Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

ENABLING LARGE SCALE AND HIGH DEFINITION SIMULATION OF NATURAL SYSTEMS WITH VECTOR MODELS AND JDEVS

Jean-Baptiste Filippi
Paul Bisgambiglia

UMR CNRS 6134
University of Corsica
Corte, 20250, FRANCE

ABSTRACT

This paper describes a new methodology to enable large scale high resolution environmental simulation. Unlike the vast majority of environmental modeling techniques that split the space into cells, the use of a vector space is proposed here. A phenomena will then be described by its shape, decomposed in several points that can move using a displacement vector. The shape also have a dynamic structure, as each point can instantiate new point because of a change in the space properties or to obtain a better resolution model. Such vector models are generating less overhead because the phenomena is recomputed only if a part of it is entering into a different space entity with different attributes, using cellular space the model would have been recomputed for each neighboring identical cells. This technique uses the DSDEVS formalism to describe discrete event models with dynamic structure, and will be implemented in the JDEVS toolkit also presented.

1 INTRODUCTION

The research in defining and creating a general purpose modeling and simulation environment began nine years ago at the UMR CNRS 6134 lab of the University of Corsica.

This research has resulted in the definition of the Object Oriented Modeling and Simulation formalism (Delhom 1996). Delhom has also proven that this formalism, based on Discrete Events formalism (DEVS) (ziegler 1990), can be used to solve any problems that can be solved by discrete event simulation.

The OOMS formalism has been successfully implemented and tested in the JDEVS environment. JDEVS is a software toolkit for environmental modeling and simulation.

A top-down design approach has been used to develop this toolkit, first trying to identify the needs from the literature to propose an up to date solution to these problems. Several tools have been implemented around a java simulation engine: A graphical modelling interface, an easy to use models library, a cellular simulation panel and a connection to GIS (Geographic Information System). The Neuro-DEVS methodology is also proposed to easily implement Artificial Neural Networks (ANN) into OO models. This feature enable a modeling of environmental systems that is based on empirical data. This toolkit has been successfully used to model and simulate several systems, such as solar power supply, bugs propagation or watershed.

Nevertheless, the simulation of large scale, high resolution and geographically distributed propagation models is not yet possible because of the techniques that are used. The way spatially distributed system is modeled (using cellular space) produce a high degree of overhead that is limiting the maximum size of a studied zone. The new modelling technique proposed here is providing a solution for this problem as the model will not be simulated on cellular space but on vector space. Thus, as the behavior of phenomena is function of the properties of the space, the use of vector space is eliminating all calculation that would have been made on neighboring cells that have the same properties.

Moreover, as most data in a GIS are stored using vector maps, this technique eliminates the need to rasterize the properties maps, avoiding the loss of precision that can be introduced by this transformation. Detail of the discrete events, vector based modelling technique will be found in the paper after a brief identification of the fundamental requirements for an environmental modeling software.

The JDEVS toolkit will also be introduced as it is the actual software basis for environmental modeling developed in our laboratory.

2 ENVIRONMENTAL MODELING SOFTWARE AND REQUIREMENTS

The massive amount of literature available on environmental modeling is reflecting strong needs in this domain, and many software approaches like the VSE (Balci et. al 1998), OOPM/RT (Lee et. al 1999), SME (Maxwell 1999) or CD++ (Wainer and Giambiasi 2000) have been developed

to satisfy those needs. Fundamental requirements have been raised from those different approaches to serve as building blocks for this software toolkit.

Environmental data storage and retrieval is a basic requirement, like in other approaches, a GIS (Geographic Information System) is used in the toolkit. But if the choice of using GIS seems obvious, the main concern was how to perform the coupling to keep the architecture as modular as possible.

The parsi-model Approach to Modular Simulation introduced by Maxwell (1999) and implemented in the SME demonstrate that environmental modeling is a collaborative process that can be simplified by using a modular and hierarchical methodology. The OOMS approach is quite similar in terms of models abstraction. Distributed modeling is enabled by the integration of a generic models database defined by (Bernardi et. al 2001).

As identified in (Gimblett et. al 1995) environmental simulations, especially for geographically distributed models, are generating a heavy computer load in terms of calculation and memory transfer, consequently another requirements is the use of high performance simulation. OOMS is satisfying this need because it is based on discrete event simulation, the memory load is reduced since the simulator is only processing significant changes of the model. Calculations can also be accelerated by the use of parallel computer with no needs to redefine the model thanks to the DEVS formalism. Unfortunately even a supercomputer cannot handle high resolution simulation on a large scale.

Gimblett et. al (1995) also identifies a need for interactive visualization, the use of java is simplifying the integration of 2d/3d runtime visualization.

Another environmental modeling domain specific problem is the lack of understanding of some natural systems. This leads to the situation where self learning algorithms, such as artificial neural networks (ANNs) can offer better results than physical models. Due of their ability to generalize from empirical data, ANN are often used in the environmental modeling field. One of the alternative software requirements is the easy integration of ANN models for specific applications.

The final requirement identified concerns geographically distributed systems (such as propagation models). The simulation using cellular space generates a heavy memory load that is limiting simulation to either low-definition models at a large scale, or a high definition at a small scale. The vector based modeling technique developed later in this paper is a technique that enable the simulation on large-scale of high-resolution models with a different way of modeling.

3 THE JAVA DEVS TOOLKIT

JDEVS toolkit is composed of five independent modules. They can interact with other modules that are already developed and some elements, including the java simulation kernel, might be changed for better performance. The toolkit has been developed by the first author and is freely available from the project website at `<http://spe.univ-corse.fr/filippiweb/>`. A complete definition of the environment can be found in (Filippi et al 2002).

3.1 Modeling and Simulation Kernel

The modeling and simulation kernel is a java implementation of the DEVS formalism. Atomics and coupled models are described as follow.

3.1.1 Atomic DEVS Models Definition

The DEVS formalism is offering well defined interfaces for the description of systems. The concept of model abstraction permits to use models that are coded in various object oriented languages. Those models are then accessed thought a software interface specified in DEVS. JDEVS is a java toolkit. Modeling atomic models directly in the toolkit can be done directly in this language. To help the modeler in this task, the GUI generates a java skeleton, stores it in the models library and compiles it.

A formal DEVS atomic model is described as:

$$\mathrm{M} = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$$

With **X** is the input events set, **S** is the state set, and **Y** is the output events set.

There are also several functions: δ_{int} manages internal transitions, δ_{ext} external transitions, λ the outputs, and ta the elapsed time. The resulted code is a generated java skeleton for :

$$\text{DevsAtom}=< X\{i1\}, Y\{o1\}, S\{A\}, \delta_{int}, \delta_{ext}, \lambda, t_a >$$

```
public class DevsAtom extends AtomicModel {
  Port i1 = new Port(this,"i1","IN");
  Port o1 = new Port(this,"o1","OUT");
public DevsAtom () {
  super("DevsAtom");
  states.setProperty("A",""); }
    EventVector outFunction(Message m) {
       return new EventVector();}
    void intTransition() {}
    EventVector extTransition(Message m) {
       return new EventVector();}
    int advanceTime(){return x;}
}
```

Output and external transition functions are returning event vectors that are appended to the job list.

3.1.2 Coupled Models Description in JDEVS

If the user wants to interact directly with the simulation engine, the coupling between models can be made directly in a java class. However, with the use of the GUI, it is possible to graphically construct the model structure that is saved in XML (Bernardi et al. 2001). A DEVS coupled model is defined as:

$$\mathrm{CM} = < X, Y, D, \{M_i\}, \{I_i\}, \{Z_{ij}\} >$$

Here, **X** is the set of input events, **Y** is the set of output events. **D** is an index of components, and for each i $\in$ **D**, $\mathbf{M}_i$ is a basic DEVS model. $\mathbf{I}_i$ is the set of influences of model i. For each j $\in \mathbf{I}_i$, $\mathbf{Z}_{ij}$ is the i to j translation function. Part of the resulted XML document type definition for a coupled model is:

```
 <!ELEMENT MODEL (TYPE, NAME, BOUNDS?,
INPUT*, OUTPUT*, CHILD*, EIC?, EOC?, IC?)>
```

With TYPE defining the kind of coupled model (Cellular, kernel, coupled...), NAME the name of the model, BOUNDS the position of the model on the screen (used only by the GUI), INPUT the set of input ports, OUTPUT the set of output ports, CHILD the index for the components of the coupled model (in the priority order), EIC is the external input coupling, EOC the external output coupling and IC the internal coupling. Each coupled model is stored in a different XML file, the parser automatically instantiates the models and creates the links during loading.

3.2 Hierarchical Block Modeling and Simulation Interface

The graphical user interface (Figure 1) is the modeling front-end of the toolkit, using this front end, the user can graphically create, compile, link and store atomic and coupled models, debug the resulting model and perform the simulation.

3.3 Generic Models Library

A complete description of the library can be found in (Bernardi et al 2001). The implementation of the library description in JDEVS is resulting in a module in the GUI. This module presents models according to its domain and sub-domain, all classified in a tree like architecture. Hierarchical block modeling and simulation interface.

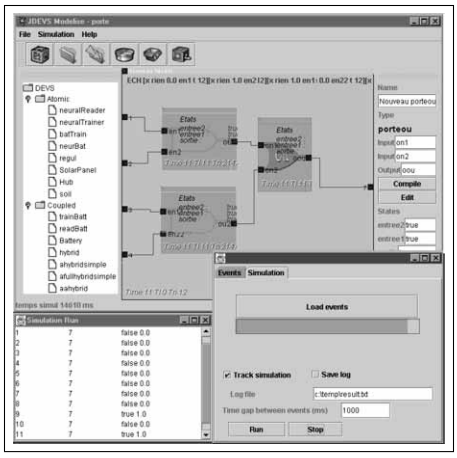

Figure 1: JDEVS Hierarchical block M&S GUI.

3.4 GIS Interconnection

(Brandmeyer and Karimi 2000) have detailed various GIS coupling methodologies. To keep the modular architecture of the toolkit, the connection to the GIS has to be made trough a loose coupling. In this kind of coupling, the data is exported from the GIS to the simulator, and results are imported back after the simulation. The simulation output is a set of discrete events (containing the cell coordinates, the cell new state, and when the change has occurred). Those events are flattened during the run to recompose discrete time maps that can be imported back to the GIS.

3.5 Cellular Simulation Panels

This module allows the user to perform (and debug) simulation of a cellular model. The user can directly interact with the simulation, as he can send events using the mouse. The general architecture shown in Figure 2 has been adopted to model cellular systems.

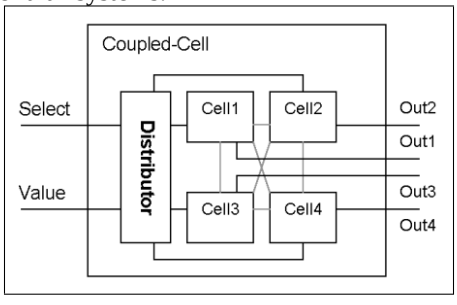

Figure 2: Cellular Models Architecture.

It is composed of a distributor and cells in a cellular coupled model. The general inputs are connected to the inputs of the distributor, then the distributor will send them to either all the cell or to the cell that would have been

selected by an event in its "Select" port. All cells are connected to the general output. 2D and 3D simulation panels (Figure 3) have been developed for the visualization of phenomena.

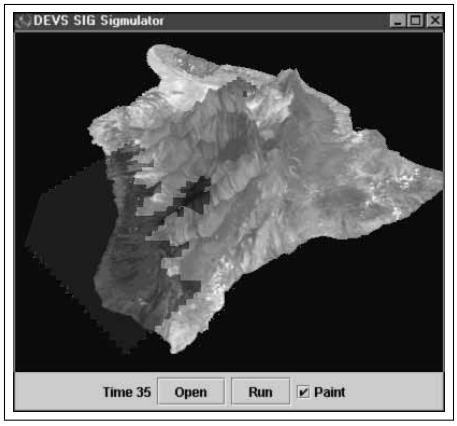

Figure 3: 3D panel

4 VECTOR SPACE MODELLING

JDEVS has been successfully used to model systems such as bugs propagation (cellular), pollution dispersion (cellular) and Solar power plant with neural network battery damage model (hierarchical blocks).

Simulation of large scale, high resolution model is still not possible using this environment, because the only way spatially distributed system are modelled is using cellular space. The vector based technique presented here has been developed to enable the simulation of such models. This technique is already functional and available in models like FARSITE (Finney and Andrews 1998) (for fire spread simulation). But FARSITE is restricted to a modified Huygens principles of wavelets propagation to determine the change in the shape of the simulated phenomenon, and simulating using variable time steps. We are presenting an attempt for a general purpose methodology that will allow to implement different structures dynamics simulated in a discrete event fashion.

4.1 Description of Vector Models

We are defining geographic agents as points that can move into space by changing their geographic attributes. A set of such points represents a shape as each point also have a structural link to its next geographical point.

A phenomena is described by its shape that is evolving in space and structure though time according to the space attributes (like a fire front). Using this methodology, the behavior of a phenomena is modeled by the definition of a basic point/segment that represents the generic point of the borderline of the phenomena. each point has a displacement vector that is used to calculate the time to next environment change.

A *geographic agent* is a basic point, it has the ability to generate new points, and adding it to its shape.

The *shape network* is a container for all interconnected points that compose a shape, it is in charge of the activation of the points. During the initialization of the simulation, the necessary number of those basic points will be instantiated to compose the initial shape of the phenomenon.

The *spatial manager* is linked to every points and has the knowledge of the space. The Spatial manager manages the different shapes that composes the model. At each structural change of a shape, the spatial manager informs the shape network of the boundaries of the new environment where the shape is evolving.

4.2 Sample Simulation of Vector Models

Figure 4 shows a vector model in its initial state. The model is decomposed in 4 points (A,B,C,D). Each point also have a "geographical link" to its next point and is aware of its position (A is linked to B, B to C, C to D and D to A). Each point also have a displacement vector that can vary in speed and direction. A point will be decomposed when it will collide another surface or if its link to another point collides another surface. Figures 4, 5 and 6 illustrate the simulation process of a simple vector model.

At the initial state the points are instantiated to represent the initial shape of the phenomena. The initial displacement vectors can also be set as initial states of the points, if no initial value is given, the points will calculate speed and direction according to the space properties.

The time to next event is then calculated for each of the points : The spatial manager sends the information of the distance to next entity in the direction of each point and segment. Once they get the information, the points are sending a message to the shape network to inform it when will occur the next collision.

In Figure 4 the point A will be the first to have a collision with a different space entity, so the model is directly moving from the initial state to the state of Figure 5.

The point A will then instantiates 2 other point (according to its decomposition policy), those 2 points are changing their behavior according to the new space attribute and the spatial manager informs them of their neighborhood. They are then calculating their time to next event, and moving to the next state of Figure 6.

A point can also send a self decomposition message, if a better resolution is needed before a collision will occur (e.g. to simulate a border effect).

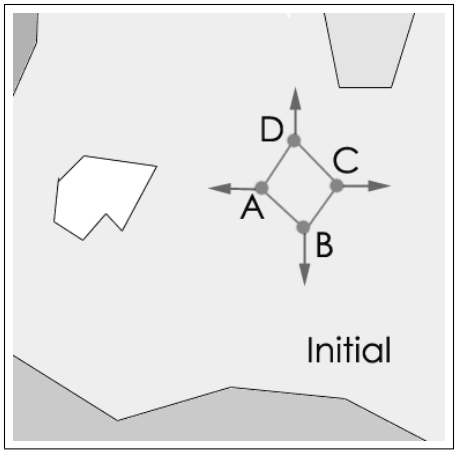

Figure 4: Initial state.

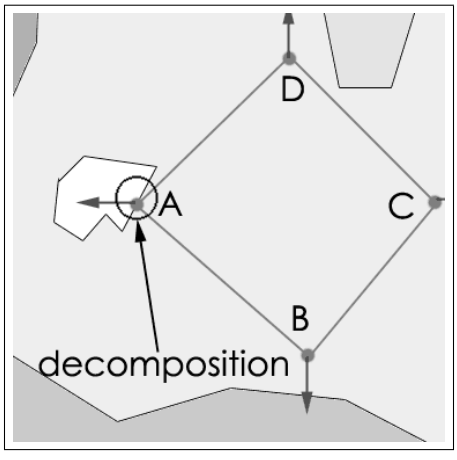

Figure 5: First collision.

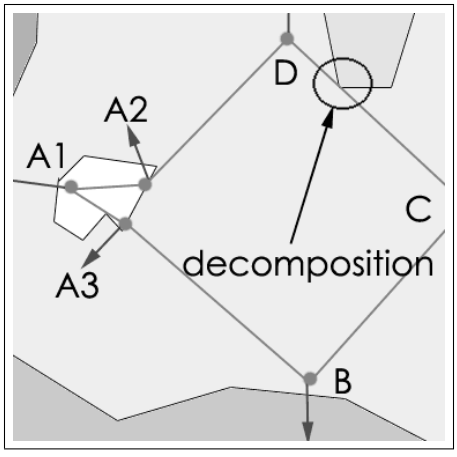

Figure 6: Second collision.

4.3 Specification of Vector Models

The formalism used to specify vector models derives from the DSDEVS formalism introduced by F. Barros (Barros 1998). DSVEVS allows the specification of dynamic structured networks of discrete event systems.

A DSDE Network is a 4 tuples

$$DSDEN_N =< X_N, Y_N, \chi, M_\chi > \text{ With}$$

- N is the network name,
- X_N is the network input values set,
- Y_N is the network output values set,
- χ is the name of the dynamic network executive,
- M_χ is the model of the executive χ.

The *network executive* χ is a special component that has the knowledge of the structure and the structure dynamics of the network. It is a modified DEVS basic model and is defined by the 9-tuples

$$M_\chi =< X_\chi, S_\chi, Y_\chi, s0_\chi, \gamma, \sum{}^*, \delta_\chi, \lambda_\chi, \tau_\chi >$$

where

- $\gamma : S_\chi \rightarrow \sum{}^*$is the structure function,
- $\sum{}^*$is the set of structures.

A structure $\sum_\alpha \in \sum{}^*$ associated partial state $s_{\alpha,\chi} \in S_\chi$, is given by

$$\sum_\alpha = \gamma(s_{\alpha,\chi}) = (D_\alpha, \{M_{i,\alpha}\}, \{I_{i,\alpha}\}, \{Z_{i,\alpha}\}) \text{ where}$$

- D_α : is the set of component associated with $s_{\alpha,\chi}$,
- for all $I \in D_\alpha$, $M_{i,\alpha}\}$ is the DEVS model of component $_i$,
- for all $I \in D_\alpha \bigcup \{\chi, N\}$, $I_{i,\alpha}\}$ is the set of components influencers of $_i$,
- for all $I \in D_\alpha \bigcup \{\chi\}$, $Z_{i,\alpha}\}$ is the input function of component $_i$. $Z_{N,\alpha}\}$ is the network output function.

A detailed explanation of the DSDE formalism can be found in Barros (1998).

The *shape network*, SN is a modified DSDE Network that contains geographical properties, is is defined by the 5 tuples.

$$SN_N =< X_N, Y_N, GS_N, \chi, M_\chi >$$

Where GS_N is a set of geographical properties that contains the boundaries of the environment where the shape is evolving.

In this shape network χ is the executive in charge of the structure of the shape. The executive links all the points of the shape together, the structure is stored in $\sum$.

χ also contains the γ function that will define the dynamic of the structure: e.g. the Huygens wavelets propagation principle can be implemented in this function to describe the propagation of a wave front.

γ can also add or retrieve points to the shape network.

The set of geographical properties are managed outside the shape network by the spatial manager. This enables to add spatial information to the environment during the simulation.

The *spatial manager* is connected to a GIS to write and retrieve geographic information for the simulated models. At each structural change of a shape network it will load the specific geographic information about the boundaries of the shapes that composes the environment where the phenomenon is evolving.

The *spatial manager* is defined in this 3 tuples

$$SM =< X_{SDB}, Y_{SDB}, D > \text{ where}$$

- SDB is the name of the spatial database,
- X_N is set of input values from the database,
- Y_N is set of output values to the database,
- D is an index of all the shape network that are managed by *SM*.

The resulting simulation tree hierarchy of vector models is shown in Figure 7.

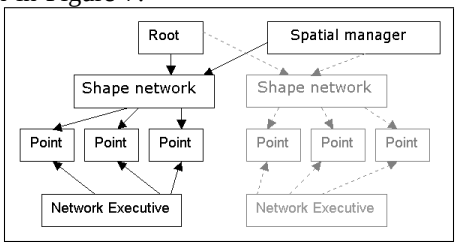

Figure 7: Vector models hierarchy

Finally the basic point is in charge of calculating the next structural change. A DEVS point model is described as follow:

$$\text{M} = < X, S(Vd, E), Y, ChV, ChS, \delta_{int}, \lambda, t_a > \text{ With}$$

- X is the input events set,
- S is the state set, (with Vd the displacement vector and E the local space property (including the coordinates)),
- Y is the output events set.

There are also the four functions that are defining the behavior of a basic point :

- ChV : is the change displacement vector function, that define the speed and direction of the displacement vector according to the new space property,
- ChS : is the change structure function that will determine the time to the next structural change structure,
- λ is outputting the new states of the point,
- δ_{int} is updating the internal states set,
- $\mathbf{t}_a$ will return the time to next change.

5 CONCLUSIONS

This paper have presented all the features already implemented in the JDEVS toolkit. The tool saves modeling and simulation time in the field of environmental modeling. It also satisfies the fundamental requirements that has been identified from other approaches in environmental modeling software. We are now extending those features by the development of vector space modeling methodology to allow the description of models using dynamic polygons and vectors rather than with cells. We hope that this method will enable the simulation of large scale, high resolution, geographically distributed systems as it generates a low degree of overhead.

This method will also serve as a basis for multi-layered models runtime cooperation that will enable a tighter coupling between models during simulation.

REFERENCES

Bernardi, F.De Gentili, E. Santucci, JF. Reusable models integration in a devs-based modeling and simulation environment. *Proceedings of the SCS ESS 2002 conference on simulation in industry*, vol 1, p 644.

Balci, Osman, Anders I. Bertelrud, Charles M. Ester-brook, and Richard E. Nance, Visual Simulation Environment. *Proceedings of the 1998 Winter Simulation Conference*, D.J. Meideiros, E.F. Watson, J.S. Carson and M.S. Mannivan, eds. Washington, DC, 13-16 December, pp. 279-287.

Barros, Fernando J. Abstract simulators for the DSDE formalism. *Proceedings of the 1998 Winter Simulation Conference*. D.J. Meideiros, E.F. Watson, J.S. Carson and M.S. Mannivan, eds. Washington, DC, 13-16 December, pp. 407-412.

Brandmeyer JE. Karimi, HA. Coupling methodologies for environmental models. *Environmental Modeling and Software*, Volume 15, Issue 5, July 2000, Pages 479-488.

Delhom, M. 1996. Modélisation et Simulation Orientées Objets. Ph.D. thesis.

Euzénat, J. 1994. Granularité dans les représentations spatio-temporelles. Tech. Rep. 2242, INRIA

Faure, X. 2001. Modélisation et simulation de la dispersion de la mouche méditerranéenne des fruits (Ceratis capitata) en Corse. Technical report UMR CNRS 6134 Lab.

Filippi, JB. Chiari, F. Delhom, M. 2002 The JDEVS environment, *Proceedings of the SCS AIS 2002 conference*, pp 317.

Filippi, JB. Bisgambiglia, P. Delhom, M. 2001 Neuro-DEVS, an hybrid methodology to describe complex systems. *Proceedings of the SCS ESS 2002 conference on simulation in industry*, vol 1, p 646.

Finney, MA. Andrews, PL. 1994 The FARSITE Fire Area Simulator: Fire Management Applications and Lessons of Summer 1994. *Proceedings of the 1994 Interior West Fire Council Meeting and Program*, p 209-216.

Gimblett, R. Ball, G. Lopes, V. Zeigler, BP. Sanders, B. Marefat, M. April 1995. Massively Parallel Simulations of Complex, Large Scale, High Resolution Ecosystem Models. *Complexity International* Vol 2, ISSN 1320-0682.

Jungst, RG. Urbina, A. L.Paez, T. 2000. Stochastic modeling of rechargeable battery life in a photovoltaic power system. Tech. Rep. 1541C, Sandia national laboratories.

Lee, Kangsun and P. A. Fishwick 1999. OOPM/RT: A Multimodeling Methodology for Real-Time Simulation. *ACM Transactions on Modeling and Computer Simulation*, Volume 9, April 1999, pp. 141-170.

Maxwell, T.A 1999, Parsi-Model Approach to Modular Simulation. *Environmental Modeling and Software*, v. 14, pp. 511-517.

Oussalah, C. 1989. A framework for modeling and linking the structure and the behavior of a system. *Artificial Intelligence in Scientific Computation*.

Sarjoughian, H.S. and Zeigler, B.P. 1998. Devsjava: Basis for a devs-based collaborative Modeling and Simulation environment. *Proceedings of the SCS Conference on Web-Based Modeling and Simulation*, vol. 5, pp. 29-36, San Diego.

Wainer, G. A., Giambiasi , N. 2001, Application of the Cell-DEVS paradigm for cell spaces modelling and simulation. *Simulation*, January 2001.

Zeigler, B.P. 1990. *Object-Oriented Simulation with Hierarchical, Modular Models*. London: Academic Press.

AUTHOR BIOGRAPHIES

JEAN BAPTISTE FILIPPI Obtained a Ba in Communication from the University of Coventry in England, graduated from Högskolan i Boras, Sweden, and obtained his Msc with "best honnours" in computer science in Corti, Corsica (FR) in 2000. He's now in his 2nd year of Ph.D. in the UMR CNRS 6134 lab of the university of Corsica. He is also working as a researcher under contract with the French Ministry of Finance and the European Union. He is the developer of several software for data mining and computer simulation. His email and web addresses are: <filippi@univ-corse.fr>, <http://spe.univ-corse.fr/filippiweb/>.

PAUL BISGAMBIGLIA is an associate Professor at the UMR CNRS 6134 lab of the University of Corsica. His research interests are fault modelling, chip validation from software description and computer simulation. His email address is: <bisgambi@univ-corse.fr>.