A Generic Framework for Environmental Modeling and Simulation*

Fabrice Bernardi, Jean-Baptiste Filippi, Jean-François Santucci
University of Corsica, SPE Laboratory, UMR CNRS 6134
B.P. 52, 20250 Corte, France
{bernardi, filippi, santucci}@univ-corse.fr

Abstract – *Because of their complexity, natural systems are often studied using various modeling paradigms. This paper describes a generic framework for natural systems studies allowing the modeler to use all these paradigms in a unique environment. This framework is composed of a DEVS based modeling and simulation environment called JDEVS, and a models library called HMLib. The associated formal framework ensures that models are reusable and interoperable components with well-defined interfaces. Integration is performed using a Web based connector allowing a distributed work. The coupling between these two tools provides a powerful framework focusing on models interoperability and reusability.*

Keywords: Discrete event simulation, natural systems modeling, modeling and simulation environment, models library, interoperability, reusability.

1 Introduction

Natural systems are among the most difficult systems to be modeled and simulated. On the one side, they are often acting over large spatial scales, long time frames and heterogeonous units of study. On the other side, experiments can be very complex and very hard to setup. To face these problems, modelers belonging to a same team often use different specific modeling paradigms, thus different specific modeling and simulation environments that are often pieces of software they themselves designed. They want also to decompose a given problem into more understandable sub-problems that can be solved using reusable models [2, 3].

DEVS (Discrete EVent System specification) is a set-theoretic formalism that includes a formal representation capable of mathematical manipulation just as differential equations serves this role for continuous systems. Our major motivation in using DEVS as an unifier paradigm is that H. Vangheluwe demonstrates recently in his meta-modeling approach that many paradigms [16, 15] (Petri-nets, ODE, State Charts, Bond Graphs,...) can be easily mapped in a DEVS representation. We propose in this article a DEVS (Discrete EVent Specification, [19, 20, 16]) based framework for natural systems modeling and simulation focusing on modeling paradigms interoperability and models reusability. The whole framework is constituted of a generic modeling and simulation environment called JDEVS associated with a hierarchical models library called HMLib. It is based on the classical DEVS formalism and a new paradigm called Feedback-DEVS allowing an empirical modeling of complex systems. We think that our approach can dramatically simplify the modelers' work since they use only one modeling and simulation environment with its integrated storage architecture to perform their complex modeling works.

JDEVS has been developed for over three years and enables object-oriented, component based, GIS connected, collaborative, visual simulation model development and execution. In our framework, it is associated with a hierarchical models library alllowing reusability of previously validated models.

2 The JDEVS Theoretical Foundations

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. 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,

*0-7803-7952-7/03/$17.00 © 2003 IEEE.

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 modeling environments that uses discrete events such as [14] or SELES [10].

DEVS is well adapted to be implemented in an object oriented framework, thus creating a component based modeling and simulation environment. It is also possible to generate a continuous time simulator out of a system modeled using the DEVS formalism as all DEVS models already maps to 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 [19].

We will not describe in detail the abstract simulators of DEVS models that can be found in [18]. Basically each basic model has a "Simulator" attached to it that trigger the execution of the functions by sending and receiving messages to a "Coordinator" in charge of its coupled model. A "Coordinator" is attached to every Coupled Model and dispatch the events to their destinations. 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 a simulation tree.

3 The HMLib Models Library

HMLib is a models library structured according to two paradigms: the application domains and the abstraction levels. Its three main characteristics are a storage independence from the considered application domain, the management of an inheritance hierarchy between the stored models, and the management of abstraction links between stored models. The genericity of use is performed through a dissociation of the contents of the model to be stored from its format using the *context* notion.

3.1 *Context-in* and *Context-out* Models

A storage architecture and a modeling and simulation environment perceive a same model in a totally different way. In the first case, the model is passive. It can not handle external entreaties. In the second case, the model is active since it is placed in a context of use. It can handle external entreaties arriving on its communication ports. Building a model storage architecture appears then as building an architecture allowing the representation of models out of their context of use.

In order to manage these two points of view, we introduce two notions, *context-in* and *context-out* models: A context-out model is an abstraction of a model. It is defined or encapsulated in a special format allowing it to be stored in a library. All the stored models share this special format.A context-in model is a context-out model extracted from its library and directly usable in its modeling and simulation environment. This extraction is in fact a format conversion.

The context-out format must be able to be adapted to all the model forms that the library designer could have to manage. This point implies very precise formal descriptions of the elements able to be stored in a library. The notion of context is very important in our approach since it dictates all our design process, allowing us for instance to introduce different storage elements totally independent from any modeling and simulation environment.

3.2 HMLib Concepts

The storage independence of the library is performed using a *Domain Parser* [8]. We call Domain Parser an object able to be used in two distinct modes, and able to analyze or to create a file that describes a context-in model. A Domain Parser relies upon a separation methodology of the extent of the model from its description format. Thus, the selected approach consists in defining a separation methodology for each domain in a library. Using such a methodology, a Domain Parser allows the user to transform a context-in model to a context-out one. The main advantage of this approach is that, never mind the model is, it can be placed context-out. Our implementation of a Domain Analyzer is based on the Builder Design Pattern used in order to "separate the construction of a complex object from its representation so that the same construction process can create different representations". In our implementation, we use the XML language [17] in order to define context-out models.

Inside HMLib, the abstraction hierarchy management is performed using the notion of *Abstraction Matrix*. The values of this square matrix are composed by the difference between the two abstraction levels of two models. If models are the same, the value is 0, the same as if they have no "abstraction relationship". We provide also the management of transfer functions allowing the modeler to perform automatically the necessary changings on the model directly in the modeling and simulation environment.

In order to avoid a properties repetition inside a same kind of models, a models library must deal with an inheritance between the stored models. This inheritance between a parent model and its children allows to store these shared properties inside some special models (parent models), dramatically simplifies the children models, and facilitates the maintenance of the whole library.

Furthermore, this inheritance hierarchy inside a library provides all the classical benefits of object inheritance: automatic properties transmission, methods overloading if components are described using algorithmic functions. For R.C Rosenberg, “the importance of a good models library is that the model designer can be supplied with reasonable alternatives”. The choice between these alternatives can be strongly facilitated if the inheritance hierarchy is structured in a smart way. In our architecture, the management of inheritance between models is performed using the characteristics of XML.

4 Implementation of the Framework

The whole framework has been implemented using an object-oriented approach and the Java language. The two parts (JDEVS and HMLib) have been designed separatly and then coupled using a network based connector.

4.1 JDEVS Object-Oriented Implementation

JDEVS is composed of four independent modules written in Java. A simulation kernel, a graphical block modeling interface, 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 1 describes the general architecture of JDEVS. We can see in this picture how the models library is integrated in the modeling and simulation environment. All JDEVS components can be stored in HMLib. These components can be JDEVS simulable models or JDEVS graphical components described using the XML language.

4.1.1 Modeling and simulation kernel

The modeling and simulation kernel is a Java implementation of the DEVS formalism. The DEVS semantics is yet mapped to a set of Java instructions until DEVS become the SISO standard currently developed by the DEVS standardization group [13]. Basic and coupled models are described as follow.

4.1.2 Basic DEVS models definition

The DEVS formalism is offering well defined interfaces for the description of systems. The concept of model abstraction permits to use models that are coded in various object oriented languages. Those models are then accessed thought a software interface specified in DEVS with Java Remote Method Invocation (RMI). Modeling basic models in JDEVS is done directly in Java. To help the modeler in this task, the GUI generates a Java skeleton, stores it in the models library and compiles it.

4.1.3 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 [6].

4.1.4 Hierarchical block modeling and simulation interface

The graphical user interface is the modeling front-end of the toolkit and, 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 modeling is made using the GUI, if different modelers works on sub-coupled models and store them in the same library, it is possible to federate those models in another graphical modeling client.

4.1.5 GIS interconnection

[9] 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 Java software library that supports the Arcview, ASCII and GML (Geographic Markup Language) formats.

4.1.6 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 an the panels are launched. Coupling between ports of a cellular and a hierarchical block model is defined in a XML parameter file similar to the coupled model description file. Yet, 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.

4.2 JDEVS Models Management in HMLib

JDEVS and HMLib have been designed separatly, but they can act together as a single application using a network based connector. This connector is defined in the HMLib theoretical specifications as a set of object-oriented interfaces. The primary goal is to define a possibility for a modeling and simulation environment to access the storage engine through a network using a Java servlet approach [11, 12, 1]. These interfaces have been implemented in JDEVS, and the whole framework enables a collaborative and Web based work.

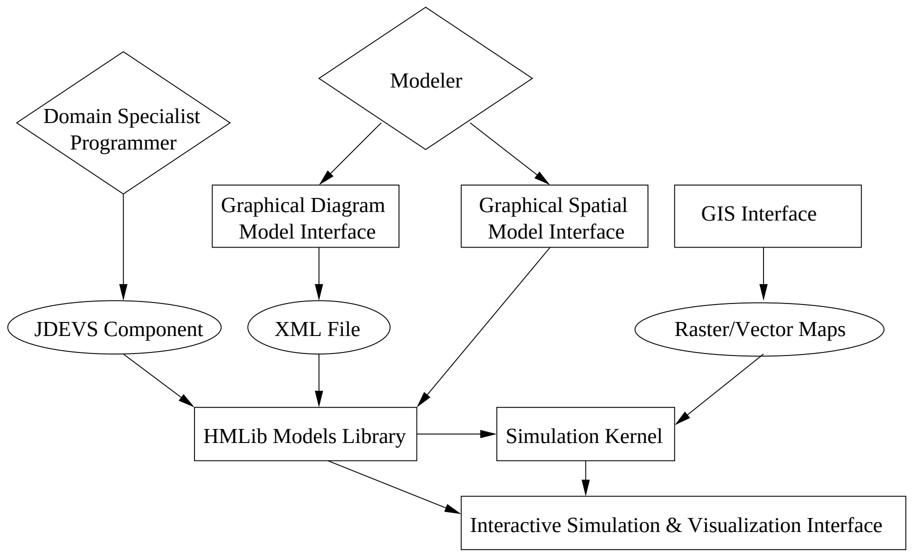
Modeler
Domain Specialist Programmer
Graphical Diagram Model Interface
Graphical Spatial Model Interface
GIS Interface
JDEVS Component
XML File
Raster/Vector Maps
HMLib Models Library
Simulation Kernel
Interactive Simulation & Visualization Interface

Figure 1: JDEVS toolkit architecture. Diamonds corresponds to human interactions, squares corresponds to the modules and circles to the interchange formats.

The basic idea of our approach is to follow the classic three tiers architecture. We use a Java application server coupled with a Web server, and a set of servlets [7]. These servlets can be directly accessed by JDEVS using a network socket.

5 Application: Fruit-fly Propagation

The section use a study of the geographic distribution of adult Mediterranean fruit fly, or medfly, to illustrate the main advantages of using JDEVS : coupling and reusability of models in a multi-paradigm framework. The purpose of this application is to illustrate the new modeling scenarios possible by the use of the formal framework to couple two different models in nature. 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 of Feedback-DEVS model, illustrates the integration of a neural network in a DEVS basic model. One is a model of spatial organization, the second is a non spatial empirical model. NevesectionApplication, Fruit-fly propagation The section use a study of the geographic distribution of adult Mediterranean fruit fly, or medfly, to illustrate the main advantages of using JDEVS : coupling and reusability of models in a multi-paradigm framework. The purpose of this application is to illustrate the new modeling scenarios possible by the use of the formal framework to couple two different models in nature. 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 of Feedback-DEVS model, illustrates the integration of a neural network in a DEVS basic model. One is a model of spatial organization, 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 are sharing the same simulation engine and the same interfaces.

5.1 The Mediterranean fruit fly

The medfly is one of the most serious economic pests of the fruit and vegetables. If control methods are not used, medfly can infest 100 percent of susceptible fruit such as apricots, pomelos and peaches and to a lesser extent, fruits such as apples and clementines. For suppressing and eradicate population of the medfly, a specific and environmentally non-polluting method of medfly control called SIT (sterile insect technique) is used increasingly. This technique consists in release a large number of sterile males over the sufficient period time at the best location. One of the main objective of the model is to estimate geographical distribution of adult medfly. This model is to be used for guidance in implementing eradication procedures and preventing spread to other locations. The onset of medfly activity is temperature dependent. In southern France (Corsica island) medfly is active in late spring, summer and autumn, when temperatures exceed an average climatic condition. Medfly can over the winter as adults, as eggs and larvae (in fruit), or as pupae in the ground. As temperatures increase in spring, adults begin to emerge from the ground and flies become active.

5.2 Cellular spread model

To manage potential geographical distribution of the Mediterranean fruit fly, it is necessary to model the phenomenon that alter those phenomena in order to quantify and qualify them. Figure 1 shows a simulation of the model developed to quantify medfly population in specific area. Like any other basic model, this cellular spread 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, in1, in2 },Y{N, S, E, W, out},S{host, ripe, Neggs, Nlarvae, Npupae, Nadults, Aeggs, Alarvae, Apupae, Aadults, Afood }> where :

N, S, E, W corresponds to the North, South, East and West ports of their neighborhood ;
in1, out corresponds to the temperature of the day and to the number of flies exchanged ;
in2 corresponds to values of the population at each stage of the life cycle (eggs, larvae, pupae, adults) ;
host : the species hosts trees apricots, pomelos, peaches, apples, clementines, other. other corresponds to a medfly insensitive cell ;
ripe = false, true depends on the ripening period of the host tree ;
Neggs, Nlarvae, Npupae, Nadults are values to the population at each stage ;
Aeggs, Alarvae, Apupae, Aadults corresponds to the average oldest of the population at each stage ;
Afood correspond to the average day of diet for adults population.

Two specials functions that the specialist (ecologist) has to implement in order to have his model working must be define : scattering( ) and development( ). The first function calculates the number of adults flies waste by the cell, the second function describe the biological model of the life cycle corresponding to interactions for each of the four types of stages. This two functions are adapted from CLIMEX model [2]. In this model :

- The λ function (output) is sending to the neighboring cells a quantity of adults flies given by the scattering() function and its Aadults and Afood values. This function is called by the simulator in case of activation.

- The δ_{ext} function (input) receives a number of adults from the neighboring cells, a temperature of the day or the values of the population at each stage and sends an activation message.

- The δ_{int} function (internal) is called when the cell receives an activation. It updates the states according to the function development() if the message is receives on the in1 port, initialise the Nvalue if the message is receives on the in2 port. In the other case the values Nadults, Aadults and Afood or changes taking into account the values received by the activation message.

- The t_a function (time advance) defines the time to the next self-activation of the cell according to the number of adults (thus defining the propagation speed).

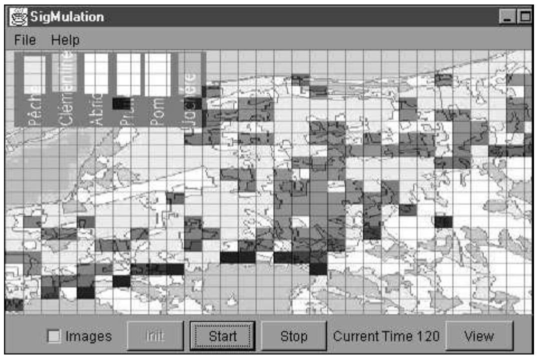

Figure 2: A Generated Cell Model.

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 initials stages : number of the flies at each stage, host type of the cell, choice of cell size). The 2d/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. To interact with the model, it is possible to click on the map during the simulation run and add flies to a specific cell. Figure 2 present a generated cell model.

6 Conclusion and Perspectives of Work

This paper has presented a generic framework for natural systems studies composed by a modeling and simulation environment called JDEVS coupled with a hierarchical models library called HMLib.

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 of a general formalism. In terms of features, abstraction, components and interfaces, JDEVS provides the advantages of a domain specific modeling environment. With JDEVS, it is also possible to couple and simulate different kinds of models without having to specify how those models should be simulated.

The HMLib models library is built on the concepts of context, abstraction hierarchy and genericity of use

[4]. It allows the modeler to perform its model design very quickly, once models have been introduced stored. We provide also this user with a set of connectors allowing him to process his models through a Web based approach [5, 7]. We have two main objectives for this work. First, we are currently studying how to manage the abstraction hierarchy to store the various information levels used in Geographical Information Systems. We want to see how we can perform automatic abstraction transformations between stored models using the context-out models. Secondly, we are beginning also to study how this context-out format could allow us to perform domain transformations between stored models. We think that this point would be a great improvment to our approach since, for example, the transformation needed to make an a model understandable by JDEVS would be performed automatically.

References

[1] K.Z. Ahmed and C.E. Umrysh. *Developing Entreprise Java Applications with J2EE and UML.* Addison-Wesley, 2001.

[2] O. Balci, A.I. Bertelrud, C.M. Esterbrook, and R.E. Nance. Developing a Library of Reusable Model Components by Using the Visual Simulation Environment. In *Proceedings of the SSC'97*, 1997. San Diego, CA, USA.

[3] D. Batory and S. O'Malley. The Design and Implementation of Hierarchical Software Systems with Reusable Components. *ACM Transactions on Software Engineering and Methodology*, 1992.

[4] F. Bernardi, J.B. Filippi, and J.F. Santucci. XML Object-Oriented Models Libraries with Web-Based Access Capacities. In *Proceedings of ICSSEA 2001*, 2001.

[5] F. Bernardi and J.F. Santucci. Developing a Web-Based Models Library for a DEVS Modeling and Simulation Environment. In *Proceedings of AIS 2002*, 2002.

[6] F. Bernardi and J.F. Santucci. Model design using hierarchical web-based libraries. In *Proceedings of the 39th conference on Design automation*, volume 1, pages 14–17, 2002. New Orleans, USA.

[7] F. Bernardi and J.F. Santucci. Model Design Using Hierarchical Web-Based Libraries. In *Proceedings of DAC 2002*, 2002.

[8] F. Bernardi and J.F. Santucci. A domain parser for a generic models library. In *Proceedings of ISCA CATA 2003*, volume 1, 2003. Honolulu, Hawaï, USA.

[9] J.E. Brandmeyer and H.A. Karimi. Coupling methodologies for environmental models. *Environmental Modelling and Software*, 15(5):479–488, 2000.

[10] A. Fall and J. Fall. A domain specific language for models of landscape dynamics. *Ecological modelling*, 141(1-3):1–18, 2002.

[11] K. Moss. *Java Servlets, Second Edition.* McGraw-Hill, 1999.

[12] P.Y. Saumont and A. Mirecourt. *Servlets et JavaServerPages, le Guide du Développeur.* Osman Eyrolles Multimedia, 2000.

[13] SISO. Simulation interoperability standards organization, http://www.sisostds.org/, 2002.

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

[15] H. Vangheluwe, J. de Lara, and P.J. Mosterman. An Introduction to Multi-Paradigm Modelling and Simulation. In *Proceedings of AIS02*, 2002.

[16] H.L. Vangheluwe. DEVS as a Common Denominator for Multi-Formalism Hybrid Systems Modelling. In *Proceedings of ISCACS 2000*, 2000.

[17] W3C Consortium. *Extensible Markup Language (XML) 1.0*, 1998.

[18] B.P. Zeigler. *Multifaceted modelling and Discrete Event Simulation.* Academic Press, 1984. ISBN: 0.12.778450.0.

[19] B.P. Zeigler. *Object-Oriented Simulation with Hierarchical, Modular Models.* Academic Press, 1990.

[20] B.P. Zeigler, H. Praehofer, and T.G. Kim. *Theory of Modeling and Simulation, Second Edition.* Academic Press, 2000.