COMPONENT-BASED SIMULATION FOR SPATIAL COMPLEX SYSTEMS IN VLE ENVIRONMENT

Paul-H. Martelloni

UMR SPE CNRS 6134
University of Corsica
Campus Grimaldi
Corti, 20250, France
martelloni_ph@univ-corse.fr

Gautier Quesnel

INRA, MIAT
24 Chemin de Borde Rouge
Auzeville CS 52627 31326
Castanet Tolosan CEDEX, France
gauthier.quesnel@inra.fr

Eric Innocenti

UMR SPE CNRS 6134
University of Corsica
Corti, 20250, France
eric.innocenti@univ-corse.fr

Paul-A. Bisgambiglia

UMR SPE CNRS 6134
University of Corsica
Corti, 20250, France
bisgambiglia@univ-corse.fr

Pierre R Gonsolin

Corsica Institute of Technology
University of Corsica
Corti, 20250, France
gonsolin@univ-corse.fr

Paul Bisgambiglia

UMR SPE CNRS 6134
University of Corsica
Corti, 20250, France
bisgambi@univ-corse.fr

ABSTRACT

This work deals with the simulation of spatial complex systems resolved with cellular models. We compare two component-based modeling implementations, based on DSDE and Multicomponent formalisms, using a Virtual Laboratory Environment framework (VLE) that is based on the DSDE formalism.

Keywords: Multicomponent, DSDE, CAM, VLE, Simulation.

1 INTRODUCTION

Spatial complex systems with Cellular Automata Models (CAMs) are the main topic of the current paper and can be found in the field of the Theory of Modeling and Simulation (TMS). We are comparing two modeling approaches based on components. A first approach is inspired by the DSDE formalism, proposed by (Barros 1997). A second one is based on Multicomponent System Specification (MC), proposed by B.P. Zeigler, in his book (Zeigler, Praehofer, and Kim 2000), and is implemented with a Virtual Laboratory Environment

(VLE). Both ensure a reliable and efficient computer modeling design. We propose their implementation in an object-modeling framework in order to simulate fire spreading. The proposed modeling has to grasp the complexity of describing spatial complex systems thanks to the use of components. DSDE describes the complex system as an assembly of Atomic DEVS components whereas MC formalism describes a complex system as an assembly of non-modular components, i.e. not relying on ports nor on message exchanges. Since a VLE does not natively contain an MC extension, we are taking this opportunity to develop it. This paper is organized as follows: in a second part, we present the the background; in a third part, we present the adaptation we porpose for VLE. In a fourth part, we use these simulators to simulate fire spreading. We discuss our work in part fifth, before bringing the paper to a conclusion.

2 BACKGROUNG

2.1 Multicomponent modeling

2.1.1 Concept of component

We distinguish two component concepts, namely component modeling, defined as prior conceptual model, and software component as the programming paradigm. There is a broad literature describing these concepts, much of the works having used components in their tools and frameworks (Buss 2002, Buss 2000, Dalle et al. 2008, Gokhale et al. 1998, Hu et al. 2005). They have been built upon the two previous concepts which are complementary to one another. In fact, there is no real consensus to define a software component. Various academic and industrial definitions are discussed in (Peschanski et al. 2000). In the context of this work, we retain the following component definition: *"A reusable software component that can be manipulated visually in a builder tool"* (Oracle. 2017). The component approach allows to dissociate system representation into two entities, a structural part on the one hand (granularity, coupling, hierarchy), and a behavioral part on the other one. We can find these different concepts in the DSDE formalism through atomic and coupled model definitions (modular approach), but also in the Multicomponent MC formalism, thanks to interacting component definition (non-modular approach).

2.1.2 Multicomponent Model

In the literature, the MC formalism proposed by B.P. Zeigler in 2000 (Zeigler, Praehofer, and Kim 2000) clearly is the most appropriate framework when it comes to modeling a spatial complex system according to a discrete time approach based on interacting components (Shiginah 2006). The MC formalism allows to express in great detail the interactions between the elements of a component-based model (CA, CAM, SMA, etc.). Formally, it describes a set $M_d|d \in D$ of indexed components. Each d component has its own set of states Q_d, a state transition function Δ_d, and an output function Λ_d. A given d component can be influenced by a set of I_d of the model's other components, which are called influencers. An M_d component can also influence other E_d components of the model thanks to its transition state function, which are influencees. Lastly, each M_d component can be influenced by inputs and contributes to the model's overall response. On the basis of these mechanics, a non-modular MC model (Here, we have not retained the modular definition, i.e. whose using communication ports between components.) is a structure:

$$MC = \langle T, X, \Omega, Y, D, \{M_d | d \in D\} \rangle \tag{1}$$

where:

- T is the time base;
- X is the set of input values;
- Ω is a set of admissible inputs, as $\Omega \subseteq (X,T)$;
- Y is the set of output values;
- D is the set of components references or their labels;
- $M_d|d \in D$ is the component set of the Multicomponent.

For each M_d component of $d \in D$ index:

$$M_d = \langle Q_d, E_d, I_d, \Delta_d, \Lambda_d \rangle \tag{2}$$

where:

- Q_d is the set of d index component states;
- $I_d \subseteq D$ is the set of components which influence d, referred to as influencers;
- $E_d \subseteq D$ is the set of components influenced by d, referred to as influencees;
- $\Delta_d : \times_{i \in I_d} \times Q_i \times \Omega \rightarrow \times_{j \in E_d} Q_j$ is the local transition function of d;
- $\Lambda_d : \times_{i \in I_d} \times Q_i \times \Omega \rightarrow Y$ is the local output function of d, it is common to

When its comes to modeling a spatial complex system, it is common to specialize MC Approach according to CAM schema.

2.2 Cellular Automata Modeling

Cellular Automata Models are MC models, where basic components are called cells. They permit the description of components interaction through multiple levels of abstraction. They have been considered as specializations of the Cellular Automata (CA) basic conceptual model, proposed by Stanislas Ulam and John von Neumann, at the end of the 1940s (Von Neumann, Burks, et al. 1966). They allow to extend the basic model's functionalities, in particular when it comes to expressing both complex dynamical structures and hierarchical behaviors. For that, they have an abstract hierarchy which may extend over multiple levels. Although theoretically hierarchy levels can be infinite, two levels are usually considered: a global level and a local level. The global level helps to describe the global behavior of a considered complex system. For example, in the case of fire spreading simulation, a propagation rule describes behavior on a global level (the global state S_t representing the fire front phenomenon in its global form), from the representation of a component subset situated at a lower hierarchy level which is the local level. The local level describes individual components of each model component. The local behaviour expression also integrates the concepts of I_d influencer and E_d influencee, particularly described in MC models. The latter allows to accurately express the interactions that can exist between CAMs' components. A CAM is a structure:

$$CAM = \langle Z_d, S, N \subseteq Z_d, \delta, \lambda \rangle \tag{3}$$

where:

- Z_d is the discrete time space of d dimension;
- S is the set of possible states of a cell component;
- N is the cell neighborhood of a c cell in the discrete d space considered (neighboring cells);
- δ_{int} is the global state transition function ("transition rule" + "propagation rule");

- $\lambda$is the output function which returns the model state at t instant;

For each cell M_c of coordinate $(x,y) \in \mathbb{N}$

$$\forall M_c \in CAM, M_c = \langle Q_c, E_c, I_c, \delta_c, \lambda_c \rangle : \tag{4}$$

- Q_c is the state of the c cell in the Z_d domain of the cellular model;
- $I_d \subseteq D$ is the set of cells influencing d, referred to as influencers;
- $E_d \subseteq D$ is the set of cells influenced by d, referred to as influencees;
- δ_c its the local state transition function of the cell;
- λ_c is the output function of the cell which returns the state value of the cell.

2.3 DSDE

Introduced in the middle of the 1990's (Barros 1995) DS-DEVS (Dynamic Structure DEVS) is a formalism using the same models as classic DEVS, but the coupled models structure can change during the simulation. At the end of the 1990's (Barros 1997) (Barros 1998) DSDE is proposed. It is an evolution of DS-DEVS but it is based on PDEVS, and takes adventage of the parallel management of PDEVS. Abstract simulators and the closure under coupling are detailed by (Barros 1995, Barros 1996, Barros 1997, Barros 1998).

2.4 VLE framework

Virtual Laboratory Experiment (VLE) is a DSDE (Barros 1998) modeling and simulation environment written in C++, under GPLv3 license, developed at the INRA Lab in Toulouse, France, a little less than ten years ago (Quesnel, Duboz, and Ramat 2009). We use version 2.0 in this work that is still under development (alpha version). VLE provides several extensions as Petrinet, Ordinary Differential Equations, or even Cell-DEVS (Wainer and Giambiasi 2001) to describe CAMs. VLE not natively having MC extension, we make use of this work to develop it.

DSDE simulator consists in executing δ_{Int}, δ_{Ext}, δ_{con} and λ functions, as well as managing time advance thanks to ta function (cf. Figure 3). Each atomic model interacts with other models through its entering event, and its output function λ.

3 VLE ADAPTATION

The modification of VLE to adapt it to the MC approach led us to add elements to the basic API and modify others. The basic API includes two parts: modeling and simulation. The modeling components include classes such as BaseModel, AtomicModel and CoupledModel, which make it possible to represent the structure of the model as objects (cf. figure 1). The simulation components contain classes such as Coordinator, Simulator and Dynamics for the simulation algorithm. To integrate our approach, we added two new model types into the modeling part Multicomponent type models and Component type models, thus we created 2 classes: "Component" and "Multicomponent". Like the Atomic models, they inherit BaseModel as shown in figure 1. To integrate our approach, the principle of MC simulation according to Zeigler requires us to implement a special MC simulator which differs a little from the VLE DSDE simulator.. For the sake of code scalability, we created an abstract class named "Simulator" and renamed the DSDE simulator "AtomicSimulator", a class derived from "Simulator". We finally created a "MCSimulator" class, also derived from "Simulator", and implemented our multicomponent simulator that mixes the roles

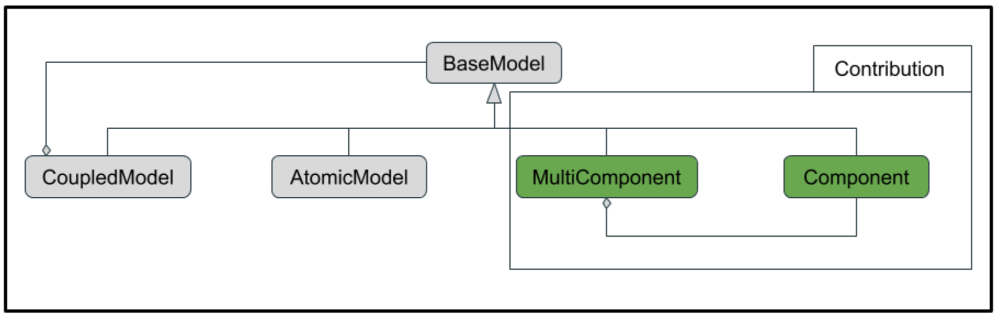


Figure 1: Main modeling components.

of an Atomic simulator because it is run as a single simulator and a coordinator because it manages the behavior or dynamics of the components that make it up. This is illustrated on figure2

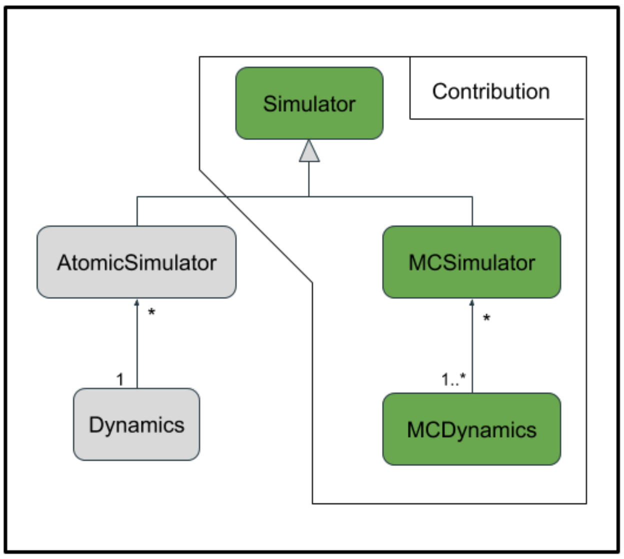


Figure 2: Main simulation components.

We have also added a new type of dynamic, DynamicComp that corresponds to a component behavior. Beside this, it was of course necessary to modify the Coordinator and the scheduler to integrate these new elements. The behaviors are defined by the modeler. According to these behaviors, the simulator updates the scheduler.

In the case of an MC model, the simulator algorithm must browse all the set of the model's components, thus updating the scheduler. It uses the MC's internal transition function, as explained in figure 3. The internal transition function of all the imminent components is executed at each event (time step). In this case, the interaction to other models is like DSDE through the entering event and the output function λ but the interaction between the components inside the MC is direct. Each component has a list of influencers and influencees components with whom there is a direct access to state variables.

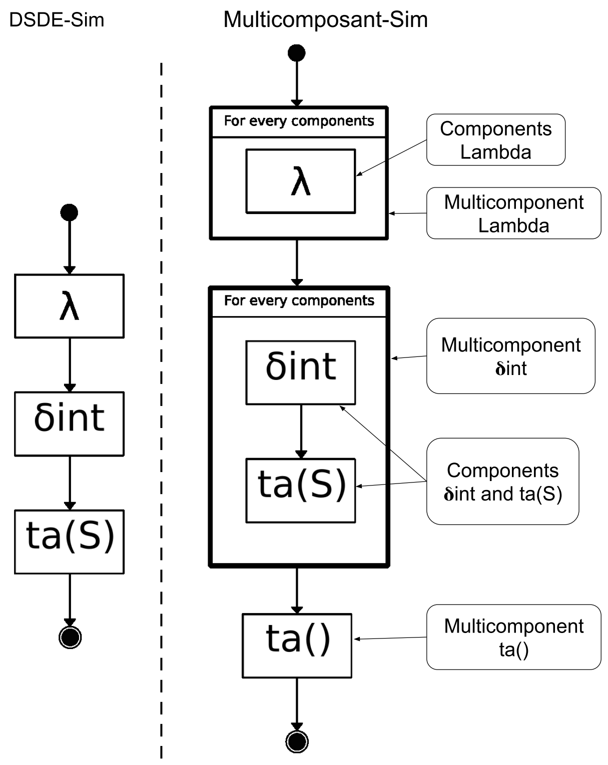


Figure 3: Simulators's algorithms.

4 FIRE MODELING

4.1 Complex system

The suggested modeling for fire spreading aims at showing the capacity of our approach to fully comprehend spatial complex systems according to a multiscale and rigorous methodology and thanks to components. We describe in this part the spatial complex system in a two-level abstract hierarchy. A global hierarchy level is used to describe the propagation phenomenon behavior in a discrete 2D space. A local level is used to describe the behavior of a $M(kg/m^2)$ of fuel at $T(K)$, under the pyrolysis physical phenomenon. The whole lot expresses a semi-physical (or semi-empirical) fire propagation model.

4.1.1 Local fire behaviour

Local fire behavior is described by the component placed in a discrete space at $P(x,y)$ coordinates. At the local level, at t time, each component has a $S(t)_{i,j}$ value among the discrete values of the S state set as: $S(t)_{i,j} \in S = \{S_A, S_H, S_{Bg}, S_C, S_{Bt}\}$,

where:

1. Ambient. $S(t)_{i,j} = S_A \Rightarrow T_{Ambient}, M(t) = M_0$, $T_{Ambient}$ is the ambient temperature (i.e. 298.15 K), and the fuel mass is M_0;
2. Heating. $S(t)_{i,j} = S_H \Rightarrow T_{Heating} = T(t) + k_1 \times t^4 + k_2 \times t, M(t) = M_0$; the temperature increases due to the amount of heat coming from components which are burning nearby. This state expires when the T temperature reaches the $T_{Ignition}$ threshold (i.e. 573.15 K); k_1 and k_2 are real constants.
3. Burning. $S(t)_{i,j} = S_{Bg} \Rightarrow T_{Burning} = T_0 + (\frac{k_3}{M_{Extinction}}) \times t^2 + k_2 - k_3 \times T, M(t) = M(t) - k_4 * M(t-1)$; We define a critical mass $M_{Extinction}$ causing the end of combustion state. We suppose combustion at constant temperature and pressure. The $M(t)$ mass of combustible decreases, to a tipping point where it causes flame extinction. When $M(t) = M_{Extinction}$, phenomenon of pyrolysis can not be maintained, the flame goes out, and fuel temperature slowly decreases; k_3 and k_4 are real constants.
4. Cooling. $S(t)_{i,j} = S_C \Rightarrow T_{Cooling} = T(t) - k_5 \times T^2 - k_6 \times T, M(t) = M_{Extinction}$. k_5 and k_6 are real constants.
5. Burnt. $S(t)_{i,j} = S_{Bt} \Rightarrow T_{Burnt} = T_{Ambiente}, M(t) = M_{Extinction}$. The remaining fuel is returned to ambient temperature. It will be left unchanged over time.

Constante	Valeur
k_1	0.03333
k_2	0.5
k_3	0.00445
k_4	0.1
k_5	0.0001
k_6	0.01

Table 1: Values of the constant used in the model

4.1.2 Global fire behaviour

The global behavior that modelers observe on output model interfaces arises from the set of state sequences of each component. At a higher hierarchical level, this behavior depends on the propagation rule formulation. We express this rule in the global component or root hierarchy component, i.e. the one containing all other components. The propagation rule takes action with the calculations at the transition phase, i.e., when calculating the next $S(time+1)$ state of the model. Its formulation needs to browse the set of the nearest elements to the one considered, called neighborhood. In our case, only the components in $S_{Heating}$ or $S_{Burning}$ states are concerned by this rule. Figure 4 gives details of state trajectory of the model's components.

We will distinguish the particular case of ignition corresponding to the simulation seed stage (initial conditions). In this example, the propagation rule mainly relies on the following statements:

- Only components which are in $S_{Heating}$ and $S_{Burning}$ states influence the neighbour components;
- Components which are in S_{Burnt} state do not evolve anymore in the simulated time.

4.2 VLE modeling

4.2.1 DSDE model

In the DSDE model each component is an atomic model. Figure 5 illustrates the connexions between the different components of the model.

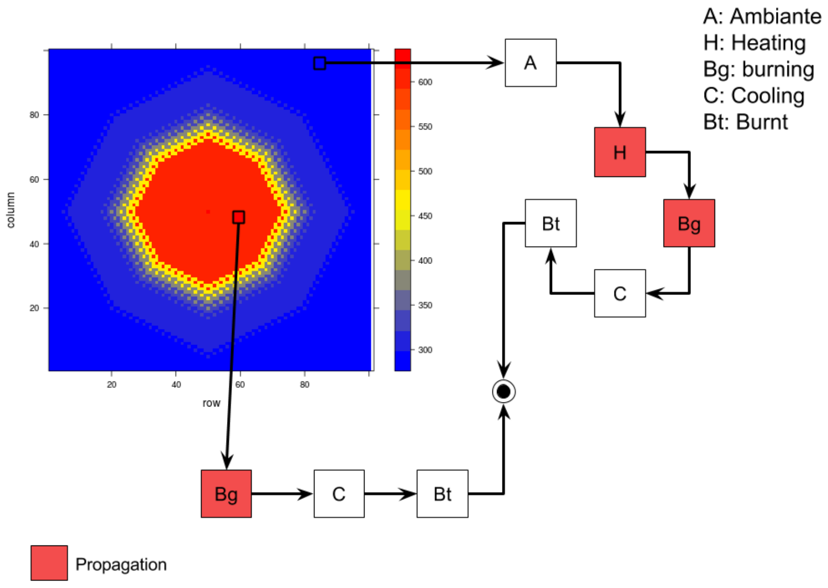


Figure 4: Components state trajectory.

The internal transition function is described in algorithm 1, where T is the cell's temperature, m the fuel mass found in cell, m_{last} is the fuel mass at $time - 1$, and $elapsed$ is the time elapsed in the S state. The influence perceived by a cell from its neighborhood is managed by the external transition function. It changes from "ambient" state to "heating" state when it receives a message coming from a neighboring cell. A cell in "heating" or "burning" state sends a message to the cells in its neighborhood, in order to exert its influence (propagation rule), in the λ output function.

Algorithm 1: DSDE model algorithm.

```
//Internal transition
if $state = heating$ then
    if $T = T_{ignition}$ then
        $state \leftarrow burning$
    else
        $T \leftarrow T + k_1 * elapsed^4 + k_2 * elapsed$
    end if
end if
if $state = burning$ then
    if $m = m_{extinction}$ then
        $state \leftarrow cooling$
    else
        $T \leftarrow T_{ambiante} + (\frac{k_3}{M_{extinction}}) * T^2 - k_3 * T$
        $m \leftarrow m - k_4 * m_{last}$
    end if
end if
if $state = cooling$ then
    if $T \leq T_{ambiante}$ then
        $state \leftarrow burnt$
    else
        $T \leftarrow T - k_5.T^2 - k_6.T$
    end if
end if
//External transition
if $state = ambiante$ then
    $state \leftarrow heating$
end if
//Output function
if $state = heating$ ou $state = burning$ then
    for each $out\_port$ do
        send $message$
    end for
end if
```

Algorithm 2: MC model algorithm.

```
if $current\_state = ambiante$ then
    $flag \leftarrow false$
    for each componant $d \in I_d$ do
        if $d.current\_state = heating$ ou $d.current\_state = burning$ then
            $flag \leftarrow true$
        end if
    end for
    if $flag$ then
        $next\_state \leftarrow heating$
    end if
end if
if $current\_state = heating$ then
    if $current\_state.T \geq T_{ignition}$ then
        $next\_state \leftarrow burning$
    else
        $next\_state.T \leftarrow current\_state.T + (k_1 * elapsed^4 + k_2 * elapsed$
    end if
end if
if $current\_state = burning$ then
    if $current\_state.m \leq m_{extinction}$ then
        $next\_state \leftarrow cooling$
    else
        $next\_state.T \leftarrow T_{amibiante} + (\frac{k_3}{m_{extinction}}) * T^2 - k_3 * T$
        $next\_state.m \leftarrow current\_state.m - k_4 * last\_state.m$
    end if
end if
if $current\_state = cooling$ then
    if $current\_state.T \leq T_{ambiante}$ then
        $next\_state \leftarrow burnt$
    else
        $next\_state.T \leftarrow current\_state.T - k_5 * current\_state.T^2 - k_6 * current\_state.T$
    end if
end if
```

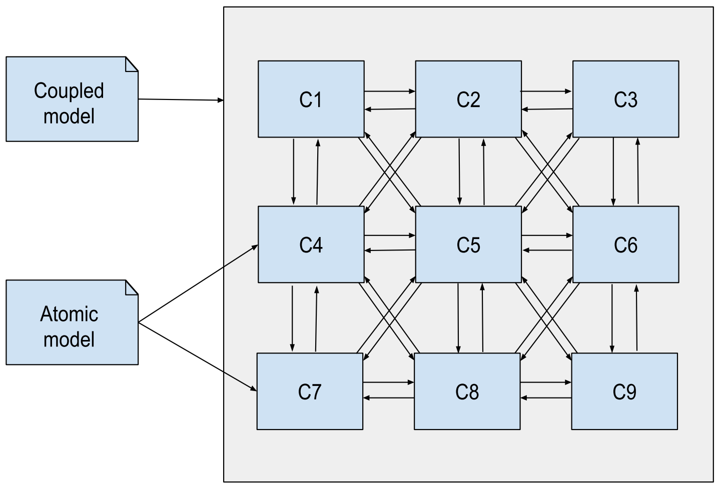


Figure 5: DSDE model.

4.2.2 Multicomponent model

In the MC model, communication between cells is directly performed according to the link between "influencer I_d/influencee E_d". Cells can directly reach the component states in the neighborhood. We do not need message exchanges, and only the δ_{int} internal transition function is executed (cf. Algorithm 2).

Figure 6 illustrates the structural design of an MC model.

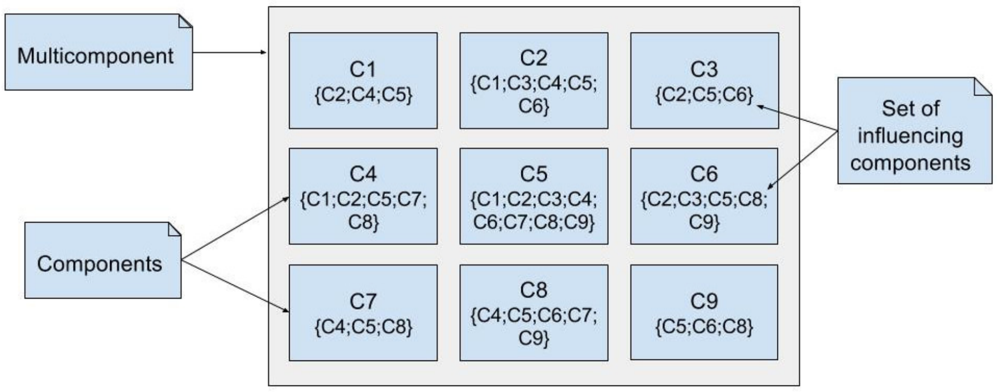


Figure 6: MC model.

5 RESULTS AND DISCUSSION

Simulations are performed on a computer with a multi-core Intel i7-2670QM CPU 2.20GHz, and 8 GB of DDR3 memory. Results are obtained on an average of 20 simulations. Experiments correspond to simulations of 300 iterations where only space size is varying, i.e. the number of cell components. We performed the simulation in three different contexts. First, the DSDE version model is simulated in the original VLE API without any of our contribution (DSDE (Original API) in table 2). Second, the DSDE version model is simulated in VLE with our contribution (DSDE in table 2). Third, the MC version model is simulated in VLE. The number of cellular components observed for each model's state, in the simulated time, for the two kinds of models is presented in figure 7. The local behavioral profile of a cellular component and its

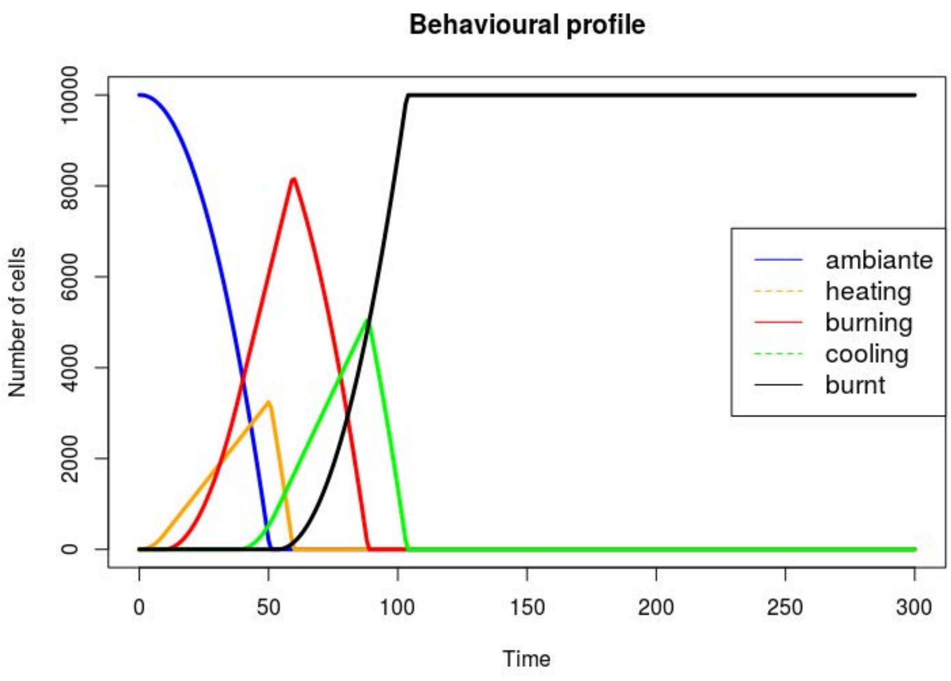


Figure 7: Evolution in time of the number of cellular components according states.

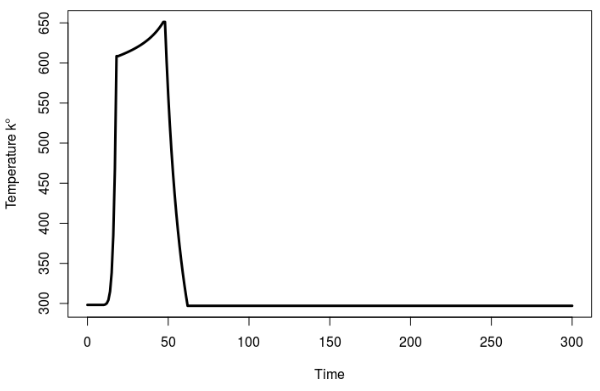
mass loss counterpart are presented in Figure 8. Simulation times are presented in Table 2 for comparisons. We observe that the Cell-DEVS specific message passing algorithm significantly slows down simulation times as soon as the number of components of the model becomes too high. On the other hand, we also notice that MC object structure integration, strongly coupled in terms of inheritance, results in reducing the performance of the original DSDE API.

Table 2: Average simulation times in seconds.

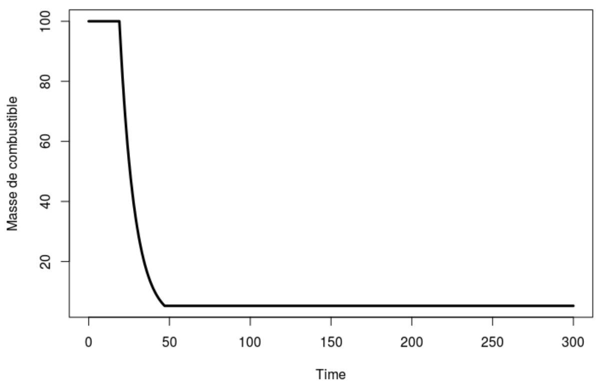
Domain size	DSDE (Original API)	DSDE	Multicomponent
100×100	10.13 s	13.00 s	1.39 s
200×200	59.08 s	62.76 s	13.10 s
300×300	187.60 s	194.59 s	31.33 s
400×400	405.33 s	426.66 s	80.65 s

6 CONCLUSION

We have implemented DSDE and Multicomponent modeling approaches for Cellular Automata models with VLE in order to simulate fire propagation. In the literature these approaches are commonly used when it comes to describing spatial complex systems with components. Since VLE does not natively have an MC extension, we developed it. In a experimental part, we observe that DSDE approach message exchanges considerably slow down the simulation process since the number of components of the model becomes higher. We also notice that the MC object structure integration strongly coupled in term of inheritance,



(a) Temperature (K).



(b) Mass of combustible (kg/m^2).

Figure 8: Local behaviour of a component.

results in lowering the performance of the original DSDE API. We will move towards a pattern-based design in our future works to improve software performance. It would aslo be interesting to use benchmark models like DEVStone (Glinsky and Wainer 2005) to confirm our results.

REFERENCES

Barros, F. J. 1995. “Dynamic Structure Discrete Event System Specification: A New Formalism for Dynamic Structure Modeling and Simulation”. In *Proceedings of the 27th Conference on Winter Simulation*, WSC ’95, pp. 781–785. Washington, DC, USA, IEEE Computer Society.

Barros, F. J. 1996, March. “The Dynamic Structure Discrete Event System Specification Formalism”. *Trans. Soc. Comput. Simul. Int.* vol. 13 (1), pp. 35–46.

Barros, F. J. 1997, October. “Modeling Formalisms for Dynamic Structure Systems”. *ACM Trans. Model. Comput. Simul.* vol. 7 (4), pp. 501–515.

Barros, F. J. 1998. “Abstract simulators for the DSDE formalism”. In *Simulation Conference Proceedings, 1998. Winter*, Volume 1, pp. 407–412. IEEE.

Buss, A. 2002. “Component based simulation modeling with Simkit”. In *Simulation Conference, 2002. Proceedings of the Winter*, Volume 1, pp. 243–249. IEEE.

Buss, A. H. 2000. “Component-based simulation modeling”. In *Simulation Conference, 2000. Proceedings. Winter*, Volume 1, pp. 964–971. IEEE.

Dalle, O., B. P. Zeigler, and G. A. Wainer. 2008. “Extending DEVS to support multiple occurrence in component-based simulation”. In *Proceedings of the 40th Conference on Winter Simulation*, pp. 933–941. Winter Simulation Conference.

Glinsky, E., and G. Wainer. 2005. “DEVStone: a benchmarking technique for studying performance of DEVS modeling and simulation environments”. In *Distributed Simulation and Real-Time Applications, 2005. DS-RT 2005 Proceedings. Ninth IEEE International Symposium on*, pp. 265–272. IEEE.

Gokhale, S. S., M. R. Lyu, and K. S. Trivedi. 1998. “Reliability simulation of component-based software systems”. In *Software Reliability Engineering, 1998. Proceedings. The Ninth International Symposium on*, pp. 192–201. IEEE.

Hu, X., X. Hu, B. P. Zeigler, and S. Mittal. 2005. "Variable structure in DEVS component-based modeling and simulation". *Simulation* vol. 81 (2), pp. 91–102.

Sun Mycrosystems Oracle. 2017, December. "JavaBeans Specification".

Peschanski, F., T. Meurisse, and J.-P. Briot. 2000. "Les composants logiciels: Evolution technologique ou nouveau paradigme". In *In Actes de la conférence OCM*, pp. 53–65.

Quesnel, G., R. Duboz, and E. Ramat. 2009, April. "The Virtual Laboratory Environment – An operational framework for multi-modelling, simulation and analysis of complex dynamical systems". *Simulation Modelling Practice and Theory* vol. 17, pp. 641–653.

Shiginah, F. A. B. 2006. *Multi-Layer Cellular DEVS Formalism for Faster Model Development and Simulation Efficiency*. Ph. D. thesis, University of Arizona, Tucson, USA.

Von Neumann, J., A. W. Burks et al. 1966. "Theory of self-reproducing automata". *IEEE Transactions on Neural Networks* vol. 5 (1), pp. 3–14.

Wainer, G. A., and N. Giambiasi. 2001. "Application of the Cell-DEVS paradigm for cell spaces modelling and simulation". *Simulation* vol. 76 (1), pp. 22–39.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press.

AUTHOR BIOGRAPHIES

PAUL-H. MARTELLONI is Ph.D. student at university of Corsica at the CNRS research laboratory UMR 6134. His research interests relate to the theory of modeling and simulation of complex systems such as marine fisheries. His email address is martelloni_ph@univ-corse.fr.

QUESNEL GAUTIER is currently a permanent researcher in computer science at INRA (French National Institute for Agricultural Research). He works on the development of the VLE software, a generic discrete event simulator based on the Theory of Modeling and Simulation. He develops planning models combining tools from Artificial Intelligence like mathematical optimization. His email address is gauthier.quesnel@inra.fr.

ERIC INNOCENTI received a Ph.D. degree (2004) from the University of Corsica. He works at the CNRS research laboratory UMR 6134. His research interests include the theory of modeling and simulation of complex systems, WSN, Environmental Metrology. His email address is eric.innocenti@univ-corse.fr.

PAUL-A. BISGAMBIGLIA born on 1981 in Peri, Southern Corsica. Currently he is an Associate Professor in Computer Science at the University of Corsica. He graduated from the University of Corsica (PhD degree in 2008). He has been working at the CNRS in UMR 6134 SPE. IEEE member since 2007. His research activities concern the Theory of Modeling and Simulation, complex systems and computational intelligence. His email address is pa.bisgambiglia@univ-corse.fr.

PIERRE REGIS GONSOLIN is the current head of the Corsica Technology Institute's department of Internet and Multimedia studies. A teacher of technical English, he holds a Master's degree from the University of Nice and a training certificate from the Ecole Nationale d'Administration. His email address is gonsolin@univ-corse.fr.

PAUL BISGAMBIGLIA is a Professor of Computer Science at the University of Corsica and is the current head of the UMRS CNRS 6134. His research interests include multi-modeling activities and the use of various techniques combination in order to develop a framework for complex natural systems. He uses modeling and simulation formalism based on a discrete event specification (DEVS) proposed by B.P. Zeigler. His email address is bisgambi@univ-corse.fr.