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

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 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 are also extending 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.

Key words: Discrete Event Simulation; Environmental modelling; Artificial Neural Networks; Vector propagation; 3d visualization.

Software availability

Name of software: *JDEVS*
Contact address and developer:

[*] Corresponding author
Email address: `filippi@univ-corse.fr` (Jean-Baptiste Filippi).
URL: `http://www.batti.org` (Jean-Baptiste Filippi).

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.*
Program 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 are often acting 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 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 to formulate but also to conceive higher level problems, whose complexity is such that they escape definition through a single metaphor (Villa, 2001). Enhancing interaction between modelling paradigms, models interoperability and model reusability is therefore 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 *multi-faceted 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 mean to embody knowledge of a real system. It concentrates on the organization of models bases for a domain. We identify a domain (like ecology) as a set of systems that shares common attributes, dynamics and representation schemes. The *multifacetted 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. Ziegler also proposed *modular, hierarchical system modelling* as a way to specify models identified by the

multifacetted 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 states transition functions, a coupled model's internal structure is specified by its components and coupling scheme, *i.e.*, how ports are connected. This modularity allows the set-up of bases of reusable components as models that shares 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 in System's theory Zeigler (1984) introduced the DEVS (Discrete EVent System specification) for modular and hierarchical modelling in discrete events simulation.
DEVS is a set-theoretic formalism that includes a formal representation capable of mathematical manipulation just as differential equations serves this role for continuous systems. It is possible to perform formal verifications of a model using DEVS formal representation (Freigassner et al., 2000) thus decreasing testing and implementation time. Recently Vangheluwe et al. (2002) demonstrates the uniqueness of DEVS in his meta-modelling approach by mapping commonly used formalisms (Petri-nets, ODE, State charts, Bond Graphs, etc..) into DEVS equivalents. In an effort to specify a standard semantic for DEVS models, a Discrete-Event Systems Specification Product Development Group has been proposed at the Simulation Interoperability Standards Organization (SISO) (SISO, 2002). 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. The Experimental Frame (EF) 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 reflects the objectives of the experimenter who performs experiments on a real system or, through simulation, on a model. Nevertheless DEVS needs to be adapted and extended when replaced in a domain-specific context. A wide set of techniques that derives from DEVS have already been developed to serve some domain specific needs (Zeigler et al., 2000). Some of those techniques such as Cell-DEVS (Wainer and Giambiasi, 2001) for simulation based on cellular automata, DSDEVS (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 supervised learning models and Vector-DEVS for the simulation of phenomena in vector space.

The goal of this paper is to motivate 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 the section 2 we make reference 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 advantages of a DEVS framework in terms of models 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 framework and present the perspectives of work.

2 A DEVS formal framework in environmental modelling

We motivate the use of a DEVS based framework by the identification of the essential features, interfaces, components, abstractions and limitations that exists in the field of environmental modelling. We are first making 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 exists in environmental systems with different scales that ranges from the plant to the planet. For each of these scales the questions are raised of 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 IMA (Integrated Modeling Architecture) (Villa, 2001) and OME (Open Modelling Engine) (Reed et al., 1999) that both provides an object oriented framework for developing modelling systems. We identified from Villa (2001), Woodbury et al. (2002) and Maxwell (1999) that such framework 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 GIS Geographical Information Systems). 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 operates in real world and real time, models requires computer metaphors for space and time. Raster, vector and agent are the most commonly used metaphors to abstract the system dynamics and to represent space in environmental modelling. Raster and vector maps are also the kind of data structures available in GIS. Ecological models also requires a time representation either continuous (discrete events) or discrete (time steps).
- Components: The basic component 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 particles of a large scaled 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 identifies the required modelling paradigms to be integrated in the framework. In IMA Villa (2001) motivates the need to focus on multi-paradigm problems by the fact that:

- More than one modelling paradigm are 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 semantic to ensure clear, high-level, non-redundant specification of models that can interoperate. He illustrates the need for a common semantic with an analogy with DNA. Like DNA provides a common language for the specification of all living beings, entities involved in a modelled system should share a simple and uniform representation in order to be used together with full interoperability.

We motivate the need to use the DEVS formalism because is has the property to be *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 the 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 shares 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 to express such system, but in discrete event systems, inputs can occur at any time, while in continuous systems inputs are piecewise continuous function of time. The discrete event representation of time is more general than the discrete time steps used 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. The 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 of using discrete event simulation is further discussed in other modelling environments that uses 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 possible to generate a continuous time simulator out of a system modelled using the DEVS formalism as all DEVS models are directly simulables by abstract simulators also defined in the DEVS framework. DEVS formalism introduces two kind of models, the basic models from which larger ones are built, and coupled models (also called network of models) that connects those models in a hierarchical fashion. Like in general systems theory, a DEVS model contains a set of states and transition functions that are triggered by the simulator.

A basic DEVS model (BM) is a structure:
$BM = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$ Where:

- X: $\{(p, v)|$(p$\in$ input ports, v$\in$ Xp)$\}$ The set of input ports and values for the reception of external events,
- Y: $\{(p, v)|$(p$\in$ output ports, v$\in$ Yp)$\}$ The set of output ports and values for the emission of events,
- S: A set of internal sequential states,
- δ_{int}: $S \rightarrow S$ The internal transition function that will move the system to the next state after the time returned by the time advance function,
- $\mathbf{t}_a$: $S \rightarrow \mathbb{R}^+$: The time advance function, that will give the life time of the current state (returns the time to the next internal transition),
- δ_{ext}: $Q \times X \rightarrow S$ The external transition function that will schedule the states changes in reaction to an input event,
- λ: $Q \times X \rightarrow S$: The output function that will generate external events just before the internal transition takes places.

Interpretation:

- Q $= \{(s, e)|(s \in S, 0 < e < ta(s)\}$ is the total state set.
- e is the elapsed time since last transition, and s the partial set of states for the duration of $ta(s)$ if no external event occur.
- δ_{int}: The model being in a state s at ti it will go into *s'*, *s'* $=\delta_{int}(s)$, if no external event occurs before $ti + ta(s)$.
- δ_{ext}: When an external event occurs, the model being in the state s since the elapsed time e goes in *s'*, *s'* $=\delta_{ext}(s,e,x)$.
  · 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 the state s is passive, the model can evolve only with an input event occurrence.

The coupled DEVS model (CM) is a structure:
$CM = < X, Y, D, \{M_d \in D\}, EIC, EOC, IC >$

- X: The set of input ports for the reception of external events,
- Y: The set of output ports for the emission of external events,
- D: The set of components (coupled or basic models),
- M_d: The DEVS model for each $_d \in D$,
- EIC: The set of input links, that connects the inputs of the coupled model to one or more of the inputs of the components that it contains,
- EOC: 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*: 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 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 that can be found in (Zeigler, 1984). Basically each basic model has a "Simulator" attached to it that trigger the execution of the functions. Each coupled model has a "Coordinator" attached to it that dispatch the events to its destination. Finally at the top of the simulation tree stands the "Root Coordinator" that has a global event list where all the input and generated events are stored and sorted chronologically until they are processed by a simulator or outputted. All these entities (Root, Coordinators and Simulators) are connected hierarchically in the simulation tree. Figure 1 presents a DEVS model with a corresponding simulation tree. In figure 1 connections between models within *Coupled model A* (left side) corresponds to the set of input *EIC*, internal *IC* and output *EOC* links, each link connect two ports (black square) that corresponds to the X and Y sets of models. The right side of figure 1 presents the corresponding simulation tree, with links between the simulators corresponding to the hierarchy links used by messages to transit from a simulator to another, each message carrying an event. The next section presents the enhancements to the DEVS formalism in order to simplify the definition of specific kind of models of use in the environmental science.

3 Modeling paradigms in the DEVS Framework

DEVS formalism is not spatial, structurally invariant and deterministic. Much work has already been done to provide extensions to the methodology so that it can express more modelling paradigm. Among the paradigms of interest in environmental science that has been the subject of extensions of DEVS we can note:

- Cell-DEVS (Wainer and Giambiasi, 2001) for simulation based on cellular automata and spatially explicit models,
- JAMES (Schattenberg and Uhrmacher, 2001) for multi-agent simulation,
- Fuzzy-DEVS (Zeigler et al., 2000) for the modelling and simulation of models with fuzzy states.

Nevertheless some of the main techniques and tools of use in environmental modelling cannot be expressed directly in DEVS. This section presents Feedback-DEVS and Vector-DEVS techniques developed to overcome some of these limitations. Feedback-DEVS helps the integration of empirical and supervised-learning models such as Neural-Networks. Vector-DEVS address the need for a modelling methodology that allows the specifications of models that could use vector maps to express the knowledge of spatial properties. Vector maps are widely used by scientists in GIS since it often provides a more accurate description of spatial information than raster maps.

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 have 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 supervised-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 a neural-network nitrogen concentration model.

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

$$\text{FM} = < X, S, Y, \delta_{int}, \delta_{ext}, \delta_{react}, \lambda, t_a > \text{Where}$$

- 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)$\}$ set of input ports and values for each ports (standard or feedback),
- S: $S' \bigcup S_f$ 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.
- δ_{react}: $Q \times X \rightarrow S$ is the reaction transition function,

In FM $< Y, \delta_{int}, \delta_{ext}, \lambda >$ are identical to 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 unmodified DEVS coupled model.

The feedback-DEVS formalism presented here enables the integration of 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 is addressing a different problem in environmental modelling, the need for a high level modelling technique suitable to study 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 Vectors-DEVS

The Vector-DEVS methodology has been designed to study the evolution of physical interfaces of phenomena. The physical interfaces of a phenomenon is the subject of observation of most propagation models and represent the outer bond of the system observed (the interface of a forest fire is the fire front). For this class of systems, Vector-DEVS enables the simulation of large scale, high resolution models. The physical interface is viewed as a set of moving vertices that constitute a shape. This technique is devoted to work at a higher abstraction level than cellular propagation models that solves the physical model expressed in partial differential equation. One of the main application of the vector propagation models is using the resolutions of the physical models for a vast number of scenarios, these resolutions are giving propagation speeds that can be stored in a database and retrieved during simulation.

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 has a structural link to its next geographical point through the *shape network*. In this section we will call "point" a geographic agent because agents are representing a shape. A phenomenon is described by its shape that is evolving in space and structure through time according to the space attributes (like a fire front). Using this methodology, the behavior of a phenomena is modelled 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 the next environmental change, an environmental change occurs when a point enters in an area that has different properties. A Spatial Manager contains all the properties of the space and provides coupling with other spatially explicit models. The formalism used to specify vector models is an adaptation from the DSDEVS formalism introduced by Barros (1996). DSDEVS allows the specification of dynamic structured networks of discrete event systems. Proofs that DSDEVS models are DEVS models and closed under composition are available in (Barros, 1996).

A 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 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. A DEVS point model (PM) is described as follow:

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

- 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 five 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.

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

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

- N is the network name,
- X_N is the network input values set,
- Y_N is the network output values set,
- GS_N is a set of geographical properties that contains the boundaries of the environment where the shape is evolving,
- χ is the name of the dynamic network executive,
- M_χ is the model of the executive χ.

In this shape network χ is an unmodified DSDEVS network executive in charge of the structure of the shape. The executive is an atomic model that contains all the coupling information for a network of models. χ contains the structure function defines the dynamic of the structure by managing, adding or deleting points to the shape network when structural changes has been scheduled by

ChS function of a point. The interested reader will find detailed description about DSDEVS and the network executive in (Barros, 1996).

In a simulation where multiple shapes evolves in space, duplicating spatial properties for each shape networks was not possible using a reasonable amount of memory. As an implementation issue to simplify the data input/output with GIS we have integrated another component to the framework, the *spatial manager*. 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 is evolving. The *spatial manager* is not a DEVS model, but a container for the states that are used in 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 will load the specific geographic information about the boundaries of the shapes that composes the environment where the phenomenon is evolving.

A similar 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 principles of wavelets propagation to determine the change in the shape of the simulated phenomenon, and simulating using variable time steps. It is not the topic of this paper to provide details about the vector propagation technique, an in depth specification of the Vector-DEVS technique is available in (Filippi and Bisgambiglia, 2002). As Vector-DEVS are DSDEVS models and thus DEVS models, coupling procedures are identical to the standard coupling procedures with other DEVS models, models are still accessible trough coupling between ports, and during simulation events passed from one type of models to another have the same format. Next section presents the implementation of the framework in the form of a toolkit.

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 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. Figure 2 describe the general architecture of JDEVS.

The only programming task that the domain specialist has to do is the redefinition of the four methods of a basic model. Once a basic model is created, it is stored in the library to be used later in a federation (coupled model). In case of a spatially distributed system, the federation is automatic in the simulation panels, atom cells or points are instantiated according to the property map exported from the GIS, interfaces of the resulted model beiing stored in XML in the library with reference to the initialization map. For a component diagram model, the coupling between models is made graphically by the modeller in the block diagram GUI, then stored in XML into the library. This section presents the details of the five modules.

4.1 Modelling and simulation kernel

The modelling and simulation kernel is a java implementation of the DEVS formalism. The DEVS semantics is currently mapped to a set of JAVA instructions until a standard semantic becomes available. Basic and coupled models are described as follow.

4.1.1 Basic DEVS models definition

The DEVS formalism is offering well defined interfaces for the description of systems. The concept of model abstraction permits use of models that are coded in various object oriented languages. Those models are then accessed thought 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. Figure 3 shows generated Java code skeleton for:

$$\text{DevsAtom} = < X\{_{Xpi\{i1\},Xpf\{f1\}}\}, Y\{o1\}, S\{A\}, \delta_{int}, \delta_{ext}, \delta_{react}, \lambda, t_a >$$

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

4.1.2 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 and Santucci, 2002). Every kind of coupled models (Cellular, Vector, Diagram...) have a specific XML structure, however, to perform coupling between different types of models they must share a minimum set of nodes (the interested reader will find details about XML language in (Means and Harold, 2001)). The resulted XML document type definition with the minimum set of nodes for a coupled model is:

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

With *TYPE* defining the kind of coupled model, *NAME* the name of the model, *BOUNDS* the position of the model on the screen (used only by the GUI), *INPUT = (PORT*)* the set of input ports, *OUTPUT = (PORT*)* the set of output ports, *CHILD = (MODEL*)* the index for the components of the coupled model (in the priority order). For the coupling a generic *LINK = (PORT, PORT)* tag is created to store couplings, *EIC = (LINK*)* is the external input coupling set, *EOC = (LINK*)* the external output coupling set and *IC = (LINK*)* the internal coupling set. Each coupled model is stored in a different XML file. Thanks to the XML format, the definition can be extended for other kind of model (such as cellular models) and new tags can be added while keeping the file backward compatible. A specific parser is defined for each type of models. The specific parser automatically instantiates the models and creates the links recursively from the top of the simulation tree to the atomics. If a model is constituted of different types of sub-models, the parser which is at the top of the simulation tree instantiates the appropriate parser for each *CHILD* node.

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 simulations. 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. Figure 4 shows the modelling and simulation interface.

At the left stands 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 stands the hierarchical block diagram representation of the model, all components can be moved with the mouse, the linking between models is performed by a click from the origin port to the destination port. On the right stand the property 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 stands the simulation panel, the user load the input events from this panel and run 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 representation of the models is then displaying the event queue and the states of the selected models during the run.

The coupling between different kind of models is done with this interface. Selection of specific kinds of models, such as Cellular models, is made in the library and results in the creation of a block component representation of the model with input and output ports. If a simulation is run, the appropriate panels are automatically generated to track the simulation.

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 has when it has been stored. Several simulation scenarios can create several "context-in" model from the same "context-out" model description.

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.

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 is exported from the GIS to the spatial manager, and results are imported back after the simulation. Input/Output operations are performed via the Geotools (2002) java software library that supports the Arcview, ASCII and GML (Geographic Markup Language) 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 represents 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 outputted 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 'flattened' is order to create maps that can be sent back to the GIS. Those maps are snapshots of the simulation at a certain time. To create those snapshots, the program takes the initial value of the map and apply all changes that has occurred before the time 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 than 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.

The cellular panel allows 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. Otherwise the interactions are predefined in an interaction event file. The general architecture adapted from Wainer and Giambiasi (2001) and shown in figure 5 has been adopted to model cellular systems.

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 cell 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 perform the coupling. Once the model is created, it is sent to the simulator that performs the simulation. During the simulation run, the evolution of the model can be observed in the 2D or 3D simulation panels (figure 6).

Those 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 stake holder, then publish the experiment on the web. The stake holder will then be able to try himself different simulation scenarios.

4.6 Vector simulation panels

The vector simulation panels are the experimental frames for the vector models. Vector models and cellular models are superposed in figure 7 to shows a sample simulation of vector model and the equivalent cellular model (vote automata (Moore, 1996)). On the figure, the two simulations are independent and superposed only to show the equivalence in terms of results.

In figure 7 (1) the model is in its initial state. The model is decomposed in four points (A,B,C,D) that shares the same location but have different displacement vectors. We chose four points to have here the 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 have a displacement vector that can vary in speed and direction. A point will be decomposed when it collides another surface or if its link to another point collides 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 activate 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, the points are sending a message containing 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 figure 7 (2) the 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).

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

We can see from the figure 7 the advantages in terms of computational cost of the technique: for this cellular voting model, and at this terrain resolution the vector models calculates 20 intersections between each point and each vertices, while the cellular model makes 46 transitions (one for each cell). Basically, both algorithms are of the same complexity (O(n)), with n = 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 which extent the vector modelling paradigm is applicable. The paradigm seems also appropriate for the modelling of propagation of fire fronts on a large scale with a reasonable level of details (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 front).

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 use a case-study to illustrate the main advantages of using JDEVS: coupling and reusability of models in a multi-paradigm framework. The two following 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 illustrates the new modelling scenarios 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 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 a multi paradigm model resulting of the coupling of these models is greatly simplified because models are sharing the same simulation engine and the same interfaces.

To manage natural resources such as water, it is necessary to model the phenomenon that alter those natural resources in order to quantify and qualify them. The sample model is adapted from Zeigler et al. (1996). Figure 8 shows a simulation of the model developed to quantify pollution in catchments 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:

<X{N, S, E, W, in },Y{N, S, E, W, out},S{poros, elev, pollut}> (N,S,E,W corresponds to the North, South, East and West ports of their neighborhood). In this model:

- The λ function (output) is sending to the neighboring cells its elevation and quantity of pollutant. This function is called by the simulator in case of an activation.
- The δ_{ext} function (input) is receiving the quantity of pollutant and altitude from the neighboring cells, and send an activation message if its elevation is significantly lower than the elevation of the neighboring cell.
- The δ_{int} function (internal) is called when the cell receive an activation. It updates the states (here the quantity of pollutant) according to 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 according to the quantity of pollutant (thus defining the flow speed). Here is the Java transcription of the output function, λ, as a code example:

```
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;}
```

Those four functions constitutes the only programming task 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, the only work that has to be done is in the data preprocessing into the GIS (generation of ASCII raster maps of the initial states: porosity, elevation and quantity pollutant, choice of cell size). The 3d simulation panel serves as the experimental frame of the simulation of these phenomena. Java3D library is used to paint the outputs of 2d or 3d cellular models. The elevation map exported from the GIS permits to reconstruct a 3d world and the 3d panel enables here the visualizations of the polluted zones. To interact with the model, it is possible to click on the map during the simulation run and add pollutant to a specific cell. General outputs of the model are a file containing a set of each output events that occurred during the simulation and a file containing raster ASCII maps generated at fixed times during the simulation run.

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 ANN (Artificial Neural Network) in a DEVS framework.
Figure 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 the 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 kind of area of the patch, OTH,
- The animal unit density, ANI,
- The daily runoff over the patch, FLO,
- The daily precipitation over the patch, PRE.

In figure 9, the box A corresponds to the Feedback-DEVS model that encapsulates the pre-trained neural-network. In this model the variable FOR, AGR, URB, WET, OTH and ANI are parameters. The model is simulating the response in terms of 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 models 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 have a Feedback input port 'FeedbackTNC' when an external event is received on this port the model triggers the learning function (here back-propagation (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 outputted 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 outputted 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 is loaded as data series in the JDEVS GUI using the simulation panel. The output of the simulation is not the quantity of pollutant at every time step (driven by the daily input data), but instead the model outputs events each time a certain amount of nitrogen is reached. The information given by the output of the model is carried by the time 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 event will be separated by a longer time. The general output of the model is an ASCII file containing a set of events that must be post processed in a spreadsheet or in a statistic processing software.

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 that the data that will pass from the a model corresponds with the data required by the model it will be connected to. In this experiment the data are compatible, the nitrogen concentration model will send a certain quantity of nitrogen at various time, while the cellular model is propagating a certain 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 of the subject of the experiment is be to test the impact of a new land use for the land patch represented by the cell, so the variables FOR, AGR, URB, WET, OTH and ANI are given before the simulation by the user.

The coupling between the two models is made in the hierarchical blocks interface by loading the two coupled models (feedback and cellular) from the library into a blank coupled model. The resulted XML tag specifying the coupling of the nitrogen and pollution model is:

```
<IC><LINK>
```

```
      <PORT model="NitroModel.xml">NitroE</PORT>
      <PORT model="cell.xml">in[40-83]</PORT>
</LINK></IC>
```

The file specifies an internal coupling between the port 'NitroE' of the nitrogen model with the 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. General outputs of the model are an ASCII file containing a set of each output events that occurred during the simulation and a file containing raster ASCII maps generated at fixed times during the simulation run.

A DEVS multi-model, combining neural-networks and cellular automata paradigms has been presented in this section to illustrate the modelling process in JDEVS. The next section concludes by briefly comparing JDEVS with other environmental modelling tools and present the current perspectives of work.

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 step by step instructions on how the model should be simulated. JDEVS, the java implementation of the framework, help 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 Perestrello, 1997) or 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 two, JDEVS compares with those environments as follow:

- Features: SWARM, SELES and ECLPSS are limited to spatially explicit modelling paradigms while OME is non spatial. Like JDEVS a specific semantic is used in IMA and SME to describe the behavior of models. Having a higher level language permits to add new modelling paradigm to the environment and still ensures compatibility between models. Nevertheless only JDEVS use a high level formalism proven to be close 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 the use of the wide variety of modelling paradigms that already extend DEVS. In particular the Feedback-DEVS and Vector-DEVS developed in this paper are extending the features of JDEVS with the integration of vector based models and supervised-learning modelling paradigms.

- Interfaces: In terms of database interfaces, most of the environmental modelling softwares provides an interface with a GIS. Like SWARM, SELES and ECLPSS, JDEVS is loosely coupling with the GIS, exchanging data with input and output files. Only SME can provide a tight coupling with GIS. JDEVS also provides an interface with a model library similar in concept to the SME or IMA models libraries. In terms of component interfaces, JDEVS uses port based interfaces that simplifies the coupling process with other DEVS models. The other modelling environment are coupled with references to models variables.
- Abstractions: SWARM, SELES and ECLPSS makes an abstraction of space in terms of cells. Although it should be possible to specify models that makes the use of a vector representation in IMA and SME only JDEVS provide a technique to specify vector based models. Like SWARM and SELES, JDEVS is using 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 are using agents and cells as basic components for the descriptions of their models. Like in OME IMA and SME it is possible to specify new kinds of components in JDEVS to implement the 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. It also exists conceptual limitations because computer models are limited by the assumptions made in the modelling paradigm used. Only IMA, SME and JDEVS enable multi-paradigm modelling and are not limited to a fixed knowledge representation scheme.

The need of a formal framework in environmental modelling extend 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 provides 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 kind of models without having to specify how those models should be simulated. As further developments, we are extending the 'spatial manager' to enable a tighter coupling between the GIS and the simulated models as well as between vector and cellular models. The next release of JDEVS will include a full integration of all the modules into an unified tool with a single interface to simplifies coupling procedures between different types of models.

Acknowledgements

This research is partially funded by a grant from the Corsican Regional Education and Research Council. We want to thank M.L. Nivet, L. Floriani, C. Paoli, D. Patel, and the 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. Vol. 1. pp. 14–17, new Orleans, USA.

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 netherland. International Journal an Geographical Information Systems 5 (1), 361 367.

Coquillard, P., Hill, D., 1997. Modélisation et Simulation des Ecosystèmes. 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. Tech. Rep. 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. pp. 1964–1970, san Diego, CA, USA.

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. Vol. 1. pp. 647–652, marseille, France.

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. pp. 209–216, coeur Alene, USA.

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, http://www.geotools.org.

Lek, S., Guiresse, M., Giraudel, J., 1999. Predicting stream nitrogen concentration from watershed features using neural networks. International Journal an 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., Perestrello, Costanza, R., 1997. An open geographic modeling environment. Simulation Journal 68 (3), 175–185.

Means, W., Harold, E., 2001. XML in a Nutshell. O Reilly, iSBN: 0-596-00058-8.

Moore, C., 1996. Majority-vote cellular automata, ising dynamics, and p-completeness. Tech. Rep. 96-08-060, Santa Fe Institute.

OpenGIS, 2002. Open geographic information system project consortium, http://www.opengis.org/.

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

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.

Santucci, J., Bisgambiglia, P., Federici, D., 1998. Behavioral fault simulation. Advanced technique for embedded systems design and test , 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, http://www.sisostds.org/.

SWARM, 2002. Swarm development group, http://www.swarm.org/.

Vangheluwe, H., Lara, J., Mosterman, P., 2002. An introduction to multi-paradigm modelling and simulation. In: Proceedings of the AIS 2002 Conference. Vol. 1. pp. 9–20, lisboa, Portugal.

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 modelling 137 (1), 23–42.

Wainer, G., Giambiasi, N., January 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 modelling 150 (3), 211–238.

Zeigler, B., 1984. Multifaceted modelling and Discrete Event Simulation. Academic Press, iSBN: 0.12.778450.0.

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

Zeigler, B., Praehofer, H., Kim, T., 2000. Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press, iSBN: 0127784551.

FIGURES

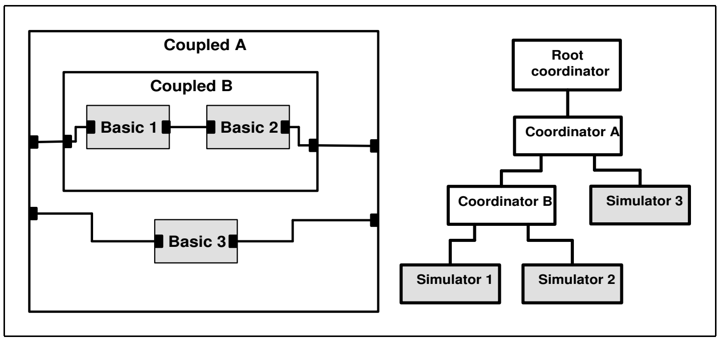


Fig. 1. A DEVS model (left) with corresponding simulation tree (right). Grey boxes corresponds to the basic models and their simulators, white boxes to the coupled models and their coordinators.

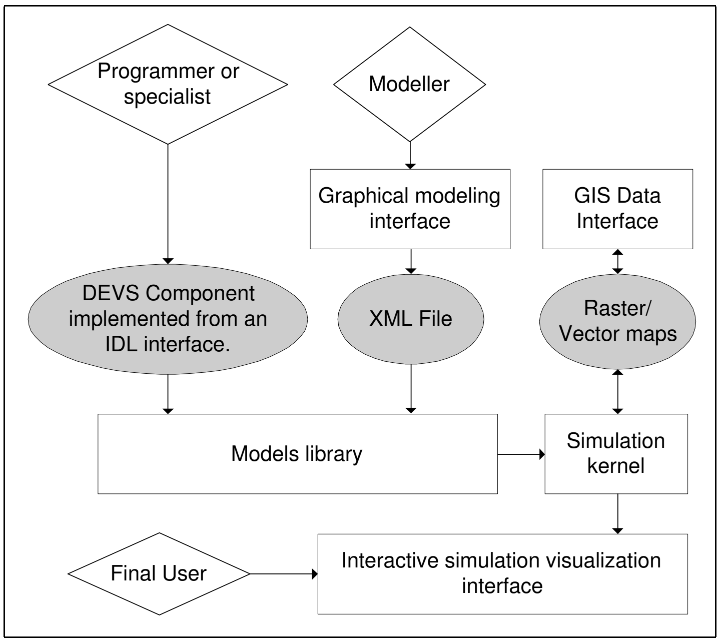
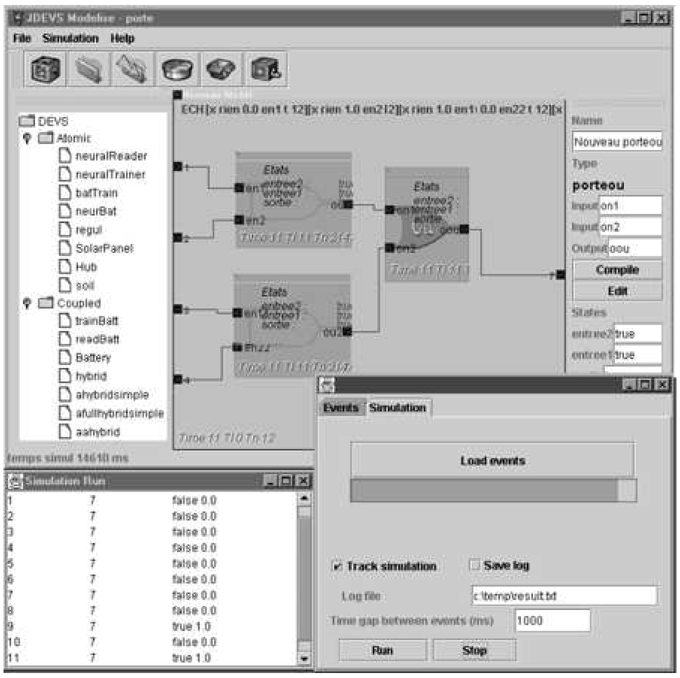

Programmer or specialist
Modeller
Graphical modeling interface
GIS Data Interface
DEVS Component implemented from an IDL interface.
XML File
Raster/ Vector maps
Models library
Simulation kernel
Final User
Interactive simulation visualization interface

Fig. 2. JDEVS toolkit architecture. Diamonds corresponds to human interactions, squares corresponds to the modules and circles to the interchange formats.

```
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;}
        }
}
```

Fig. 3. Java skeleton code for DevsAtom



ECH [x rien 0.0 en1 t 12][x rien 1.0 en2 t 2][x rien 1.0 en1 0.0 en22 t 12][x

Name: Nouveau porteou
Input: on1
Input: on2
Output: oou
entree2: true
entree1: true

temps simul 14610 ms

Simulation Run

```
1            7            false 0.0
2            7            false 0.0
3            7            false 0.0
4            7            false 0.0
5            7            false 0.0
6            7            false 0.0
7            7            false 0.0
8            7            false 0.0
9            7            true 1.0
10           7            false 0.0
11           7            true 1.0
```

[✓ Track simulation]   Save log
Log file: c:\temp\result.txt
Time gap between events (ms): 1000

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

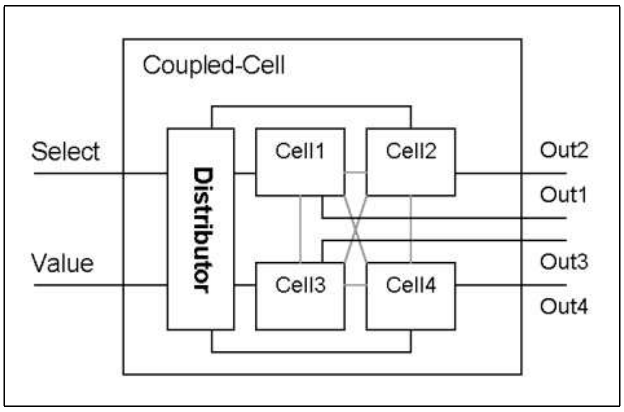


Fig. 5. Cellular models architecture.

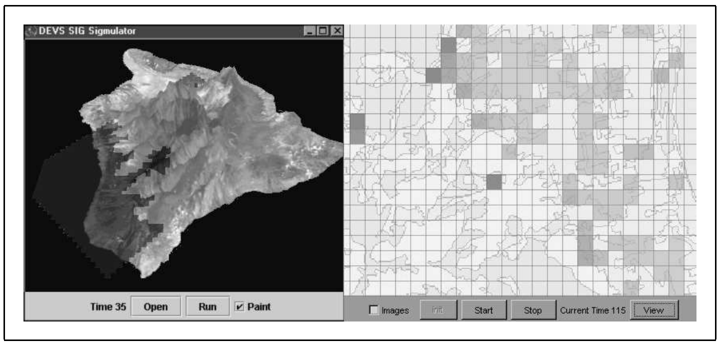


Fig. 6. 3D and 2D interactive visualization panels

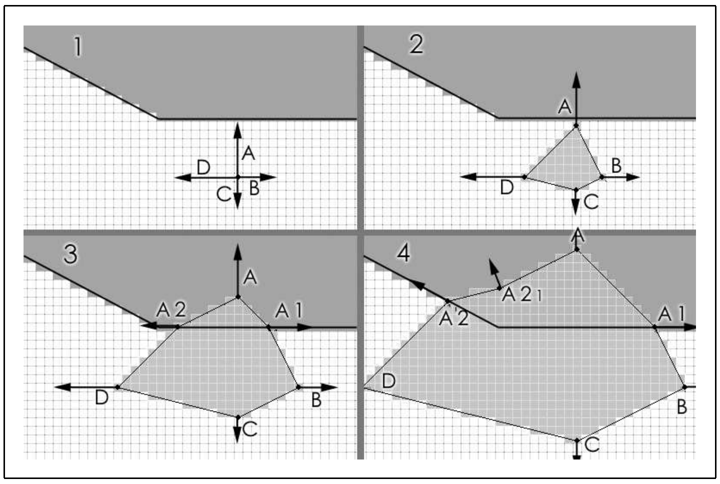


Fig. 7. Vector propagation, in its initial state (1), first collision(2), second collision for point A2(3) and last collision (4)

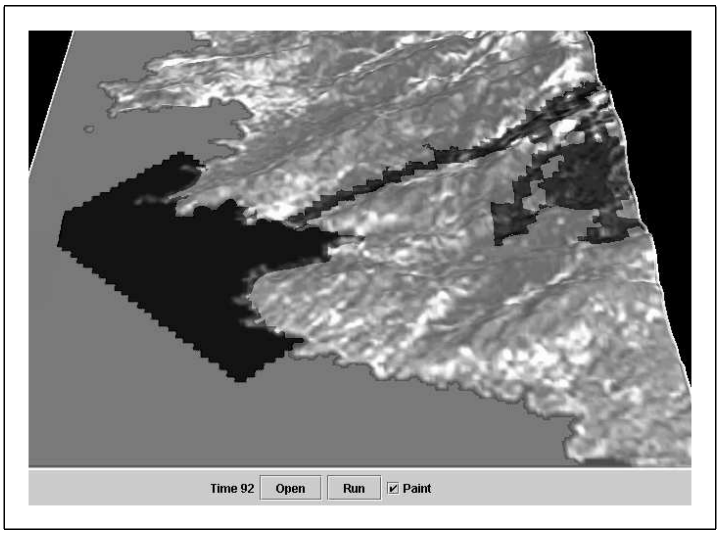
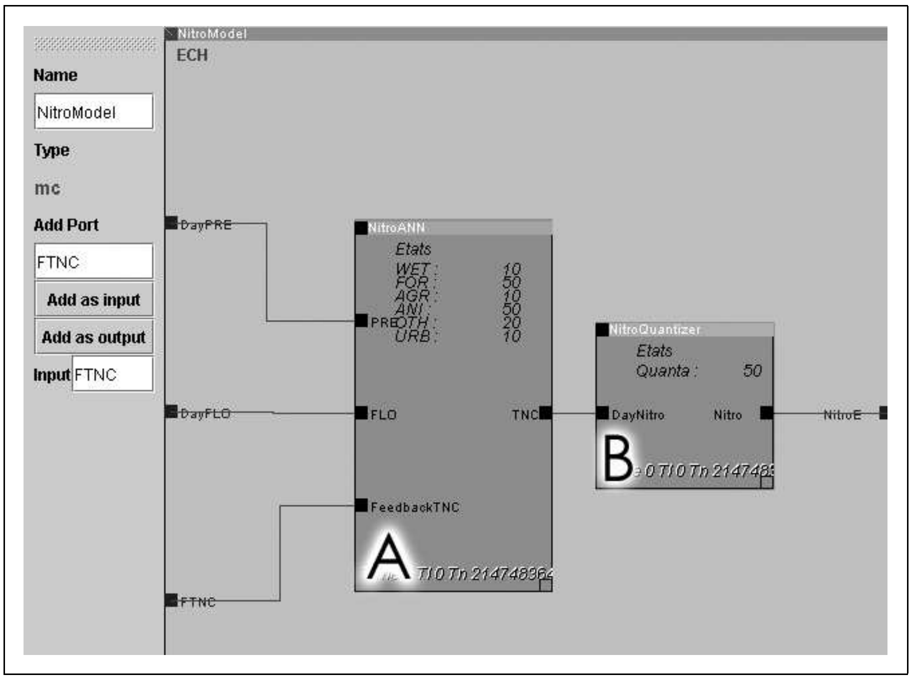


Fig. 8. Pollution model in the 3D panel



Name: NitroModel
Type: mc
Add Port: FTNC
Input: FTNC

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