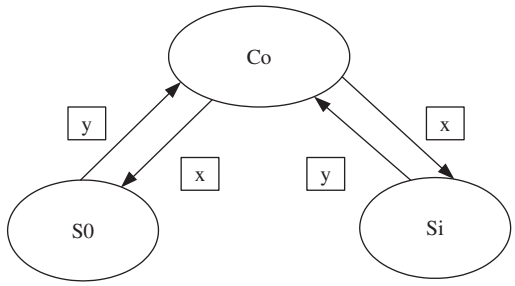

Figure 14. Cells' communication.

Once the simulation has been optimised on a single processor, parallel and distributed simulation will be the only issue to simulate this type of processes on large scales under real time deadlines. In this research area, some recent parallel implementations are showing good results anyway (He and Wu 2002).

References

F.A. Albini, "A model for fire spread in wildland fuels by radiation," *Combustion Sci. Techn.*, 42, pp. 229–258, 1985.

J. Ameghino, A. Troccoli and G. Wainer, "Models of complex physical systems using Cell-DEVS", in: *34th IEEE/SCS Annual Simulation Symposium*, Seattle, USA, 2001.

J.H. Balbi, P.A. Santoni and J.L. Dupuy, "Dynamic modelling of fire spread across a fuel bed," *Int. J. Wildland Fire*, 9, pp. 275–284, 1998.

F. Barros and G.L. Ball, "Fire modelling using dynamic structure cellular automata, III International Confer. On Forest Fire Research," in: *14th Conference on Fire and Forest Meteorology, Portugal*, 1, pp. 879–888, 1998.

J.B. Filippi, F. Chiari and P. Bisgambiglia, "Using JDEVS for the modelling and simulation of natural complex systems, AIS 2002 – Simulation and planning in high autonomy systems, Portugal, pp. 317–322, 2002.

F. He and J. Wu, "An efficient parallel implementation of the Everglades Landscape Fire Model using checkpointing," *Parallel Comput.*, 28, pp. 65–82, 2002.

A.M.G. Lopes, M.G. Cruz and D.X. Viegas, "FireStation: an integrated software system for the numerical simulation of fire spread on complex topography," *Environmental Modelling & Software*, 17, pp. 269–285, 2002.

T. Marcelli, A. Aïello, P.A. Santoni and J.H. Balbi, An object oriented environment applied to the fire fire spread across a fuel bed under local wind condition, Advanced Technology Workshop, 10–11 June 1999.

J. Margerit and O. Séro-Guillaume, "Modelling forest fires. Part II: reduction to two-dimensional models and simulation of propagation," *Int. J. Heat Mass Transfer*, 45, pp. 1723–1737, 2002.

A.G. McArthur, *Weather and Grassland Fire Behavior*, Australian Forest and Timber Bureau Leaflet, Canberra, 100, 1966.

A. Muzy, T. Marcelli, A. Aiello, P.A. Santoni, J.F. Santucci and J.H. Balbi, "Application of DEVS formalism to a semi-physical model of fire spread across a fuel bed", *13th European Simulation Symposium and Exhibition*, South France, pp. 641–643, 2001.

R.C. Rothermel, A mathematical model for predicting fire spread in wildland fuels, USDA, Forest Service Research, Paper INT-115, 1972.

P.A. Santoni, "Propagation de feux de forêt, modélisation dynamique et résolution numérique, validation sur des feux de litière", PhD thesis of the University of Corsica, South France, 1997.
P.A. Santoni, "Elaboration of an evolving calculation domain for the resolution of a fire spread model," *Numer. Heat Transfer, Part A*, 33, pp. 279–298, 1998.
A. Simeoni, P.A. Santoni, M. Larini and J.H. Balbi, "Reduction of a multiphase formulation to include a simplified flown in a semi-physical model of fire spread across a fuel bed," *Int. J. Thermal Sci.*, 42, 2002.
M.J. Vasconcelos, J.M.C Pereira and B.P. Zeigler, "Simulation of fire growth using discrete-event hierarchical modular models," *EARSeL Adv. Remote Sens.*, 4, pp. 54–62, 1995.
R.O. Weber, "Modelling fire spread through fuel beds," *Prog. Energy Combustion Sci.*, 17, pp. 67–82, 1990.
B.P. Zeigler, *Multifaceted Modelling and Discrete-event Simulation*, London: Academic Press, 1984.
B.P. Zeigler, H. Praehofer and T.G. Kim, *Theory of Modelling and Simulation*, 2nd ed., Orlando: Academic Press, 2000.

Alexandre Muzy received his PhD Degree from the Università di Corsica – Pasquale Paoli in 2004. He is currently a cresearcher at the CNRS research laboratory UMR 6134 of the Università di Corsica. His current research interests relate to the theory of modelling and simulation of complex systems, multi-agent systems and cellular models, with applications to fire spread and economic problems.

Eric Innocenti received his PhD Degree from the Università di Corsica – Pasquale Paoli in 2004. He worked as software developer for the past 3 years. He is currently assistant professor and researcher at the CNRS research laboratory UMR 6134 of the Università di Corsica. His current research interests relate to the theory of modelling and simulation of complex systems, the DEVS formalism and parallel and distributed processing.

Antoine Aiello received the PhD (1997) of the University of Corsica, France. He is Professor and President of the University of Corsica. His current research interests relate to the modelling and simulation of natural complex systems and the development of multimedia architectures. He is a member of the CNRS research laboratory UMR 6134 of the University of Corsica.

Jean-Francois Santucci obtained his PhD in March 1989 at the Université d'Aix-Marseille – topic: a knowledge based system for testing of digital system. He has been an Associated Professor at the Université de Nîmes, France, from 1989 to 1995 and from 1995 to 1996 at the University of Corsica, France. He is Professor in Computer Sciences since 1996 at the University of Corsica. His research interests are modeling and simulation of complex systems and high level digital testing. He published about 100 papers in international conferences and journals. He has been scientifically responsible for several European and international industrial contracts. Since 1998 he is Adjunct Director of the CNRS Research Laboratory UMR CNRS 6134, University of Corsica.

Paul-Antoine Santoni obtained his PhD in 1996 at the Università di Corsica – Pasquale Paoli. He is currently Professor in Physics at the Università di Corsica – Pasquale Paoli. His research interests concern physics-based models of fire spread. He is currently a researcher at the CNRS research laboratory UMR 6134 of the Università di Corsica.

Thierry Marcelli obtained his PhD at the Università di Corsica – Pasquale Paoli. He is currently Assistant Professor in Physics at the Università di Corsica – Pasquale Paoli. His research interests concern physics-based models of fire spread. He is currently a researcher at the CNRS research laboratory UMR 6134 of the Università di Corsica.