Available online at www.sciencedirect.com
SCIENCE DIRECT®
Environmental Modelling & Software 19 (2004) 261–274
Environmental Modelling & Software
www.elsevier.com/locate/envsoft

JDEVS: an implementation of a DEVS based formal framework for environmental modelling

Jean-Baptiste Filippi *, Paul Bisgambiglia

UMR CNRS 6134, University of Corsica, Corte 20250, France

Received 28 February 2003; received in revised form 10 August 2003; accepted 14 August 2003

Abstract

The development of models using multiple modelling paradigms is necessary to formulate and study current problems in environmental science. To simplify the coupling of those models, a formal basis for a high-level specification of such models must be set up. In this paper, we propose a discrete event system specification (DEVS) based modelling framework as a formal basis in environmental modelling. The formal framework ensures that the models are reusable and interoperable components with well defined interfaces. Moreover, a wide variety of modelling paradigms can be expressed in the DEVS formalism. We also extend the modelling paradigms that can be expressed in the DEVS framework with two techniques: Feedback-DEVS for the specification of supervised-learning models and Vector-DEVS for the specification of models in vector space. JDEVS is the Java implementation of the framework. It enables discrete event, general purpose, object oriented, component based, GIS connected, collaborative, visual simulation model development and execution. A Feedback-DEVS neural-network model and a cellular infiltration model are described as experiments using JDEVS. Those models are later coupled to show the new modelling scenarios enabled by the use of a formal framework and the flexibility of the software.

Keywords: Discrete event simulation; Environmental modelling; Artificial neural networks; Vector propagation; 3D visualization

Software availability

Name of software: JDEVS
Developer and contact address: Jean-Baptiste Filippi, UMR CNRS 6134, University of Corsica, Corte, 20250, France. Tel.: +33-495-450-209
Year first available: 2002
Hardware required: Any Java enabled computer with 64 MB of RAM and OpenGL or DirectX accelerated graphic card
Programming language: Java
Program size: 5 MB
Availability and cost: Free for non-commercial use. Sample Java applets and download from the project site: http://spe.univ-corse.fr/filippiweb/

1. Introduction

Environmental systems often act over large spatial scales, long time frames and heterogeneous units of study. The modelling and computer simulation of such complex systems have become essential tools for understanding those processes, predicting the future state of those systems and developing new theories. To conceptualize the world, the modellers need systems of computer metaphors in the form of modelling paradigms. A wide variety of modelling paradigms, such as multi-agent or cellular modelling and simulation, are in use today and have been implemented in modelling and simulation environments such as SELES (Fall and Fall, 2002), SWARM (SWARM, 2002) or ECLPSS (Woodbury et al., 2002). What is becoming difficult is not only formulating but also conceiving higher-level problems, whose complexity is such that they escape definition through a single metaphor (Villa, 2001). Enhancing interaction between modelling paradigms, model interoperability and model reusability is therefore

* Corresponding author.
E-mail addresses: filippi@univ-corse.fr (J.-B. Filippi); http://www.batti.org (J.-B. Filippi).

1364-8152/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2003.08.016

an important issue that can be addressed by the use of a formal framework that is sufficiently open and flexible.

We propose to base our formal framework on a methodology called *multifaceted modelling* introduced by Zeigler (1984). Based on system theory concepts, the methodology recognizes the multiplicities of objectives and models in the field of modelling and simulation. This methodology regards a model as a means of embodying knowledge of a real system. It concentrates on the organization of model bases for a domain. We identify a domain (like ecology) as a set of systems that share common attributes, dynamics and representation schemes. The *multifaceted modelling* methodology is similar to the domain analysis and modelling method (Praehofer et al., 2000) intended to identify the essential features, interfaces, components, abstractions and limitations of a family of systems in one domain. The core of Zeigler's methodology is the *system entity structure* knowledge representation scheme, which recognizes the need to represent:

- Decomposition, how a system may be broken down into component systems.
- Coupling, how these components may be combined to reconstitute the original system.
- Taxonomy, the admissible variants of a component and their specialization.

Ziegler also proposed *modular, hierarchical system modelling* as a way to specify models identified by the *multifaceted modelling* methodology.

Modular, hierarchical system modelling is an approach to complex systems dynamics where modular building blocks (system components with well defined interfaces) are coupled in a hierarchical manner to form complex systems. Modular building blocks are defined by specifying their input and output interface in the form of input and output ports through which all interaction with the environment occurs. *Atomic* and *coupled* models are distinguished in system modelling. While an atomic model specifies its internal structure in terms of its states and state transition functions, a coupled model's internal structure is specified by its components and coupling scheme, i.e. how the ports are connected. This modularity allows the setting up of bases of reusable components, as models that share the same interfaces are interchangeable. Modular hierarchical system modelling concepts have been applied in several domains, most notably hardware design (Santucci et al., 1998), physical systems modelling (Wainer and Giambiasi, 2001), and forest modelling (Vasconcelos et al., 1993). Inspired by systems theory, Zeigler (1984) introduced the discrete event system specification (DEVS) for modular and hierarchical modelling in discrete event simulation.

DEVS is a set-theoretic formalism that includes a formal representation capable of mathematical manipulation just as differential equations perform this role in continuous systems. Another interesting feature is that it is possible to perform formal verifications of a model using the DEVS formal representation (Freigassner et al., 2000), thus decreasing testing and implementation time. DEVS formalism also presents an explicit separation between modelling and simulation, DEVS simulators are generated automatically from a DEVS model description in an experimental frame (EF). The experimental frame describes a limited set of circumstances under which a system (real or model) is to be observed or subjected to experimentation. As such, the experimental frame rejects the objectives of the experimenter who performs the experiments on a real system or, through simulation, on a model. Recently, Vangheluwe et al. (2002) demonstrated in their meta-modelling approach that many formalisms (Petri-nets, ODE, state charts, bond graphs, etc.) can be mapped into a DEVS representation. Vangheluwe, one of the founders of Modelica (MODELICA, 2002), is currently developing the Atom3 tool (Vangheluwe et al., 2002) to perform automatic transformations from any formalism into DEVS at a semantic level. The DEVS semantics is in the process of being standardized by the Simulation Interoperability Standards Organization (SISO) (SISO, 2002). Nevertheless, DEVS needs to be adapted and extended when replaced in a domain specific context. A wide set of techniques such as F-DEVS (Kofman et al., 2000), G-DEVS (Giambiasi et al., 2001) or Fuzzy-DEVS (Zeigler et al., 1984) that derive from DEVS have already been developed to serve some domain specific needs. Some of those techniques such as Cell-DEVS (Wainer and Giambiasi, 2001) for simulation based on cellular automata, DS-DEVS (Barros, 1996) for the modelling and simulation of models with variable structures or JAMES (Schattenberg and Uhrmacher, 2001) for multi-agent simulation are well adapted in environmental modelling, but to cover the needs not previously tackled, we have also developed two new techniques, Feedback-DEVS for the modelling and simulation of self-learning models and Vector-DEVS for the simulation of phenomena in vector space.

The goal of this paper is to fill the need for a DEVS based formal framework in the domain of environmental modelling. We also propose JDEVS as an implementation of the framework. In Section 2, we refer to domain analysis in environmental modelling and introduce the formal framework and its semantics. In Section 3, we specify in detail the new techniques that we have integrated to the formal framework, Feedback-DEVS and Vector-DEVS. In Section 4, we present our Java implementation of the formal framework, JDEVS. Finally, in Section 5, we illustrate the advantages of a DEVS framework in terms of model reusability and interchange with the coupling of two different pollution models that use different modelling paradigms. In the

last section, we conclude by comparing JDEVS with other environmental modelling frameworks and present the perspectives of our work.

2. A DEVS formal framework in environmental modelling

We advance the use of a DEVS based framework by the identification of the essential features, interfaces, components, abstractions and limitations that exist in the field of environmental modelling. We first make a domain analysis in the field of environmental modelling, then motivate and introduce the discrete event formal framework and the DEVS semantics.

2.1. Environmental modelling domain analysis

Different abstraction levels exist in environmental systems with different scales that range from plant to planet. For each of these scales, questions are raised on the spatial organization of the system (spatial structure or geometry), its organization in sub-systems (topology), or its physical behavior (energy flow through the system) (Coquillard and Hill, 1997). Our objective for the formal framework is to provide a set of techniques to study those systems at different levels. The approach is similar to integrated modelling architecture (IMA) (Villa, 2001) and open modelling engine (OME) (Reed et al., 1999) that both provide an object oriented framework for developing modelling systems. We identified from Villa (2001), Woodbury et al. (2002) and Maxwell (1999) that such a framework should present the following:

- Features: Environmental models are often described as either non-spatial or spatially explicit, structurally invariant or dynamic, stochastic, empiric or deterministic, process based or agent based. A general framework should not be restricted to a few of those features. It should provide a set of modelling paradigms that can be coupled at a higher-level of specification.
- Interfaces: The framework should provide interface with non-spatial or spatially explicit databases (such as geographical information systems, GIS). Initiatives like the open geodata interoperability specification (OpenGIS, 2002) have paved the way by providing an open framework for universal geographic data access and processing. Models should also share the same interfaces to be successfully integrated in a library of reusable components.
- Abstractions: If the environmental systems to be modelled operate in the real world and in real time, models require data structures for space and time. Raster, vector and agent are the most common space representations in environmental modelling and also the kind of data structures available in GIS. Ecological models also require a time representation that is either continuous (discrete events) or discrete (time steps).
- Components: The basic components of a model are different depending on the paradigm used, agents in agent simulation, cells in cellular simulation, shapes for simulation in vector space and building blocks in process based simulation. Another important component of a modelling framework is the experimental frame that describes the circumstances under which a model is to be observed.
- Limitations: Even though current computers are capable of billions of operation per second, it is not possible to simulate every particle of a large scale system. Current computer models are limited in terms of purposes by the assumptions made in the modelling paradigm used (Fall and Fall, 2002). The quality of the results is also limited by the quality of the input data available (Bregt et al., 1991).

The essential features, interfaces, components, abstractions and limitations of a family of systems in environmental modelling identify the required modelling paradigms to be integrated in the framework. In IMA, Villa (2001) points out the need to focus on multi-paradigm problems by the fact that:

- More than one modelling paradigm is often needed to capture the minimal description of a complex system.
- Science needs new concepts to formalize and understand the complexity of nature.
- The availability of an integrating framework to run and link multi-paradigm models along with model analysis tools is essential for decision makers to profit from the use of models and evaluate their appropriateness.

Villa (2001) proposed the IMA semantics to ensure clear, high-level, non-redundant specification of models that can interoperate. He illustrates the need for a common semantics with an analogy with DNA. Just as DNA provides a common language for the specification of all living beings, the entities involved in a modelled system should share a simple and uniform representation in order to be used together with full interoperability.

We emphasize the need to use the DEVS formalism because it has the property of being *closed under composition*. A formalism is said to be *closed under composition* if any composite system obtained by coupling components specified by the formalism is itself specified by the formalism. The differential equation and sequential state machine formalism are known to be closed under composition. The significance of such closure is that it facilitates hierarchical construction of models by recursive application of coupling procedures (Zeigler, 1984). Moreover, many of the modelling paradigms to be used in an environmental modelling framework have

already been adapted to DEVS and share the same properties and semantics. Nevertheless, we have added new techniques that the field still lacks: modelling paradigms for empirical models and for the simulation of phenomena in vector space.

2.2. Discrete event systems and DEVS semantics

In general systems theory, a system is defined by its inputs, outputs, states, time base and transition functions to provide new states and outputs from inputs. Like continuous systems, discrete event systems are a way of expressing such a system, but in discrete event systems, inputs can occur at any time, while in continuous systems inputs are piecewise continuous functions of time. The discrete event representation of time is more general than the discrete time steps used in continuous systems. It gives the ability to fix time steps by specifying a given time for a model to stay in a stable state, an activation event being generated for the model for every fixed time step. Simulation is then possible for models working at different time steps as they will share the same event list that is sorted chronologically. Having this representation of the system could also save simulation time. In discrete event simulation, if the state of the system is stable, it may not be evaluated until an event arrives, while the state of a model would be evaluated at every time step in a discrete time simulation. The interest in using discrete event simulation is further discussed in other modelling environments that use discrete events such as SWARM (2002) or SELES (Fall and Fall, 2002).

DEVS is well adapted to be implemented in an object oriented framework, thus creating a component based modelling and simulation environment. It is also possible to generate a continuous time simulator out of a system modelled using the DEVS formalism as all DEVS models already map to abstract simulators also defined in the DEVS framework. DEVS formalism introduces two kinds of models, basic models from which larger ones are built, and coupled models (also called network of models) that connect those models in a hierarchical fashion. Fig. 1 presents a DEVS model with a corresponding simulation tree. Like in general systems theory, a DEVS model contains a set of states and transition functions that are triggered by the simulator.

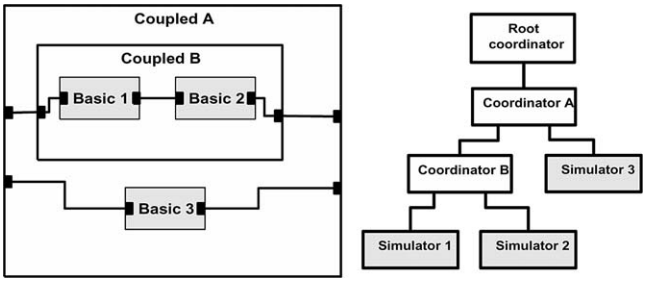

Fig. 1. A DEVS model (left) with corresponding simulation tree (right). Gray boxes correspond to the basic models and their simulators, white boxes to the coupled models and their coordinators.

A basic DEVS model (BM) is a structure:

$$\mathrm{BM} = \langle X, S, Y, \delta_{\mathrm{int}}, \delta_{\mathrm{ext}}, \lambda, t_{\mathrm{a}} \rangle$$

where:

- $X = \{((p,v)) | (p \in \text{input ports}, v \in X_p)\}$ is the set of input ports and values for the reception of external events,
- $Y = \{(p,v) | (p \in \text{output ports}, v \in Y_p)\}$ is the set of output ports and values for the emission of events,
- S is a set of internal sequential states,
- $\delta_{\mathrm{int}} = S \rightarrow S$ is the internal transition function that will move the system to the next state after the time returned by the time advance function,
- $t_{\mathrm{a}} = S \rightarrow \mathbb{R}^+$ is the time advance function, which will give the life time of the current state (returns the time to the next internal transition),
- $\delta_{\mathrm{ext}} = Q \times X \rightarrow S$ is the external transition function that will schedule the state changes in reaction to an input event,
- $\lambda = Q \times X \rightarrow S$ is the output function that will generate external events just before the internal transition takes place.

Interpretation:

- $Q = \{(s,e) | (s \in S,\ 0 < e < ta(s)\}$ is the total state set.
- e is the time elapsed since last transition, and s the partial set of states for the duration of $ta(s)$ if no external events occur.
- δ_{int}: The model being in state s at ti, it goes into s', $s' = \delta_{\mathrm{int}}(s)$, if no external event occurs before $ti + ta(s)$.
- δ_{ext}: When an external event occurs, the model being in state s since elapsed time e goes into s', $s' = \delta_{\mathrm{ext}}(s,e,x)$.
  - The next state depends on the elapsed time in the present state.
  - At every state change e is reset to 0.
- λ: The output function is executed before an internal transition; before emitting an output event, the model remains in a transient state.
- A state with an infinite life time is a passive state (steady state), else, it is an active state (transient state). If state s is passive, the model can evolve only with an input event occurrence.

The coupled DEVS model (CM) is a structure:

$$\mathrm{CM} = \langle X, Y, D, \{M_d \in D\}, \mathrm{EIC}, \mathrm{EOC}, \mathrm{IC} \rangle$$

- X is the set of input ports for the reception of external events,

- Y is the set of output ports for the emission of external events,
- D is the set of components (coupled or basic models),
- M_d is the DEVS model for each $d \in D$,
- EIC is the set of input links that connects the inputs of the coupled model to one or more of the inputs of its components,
- EOC is the set of output links that connects the outputs of one or more of the contained components to the output of the coupled model,
- IC is the set of internal links that connects the output ports of the components to the input ports of the components in the coupled models.

In a coupled model, an output port from a model $M_d \in D$ can be connected to the input of another $M_d \in D$ but not directly to itself. Both the coupled and basic models can stand alone and are stored in a models library for reuse and archiving. Depending on the implementation of the formalism, it is possible to hide the internals of coupled models and create higher-level components.

We will not describe in detail the abstract simulators of DEVS models, which can be found in Zeigler (1984). Basically, each basic model has a "simulator" attached to it that triggers the execution of the functions. Each coupled model has a "coordinator" attached to it that dispatches the events to their destination. Finally, at the top of the simulation tree stands the "root coordinator" that has a global event list where all the inputs and generated events are stored and sorted chronologically until they are processed by a simulator or output. All these entities (root, coordinators and simulators) are connected hierarchically in the simulation tree.

3. Modelling paradigms in the DEVS framework

DEVS formalism is not spatial, structurally invariant and deterministic. Much work has already been done to provide enhancements to the methodology, so that it can express more modelling paradigms:

- Cell-DEVS (Wainer and Giambiasi, 2001) for simulation based on cellular automata and spatially explicit models,
- DSDEVS (Barros, 1996) for the modelling and simulation of models with eight variable structures,
- JAMES (Schattenberg and Uhrmacher, 2001) for multi-agent simulation,
- Fuzzy-DEVS (Zeigler et al., 1984) for the modelling and simulation of models with fuzzy states.

Nevertheless, the domain still lacks an empirical modelling paradigm and a way of expressing vector space. We propose the Feedback-DEVS and Vector-DEVS techniques to meet those requirements.

3.1. Feedback-DEVS

A limitation of using DEVS is that the modeller must have a deep knowledge of the physical behavior of the studied system in order to get good results. To get around this limitation, the modeller might want to use empirical models. Feedback-DEVS extends the basic DEVS formalism to permit an explicit definition of self-learning models (such as neural networks). This technique has been successfully implemented in a model of a photovoltaic power system with a Feedback-DEVS neural network battery damage model (Filippi et al., 2002) and is illustrated in Section 5 in the neural-network nitrogen concentration model.

A basic Feedback-DEVS model (FM) is a structure:

$$\mathrm{FM} = \langle X, S, Y, \delta_{\mathrm{int}}, \delta_{\mathrm{ext}}, \delta_{\mathrm{react}}, \lambda, t_{\mathrm{a}} \rangle$$

where

- $X = \{(p_i, v_i)(p_f, v_f) | p_i \in$ standard input ports, $p_f \in$ feedback input ports, $v_i \in Xpi$, $v_f \in Xpf)\}$ is a set of input ports and values for each port (standard or feedback),
- $S = S' \cup S_f$ is the internal state set with S' the state set for a normal behavior and S_f the states that the model could reach in a reaction to a feedback,
- $\delta_{\mathrm{react}} = Q \times X \rightarrow S$ is the reaction transition function.

In FM, $\langle Y, \delta_{\mathrm{int}}, \delta_{\mathrm{ext}}, \lambda \rangle$ are identical to the basic DEVS components, the difference with the standard basic model being the explicit separation between standard and feedback inputs in the input set. This enables a different processing of the events arriving in *Xpi* and *Xpf*. The *Xpi* inputs are processes by δ_{ext}, while *Xpf* inputs are processes by δ_{react}. Feedback-DEVS basic model can be coupled in the unmodified DEVS coupled model.

The Feedback-DEVS formalism presented here enables the integration of the adaptive modelling technique into a general purpose simulation framework. This permits the study of interactions between physical systems modelled using different techniques. The Vector-DEVS technique presented next addresses a different problem in environmental modelling, the need for a high-level modelling technique suitable for studying spatial organization on a large scale. Furthermore this technique introduces the "spatial manager", a simulation component that manages the interaction of spatially distributed models (such as cellular, entity or vector propagation models).

3.2. Vector-DEVS

Simulation of large scale, high-resolution model is still not possible using this environment, because the

only way spatially distributed systems are modelled is by using cellular space. We have developed the vector based technique to enable the simulation of such models.

The vector propagation technique is designed to work at a higher abstraction level than the cellular propagation models that solve the physical model expressed in partial differential equations. One of the main applications of the vector propagation models is using the resolutions of the physical models for a vast number of scenarios; these resolutions give propagation speeds that can be stored in a database and retrieved during simulation. This technique is already functional and available in models like FARSITE (Finney and Andrews, 1994) (for fire spread simulation). But FARSITE is restricted to a modified Huygens principle of wavelet propagation to determine the change in the shape of the simulated phenomenon, and simulating using variable time steps. We present an attempt at a general purpose methodology that will allow implementation of different dynamics of structures simulated in a discrete event fashion.

3.2.1. Description of vector models

We define *geographic agents* as points that can move in space by changing their geographic attributes. A set of such points represent a shape, as each point also has a structural link to its next geographical point through the *shape network*. In this section, we will call a "point" a geographic agent because agents represent shape. A phenomenon is described by its shape that evolves in space and structure through time according to the space attributes (like a fire front). Using this methodology, the behavior of a phenomenon is modelled by the definition of a basic point/segment that represents the generic point of the borderline of the phenomenon. Each point has a displacement vector that is used to calculate the time to the next environmental change. A spatial manager contains all the properties of the space and provides coupling with other spatially explicit models. Fig. 2 presents the spatial manager containing a Vector-DEVS and a cellular model hierarchy.

A *geographic agent* is a basic point; it contains the definition of its dynamics over space (speed and direction). It is a modified basic DEVS model with the standard transition functions and an additional ChV (change displacement vector function) and ChS (change structure function). The point evolves in space by changing to a new state from a set of internal states and a set of spatial properties managed outside the agent.

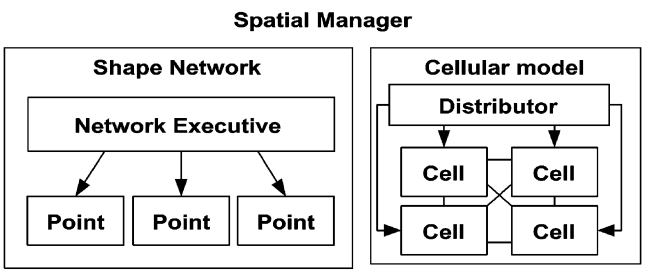

Fig. 2. Vector-DEVS and cellular models hierarchy in a spatial manager.

The *shape network* is a container for all interconnected points that compose a shape; it is in charge of the activation of the points (ordering the actions 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 shape network also contains a special component, the *network executive*, that manages the interconnection between points.

The *spatial manager* is linked to every shape network or other spatial models that compose the simulation space. It manages all properties of space that are used in the basic spatial components (cell or point). At each structural change of a shape, the spatial manager informs the shape network of the boundaries of the new environment where the shape evolves. The spatial manager is not a DEVS model, but a container for the states that are used in the DEVS spatial models. It is connected to a GIS to retrieve geographic information for the simulated models. At each structural change of a shape network, it loads the specific geographic information about the boundaries of the shapes that compose the environment where the phenomenon evolves.

Vector-DEVS uses a DSDEVS (Barros, 1996) basis for the definition of spatially explicit and structurally evolving shapes. The DSDEVS technique is not spatially explicit and is a general DEVS based formalism for the definition of variable structure models. It is not the objective of this paper to provide details on the vector propagation technique; the interested reader can find an in-depth specification of the Vector-DEVS technique in Filippi and Bisgambiglia (2002).

4. The Java DEVS toolkit

JDEVS is the Java implementation of the formal framework. JDEVS is composed of five independent modules: a simulation kernel, a graphical block modelling interface, a models library, a connection to a GIS and the simulation panels. They can interact with other modules that are already developed and some elements, including the Java simulation kernel, might be changed for better performance. Fig. 3 describes the general architecture of the JDEVS.

The only programming task that the domain specialist has to do is the redefinition of the four methods of a basic model. Once the atom is created, it is stored in the library to be used later in a federation (coupled model). In the case of a spatially distributed system, the federation is automatic in the cellular panel; the atom cells are instantiated according to the raster property map

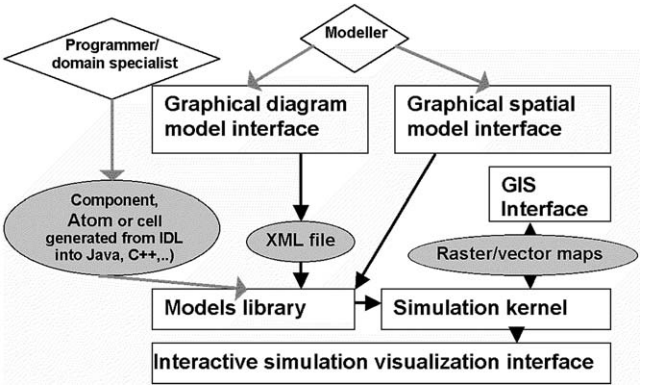

Fig. 3. JDEVS toolkit architecture. Diamonds correspond to human interactions, squares correspond to the modules and circles to the interchange formats.

exported from the GIS. For a component diagram model, the coupling between models is made graphically by the modeller in the block diagram GUI, then stored in XML in the library.

4.1. Modelling and simulation kernel

The modelling and simulation kernel is a Java implementation of the DEVS formalism. The DEVS semantics is to be mapped to a set of Java instructions only after the DEVS becomes the SISO standard currently developed by the DEVS standardization group (SISO, 2002). Basic and coupled models are described as follows.

4.1.1. Basic DEVS model definition

The DEVS formalism offers well defined interfaces for the description of systems. The concept of model abstraction permits the use of models that are coded in various object oriented languages. Those models are then accessed through a software interface specified in DEVS with Java remote method invocation (RMI). Modelling basic models in JDEVS is done directly in Java. To help the modeller in this task, the GUI generates a Java skeleton, stores it in the models library and compiles it.

As an example, the resulted code is a generated Java skeleton for:

DevsAtom

$$= \langle X\{i1\}, F\{f1\}, Y\{o1\}, S\{A\}, \delta_{\text{int}}, \delta_{\text{ext}}, \delta_{\text{react}}, \lambda, t_a \rangle$$

```
public class DevsAtom extends BasicModel {
  Port i1 = new Port(this,"i1","IN");
  Port f1 = new Port(this,"f1","FEEDBACK");
  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();}
  EventVector react(Message m) {
   return new EventVector();}
  Int advanceTime(){return A;}
}
```

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

4.1.2. Coupled model description in JDEVS

If the user wants to interact directly with the simulation engine, coupling between models can be done 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 and Santucci, 2002). Part of the resulting 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, etc.), 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 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.

4.2. Hierarchical block modelling and simulation interface

The graphical user interface is the modelling front-end of the toolkit; using this front end, the user can graphically create, compile, link and store basic and coupled models, debug the resulting model and perform the simulation. Distributed modelling is made using the GUI; if different modellers work on sub-coupled models and store them in the same library, it is possible to federate those models in another graphical modelling client. Fig. 4 shows the modelling and simulation interface.

On the left is the models library (with basic and coupled models); with a mouse click, the selected model is added to the selected coupled model. In the center is the hierarchical block diagram representation of the model; all components can be moved with the mouse,

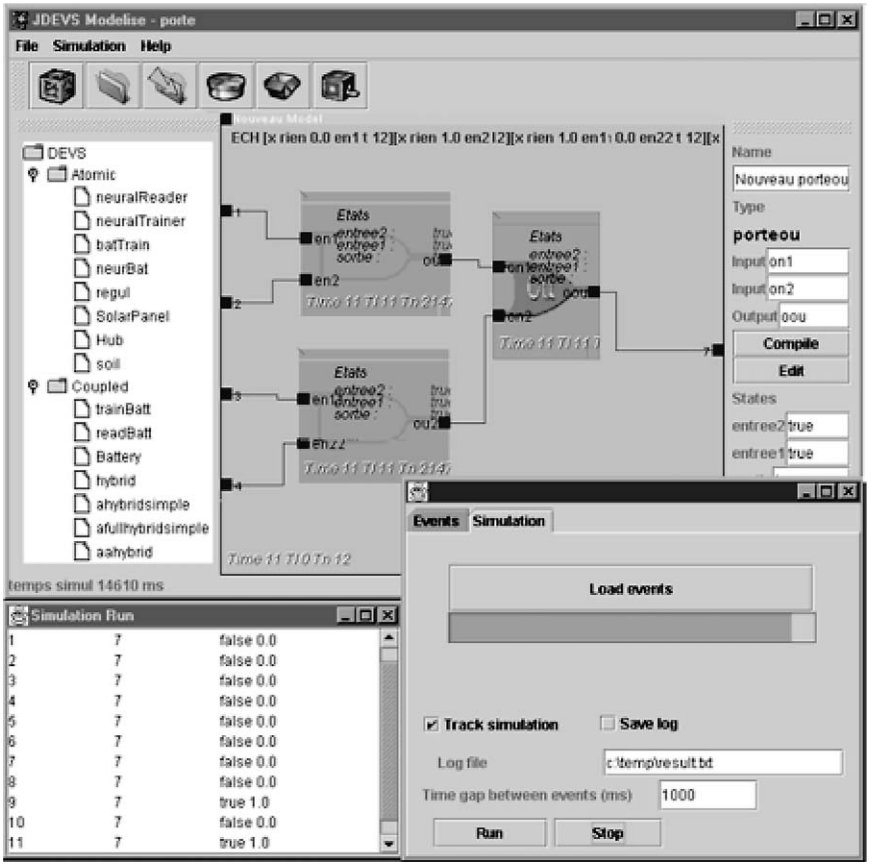

Fig. 4. JDEVS hierarchical block M&S GUI.

the linking between models being performed by a click from the origin port to the destination port. On the right is the properties panel of the selected component. If it is a basic model, the user can edit and compile it from this properties panel. At the bottom of the figure is the simulation panel; the user loads the input events from this panel and runs the simulation to the screen or to a file. To debug the model, it is possible to track the simulation. In this mode, a chosen delay is set between the processing of each event. The diagram of the models then displays the event queue and the states of the selected models during the run.

4.3. Generic models library

A complete description of the library can be found in Bernardi and Santucci (2002). The purpose of the library is to provide easy model storage and reusability. The software design in itself can be seen as an object oriented database.

In addition to the structural links, the library stores the inheritance and the abstraction link between models. The models stored in the library are called “context-out” models (usually the source code). To retrieve a model, it is instantiated, and then put “in context”. During this phase, the state of the model is set back to the state it had when it was stored. Several simulation scenarios can create several “context-in” models from the same “context-out” model description.

The implementation of the library description in JDEVS results in a module in the GUI. This module presents models according to its domain and subdomain, all classified in a tree-like architecture.

4.4. 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 is made through a loose coupling. In this kind of coupling, the data are exported from the GIS to the spatial manager, and the results are imported back after the simulation. Input/output operations are performed via the Geotools (2002) Java software library that supports Arcview, ASCII and Geographic Markup Language (GML) formats.

As the simulation is event driven, the map is never entirely updated during the run. The output of the simulation is a set of events that represent only what has changed on the map over time. This greatly reduces the size of the log file and accelerates the execution, but this format cannot be output directly to the GIS. Here is an

example of the output of a cellular model of bug propagation:

Time	Cell	Value
20	546	10.0
22	778	20.0
25	837	20.0

where “time” is the time when the change has happened, “cell” is a reference to the cell where the change has happened and “value” is the new value for the cell.

The simulation events must be “fattened” in order to create maps that can be sent back to the GIS. Those maps are snapshots of the simulation at a certain date. To create those snapshots, the program takes the initial value of the map and applies all changes that have occurred before the date defined by the user.

4.5. Cellular simulation panels

The cellular simulation panels are the experimental frames for the cellular models. The panels share the same simulation engine as that of the other modules, so they have access to the general input and output ports of the models loaded in the JDEVS GUI when both the GUI and the panels are launched. Coupling between ports of a cellular model and a hierarchical block model is defined in an XML parameter file similar to the coupled model description file. However, the parameter file must be written in a text editor, but we have planned to simplify the coupling procedure in the next version of JDEVS.

The cellular panel allows one to perform (and debug) simulation of a cellular model. The user can directly interact with the simulation, as one can send events using the mouse. Otherwise the interactions are predefined in an interaction event file. The general architecture from Wainer and Giambiasi (2001) and shown in Fig. 5 has been adopted to model cellular systems.

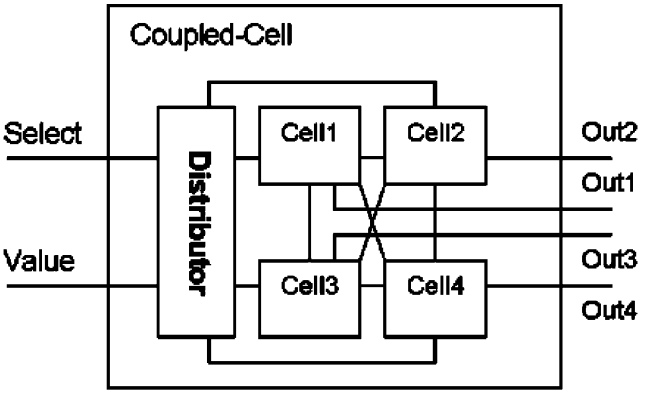

Fig. 5. Cellular model architecture.

It is composed of a distributor and cells in a cellular coupled model. The general input is connected to the “value” input port of the distributor. The distributor can send a value to either all the cells or to the cell that is selected by an event in its “Select” port. All cells are also connected to a general output (one output port for each cell).

Using the GIS connector, the cellular panel load the maps, automatically instantiates one cell for each point of the grid and performs coupling. Once the model is created, it is sent to the simulator which performs the simulation. 2D and 3D simulation panels (Fig. 6) have been developed for the visualization of phenomena.

These simulation panels can also run as applets in any Java enabled browser, thus creating a wide range of new applications. As an example of such applications, models developed in JDEVS can be directly published and experimental frames set up for a certain type of user. The modeller can decide to define an experimental frame (a chosen model and a site of interest) for a stakeholder, then publish the experiment on the Web. The stakeholder himself will then be able to try different simulation scenarios.

4.6. Vector simulation panels

The vector simulation panels are the experimental frames for the vector models. The vector models and cellular models are superposed in Fig. 8 to show a sample simulation of vector model and the equivalent cellular model (vote automata; Moore, 1996). In the figure, the two simulations are independent and superposed only to show the equivalence in terms of results.

In Fig. 7(1), the model is in its initial state. The model is decomposed into four points (A, B, C, D) that share the same location but have different displacement vectors. We chose four points to have equivalence with the cellular Von Neumann neighborhood (north, east, south, west). Each point also has a “geographical link” to its next point and is aware of its position and neighborhood (A is linked to B, B to C, C to D and D to A). Each point also has a displacement vector that can vary in speed and direction. A point will be decomposed when it collides with another surface or if its link to another point collides with another surface.

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 taken until the next event is then calculated for each of the points: the shape network activates the points by sending the position of all points constituting its neighborhood (any spatial entities in the direction of the point and segment). Once they get the information,

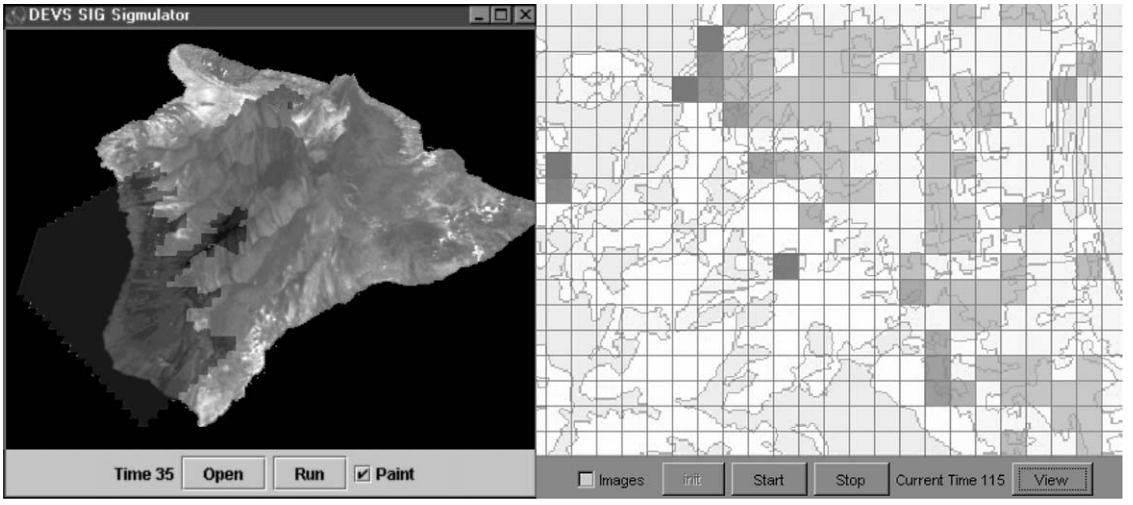

Fig. 6. 3D and 2D interactive visualization panels.

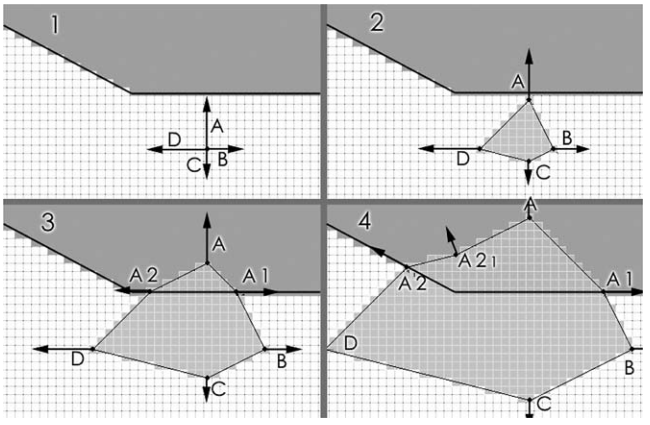

Fig. 7. Vector propagation, in its initial state (1), first collision (2), second collision for point A2 (3) and last collision (4).

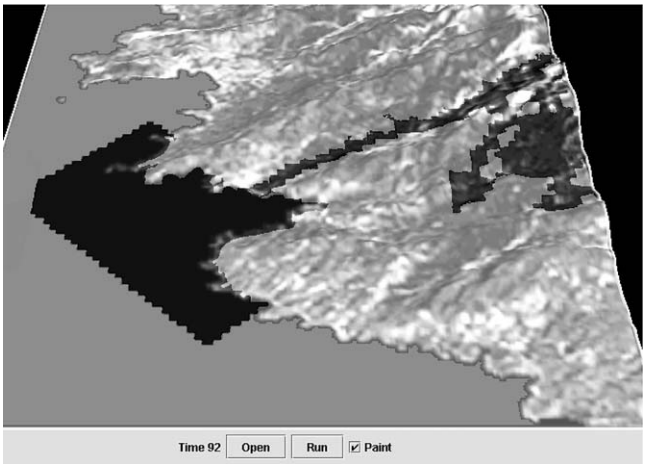

Fig. 8. Pollution model in the 3D panel.

the points send a message indicating their current position to the shape network. The time to collision is calculated by dividing the distance to the closer vertices of the neighborhood by the speed of the point.

In Fig. 7(2), point A will be the first to have a collision with a different space entity, so the model directly moves from the initial state to the first collision (2).

Point A will then schedule a change and the shape network instantiates two other points (according to its decomposition policy); those two points change their behavior according to the new space attribute and the spatial manager informs the shape network of their neighborhood. They then calculate their time to the next event (a change in direction and decomposition for point (A2) (3), and the last collision for point D (4)).

We can see from Fig. 7 the advantages in terms of computational cost of the technique: for this cellular voting model, and at this terrain resolution, the vector model calculates 20 intersections between each point and each vertex, while the cellular model makes 46 transitions (one for each cell). Basically, both algorithms are of the same complexity (O(n)), with n being the number of cells for the cellular automata (Moore, 1996) and n the number of vertices for the vector model.

We are currently developing some models of wave propagation to validate to what extent the vector modelling paradigm is applicable. The paradigm also seems appropriate for the modelling of propagation of fire fronts on a large scale with a reasonable level of detail (Filippi and Bisgambiglia, 2002). Nevertheless, this technique is limited by the fact that a shape only represents an homogenous phenomenon, so this modelling paradigm is only of interest to study physical interfaces (such as fire fronts).

JDEVS has been successfully used for the modelling and simulation of fruit fly propagation in an orchard

(Faure, 2001), fire propagation and a photovoltaic power system with a neural network battery damage model (Filippi and Bisgambiglia, 2002). The following section presents an experiment that illustrates a multi-paradigm modelling process with JDEVS.

5. Applications

This section uses a case study to illustrate the main advantages of using JDEVS: coupling and reusability of models in a multi-paradigm framework. The following two applications, a cellular pollution infiltration model and a nitrogen concentration model, are simplified implementations of Zeigler et al. (1996) and Lek et al. (1999). The purpose of these models is to illustrate the new modelling scenarios made possible by the use of the formal framework. With the first model, we show an implementation of a cellular model in JDEVS and give an overview of the effort needed to implement such a model in the framework. The second model, a hierarchical block Feedback-DEVS model, illustrates the integration of a neural network in a DEVS basic model. The two models are different in nature. One is a model of spatial organization, while the second is a non-spatial empirical model. Nevertheless, the last part of this section shows that the coupling of these models is greatly simplified because the models share the same simulation engine and the same interfaces.

5.1. Cellular pollution infiltration model with 3D visualization

To manage natural resources such as water, it is necessary to model the phenomena that alter these natural resources in order to quantify and qualify them. The sample model is adapted from Zeigler et al. (1996). Fig. 8 shows a simulation of the model developed to quantify pollution in catchment basins. Like any other basic model, this cellular pollution model is described in one file, the atom cell description file. The behavior is described in programming code. The skeleton for the file is generated by the GUI; it contains the four functions of the basic model as well as the following state set:

$\langle X\{\text{N,S,E,W,in}\},Y\{\text{N,S,E,W,out}\},S\{\text{poros,elev,pollut}\}\rangle$

(N, S, E, W correspond to the north, south, east and west ports of their neighborhood). In this model:

- The λ function (output) sends its elevation and quantity of pollutant to the neighboring cells This function is called by the simulator in case of activation.
- The δ_{ext} function (input) receives the quantity of pollutant and altitude from the neighboring cells, and send an activation message if its elevation is significantly lower than that of the neighboring cell.
- The δ_{int} function (internal) is called when the cell receives an activation. It updates the states (here, the quantity of pollutant) depending on the quantity received and the porosity of the ground.
- The t_a function (time advance) defines the time to the next self-activation of the cell depending on the quantity of pollutant (thus defining the flow speed). The Java transcription of the output function as a code example is as follows:

```
EventVector outFunction(Message m){
 e=new EventVector();
 e.add(new Event(N,“Elev”,“Pollut”));
 e.add(new Event(S,“Elev”,“Pollut”));
 e.add(new Event(E,“Elev”,“Pollut”));
 e.add(new Event(W,“Elev”,“Pollut”));
 return e;}
```

This is the four-function programming code that the specialist (mathematician, physicist, ecologist) has to implement in order to have his model working. Once the behavior of the basic cell model is described, only data preprocessing into the GIS (generation of ASCII raster maps of the initial states: porosity, elevation and quantity pollutant, choice of cell size) needs to be performed. The 3D simulation panel serves as the experimental frame of the simulation of these phenomena. The Java3D library is used to paint the outputs of 2D or 3D cellular models. The elevation map exported from the GIS permits reconstruction of a 3D world and the 3D panel enables visualization of the polluted zones. To interact with the model, it is possible to click on the map during the simulation run and add a pollutant to a specific cell.

5.2. A neural-network model of nitrogen concentration

The model of nitrogen concentration is adapted from Lek et al. (1999). It uses Feedback-DEVS for the implementation of an artificial neural network (ANN) in a DEVS framework.

Fig. 9 presents the model in the JDEVS GUI. The neural network has been trained to provide the daily quantity of nitrogen (TNC) produced by a patch of land. The independent input variables for the models are:

- The percentage of patch area under forest, FOR,
- The percentage of cultivated area of the patch, AGR,
- The percentage of urban area of the patch, URB,
- The percentage of wetland area of the patch, WET,
- The percentage of other kinds of area of the patch, OTH,
- The animal unit density, ANI,

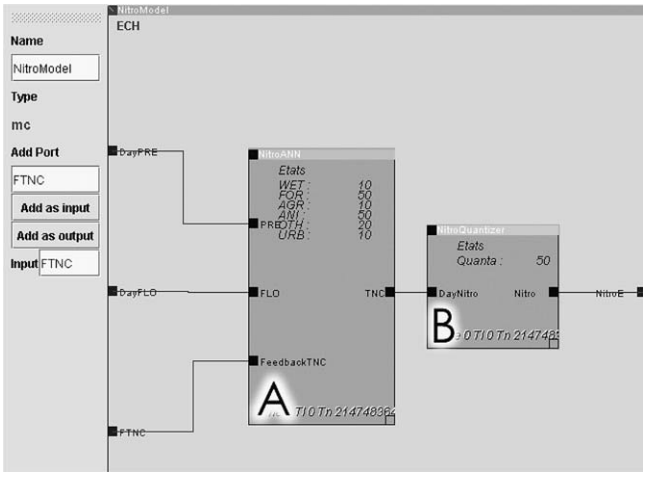

Fig. 9. Nitrogen ANN model in JDEVS, with the neural-network model (A) and the nitrogen quantizer (B).

- The daily runoff over the patch, FLO,
- The daily precipitation over the patch, PRE.

In Fig. 9, box A corresponds to the Feedback-DEVS model that encapsulates the pretrained neural network. In this model, the variables FOR, AGR, URB, WET, OTH and ANI are parameters. The model simulates the response in terms of the quantity of nitrogen for a patch with a fixed repartition of forest, cultivated, urban, wetland or other area and with a fixed animal density. This basic model has two input ports, “PRE”, the daily precipitation, and “FLO”, the average water flow. The daily quantity of nitrogen is emitted by the output port “TNC”. This model also has a feedback input port, “feedbackTNC”. When an external event is received by this port, the model triggers the learning function (here backpropagation; Lek et al., 1999) to learn the new nitrogen quantity for the value of rainfall and runoff of ports “PRE” and “FLO”.

Box B corresponds to a basic model of a quantizer. The output quantizer model sums the quantity of nitrogen received in its “DayNitro” port (here output by the neural network). When the cumulated quantity of pollutant reaches a certain amount, an output event is generated in the “Nitro” output port. The amount of nitrogen to be reached before an event is output is defined by the “Quanta” property of the quantizer model. The “Nitro” output port is connected to the general output port of the model “NitroE”.

For the simulation of this model, the daily data of rainfall and runoff for the patch are loaded as data series in the JDEVS GUI using the simulation panel. The output of the simulation is not the quantity of pollutant at each time step (driven by the daily input data), but instead the model output events each time a certain amount of nitrogen is reached. The information given by the output of the model is carried by the time taken between two output events. If the patch outputs a lot of nitrogen, the time between each output will be very short. If the patch does not output much nitrogen, the events will be separated by a longer period of time.

5.3. Coupling the model of nitrogen concentration with the cellular pollution infiltration model

Although the pollution infiltration model is a cellular model, and the nitrogen concentration model is a neural-network model, once integrated in the framework they both share the same port based interfaces. Thus, it is possible to couple these models and simulate the impact of a new land use for a patch of land.

Before performing the coupling, the modeller has to verify whether the data that will pass from the model will correspond to the data provided by the model it will be connected to. In this experiment, the data are compatible; the nitrogen concentration model will send the quantity of nitrogen at various times, while the cellular model specifies the quantity of pollutant.

Then, a cell must be chosen for study, and the rainfall and runoff data for the cell extracted from a GIS. As the subject of the experiment is to test the impact of new land use on the land patch represented by the cell, the variables FOR, AGR, URB, WET, OTH and ANI are given before simulation by the user.

Next, the modeller must define a coupling parameter file to couple the two models. The XML parameter file created for the coupling of the nitrogen and pollution model is:

```
⟨?xml version = “1.0” encoding=“UTF-8”?⟩
⟨IC⟩⟨LINK⟩
⟨ PORT model = “NitroModel.xml”⟩NitroE⟨/PORT⟩
⟨ PORT model = “cell[40 - 83]”⟩in⟨/PORT⟩
⟨/LINK⟩⟨/IC⟩
```

The file specifies an internal coupling between port “NitroE” of the nitrogen model and port “in” of the chosen cell (here, the cell at line 40 and row 83).

For the simulation, the nitrogen model is loaded first in the GUI, and the cellular pollution infiltration model is loaded in the cellular panel with the coupling parameter file, the simulation being finally launched by the simulation panel of the GUI.

6. Conclusion

A DEVS based formal framework provides a well defined architecture to specify models of a wide variety of modelling paradigms. Moreover, as the generation of a simulator from a DEVS model is automatic, it is not

necessary to specify instructions step by step on how the model should be simulated. JDEVS, the Java implementation of the framework, helps researchers to build and simulate environmental models using the DEVS formalism. Compared to other modelling softwares such as SWARM (2002), SELES (Fall and Fall, 2002) or ECLPSS (Woodbury et al., 2002), JDEVS is not limited to a single modelling paradigm (multi-agent or cellular automata) or to spatially explicit models. The approach is similar in terms of objectives to IMA (Villa, 2001), SME (Maxwell and Costanza, 1997) and OME (Reed et al., 1999), as those environments provide an integrated framework for models interoperability. According to the points outlined in the domain analysis of Section 2, JDEVS compares with those environments as follows:

- *Features*: SWARM, SELES and ECLPSS are limited to spatially explicit modelling paradigms, while OME is non-spatial. Like JDEVS, a specific semantics is used in IMA and SME to describe the behavior of models. Having a higher-level language permits addition of new modelling paradigms to the environment and still ensures compatibility between models. Nevertheless, only JDEVS uses a high-level formalism proven to be closed under composition. Moreover, it is necessary to use specific application programming interfaces to add new modelling paradigms to SME or IMA, while JDEVS can make use of the wide variety of modelling paradigms that already extend DEVS. In particular, the Feedback-DEVS and Vector-DEVS developed in this paper extend the features of JDEVS with the integration of vector based models and self-learning modelling paradigms.
- *Interfaces*: In terms of database interfaces, most of the environmental modelling softwares provide an interface with a GIS. Like SWARM, SELES and ECLPSS, JDEVS is loosely coupled with GIS, exchanging data with input and output files. Only SME can provide a tight coupling with GIS. JDEVS also provides an interface with a models library that is similar in concept to the SME or IMA models libraries. In terms of component interfaces, JDEVS uses port based interfaces, which simplifies the coupling process with other DEVS models. The other modelling environment is coupled with references to model variables.
- *Abstractions*: SWARM, SELES and ECLPSS make an abstraction of space in terms of cells. Although it should be possible to specify models that make use of vector representation in IMA and SME, only JDEVS provides a technique to specify vector based models. Like SWARM and SELES, JDEVS uses discrete events for the simulation; this provides the most general description of time as discussed in Section 2.2.
- *Components*: In terms of modelling components, SWARM, SELES and ECLPSS use agents and cells as basic components for the description of their models. Like OME, IMA and SME, it is possible to specify new kinds of components in JDEVS to implement a variety of approaches in environmental modelling. In terms of software components, JDEVS, like SWARM, makes an explicit separation between the experimental frame and the model. It is possible to visualize a running simulation in 2D, 3D in a Web applet or directly store the result for post-treatment in a GIS.
- *Limitations*: The quality of the simulation results of any modelling environment is limited by the quality of the data available. There are also conceptual limitations, because computer models are limited by the assumptions made in the modelling paradigm used. Only IMA, SME and JDEVS enable multiparadigm modelling and are not limited to a fixed knowledge representation scheme.

The need of a formal framework in environmental modelling extends beyond the choice of the DEVS formalism or the JDEVS implementation. Because it is based on a DEVS based formal framework, JDEVS provides a different approach than the existing tools. In terms of flexibility and genericity of use, it can provide the high-level approach of a general formalism. In terms of features, abstraction, components and interfaces, JDEVS provides the advantages of a domain specific modelling environment. With JDEVS, it is also possible to specify, store, retrieve, couple and simulate different kinds of models without having to specify how those models should be simulated. As further developments, we extend the “spatial manager” to enable a tighter coupling between the GIS and the simulated models as well as between the vector and cellular models. The next release of JDEVS will also include a full integration of all the modules into an unified tool with a single interface. This will simplify the coupling procedures between spatially explicit and non-spatial models.

Acknowledgements

This research is partially funded by a grant from the Corsican Regional Education and Research Council. We wish to thank M. Nivet, L. Floriani, Deepa Patel, and two anonymous reviewers for comments that helped to greatly improve this paper.

References

Barros, F., 1996. Dynamic structure discrete event system specification formalism. Transactions of the Society for Computer Simulation 13 (1), 35–46.

Bernardi, F., Santucci, J., 2002. Model design using hierarchical web-

based libraries. In: Proceedings of the 39th Conference on Design Automation, New Orleans, USA, vol. 1., pp. 14–17.

Brandmeyer, J., Karimi, H., 2000. Coupling methodologies for environmental models. Environmental Modelling and Software 15 (5), 479–488.

Bregt, A., Denneboom, J., Gesink, H., Van Randen, Y., 1991. Determination of rasterazing error: a case study with the soil map of the Netherlands. International Journal on Geographical Information Systems 5 (1), 361–367.

Coquillard, P., Hill, D., 1997. Modélisation et Simulation des Ecosystémes. Masson (ISBN: 2-225-85363-0).

Fall, A., Fall, J., 2002. A domain specific language for models of landscape dynamics. Ecological Modelling 141 (1–3), 1–18.

Faure, X., 2001. Modélisation et simulation de la dispersion de la mouche méditerranéenne des fruits (*Ceratis capitata*) en Corse. Technical Report 182, UMR CNRS 6134 Lab.

Filippi, J., Bisgambiglia, P., 2002. Enabling large scale and high definition simulation of natural systems with vector models and JDEVS. In: Proceedings of the 2002 Winter Simulation Conference, San Diego, CA, USA,, pp. 1964–1970.

Filippi, J., Bisgambiglia, P., Delhom, M., 2002. Neuro-devs, an hybrid methodology to describe complex systems. In: Proceedings of the SCS ESS 2002 Conference on Simulation in Industry, Marseilles, France, vol. 1., pp. 647–652.

Finney, M., Andrews, P., 1994. The farsite fire area simulator: fire management applications and lessons of summer 1994. In: Proceedings of the 1994 Interior West Fire Council Meeting and Program, Coeur Alene, USA,, pp. 209–216.

Freigassner, R., Praehofer, H., Zeigler, B., 2000. Systems approach to validation of simulation models. Cybernetics and Systems 1, 52–57.

Geotools, 2002. Geotools Users Group. Available from <http://www.geotools.org>.

Giambiasi, N., Escude, B., Gosh, S., 2001. Gdevs: a generalized discrete event specification for accurate modeling of dynamic systems. In: Fifth International Symposium on Autonomous Decentralized Systems, Dallas, TX, vol. 1., pp. 464–502.

Kofman, E., Giambiasi, N., Junco, S., 2002. Fdevs: A general devs-based formalism for fault modeling and simulation. In: Proceedings of the 2000 European Simulation Symposium, Hamburg, Germany, vol. 1., pp. 77–82.

Lek, S., Guiresse, M., Giraudel, J., 1999. Predicting stream nitrogen concentration from watershed features using neural networks. International Journal on Geographical Information Systems 33 (16), 3469–3478.

Maxwell, T., 1999. A parsi-model approach to modular simulation. Environmental Modeling and Software 14 (6), 511–517.

Maxwell, T., Costanza, R., 1997. An open geographic modeling environment. Simulation Journal 68 (3), 175–185.

MODELICA, 2002. Modelica Users Group. Available from <http://www.modelica.org/>.

Moore, C., 1996. Majority-vote cellular automata, using dynamics, and p-completeness. Technical Report 96-08-060, Santa Fe Institute.

OpenGIS, 2002. Open Geographic Information System Project Consortium. Available from <http://www.opengis.org/>.

Reed, M., Cuddy, S., Rizzolli, A., 1999. A framework for modelling multiple resource management issuesan open modelling approach. Environmental Modeling and Software 14 (6), 503–509.

Sametinger, J., Stritzinger, A., Praehofer, H., 2000. Building reusable simulation components. In: Proceedings of WEBSIM2000, Web-Based Modelling & Simulation, San Diego, CA, USA, vol. 1., pp. 1–7.

Santucci, J., Bisgambiglia, P., Federici, D., 1998. Behavioral fault simulation. In: Advanced Technique for Embedded Systems Design and Test., pp. 261–285.

Schattenberg, B., Uhrmacher, A., 2001. Planning agents in JAMES. Proceedings of the IEEE 89 (2), 158–173.

SISO, 2002. Simulation Interoperability Standards Organization. Available from <http://www.sisostds.org/>.

SWARM, 2002. Swarm Development Group. Available from <http://www.swarm.org/>.

Vangheluwe, H., Lara, J., Mosterman, P., 2002. An introduction to multiparadigm modelling and simulation. In: Proceedings of the AIS 2002 Conference, Lisboa, Portugal, vol. 1., pp. 9–20.

Vasconcelos, M., Perestrello, J., Zeigler, B., 1993. Simulation of forest landscape response to fire disturbances. Ecological Modelling 65, 177–198.

Villa, F., 2001. Integrating modelling architecture: a declarative framework for multi-paradigm, multi-scale ecological modelling. Ecological modeling 137 (1), 23–42.

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

Woodbury, P., Beloin, R., Swaney, D., Gollands, B., Weinstein, D., 2002. Using the eclpss software environment to build a spatially explicit component based model of ozone effects on forest ecosystems. Ecological modeling 150 (3), 211–238.

Zeigler, B., 1984. Multifaceted Modelling and Discrete Event Simulation. Academic Press (ISBN: 0.12.778450.0).

Zeigler, B., Praehofer, H., Kim, T., 1984. Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press (ISBN: 0127784551).

Zeigler, B., Moon, Y., Lopez, V., Kim, J., 1996. Devs approximation of infiltration using genetic algorithm optimization of a fuzzy system. Mathematical and Computer Modelling 23 (11/12), 215–228.