Stochastic modeling strategies for the simulation of large (spatial) distributed systems: Application to fire spread

Alexandre Muzy* and David R.C. Hill**

* UMR LISA CNRS 6240, Università di Corsica – Pasquale Paoli, B.P. 52, Campus Mariani, 20250 Corti. France.

**Blaise Pascal University – Clermont Université, LIMOS, UMR CNRS 6158, 24 ave des Landais, BP 10125, Campus des Cézeaux, 63173 Aubière CEDEX, France

1. Introduction

The implementation of large (spatial) distributed systems is usually based on either:

1. Mathematical models, which are generally obtained from modeling discipline hypotheses (e.g., Physics, Economics, and Biology). Mappings are then defined from mathematical structures to computational ones. Modeling capabilities of computers are thus constrained by previous discipline hypothesis of usual scientific disciplines and/or by the discretization methods used to implement the model.
2. Fully computational models, which are often derived from heuristic hypotheses. Heuristics correspond to the modeler or expert knowledge. The latter has to try carefully to avoid usual modeling techniques of disciplines. Using this strategy, the modeler is expected to be in a better position to explore new heuristics finding new modeling hypotheses. Not using usual discipline knowledge allows creating new knowledge and original tools to interact with both system and usual models already developed. However, modeling frameworks are guides through these new unknown modeling paths. This chapter discusses this kind of framework.

Computers have been developed to achieve massively, quickly, and automatically calculus performable by humans by hand. Currently, as a result of the increase of computational and communication capabilities, new scientific achievements can be reached by storage of large amounts of data and distributed computing. Developing new modeling and simulation frameworks able to deal with large spatial data is now common in many high performance computing projects.

However, despite powerful computational capabilities, a high number of interacting sub-systems requires the efficient exploitation of computational resources. Therefore, a framework for the modeling and simulation of large (spatial) interacting systems requires: (1) guiding the modeler sufficiently to deal with heuristic modeling hypotheses, and (2) being efficient enough to implement tractable solutions.

A last refinement about framework characteristics concerns its first requirement and its ability to guide the modeler sufficiently to deal with “heuristic modeling hypotheses.” Trying to escape from discipline modeling techniques (usually inherited from analytical techniques) necessitates providing another computation-based technique. Modeling and simulation can be done based on data of one or many systems. The more experimental data are collected, the more “precise” or “certain” the simulation model can be.

Human knowledge about large (spatial) distributed systems is far from complete and actually uncertain (notably because human capabilities are limited). Considering uncertainty and the opposite "certainty", the Monte Carlo technique is a powerful means to make explicit the exploration of model parameters (Fishman, 2000). Probabilities and distribution laws are assigned to parameter values. Investigation of state spaces through parameter changes can be automatically achieved using experimental plans (Kleijnen, 1997; Amblard et al., 2003). Finally, a confidence interval can be obtained on probability results.

The topic of study in this chapter is large (spatial) interacting systems. In that context, we introduce: (i) efficient design strategies, (ii) stochastic modeling approaches, and (iii) the development of fire spread computational models (*vs.* physics-based models).

2. Design and performance balance

A good design should lead to an as simple and as evolvable as possible computational model. Good performances are, for large distributed systems, inversely proportional to the design quality. Performances require the removing and/or simplification of redundant and non efficient (even if reusable and more understandable) code or structures. A trade-off has to be made between efficiency and design quality. To optimally choose between corresponding solutions of efficiency and design, the activity paradigm can be used for both design (at modeling level) and efficiency (at simulation level) tasks.

2.1 Background on world-views and simulator efficiency

There are three common types of strategy to implement the kernels of discrete event simulations (Balci, 1998). These strategies, also called world views, consist of: *event-scheduling*, *activity-scanning* (including the three phases optimization), and the *process-oriented* strategy introduced by the Simula language (Dahl and Nygaard, 1966). A strategy makes certain forms of model description more naturally expressible than others. In all of these world-views, an event corresponds to an instantaneous change in the state of a system at a particular time. Event scheduling models work with pre-scheduling of all events without provision for activating events. In contrast, in the activity scanning approach, events can be conditioned on a contingency test in addition to being scheduled. A model is said to be active when its scheduling time has occurred and when its contingency test is satisfied. An optimization of the activity scanning strategy is named the three phases approach (Pidd, 1992; Coquillard and Hill, 1997). The interaction process world view is a combination of both event-scheduling and activity scanning strategies.

An additional classification consists of considering the two kinds of time management: discrete-time and discrete-event simulations. In discrete-time simulations, a clock advances the simulation by a fixed time step. At every time step, states are computed. In discrete-event time based management, events drive the simulation. The simulation time advances from one event time-stamp to another according to the events scheduled.

All these techniques and strategies focus on particular concepts (events, activities, and processes) and simulation time managements. On the other hand, source systems are also usually described through a system-based decomposition. According to interactions and autonomy of systems, systems are identified and connected. Discrete Event System Specification (DEVS) (Zeigler et al., 2000) is the soundest framework aiming at tackling computational systems, discrete-events, and simulation time managements. Through its specification hierarchy, depending on the observed behavior of the entire system, the internal structure is progressively described up to the definition of interfaces (or ports), in a modular way. At the final specification level, a sub-model is fully modular. At the previous level, the states of sub-models can directly influence the transitions of other sub-models. The choice of specification level is crucial in terms of reusability and efficiency. Currently, three kinds of approaches discuss both modularity and efficiency of DEVS models:

1. At the simulation level, (Wainer and Giambiasi, 2001) propose to flatten the hierarchy of cellular models. (Hu and Zeigler, 2004) aim at taking advantage of spatially distributed causal events in cellular models. (Muzy and Nutaro, 2005) suggest activity tracking mechanisms only focusing on active components, reducing the data structure overheads resulting from discrete-event managements.

2. At the compilation level, (Lee and Kim, 2003) only account for active event paths. Sub-systems that will not receive events or compute transitions during the simulation are not compiled. This results in a reduction of memory size and a simulation speed-up.
3. At the modeling level, wrapping and modularity are compared in terms of reusability and efficiency (Muzy et al., 2003; Shighina, 2006; Sun and Hu, 2008). Modularity increase reduces performances (because of discrete-event managements) but improves reusability. A weaker modularity (reducing state encapsulation) reduces the reusability of atomic systems, while improving performance. Obviously, the use of port interfaces induces an increase in the number of discrete-events and overhead of data structure management.

2.2 Modularity and efficiency

As depicted in Figure 1, various choices of modularity can be made for describing a spatial system. At the modeling level, models can be implemented in a modular or in a non-modular way. Modular models consist of cells whose state is encapsulated. Cells interact through interface ports and every cell is designed as receiving and sending events. Non-modular cells do not have ports. They directly access the state of influencing neighbors.

At the simulation level, a full aggregation (corresponding to a non-modular implementation) corresponds to a single simulator driven by the root coordinator. A distributed solution consists in assigning one simulator to every cell. This solution increases the number of messages exchanged (decreasing efficiency) but enhances reusability.

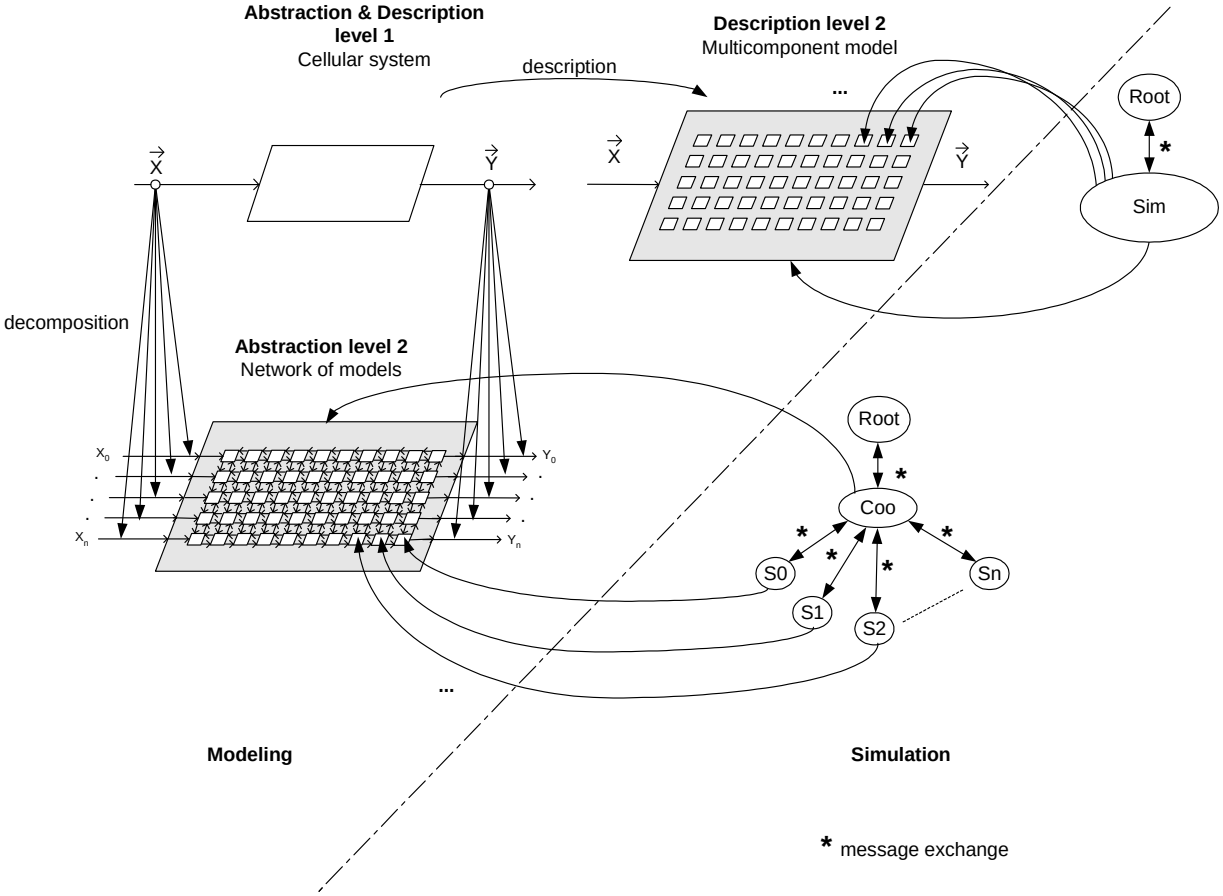


Figure 1. Comparison between modular and non-modular modeling and simulation (Muzy et al., 2003)

A first full modular implementation at both model and simulator levels has been implemented through the Cell-DEVS formalism (Wainer and Giambiasi, 2001) and the corresponding CD++ environment (Wainer, 2002). To illustrate some performance aspects, consider a laboratory experiment that consists of a

combustion table of 30 cm long and 60 cm wide for a line-ignition, the prediction of spread rate (2.96 mm/s). For a real propagation of 150 s of a 100 x 100 cell space, the execution time of the Cell-DEVS simulation was about 21h20min (with an old *Pentium III* at 500 Mhz). Even if this modeling and simulation experiment is elegant and grounded formally, the exchange of many messages produces significant overhead because of data structure management (of schedulers). This results in an execution time much greater than the actual propagation time.

On the other hand, a fully aggregated simulator and non-modular model has been implemented (Muzy et al., 2003). An ignition point has been simulated by initializing center cells with a temperature gradient. For a real propagation of 200 s, execution times decreased to 160 s. The drawback, however, is that the entire modeling and simulation model is less reusable.

Many choices can be considered at the modeling and simulation levels for enhancing efficiency. According to the degree of reusability requested, modularity can be achieved at the modeling and/or simulation levels. Reducing the degree of modularity reduces the degree of reusability of components. Full modular components are designed in an autonomous way. They react to and send external discrete-events. Take the example of a cellular automaton. In a non-modular cellular automaton, simple neighboring rules can be implemented for every cell as: “If my neighboring cell is alive, then I become alive.” In a modular cellular automaton, the specification is different. It would be: “If the message received from my neighboring cell indicates that it is alive, then I become alive.” The internal structure of non-modular models depends more directly on the state of its influencing models.

2.3 Activity paradigm

Beyond choosing the modularity of a model, the modeler can use a new technique to enhance efficiency: The activity paradigm. Using the latter, models and simulators are designed to automatically track activity (state changes of components.) Activity detection rules are added to models. Conversely, this technique increases explicit modularity and non-modularity aspects. Indeed, DEVS is a powerful framework, however its model and simulator structures have to be exploited to map activity of components. Sub-systems can be modeled as active or inactive, the entire simulation only directly tracking and computing activity while ignoring inactivity. Simulation resources can be characterized through activity comparing resource usages of modular and non-modular models .Let’s take a simple example of fire spread in a 100 x 100 cell space (*cf.* Figure 2.) Inactive cells (burnt in black or unburnt away ahead of the fire front) and active cells (of the fire front) can easily be identified.

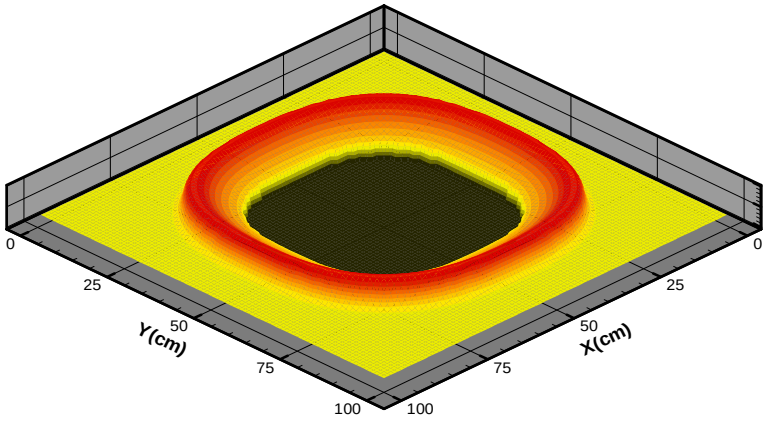


Figure 2. Fire spread example with an ignition point at the center of the square

Activity tracking consists in only including burning or heated cells in the main simulation loop. A mean relative activity measure can be studied:

$$Å = \frac{Activity_{int} + Activity_{ext}}{max(Activity_{int}).T_{sim}}$$

Where $Å$ is the percentage of mean activity measure relatively to $max(Activity_{int})$, which represents the maximum number of internal transitions (accounting for all active as well as inactive internal transitions of cells over the entire propagation domain), $Activity_{ext}$ is the number of external transitions, T_{sim} is the simulation time period.

Then, as depicted in Figure 3, applying this simple mean relative activity metrics (counting the number of external and internal transitions of components over a total simulation time of 221 s, which corresponds to the end of fire spread propagation when using a discrete-time step of 0.01 s) allows investigating modularity and activity tracking combinations—experimentally and theoretically:

- (no activity tracking, no modularity): A total of 10,000 internal transitions for all cells (active and inactive) are achieved at every time step. This corresponds to a maximum of $Å = 100\%$.
- (activity tracking, no modularity): The average number of active cells is experimentally determined (Muzy et al., 2008b): $Å = 23.29\%$.
- (activity tracking, modularity): $Å = 93.14\%$, simply multiplying the (activity tracking, no modularity) case by four (corresponding to the number of neighbors per cell).
- (no activity tracking, no modularity): $Å = 400\%$, simply multiplying the (no activity tracking, no modularity) case by four (corresponding to the number of neighbors per cell).

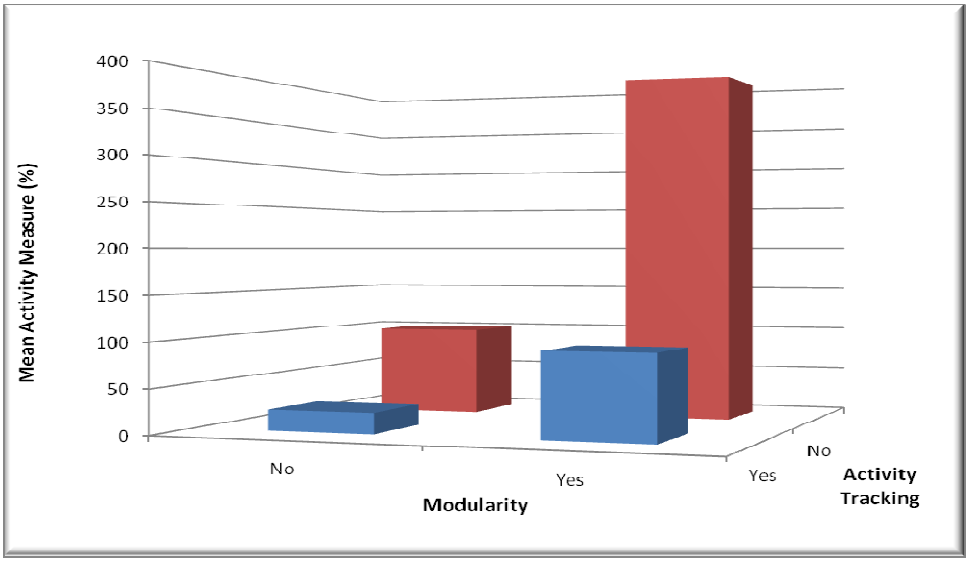


Figure 3. Activity and modularity comparison of efficiency

This simple comparison metrics pinpoints the efficiency of activity tracking approaches through modularity. One can notice that metrics for both (activity tracking, modularity) and (no activity tracking, no modularity) are very close. Notice that the efficiency of the (activity tracking, modularity) case will improve linearly with the number of inactive components. According to the superiority of the (activity tracking, no modularity) case, no doubts arise concerning the efficiency supremacy of this solution. Conversely, the (no activity tracking, modularity) case seems to not have any benefits that would merit an implementation.

Here are the lessons that can be learned from this simple example:

1. Do not use explicit discrete-event exchanges for massively parallel discrete-time implementation. Discrete-events should be used with a discrete-event (continuous) time base in mind.
2. Use activity tracking when the mean number of active components is low compared to the total number of components,
3. Discrete-event overhead can be compensated by a high number of inactive components.

More recently the activity tracking paradigm has been introduced (Muzy and Zeigler, 2008). This paradigm describes and proposes structures focusing on active sub-systems. Usual time flows and world-views are embedded. The formal description of DEVS models is discussed through activity-based simulation algorithms and new structures accounting for dynamic structures. Using the activity-tracking paradigm, components can be modeled and simulated in two steps:

1. The propagation activity is tracked. Information exchanged between components is routed and computed. The current set of active components is scanned. Events are routed and output transitions are computed. Final receivers are detected in the hierarchy using a recursive routing function (Muzy and Nutaro, 2005). The active set is then updated including imminent components for external transitions. The order of the active set depends on a tie-breaking function of imminent components (Zeigler et al., 2000).
2. According to current states and to new inputs, new states are computed. External and internal transitions (because of external and internal events) of active components are computed. Components changing state significantly are marked to be added to the new (ordered) active set. In a discrete-event driven simulation, the new active set corresponds to a scheduler and active components are marked to execute further their internal transition function (corresponding to an internal event occurrence). In this case, the current active (ordered) set is a sub-set of the scheduler, which corresponds to components active at the current simulation time.

Models can be designed for tracking activity during the entire simulation loop. Metaphorically, distributed sub-models (constituting the entire model) can be considered as filtering gates orienting and computing activity. As depicted in Figure 4, models are activated or not by the flows of external events or by scheduled internal events (bottom right corner). This activation depends on the adjustment of a sensitivity parameter determining the activation of the model. Activity in a network of models depends on: (i) state transitions or internal events, and (ii) external events received and sent.

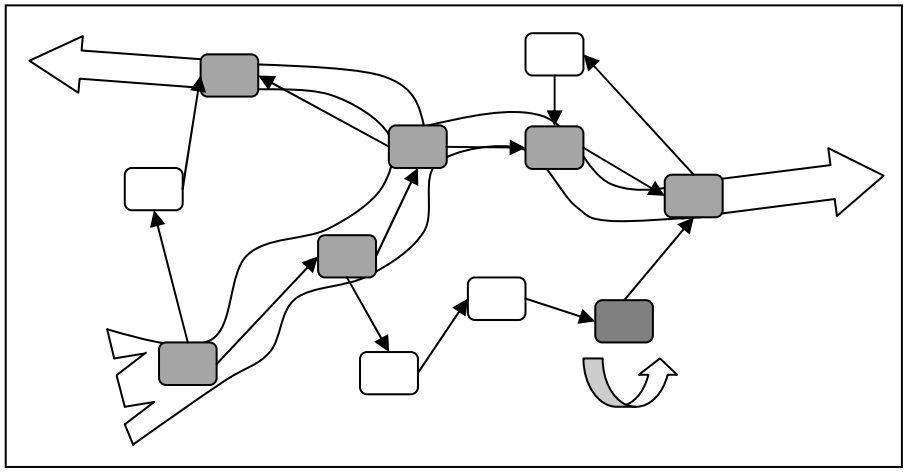


Figure 4. Active components

This algorithm makes explicit the distinction between activity tracking of (i) events, and (ii) components. Using only the second phase corresponds simply to a three-phase algorithm: The three-phase approach executes sequentially (Pidd, 1984): (1) the time-scanning (A Phase); (2) the bound-to-occur or book-keeping activities that represent the unconditional state changes (unconditional events) that can be scheduled in advance (B phase); and (3) the conditional or co-operative activities that represent the state changes that are conditional upon the co-operation of different objects or the satisfaction of specific (compound) conditions (C phase).

The activity paradigm renders explicit the many strategies offered to the modeller for the design of large (spatial) distributed systems. According to his own design and efficiency constraints, he can alternatively opt for tracking activity or not, as well as choosing to use modularity or not.

3. Modeling of stochastic aspects in (spatial) distributed systems

In many domains, stochastic aspects of spatial spreading have to be considered. At the beginning of the millennium, colleagues, like Lewis, were surprised that despite the recognized importance of stochastic factors, recent models for ecological invasions were almost exclusively formulated using deterministic equations (Lewis, 2000). In forestry, the forest growth can rely for instance on the distant spread of seeds. It has been shown that pioneering trees, whose seeds were propagated by heavy winds, do have an impact on the colonization of landscapes by particular species (Coquillard and Fain, 1995; Coquillard, 1995). In oceanography, the spreading of a tropical alga, *Caulerpa taxifolia*, introduced by mistake in the Mediterranean Sea is based on the spatial spreading of cuttings. This spreading is in essence stochastic, and occurs at a stochastic and discrete distance from an original point of settlement. Only stochastic models under spatial constraints were able to reproduce maps of the spreading of this alga (Hill et al., 1998).

In fundamental domains such as physics, stochastic and spatial modeling enables the understanding of phenomena in which determinism does not offer the same precision [e.g., the spreading of spiral waves (Jung and Mayer Kress 1995)]. Here, the effects observed can be understood as a generalization of the concept of stochastic resonance in spatially extended systems. In (Falcke et al., 2000) a study is documented of intracellular spreading of calcium-induced calcium release with the stochastic DeYoung-Keizer-model. The system under study presents a state characterized by backfiring. The backfiring occurs because the steadily propagating pulse solution undergoes a heteroclinic bifurcation (in dynamical systems this type of bifurcation is a global bifurcation involving a heteroclinic cycle which is an invariant set in the phase space of a dynamical system. The use of spatial stochastic modeling was also choosen to argue that quantum-gravitational fluctuations in the space-time background give the vacuum non-trivial optical properties that include diffusion and consequent uncertainties in the arrival times of photons, causing stochastic fluctuations in the velocity of light in vacuo (Ellis et al., 2000). In nuclear medicine, we have shown that the precise detection of small tumors with an error less than 10% can currently only be achieved by spatial Monte Carlo simulations (ElBitar et al., 2006).

Spatial simulations are generally displayed with computer graphics. Visualization and animation is beneficial because of the human ability to capture spatial relationships. However, in the context of stochastic simulations, when we observe the results of a single replication, it is very important to avoid concluding something with only a small sample of a complex random variable. This does not mean that the study of transient behavior is not useful, but this implies more complex computing when we want to observe the results of many replications. For instance, in the *Caulerpa taxifolia* spreading simulation, the spatial auto-correlation is strong since *Caulerpa taxifolia* contaminated zones tend to form aggregated spots. On the contrary, from one replicate to the other, peripheral spots distribution is completely different without apparent correlation. Thus, it can be interesting to sum up the results of a large number of replicates in a single representation of the frequencies. This constituting a Discrete Spectral Analysis we already defined (Hill, 1997 ; Hill et al., 1998). Spectral analysis helps us to point out areas that have a high probability of invasion by *Caulerpa taxifolia*. However, this leads to difficulties in result interpretation, since the visualized result is a sum in the space of possible solutions. Furthermore, the existence of a peak is not easy to analyze and sometimes needs deeper study to discover possible spatial attractors (Hill *et al.*, 1996). Indeed, this technique itself is not able to detect some multiple spatial attractors because of spatial correlations. Consequently, in some cases the system could significantly diverge from the predicted results. Unfortunately, no reliable mathematical or statistical techniques are available at this time to detect such circumstances. Accurate knowledge and the functional analysis of the ecosystems seem to be the only ways to avoid incorrect forecasts and interpretations.

4. Fire spread modeling through a virtual laboratory

In Physics, equation-based structures are used to represent fundamental identified physical mechanisms involved in both fire and diffusion processes. Parameters of equations have been (or are) identified through laboratory experiments. These parameters do not correspond directly to the parameters intuitively identified as influencing the fire spread (slope, wind, biomass, humidity). These parameters are split into many other parameters, which aim at describing very finely physicochemical mechanisms (gas dynamics, temperature

radiation, etc.) These models are more adapted to fine-grain physics-based laboratory experiments than actual fire spread.

Computer scientists have a different approach. First their laboratory consists only of one or many computers. They directly build models and simulate them. Then, they verify if the model behavior fits their expectations describing consistent dynamics. They calibrate their model by adjusting parameters and exploring parameter values through experimental plans. Afterwards, the first constructive experiments designed by computer scientists are described.

4.1 Fire spread modeling

Fire spread models are usually categorized into forest and urban fire spread models. Forest fires are considered here. Forest fire models can be separated into stochastic and deterministic models. Stochastic models aim at predicting the most probable fire behavior in average conditions. In contrast, in analytical models, the fire behavior is usually deduced from the deterministic physical laws driving the evolution of the system. Recently, several sophisticated models were proposed (Barros and Mendes, 1997; Karafyllidis and Thanailakis, 1997; Hernández Encinas et al., 2007; Yassemia et al., 2008) and successfully validated by comparison with real fires. All these models use either simple or Dynamical Structure Cellular Automata (DSCA).

With reference to deterministic models, based on Weber's classification (Weber 1990), three kinds of mathematical models for fire propagation can be identified according to the methods used in their construction. The first type of models are statistical models (McArthur 1966), which make no attempt at including specific physical mechanisms, being only a statistical description of test fires. The results can be very successful in predicting the outcome of fires similar to the test fires. However, the lack of a physical basis means that the statistical models must be used cautiously outside the test conditions. The second category of models is composed of semi-empirical models (Rothermel, 1972) based on the principle of energy conservation but which do not distinguish between the different mechanisms of heat transfer. Rothermel's stationary model is a one-dimensional model, in which a second dimension can be obtained using propagation algorithms (Richards, 1990) integrating wind and slope. Finally, physics models (Albini, 1985) integrate wind and slope effects in a more robust manner by describing the various mechanisms of heat transfer and production. Physical mechanisms are described using a chemical, thermal, and mechanical definition of basic fire phenomena. Hence, physical and semi-empirical models use the definition of basic fire phenomena to physically describe fire propagation.

With reference to stochastic models, few works are available. In (Hargrove et al., 2000), authors explore the results of various stochastic experimental designs (according to various classes of moisture and ignition probabilities based on percolation thresholds) of a replicated fire spread simulation. No fuel biomass is taken into account. Besides, authors argue that "each simulation was replicated five times", without substantiating the choice of the number of replications although serious studies of the stochastic variability of results have to be performed (Kleijnen and Groenendaal, 1992). In (Jimenez et al., 2007), a sensitivity analysis of Rothermel's model using Monte Carlo simulation is provided. In (Gu and Hu, 2008), Monte Carlo simulation is used to evaluate the error of fire spread data acquisition during a fire spread simulation. This error is then integrated in the fire spread simulation. On stochastic fire spread modeling two major approaches have emerged that must be mentioned. The first is a very precise mathematical framework for stochastic fire spread modeling (Vorob'ov, 1996). In the latter a very clear presentation of ellipse based stochastic model is presented. An interesting concept of random spread process is introduced. However, here too, no discussions on the stochastic variability of results, neither on biomass modeling, are provided. More recently, in the reference journal *Combustion and Flame*, a very interesting and promising article (Porterie et al., 2007) has been introduced (and apparently discussed with the Nobel Prize winner P.G. de Gennes). Small world networks, percolation, and stochastic simulations are designed. This demonstrates a new interest of the physicist community for computer-based techniques. Long range spotting is modeled and analytic physics-based equations are used to predict firebrand distances and numbers.

Concerning the discrete-event design of fire spreading models, much work can be cited:

- In (Vasconcelos et al., 1995), a first discrete-event design of Rothermel's model is proposed. Discrete-events correspond to the patch burning times provided by Rothermel's model. An

experimental frame is defined. This frame embeds ecological data of the usual fuel classes provided by Rothermel's model.

- In (Barros and Mendes, 1997), dynamic structure cellular automata are used to store in memory only the burning active cells obtained through Rothermel's model. A dynamic structure specification is provided. Memory reductions are discussed.
- In (Ameghino et al., 2001), a high-level specification of (Vasconcelos, 1995) is presented. The specification exemplifies the delay-focused and cell-focused macro-instructions of the Cell-DEVS formalism. As macro-specifications and instructions are automatically embedded in Cell-DEVS, a reduction in code size is obtained.
- In (Muzy et al., 2005), an intensive fine-grained semi-physical model of fire spread has been investigated through discrete-event design. Optimizations (at the implementation and specification levels) are investigated in (Muzy, et al. 2003) to account for dynamic structures and discrete-time modeling. Recently, in (Muzy et al., 2008b), the quantization technique has been investigated and a parallelization of the model has also been considered (Innocenti et al., 2009)
- In (Ntaimo et al., 2008), previous work in (Vasconcelos, 1995) has been worked out in detail and extended to many scenarios including wind, slope and fire fighting. Dynamic structures have been investigated in (Sun and Hu, 2007). A stochastic implementation comparing experimental and simulation data (obtained through Rothermel's model) is described.

The complexity of fire spread modeling introduces bottlenecks at both physical and computer modeling levels. At the physical level, physicists use concepts (convection, radiation, diffusion) and corresponding usual mathematical models (partial differential equations and ordinary differential equations) to describe fire spread models. After a discretization through usual numerical methods (Euler, Runge-Kunta, etc.), computer-scientists then use design techniques (object-oriented, discrete-events, meta-modeling, dynamic structure, etc.). They discuss design and execution time advantages of the techniques used. However, many scientific disadvantages emerge from this approach. The design techniques may evidence issues in an attempt to follow the evolutions of discretized physics-based models. A first drawback is that the computer design is totally dependent on discretization techniques [except perhaps now with quantization techniques: (Zeigler et al., 2000)]. A second drawback is that computer scientists usually do not master physics-based modeling techniques and physicists do not exploit the full advantages of computer-based modeling techniques that, moreover, they sometimes do not even know. A third drawback is that the filters constituted by both physics-based concepts and discretization techniques result in multidimensional parameters, which do not directly represent physical, topological, and biological properties of fire fronts and fuels. Modeling levels are not grasped in a single consistent and complete top-down approach (from problem to implementation). Using a consistent stochastic framework, embedding experimental plans, to model (spatial) distributed systems, scientists are able to master and experiment with the full modeling and simulation process (even if they usually need expert advice).

4.2 Experimental model 1: Near-to-near propagation including firebrands

Because of the many (spatially distributed) phenomena occurring uniformly and in parallel in a fire spread, cellular automata constitute an appropriate paradigm to model and simulate them. A first very simple program with a textual interface is sufficient to design a first fire spread model.

Cells of the cellular automaton hold a discrete state constituted of symbols as follows:

'.' : The cell is unburnt,
'*': The cell is burning,
'x': The cell is burnt.

Fire propagation is twofold:

1. Near-to-near propagation: Every burning cell gets one chance over eight (for the eight neighbors) to ignite a neighbor. In case of wind, neighbors are selected according to the strength and direction of the wind.

2. Firebrand propagations: Firebrands are the sparks, burning wood expulsed from burning trees during a fire spread, igniting other trees. Every burning cell produces one firebrand in the direction of the wind. The distance depends on the wind strength. Distances of the firebrand are calculated as follows:

$$\begin{cases} dx = i + w_x[\theta(\varepsilon + \vartheta)] \\ dy = j + w_y[\theta(\varepsilon + \vartheta)] \end{cases}$$

Where, dx and dy are the distances of firebrand projection in directions x and y, $\theta\epsilon[0,1]$ is a random real number, *(i,j)* are the coordinates of cells, w_x and w_y are the wind directions, ϑ is the wind strength and ε is a parameter adjusted to observe modification of fire spread.

This very simple model obtains consistent results as represented in Figure 5.

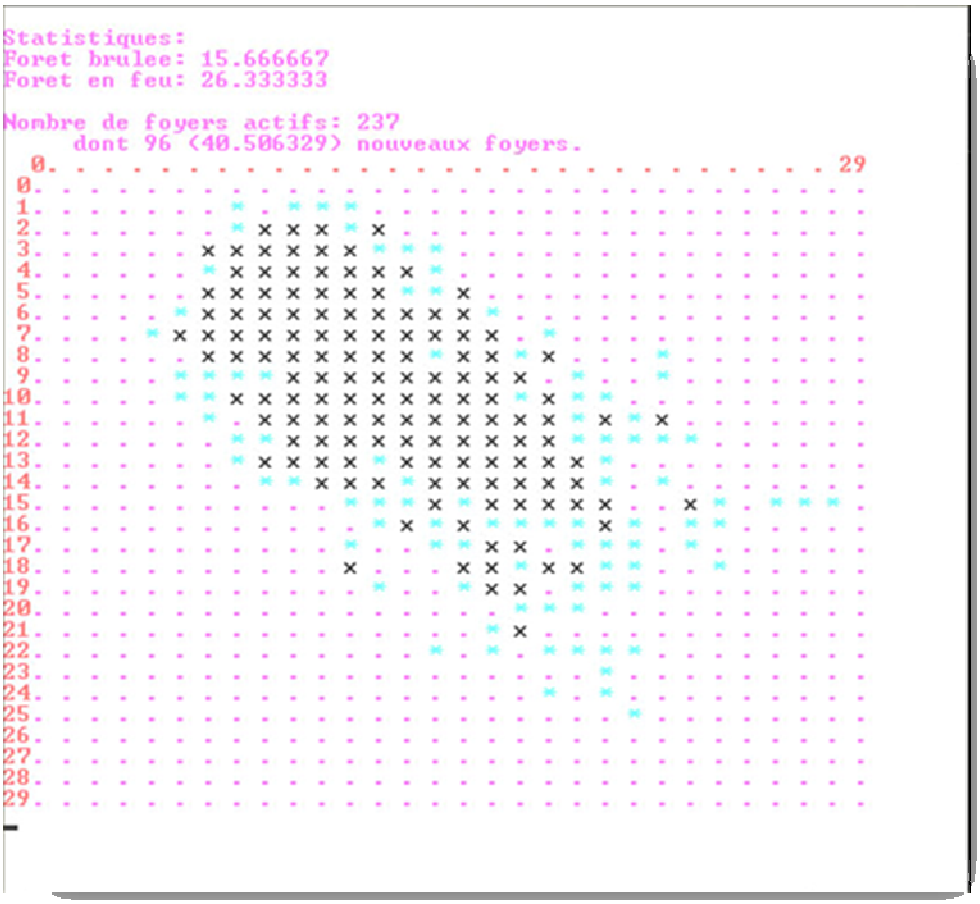


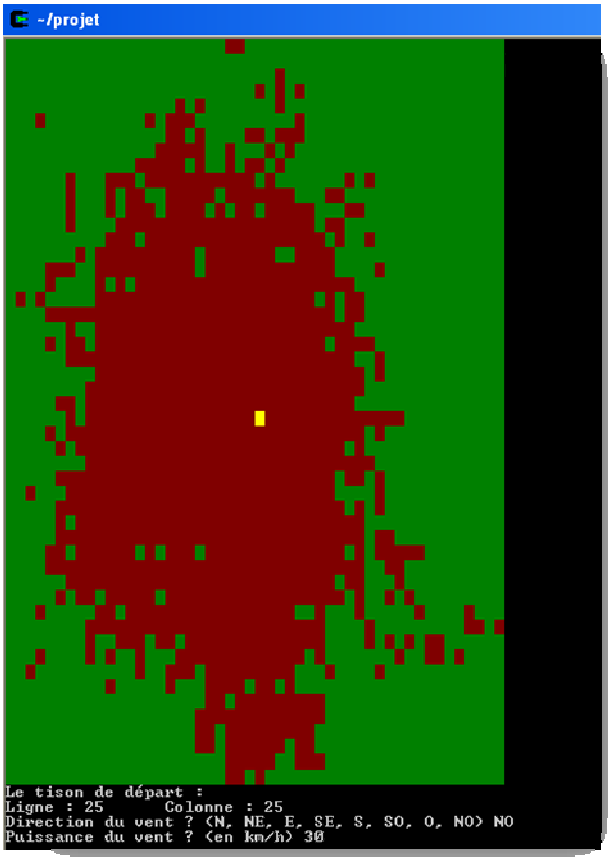
Figure 5. Fire spread under a wind of force 2 in South-West direction

4.3 Experimental model 2: Pseudo-random distributions of firebrands

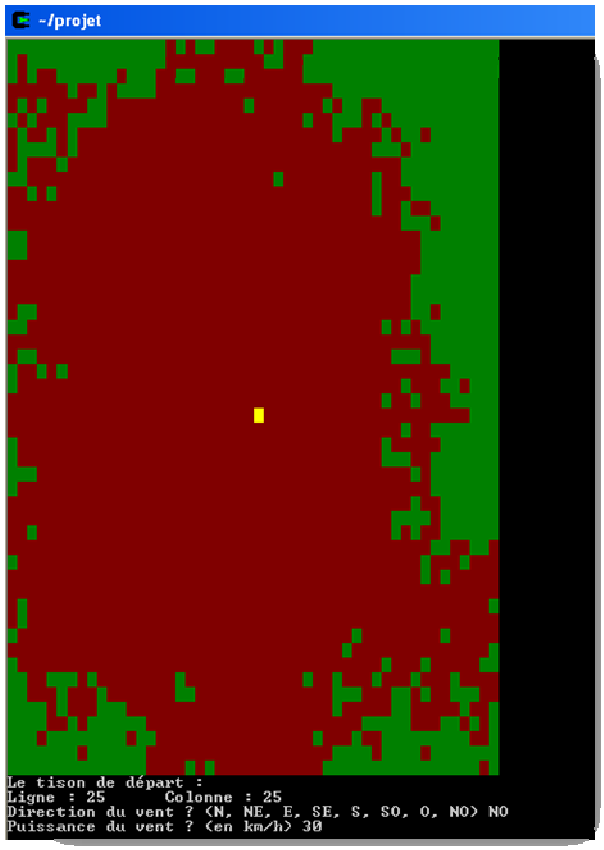
Here, cells have three states: unburnt, burnt, and burning. Parameters correspond to: (i) the strength of the wind, and (ii) the wind direction. The direction corresponds to the angle value with respect to a horizontal wind. Fire propagates only by firebrands. The distance *D* of projection is determined through a negative exponential law whose mean is proportional to wind strength. Thus, the greater the wind strength, the more distant would be the firebrand. The direction is calculated according to wind direction and to a noise parameter $\beta\epsilon[-1,1]$ through a Gaussian law of mean equal to zero and deviation inversely proportional to *D*.

Thus, when the distance is large, the firebrand direction corresponds to the wind direction (β is minimum). When the distance is small, the firebrand is projected in any direction. This latter case corresponds to a near-to-near propagation.

Results are presented in Figure 6. Some imperfections of the model can be observed, but remember that we observe only one sample of a complex stochastic variable. Some cells inside the fire front remain unburnt. However, the firebrand method provides acceptable results. As it can be noticed in Figure 6, the shape of the fire front is consistent with wind strength directions.



After 35 itérations



After 45 iterations

Figure 6. Fire spread under a wind of 30 km/h in North-West direction

4.4 Experimental model 3: A more sophisticated approach

The two previous simple models allowed to design and implement a more complete simulator. In the latter, the finite states used can be refined using a biomass decrease in cells. According to the level of degradation of cells, corresponding states are selected (unburnt, heated, burning, and burnt). A difficulty relates to the number of parameters to calibrate. This number needs to be limited to explore the full state space of the final model. This modelimplements the near-to-near propagation of model 1 and firebrand propagation of model 2.

A Model-View-Controller (MVC) design pattern is used to facilitate the development of a graphical user interface (as shown later) (Gamma et al., 1994). In addition the MVC design pattern allows:

- separating variable aspects from static aspects of the implementation,
-
- favoring composition over inheritance in this case where no classification is needed,

- programming with separated interfaces, reducing couplings between objects.

Entities of the fire spread simulator according to the Model-View-Controller are:

1. *Model*, which contains the entire simulation engine *View*, which provides the window interface,
2. *Controller*, which manag*es e*vents from the window interface and transfers changes from view to model.

We have also used an observer pattern to notify the view when we have a state change in the fire model. Indeed the observer patterns helps in maintaining a list of observers automatically notified of any state change. This is particularly useful in implementing distributed event handling systems. The strategy pattern has also been used to implement a dynamic swap of algorithm particularly when we tested different variants of fire spread. More details on this use for fire spread can be found in (Innocenti et al 2009).

The pseudo-random generator used is the Mersenne Twister (Matsumoto and Nishimura, 1997) from the SSJ Java library (L'Ecuyer and Buist, 2005). Figure 7 depicts the main diagram of the entire implementation. At the design level, the elements have been divided into burnable and unburnable. The class `Simulator` is composed of Tree(s) and Shrub(s) to be more reusable. In addition, replications are used for representing spectral analyses (Hill, 1997). By replicating simulations, the entire span of exponential and Gaussian laws can be explored. Every replication is saved. At the end of the simulation, spectral analyses are computed. Spectral analyses correspond to a spatial sum of probabilities of a fire spread experiment. In the Simulator class a method for image analysis is implemented. This method converts a plane or remote sensing image in a software `Element`(s) (with basic burnable and unburnable cell states).

The Simulator class has two main attributes:

- `Map` representing the propagation domain. It is an array of objects from class `Element`.
- `Element` is an abstract class implementing `Tree` and `Shrub` classes. Each pixel from the map corresponds to an element.

There are three main functions:

- `run()`, which corresponds to the main simulation loop (including resume actions.)
- `imageProcessing()`, which transforms picture pixels in an identified element (tree or shrubb) stored in a table of `Element(s)` of `Map`.
- `burnMap()`, which processes each Element in the Map that is in its 'burning' state'. Each call to `burnMap()` corresponds to an iteration of the simulator.

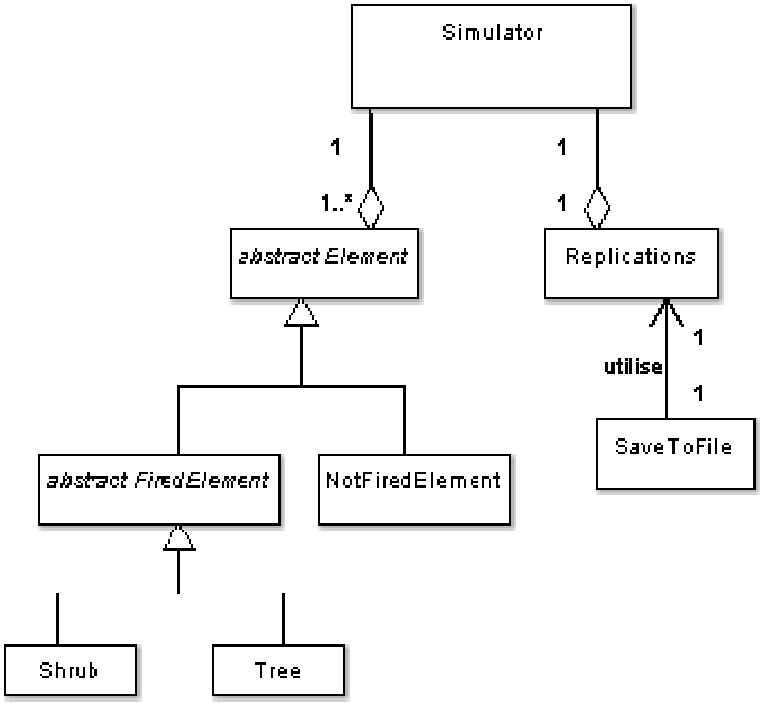
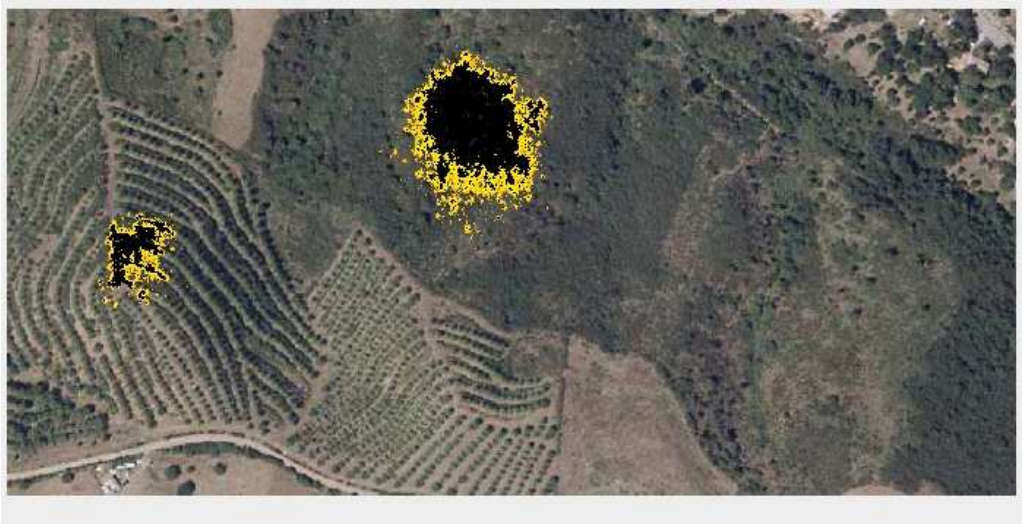


Figure 7. Simulator class diagram

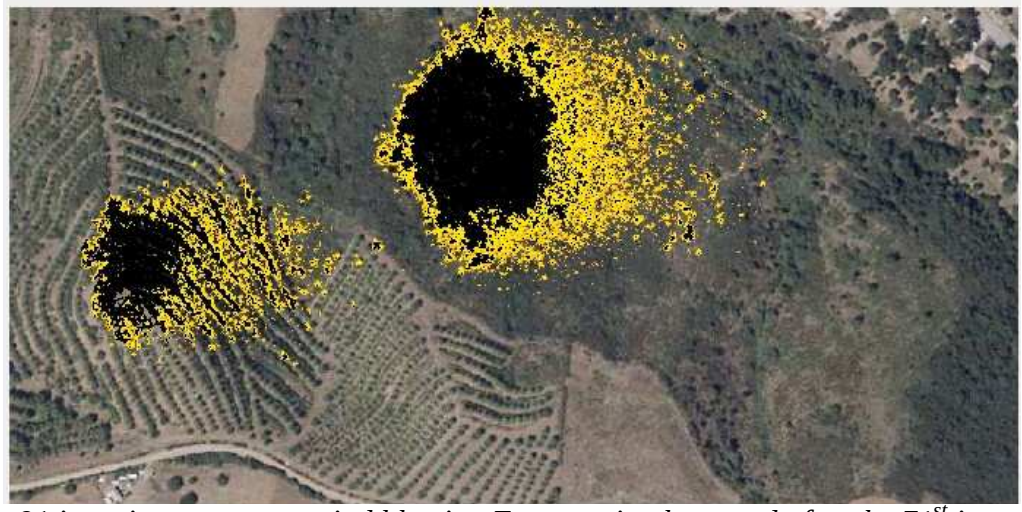
The structure of this simulator differs from the structure of (Muzy and Nutaro, 2005) in many aspects:

- Replications are implemented (i.e., rerun many simulations). Replication mechanisms still need to be investigated in order to be formally embedded in abstract DEVS simulators.
- No modularity is used for components.

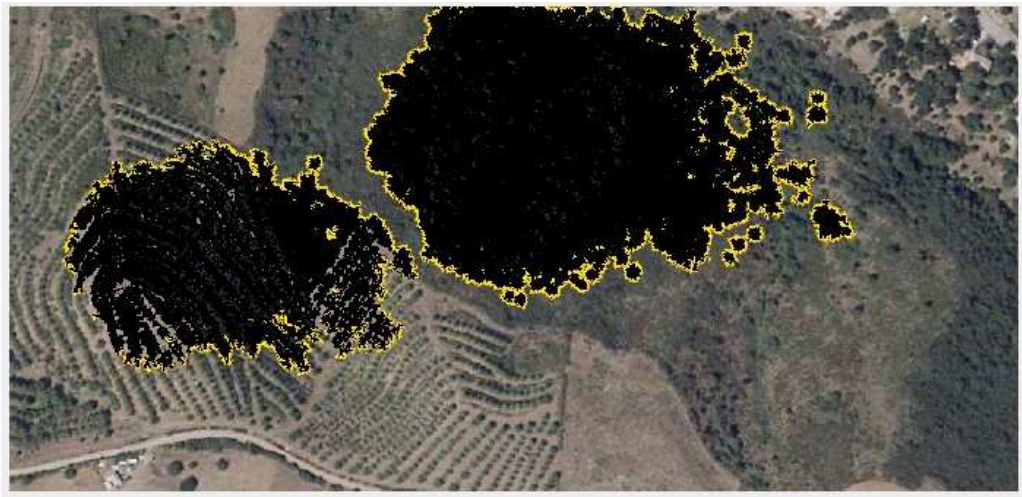
Parameters of the interface are: the strength and direction of the wind, as well as the real-time simulation speed. Actual aerial pictures can be uploaded. Figures 8 and 9 depict a fire spreading, somewhere on the island of Corsica. It can be noticed that the unburnable elements constituted by the road are not burnt.



After 71 iterations, with a slight wind blowing South, with two initial ignitions



After 91 iterations, a strong wind blowing East was implemented after the 71st iteration.

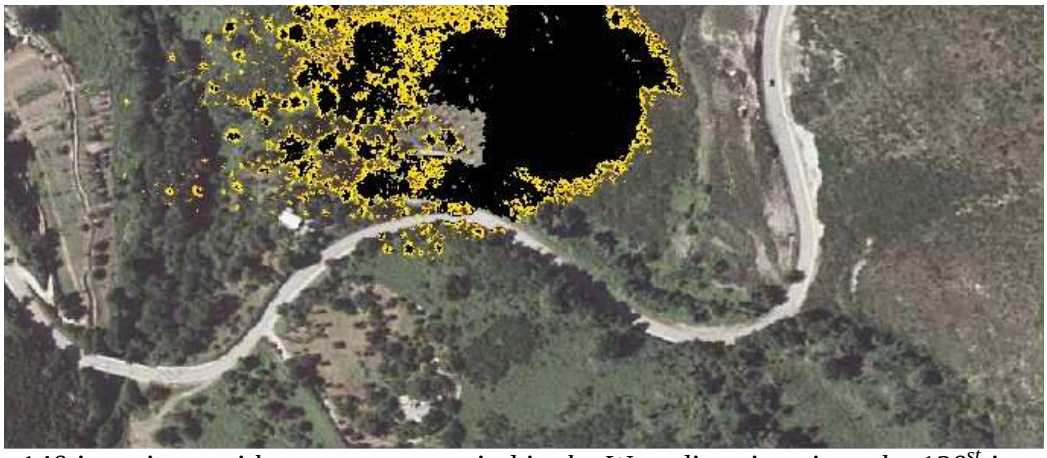


After 125 iterations, without wind since the 91st iteration

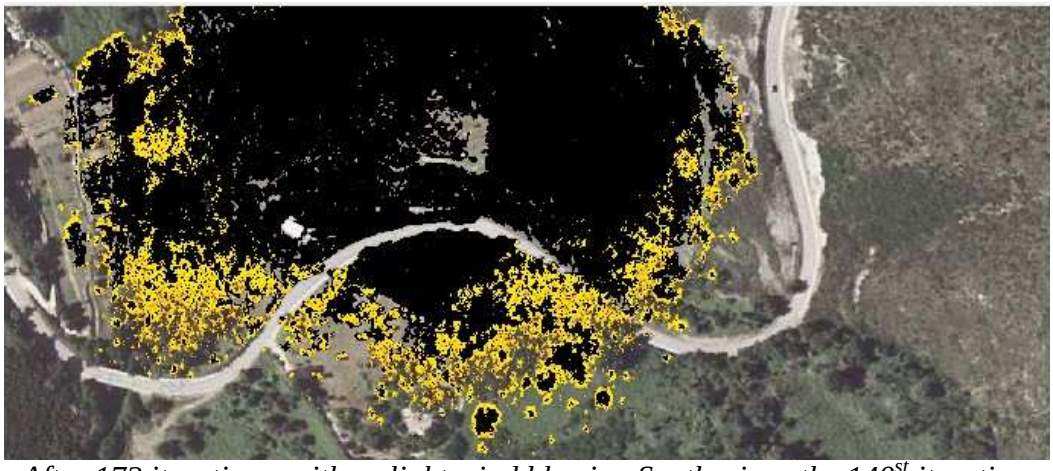
Figure 8. An example of fire spread in a visual interactive simulation

After 41 iterations, with a slight wind in South-West direction

After 120 iterations, without wind since the 41st iteration



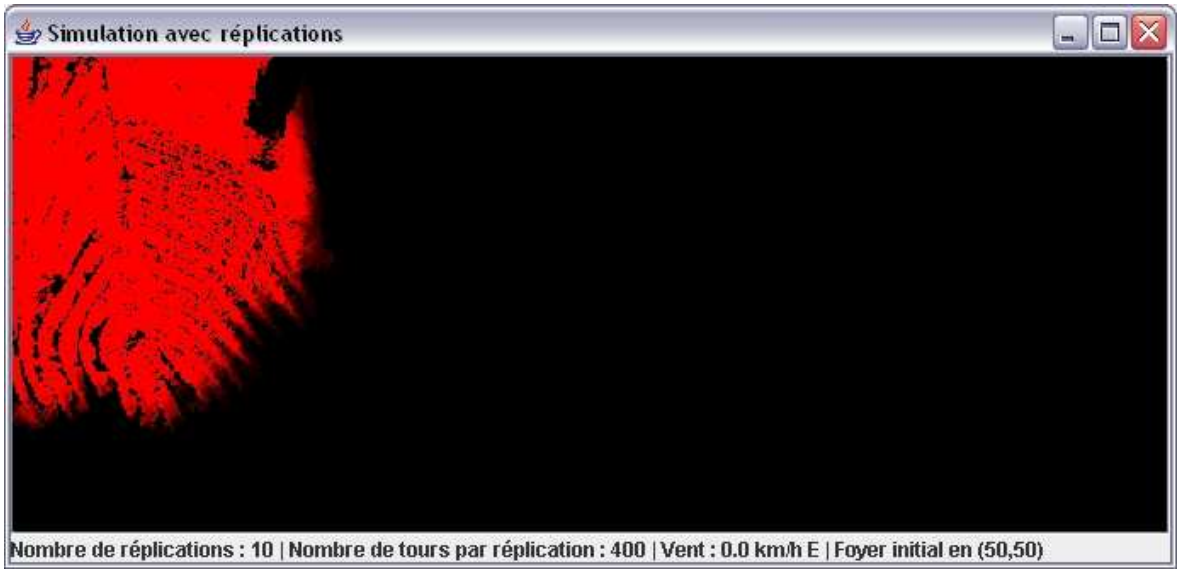
After 140 iterations, with a very strong wind in the West direction since the 120st iteration



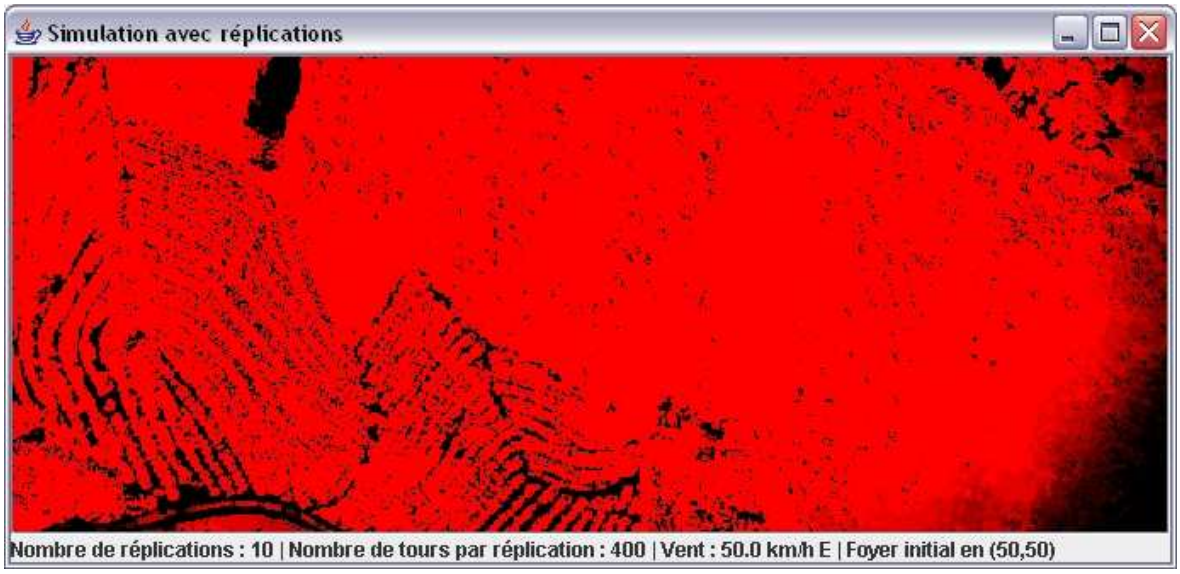
After 172 iterations, with a slight wind blowing South, since the 140st iteration

Figure 9. An example of fire spread. Notice that the fire does not burn the road except when there are trees over the road.

In the figure 10 below we present two screen shots with the sum of 10 replications of fire propagation. Of course, spatial spectral analysis of this kind is not performed when the interactive simulation mode is active (the only difference between two replications being the random stream)



(a)



(b)

Figure 10. (a) Ignition on the top left corner after 10 replications of 400 iterations each without wind (b) Ignition on the top left corner after 10 replications of 400 iterations each with a wind of 50 km/h in East direction

Lastly, Figure 11 represents possible fire propagation in 3D in a virtual environment (near the ISIMA building in France).

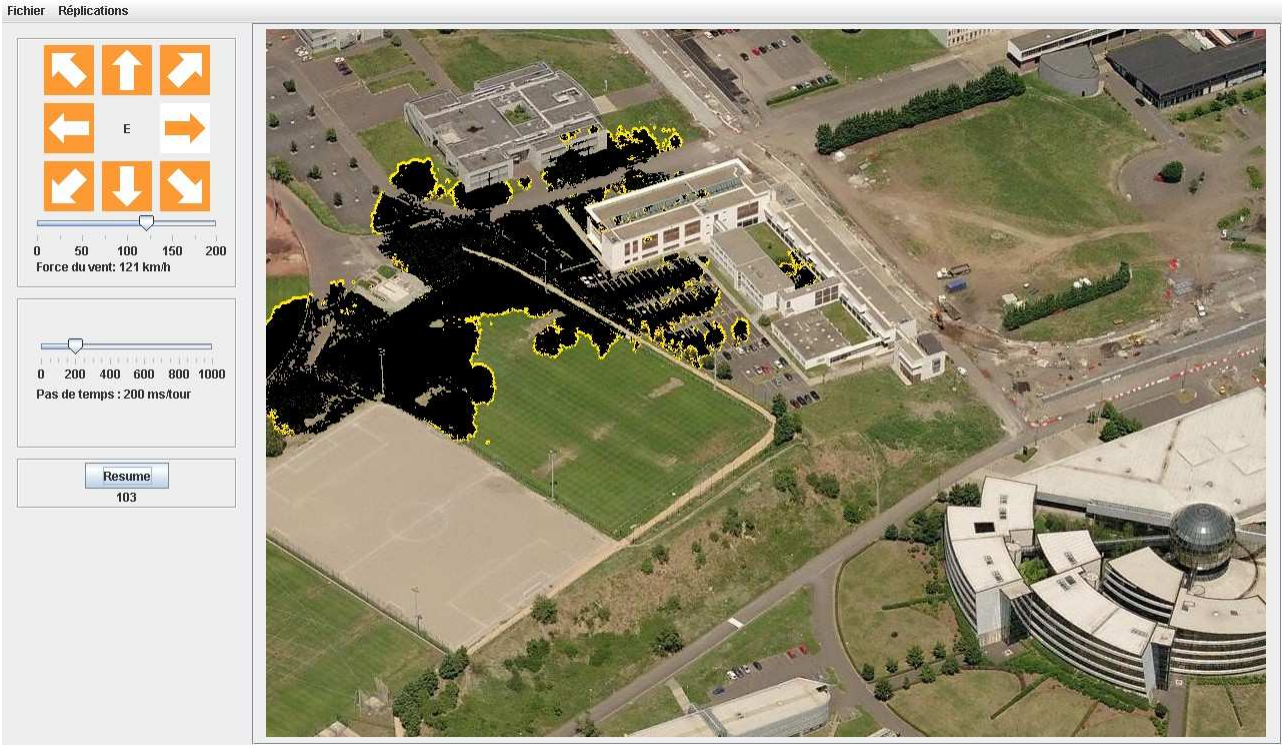


Figure 11. Wind in East direction of 121 km/h.

4.5 Efficiency and activity

We used the Test and Performance Tools Platform (TPTP) for the Eclipse development environment to test the execution time performances of the implementation. As depicted in Figure 12, we noticed that the refresh method of the simulator class lasts 0.8 s at each execution.

>Package	Base Time (seconds)	Average Base Time (seconds)	Cumulative ...	Calls
Element	17.836966	0.000003	17.836966	5272...
Element()	0.774271	0.000003	0.774271	239553
isActive() boolean	5.666641	0.000003	5.666641	1676...
isActiveNextTime() b	11.389609	0.000003	11.389609	3353...
setActive(boolean) v	0.000779	0.000004	0.000779	197
setActiveNextTime(b	0.005667	0.000003	0.005667	2061
Ellipse	0.001024	0.000008	0.001024	128
FiredElement	5.043482	0.000004	6.031027	1398...
Forest	19.780943	0.520551	40.985431	38
burn() void	5.926146	0.846592	12.207322	7
Forest(java.awt.imag	0.031390	0.031390	5.107648	1
imageProcessing(jav	1.569360	1.569360	5.076259	1
process() void	0.000320	0.000023	35.877760	14
refresh() void	12.253706	0.875265	23.670118	14
setDefaultAllParams(	0.000022	0.000022	0.000022	1
InfluencedArea	0.162283	0.000019	0.366520	8745
NotFiredElement	1.956256	0.000004	2.177088	553994
Shrub	0.100063	0.000006	0.299124	16295
Simulator	12.989674	0.000004	77.587404	3622...
Tree	1.050821	0.000007	2.523944	158321
erifon.util	1.615908	0.000062	9.802466	26017

Figure 12. Performances report of the simulation methods

The `refresh` method uses the `isActiveNexTime` method of the `Element` class to determine if this element will be on fire at the next iteration. If that is the case, the element is set to active. As it can be noted in Figure 13, this method is not optimized and every element of the propagation domain is tested at each iteration. However, remember the mean relative activity measure. Applying this measure, we find that Å *is below* 1% . Therefore, activity tracking is highly recommended here to enhance performances.

```
public  void refresh()
{
      for(int i=0; i<size_.width; i++)
      {
            for(int j=0; j<size_.height; j++)
            {
                  if(map_[i][j].isActiveNextTime())
                  {
                        map_[i][j].setActive(true);
                  }
            }
      }
}
```

Figure 13. refresh() non optimized method.

As described in Figure 14, a new method `setActiveNextTime()` is used to embed a HashSet containing the active elements is implemented. Then, the method `refresh()` only browses now active elements through the list `elementsActiveNextTime`. After implementing the activity tracking method, executions times have been reduced by a factor 400! Even with this kind of elementary modeling we see the value of the activity tracking approach.

```
public void setActiveNextTime(boolean inActive)
{
      activeNextTime_ = inActive;
      if(inActive  && (!container_.elementsActiveNextTime_.contains(this)))
      {
            container_.elementsActiveNextTime_.add(this);
      }
}
```

Figure 14. Activity tracking method

5. DEVS Discussion

In this chapter, first the activity paradigm was discussed for the design of large (spatial) distributed system. Through this paradigm, the modularity aspects and efficiency measures were explicitly articulated. Then, a non-modular activity tracking model of fire spread was implemented. Though more formal work is still required to extend usual DEVS abstract simulators to replication-based problems, advantages of stochastic modeling for implementing spatial constraints could be presented. In several cases, stochastic modeling is the only strategy available to represent the spatial distant and discrete interactions.

This worked leveraged the observation that formalization of DEVS for stochastic modeling is possible. In (Zeigler, 1976), a formal specification of pseudo-random generators is described. Pseudo-random generators being deterministic, DEVS describes stochastic models in terms of systems embedding pseudo-random generators. Recently, in (Kofman and Castro, 2006), a formal extension of DEVS has been provided. A DEVS-based description of stochastic models through STDEVS would allow the modeling and simulation community to formally and explicitly explore this long existing modeling technique. DEVS popularity to a large extent derives from the useful level of definition. It is abstract enough to allow the

development of new concepts and structures. It is also precise enough to avoid ambiguities and guide modelers. Since its first book definition (Zeigler, 1976), many developments have been achieved. A constant critical evaluation of these developments is necessary to enhance the entire framework. However, these developments have to be at the best level of abstraction. Because of the increasing complexity of hardware and software architectures, metalevels (and automatic mappings from higher to lower levels) need to be designed. Considering complex systems as multilevel and highly composed, detecting and mapping simulation structures on activity will increasingly be necessary. Activity tracking mechanisms will allow improving DEVS efficiency and abstraction.

Integrating explicitly modularity and efficiency issues, as well as experimental plans and stochastic modeling strategies in DEVS, (computer) scientists will gain freedom and dispose of a more complete and powerful modeling and simulation framework.

References

Albini, F. A., 1985. A model for fire spread in wildland fuels by radiation. Combustion Science and Technology 42: 229-258.

Amblard, F., Hill D., S. Bernard, J. Truffot, and G. Deffuant, 2003, MDA compliant design of SimExplorer, A software to handle simulation experimental frameworks, in Proc. Summer Simulation Conf. (SCSC) 2003, Montréal, QC, Canada, Jul. 20–24, 279–284.

Ameghino, J.; Troccoli, A.; Wainer, G., 2001. Models of complex physical systems using Cell-DEVS. Proceedings of Annual Simulation Symposium, Seattle, WA. U.S.A.

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

Balci O., 1988. The Implementation of four conceptual Frameworks for Simulation Modeling in High-Level Languages, Technical Report SRC-88-009, System Research Center, Virginia State University.

Barros, F. J. and M. T. Mendes, 1997. Forest fire modelling and simulation in the DELTA environment. Simulation Practice and Theory 5(3): 185-197.

Coquillard P., 1995, Simulation of the cyclical process of heathlands. Induction of mosaics structures, evolution to irreversible states, Ecological Modelling, Vol. 80, 97-111.

Coquillard P., Fain J., 1995. Monte-carlo simulation of colonization by pinus sylvestris of heathland on degenerate phase on volcanic substrate. In Bellan-Santini D., Bonin G., C. C. Emig (Eds), Functioning And Dynamics Of Natural And Perturbed Ecosystems,. Lavoisier Pub., Paris, 274-289.

Coquillard, P., Hill D., 1997, Modélisation et simulation d'écosystèmes — Des modèles déterministes aux simulations à événements discrets, Collection Ecologie, Masson, Paris, 1997.

Dahl, O. J., Nygaard K., 1966, Simula - An algol based simulation language. Communication of the ACM. Vol. n°9, 671-678.

El Bitar Z., Lazaro D., Breton V., Hill D., Buvat I., 2006, Fully 3D Monte Carlo image reconstruction in SPECT using functional regions. Nucl. Instr. Meth. Phys. Res. 569, 399-403.

Faure, T. and G. Deffuant, 2006, SimExplorer: A software tool for programming and executing experiemental designs on complex models," in Proc. Summer Simulation Multiconf., Calgary, Canada, 218–225.

Fishman G.S., 2000, Monte Carlo. Concepts, Algorithms and Applications, 2000, Springer.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J., 1994. Design patterns. Elements of Reusable Object-Oriented Software. Addison Wesley. ISBN:0-201-63361-2.

Gu, F., X. Hu, 2008, Towards Applications of Particle Filter in Wildfire spread Simulation, Proc. 2008 Winter Simulation Conference (WSC'08).

Hargove, W. W., R. H. Gardner, M. G. Turner, W. H. Romme and D. G. Despain, 2000. Simulating fire patterns in heterogeneous landscapes. Ecological Modelling 135: 243-263.

Hernández Encinas Hoya White S., Martín del Rey, A., Rodríguez Sánchez G., 2007. Modelling forest fire spread using hexagonal cellular automata. Applied Mathematical Modelling 31: 1213–1227.

Hill D., Object-Oriented Analysis and Simulation, Addison-Wesley Longman, 1996, 291p.

Hill, D., Mazel, C. and Coquillard, P., 1996. Integrating V&V in the object-oriented life cycle of ecological modelling simulation projects. In proceedings of the European Simulation Symposium 96, Oct. 24-26, Genova. Italy. Vol II, pp. 21-25.

Hill D., Coquillard P., De Vaugelas J., Meinesz A., 1998, An algorithmic Model for Invasive Species Application to Caulerpa taxifolia (Vahl) C. Agardh development in the North–Western Mediterranean Sea. Ecological modelling, Vol. 109, pp. 251-265.

Hill D., 1997, Modélisation des processus d'expansion : application à Caulerpa taxifolia", Conférence Internationale sur la " Dynamique des Espèces invasives ", 13-15 mars 1997, Paris, Académie des Sciences. Tec & Doc, 219-230.

Hu, X., and B. P. Zeigler, 2004. A high performance simulation engine for large-scale cellular DEVS models. High Performance Computing Symposium (HPC'04), Advanced Simulation Technologies Conference (ASTC), 2004, p. 3-8.

Hu, X., B. P. Zeigler, 2004. A high performance simulation engine for large-scale cellular DEVS models. High Performance Computing Symposium (HPC'04), Advanced Simulation Technologies Conference (ASTC), Arlington, USA, 3-8.

Innocenti, E., Muzy, A., Aïello, A., Santucci, J., F., 2004. Discrete Event Simulation of fire spread: A modular Approach for the Choice of a Strategy of Active Elements Management, In proceedings International Carpathian control conference ICCC'2004, Zakopane Poland, 69-74.

Innocenti, E., Muzy, A., Hill, D., Aïello, A., Santucci, J.F., 2004. Active-DEVS : A computational model for the simulation of fire propagation. In proceedings IEEE, SMC 2004, International Conference on Systems, Man and Cybernetics, October 10-13. The Hague, The Netherlands, 1857-1863.

Innocenti E, Silvani X., Muzy A., Hill D., "A software framework for fine grain parallelization of cellular models with OpenMP: Application to fire spread", Environmental Modelling & Software, Vol 24, 2009, pp. 819–831.

Jimenez, E., Hussaini, M.Y., and Goodrick, S., 2007. Quantifying parametric uncertainty in the Rothermel model, International Journal of Wildland Fire 17(5) 638–649.

John Ellis, N. E. Mavromatos1, and D. V. Nanopoulos, 2000, Quantum-Gravitational Diffusion and Stochastic Fluctuations in the Velocity of Light, General Relativity and Gravitation, Volume 32, Number 1 / janvier, 127-144

Karafyllidis, I., Thanailakis, A., 1997. A model for predicting forest fire spreading using cellular automata. Ecological Modeling 99, 87-97.

Kleijnen J. Groenendaal W. V., 1992. Simulation a Statistical Perspective. Wiley.

Kleijnen J.P.C. ,1987, Statistical tools for simulation practionners. Marcel Dekker Inc. Pub., New York, USA.

Kofman, E. and Castro, R., 2006. STDEVS. A Novel Formalism for Modeling and Simulation of Stochastic Discrete Event System. Technical Report LSD0603, LSD, Universidad Nacional de Rosario. Accepted in AADECA 2006.

L'Ecuyer P. and Buist E., 2005, Simulation in Java with SSJ, Proceedings of the 2005 Winter Simulation Conference, 611-620.

Lee, W. B., and T. G. Kim, 2003. Simulation speedup for DEVS models by composition-based compilation. Summer Computer Simulation Conference, 395-400.

Lewis, M.A., 2000, Spread rate for a nonlinear stochastic invasion, Journal of Mathematical Biology, Volume 41, Number 5, 430-454.

Martin Falcke, Lev Tsimring, and Herbert Levine, 2000, Stochastic spreading of intracellular Ca2+ release", Physical Review, 62 (2) 2636-2643.

Matsumoto M. and Nishimura T. Mersenne Twister, 1997, A 623-dimensionally equidistributed uniform pseudorandom number generator, Proceedings of the 29th conference on Winter simulation, 127-134.

McArthur, A. G., 1966. Weather and grassland fire behaviour. Australian Forest and Timber Bureau Leaflet 100.

Mell, W.E., Jenkins, M.A., Gould, J. and Cheney, P. 2007. A Physics-based approach to modeling grassland fires. International Journal of Wildland Fire 16(1) 1-22.

Morvan D., Dupuy J. L., 2004. Modeling the propagation of a wildfire through a Mediterranean shrub using a multiphase formulation, 2004. Combustion and flame. 138 (3) 199 – 210.

Muzy A., E. Innocenti, G. Wainer, A. Aiello, and J. F. Santucci, 2002. Cell-DEVS quantization techniques in a fire spreading application, Winter Simulation Conference (WSC) - Exploring new frontiers, IEEE/ACM/SIGSIM/SCS, San Diego, USA, 542-549.

Muzy, A. , E. Innocenti, G. Wainer, A. Aiello, and J. F. Santucci, 2005. Specification of discrete event models for fire spreading, Simulation: transactions of the society of modeling and simulation international, SCS, 81, 103-117.

Muzy, A. , G. Wainer, E. Innocenti, A. Aiello, and J. F. Santucci, 2002. Comparing simulation methods for fire spreading across a fuel bed, Simulation and planning in high autonomy systems conference (AIS), SCS, Lisbon, Portugal, 219-224.

Muzy, A. and B. P. Zeigler, 2008a. Introduction to the Activity Tracking Paradigm in Component-Based Simulation."The Open Cybernetics and Systemics Journal 2: 48-56.

Muzy, A., E. Innocenti, D. Hill and J. F. Santucci, 2003. Optimization of cell spaces simulation for the modelling of fire spreading. 36th Annual Simulation Symposium, Orlando, USA, IEEE/SCS/ACM, 289-296.

Muzy, A., J. J. Nutaro, 2005. Algorithms for efficient implementation of the DEVS & DSDEVS abstract simulators. 1st Open International Conference on Modeling and Simulation (OICMS), Clermont-Ferrand, France.

Muzy, A.,J. Nutaro, B.P. Zeigler, and P. Coquillard, 2008b. Modeling and simulation of fire spreading through the activity tracking paradigm. Ecol. Model., 219(1-2). p. 212-225.

Natimo L., X. Hu, Y. Sun, DEVS-FIRE: Towards an Integrated Simulation Environment for Surface Wildfire Spread and Containment, Simulation, 84 (4) 137-155, 2008.

Ntaimo, B. P. Zeigler, et al. 2004. Forest fire spread and suppression in DEVS. Simulation. 80 479-500

Peter Jung and Gottfried Mayer-Kress, 1995, Spatiotemporal Stochastic Resonance in Excitable Media, Phys. Rev. Lett. 74, pp. 2130 – 2133.

Pidd, M., 1992, Object-orientation and three phase simulation, in: Winter Simulation Conference, 689–693.

Porterie P., Zekri N., Clerc J.P., Loraud J.P., Modeling forest fire spread and spotting process with small world networks, 2007. Combustion and Flame. 149 (1-2) 63-78.

Posse, E., Muzy, A., Vangheluwe, H. 2006. "A framework for visual specification and simulation of cellular systems". In Proceedings of the 2006 DEVS Integrative M&S Symposium (DEVS'06). 2006.

Richards, G. D., 1990. An elliptical growth model of forest fire fronts and its numerical solution. International Journal of Numerical Method Engineering 30: 1163-1179.

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

Shiginah, F. A. S. B. 2006. Multi-Layer Cellular DEVS Formalism for Faster Model Development and Simulator Efficiency. PhD Thesis. Electrical and Computer Engineering Dept., University of Arizona.

Sun, Y., X. Hu, 2008, Partial-modular DEVS for improving performance of cellular space wildfire spread simulation, Proc. 2008 Winter Simulation Conference (WSC'08).

Sun, Y., X. Hu, Performance Measurement of DEVS Dynamic Structure on Forest Fire Spread Simulation, Proc.14th AI, Simulation and Planning in High Autonomy Systems (AIS 2007), 2007.

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

Vorob'ov, O. Yu. , 1996. Random set models of fire spread ***Fire Technology***, Springer Netherlands, *Volume 32, Number 2 / avril 1996*.

Wainer, G. 2002. CD++: a toolkit to develop DEVS models. Software—Practice & Experience 32 (13).

Wainer, G. and N. Giambiasi 2001. Application of the Cell-DEVS paradigm for cell spaces modeling and simulation. Simulation 76(1): 22-39.

Weber, R. O., 1990. Modelling fire spread through fuel beds. Progress in Energy and Combustion Science 17: 67-82.

Yassemia S., Dragićević S. ,Schmidt M., 2008. Design and implementation of an integrated GIS-based cellular automata model to characterize forest fire behaviour. Ecological Modelling 210 : 71–84.

Zeigler, B.P., 1976. Theory of Modelling and Simulation. Wiley Interscience, first edition.

Zeigler, B.P., Praehofer, H., Kim, T.G., 2000. Theory of Modelling and Simulation, second ed. Academic Press.