DEVS Coupling of Spatial and Ordinary Differential Equations: VLE Framework

Gauthier Quesnel
LIL
50, rue Ferdinand Buisson
BP 719
62228 Calais, France
quesnel@lil.univ-littoral.fr

Raphaël Duboz
UR RAP – IRD
Avenue Jean Monnet
BP 171
34203 Sète, France
duboz@ird.fr

David Versmisse
LIL
50, rue Ferdinand Buisson
BP 719
62228 Calais, France
versmisse@lil.univ-littoral.fr

Éric Ramat
LIL
50, rue Ferdinand Buisson
BP 719
62228 Calais, France
ramat@lil.univ-littoral.fr

ABSTRACT
In this article, we study the different ways to develop a multi-paradigm model of a hybrid system. VLE (Virtual Laboratory Environment) offers two ways: the DEVS mapping and the DEVS wrapping in order to manage existing models in VLE framework. We deal with the wrapping technique which consists in connecting existing simulators models to DEVS compliant simulators. Our approach is illustrated with several simple examples: an ordinary differential equations system in the case of wrapping and a spatial differential equations system in the case of mapping. The second aim of this article is to propose an approach to develop a multimodel of hybrid system in the formal and efficient application programming interface of VLE.

KEYWORDS
DEVS, Wrapping, Mapping, ODE, PDE, Quantized System, Cell-DEVS

1. Introduction

Nowadays, it is recognized that multimodelling is a powerful concept for modelling and simulation of large complex systems. At the end of 80's, P.A. Fishwick and B.P. Zeigler [Fishwick 1992] introduced the multimodelling basis concepts. One can define multi-models as large models which are composed of different types of models (i.e. different paradigms) [Fishwick 1995]. Concepts like refinement and hierarchical composition are basis of multimodelling. The first describes the decomposition of one model into several other ones in order to refine the behaviour of the composed model. The last defines the opposite process: it is say models aggregation. In this context, a major issue is how to deal with the coupling of heterogeneous models. Several works deal with the coupling of heterogeneous models. For a review of concepts and techniques, see the book of B.P. Zeigler *et al.* [Zeigler *et al.* 2000]. With DEVS, Discrete Event System Specification [Zeigler 1976], B.P. Zeigler has provided formal basis for coupled model construction in a network or graph manner. If models we want to couple are specified in a unique formalism, it does not appear unsolvable problems, whereas the coupling of models which are formally different can lead to several coupling strategies.

The work presented here takes place in the field of formally heterogeneous models coupling. To tackle this issue, we briefly recall concepts like mapping and wrapping. Then we give example of "formal" wrapping and show the feasibility with a particular and very simple application.

1.1. Heterogeneous Models Coupling

In their book, B.P. Zeigler *et al*. give formal basis for the specification of multi-formalism models. In the same way, H. Vangheluwe proposes a framework for multi-paradigm modelling and simulation [Vangheluwe *et al.* 2002]. In all those works, the DEVS formalism appears as the common denominator for the integration of heterogeneous models. DEVS provides two ways for integration of non-DEVS models into DEVS simulator: Mapping and Wrapping:

> ***Mapping***: Mapping concept means to find a total equivalence between an existing formalism and DEVS. We can find an example in Jacques and Wainer's works for the mapping of Petri nets into DEVS [Jacques 2002].
>
> ***Wrapping***: Wrapping means to build an algorithm that is compatible with DEVS abstract simulators i.e. to develop a functional interface between a particular model and a DEVS simulator.

In this paper, we propose to deal with wrapping. As we have applications in ecology [Duboz 2003], we need to address systems complexity. Wrapping seems to be a powerful mean to achieve it. We develop a free and open source platform: VLE, available on sourceforge.net website. Initially, VLE only deals with agent based modelling. This platform is now oriented

toward the integration of heterogeneous models. Actually, VLE uses the DEVS formalism and abstract simulators [Zeigler *et al.* 2000] for coupling models by integrating the concept of DEVS-Bus [Kim 1996]. VLE provides a framework for heterogeneous simulators coupling. Furthermore, VLE integrates XML applications to describe experiments and model coupling [Duboz 2002]. VLE is oriented toward the managing of existing models. Figure 2 shows the DEVS-Bus framework in the context of VLE.

We distinguish two layers: the top layer which is in concern with the formal integration and the simulation layer which is in concern with abstract simulators integration into DEVS-Bus.

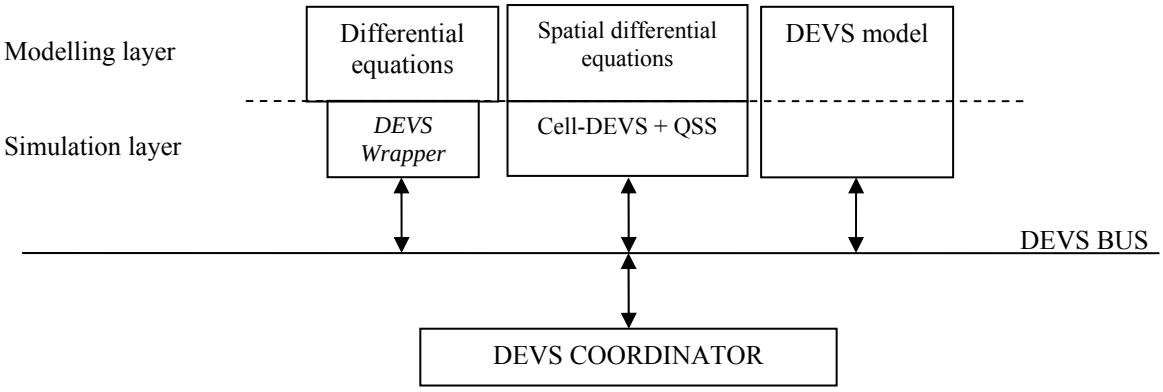


Figure 2: The use of DEVS-Bus for operational and formal coupling in VLE.

In this article, we discuss the articulation between the modelling and the simulation layer (Figure 2). It means that we address the issue of the wrapping of different pre-defined formal models into the VLE framework in order to simulate them.

In such a wrapping, the main difficulty is in concern with the connection of continuous time models or non timed models within the DEVS-Bus framework. Indeed, DEVS specifies discrete events based simulations. It may be difficult to connect simulators with another time scheduling or without explicit representation of what is an event. In this paper, we take an example with the coupling of a DEVS model with a Petri net and an ordinary differential equations system to illustrate this issue. Before presenting our example, we briefly present the formal basis of our works.

1.2. **DEVS and VLE**

A DEVS atomic model is defined as a structure:

$$M = \langle X, Y, S, \delta_{\text{int}}, \delta_{ext}, \lambda, ta \rangle$$

Where:

$X = \{(p,v) \mid p \in InPorts, v \in X_p\}$, is the set of input ports and values.

$Y = \{(p,v) \mid p \in OutPorts, v \in Y_p\}$, is the set of output ports and values.

S, is the set of sequential states.

$\delta_{\text{int}} : S \to S$, is the internal transition function.

$\delta_{ext} : Q \times X \to S$, is the external transition function.

$\lambda : S \to Y$, is the output function.

$ta : S \to \Re_0^+$, is the time advance function.

$Q = \{(s,e) \mid s \in S, 0 < e \leq ta(s)\}$,

Q is the set of total states,

e is the time elapsed since last transition.

A DEVS coupled model defines how to couple atomic models in order to build a new model. The property of closure under coupling guaranties that a DEVS coupled model is equivalent to an atomic one in an upper hierarchical level. Hierarchical decomposition and modularity are fundamental in the DEVS formalism [Zeigler *and al.* 2000]. In VLE, as in DEVS, we consider that δ_{int}, δ_{ext}, λ and ta are functions which encapsulate the dynamic and the behaviour of a model. We are particularly interested in formalism wrapping. Two questions are addressed:

1. "How to translate input and output event notions?"
2. "How to translate, δ_{int}, δ_{ext}, λ and ta?".

Indeed, if we are able to specify those functions considering a particular formalism, we can connect the associated simulators. Into the framework VLE, interaction between simulators and coordinators is given by the five "classical" DEVS functions [Zeigler *and al.* 2000].

```
EventList getOutputFunction(Time currentTime)
Time getTimeAdvance()
void init()
void processInternalEvent(InternalEvent event)
void processExternalEvent(ExternalEvent event)
```

1.3. **Wrapping**

Input and output events for non-DEVS formalism: If the designed formalism is based on the discrete events concept, it does not appear major problems. We just have to check that exchanged events are compatible in term of data type and semantics. If there is no notion of input and/or output event in the formalism we want to couple with a DEVS simulator, we need to find equivalence for those notions. We can propose various strategies for each formalisms, lets us give an example. What does an input or output event "means" for an Ordinary Differential Equation (ODE). In this case, input event can be seen as a perturbation on a particular variable of the ODE system. Therefore, the ODE model must incorporate the adequate behaviour in order to react in a good manner. For instance, the perturbation can be viewed as the definition of new initial conditions. As a consequence, the wrapper has to specify the type of input and output events it can accept. Then, some rules of translation (analogy) explicit how the data of the input events are interpreted in the target formalism.

In a more general way, the exchanged data related to events belong to a mathematical domain (For instance: $\Re^3$ or $\aleph^2 \times \Re$). When using a wrapper, this domain is constrained by the formalism encapsulated in the wrapper. If we consider a Coloured Petri nets (CPN) [Jensen 1997], we can imagine that attributes of coloured tokens take there values from information shared by input events. For outputs events, the wrapper of CPN must translate the arrival of a coloured token in a particular format to generate an output event. The rules of translation depend on the context of coupling, i.e. the models which are coupled with the wrapper [Quesnel 2005].

DEVS transition functions for non-DEVS formalisms: Transition functions specify the dynamic of state changes. These changes are triggered by two functions: the external transition function and the time advance function. Considering non-DEVS formalism, these functions we want to incorporate in a DEVS model must find an equivalent in the targeted formalism. In the case of the external transition function, we have to specify which functions are activated if an external event occurs. As the concept of state is specific to formalism, the state changes are described by the formalism. In other words, there are no major difficulties. In the case of time advance function, it can be more difficult. Indeed, we are facing the definition of time in formalisms. Time can be continuous, discrete or simply missing. In the latter case, the definition of time is given in the context of model coupling. The answer is then given by the context of model coupling.

2. **Wrapping and mapping in VLE**

In this section, we describe the shift from classical formalisms to a wrapper based on two examples: an ordinary differential equations system and a spatial differential equations system.

2.1. **ODE Wrapper**

In this section, we develop a wrapper to encapsulate an Ordinary Differential Equation system. The ODE system is resolved by the classical 4th order Runge-Kutta integration. This method is used in this paper in order to simplify the presentation. We can use any integration method. In VLE, we can deal with any ODE system following the form:

$$\frac{dx_i}{dt} = f_i\left(x_0,...,x_i,...,x_n\right)$$

Where equations are composed with different elements: variables, parameters, functions with parameters, operators (+, −, × or ÷) and real constants. The Runge-Kutta integration method can be change by another integration method like quantized methods QSS1 or QSS2 [Kofman 2001]. These last methods have the advantage to express with DEVS abstract simulator.

In the following sections, we discuss on wrapper signification for an ODE system, we define formally ODE Wrapper and we show an example.

2.1.1 Presentation

First, the equation parameters are initialised at time t_{min} and they don't change during the system resolution. This parameters are useful for us to perform some experiences with just modify their values.

The system variables can be modified at any moment. These modifications are named perturbation. In this wrapper, we define two types of perturbation: *additive* and *reset*. The aim of this property is to apply change on current state of model to manage incoming events. On each perturbation, the resolution algorithm is restarted (i.e. the perturbation defines new initial conditions).

Considering output event, we borrow an example from combined discrete and continuous formalism DEV&DESS [Zeigler *et al.* 2000]. We consider that output events are generated on thresholds. We distinguish three types of thresholds: a threshold on a value, on first derivative and on second derivate. Furthermore, we consider that an output event is generated at each time step of resolution of the integration method.

In the case of output events are generated when a threshold is reached, they are associated with a set of data which are built starting from the state of the system. The data types attached with event are specified by the modeller and they can depend on others models which are connected with the ODE wrapper.

After the translation of input and output events, we must define the transition functions. The internal transition function and the time advance function are related to the integration time step. At each time step, the state of system changes according to a numerical scheme.

According with the DEVS formalism, the output function is computed just before the internal transition function. In our example, the output function must generate an event if a threshold is reached. This computation is made possible thanks to an on line estimation of the derivatives. One can formalise this operation using a DEVS model.

2.1.2 Formal description

In this part, we formally define the ODE Wrapper:

$$M_{ode} = \left\langle X, Y, S, \delta_{\text{int}}, \delta_{ext}, \lambda, ta, \Omega, O \right\rangle$$

The set of states S, is define like:

$$S = \left(\vec{X}, \dot{\vec{X}}, \ddot{\vec{X}}, T \right)$$

Where:

$$\dot{\vec{X}} = \begin{pmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{pmatrix}$$

With $\omega \in \Omega$ set of definitions of thresholds, T a set of reached thresholds, $T \in \omega^p, T \subset \Omega, p \in \aleph$ number of reached threshold. $\omega = (i, \alpha, \sigma, v)$ defines of threshold with: i index of variable, α type of threshold (constant, derive1 and

derive2), $\sigma \in \{+,-\} \bigcup 0$. σ represents the direction of variation and v value of variable. $O = \{(p,\omega)\}$ defines the set of output port and threshold couple.

The internal transition is defined:

$$\delta_{\text{int}}\left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},\emptyset\right) = \left(\vec{X}',\dot{\vec{X}}',\ddot{\vec{X}}',T\right)$$

Where $\vec{X}'$, $\dot{\vec{X}}'$, $\ddot{\vec{X}}'$ are estimated variable at next time. We use Range-Kutta integration method to build estimated variables from $\vec{X}$ to $\vec{X}'$, $\dot{\vec{X}}$ to $\dot{\vec{X}}'$ and $\ddot{\vec{X}}$ to $\ddot{\vec{X}}'$. But all other integration method can be use.

$$\lambda\left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},\emptyset\right) = \varnothing$$

$$ta\left(\left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},\emptyset\right)\right) = \Delta$$

With $\Delta \in \Re$ is the integration step.

The previous functions show the integration of ODE. The following functions allow the management of events on the thresholds i.e. when $T \neq \varnothing$

$$\delta_{\text{int}}\left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},T\right) = \left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},\emptyset\right)$$

$$\lambda\left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},T\right) = \{(p,threshold(i,Xi))\}$$

$$ta\left(\left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},T\right)\right) = 0$$

Where i such as $\omega = (i,\alpha,\sigma,v) \in T$ and $(p,\omega) \in O$. In this article, we generalise the $ta(S)$ function if integration method don't use a synchronous method (for instance QSS2 [Kofman 2001]). If we use only a Range-Kutta integration, we can produce output events and next state of system in same time. In this case,

$$ta\left(\left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},T\right)\right) = \Delta$$

The external transition function is:

$$\delta_{ext}\left(\left((X_0,\ldots,X_i,\ldots,X_n),\dot{\vec{X}},\ddot{\vec{X}},T\right),e,(p,(i,v,\alpha))\right) = \begin{cases} \left((X_0,\ldots,X_i+v,\ldots,X_n),\dot{\vec{X}},\ddot{\vec{X}},T\right) if\ \alpha = additive \\ \left((X_0,\ldots,v,\ldots,X_n),\dot{\vec{X}},\ddot{\vec{X}},T\right) if\ \alpha = reset \end{cases}$$

$$ta\left(\left(\vec{X},\dot{\vec{X}},\ddot{\vec{X}},T\right)\right) = \sigma - e$$

Such as:

$$X = \{(p,(i,v,\alpha))\}, p \in InPorts.$$

Where $\dot{\vec{X}}'$ and $\ddot{\vec{X}}'$ are re-initialised. If $T \neq \varnothing$ then conflict between internal and external event. The modeller must determine priority management between internal and external events. σ is the time that rest in current state. This method allows system to advance if the wrapper receive external event between two steps.

2.1.3 Algorithms

In this part, we describe algorithms written into the ODE Wrapper of VLE framework. For the *processInternalEvent* function, we present only the detection of constant and positive threshold. The others cases are similar.

```
processInternalEvent(internalEvent : Event)

   If T = ∅ Then
       S' = S
       ODE.solveSystem()
       For each ω = (i, α, σ, ν) ∈ Ω Loop
              If σ = +, α = 'constant', X_i' ∈ S', X_i ∈ S, X_i' < ν < X_i Then T = T ∪ {ω}
              ...
       End Loop
   Else T = ∅
   σ = Δ
   End If
```

Two cases for advance time function: the transient state for the generation of output (ta is null) and the state of numerical integration. The variable σ is equal to Δ, the integration step, in the majority of case. If an external event occurs then σ is decrease with the time elapsed since last transition (e).

```
Time getTimeAdvance()

   If T = ∅ Then Return σ
   Else Return 0
   End If
```

The output function is active when the set of reached thresholds isn't empty. The outputs of the model are building from the set of threshold and are also only discrete. Contrary to HFS (Heterogeneous Flow Systems – [Barros 2002]) or FSPN (Fluid Stochastic Petri Nets – [Kulkarni 1993] [Horton 1998] [Tuffin 2001]), our ODE wrapper didn't process continuous outputs and continuous inputs. The aim of this paper is to show how to build a DEVS wrapper in VLE framework. HFS and FSPN are more efficient methods for modelling and simulation of hybrid systems. Our ODE wrapper is more similar to DTSS/DESS (Discrete Time System Specification / Differential Equation System Specification [Zeigler 2000]).

```
EventList getOutputFunction(currentTime : Time)

   Y = ∅
   If T ≠ ∅ Then
       For each ω = (i, α, σ, ν) ∈ T Loop
              Y = Y ∪ {(p, X_i) / (p, ω) ∈ O}
       End Loop
   End If
   Return Y
```

When an external event occurs on a variable of the differential equations system, a discontinuity appears. The derivative and the second derivative are undefined. The numerical integration method is reset.

```
processExternalEvent(externalEvent : Event)

   Let x ∈ X / x = (p, (i, ν, α))
   If α = additive Then
        X_i = X_i + ν
   Else If α = reset Then
        X_i = ν
   End If
   ∀Ẋ_i, ∀Ẍ_i are undefined
   σ = σ − e
```

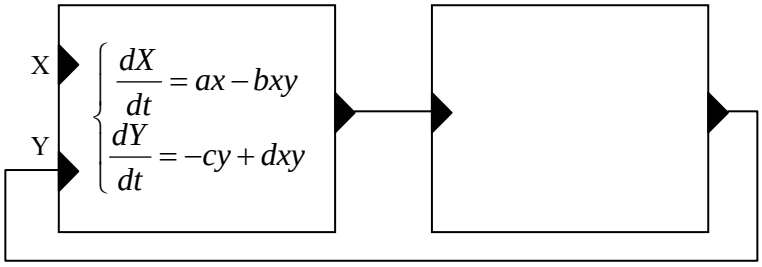
2.1.4 Example

In this section, we develop an example of model for using ODE wrapper. We construct a toy model simulating the regulation of prey and predator population dynamic by an external intervention. All unit and parameters values are arbitrary.

We just want an illustration of our approach. We first consider a the prey-predator model defined by Lotka-Volterra [Lotka 1925]:

$$\frac{dX}{dt} = ax - bxy$$
$$\frac{dY}{dt} = -cy + dxy$$

With X the concentration of preys and Y the concentration of predators. Parameters $a=0.2$, $b=0.2$, $c=0.5$ and $d=0.5$ are arbitrary. We use these values in order to obtain a stable equilibrium between preys and predatory. Initial condition are $X=1$ and $Y=0.1$.



The ODE wrapper can receives disturbances from connected model on its input ports. This model send external event to the wrapper to decrease X variable by a constant: 1.2. ODE wrapper sends external events when Y variable reaches a constant threshold with value 3. The connected model who receives this events, wait 3 events and send an event to decrease X variable.

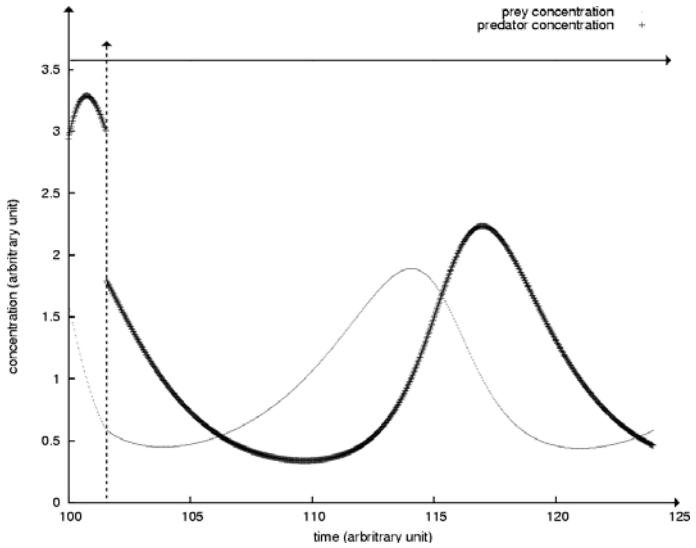


Figure 3: This figure shows a discontinuity in the predator concentration evolution. It corresponds to an event of predator uptake. At the same date, we can see another discontinuity in the evolution of prey concentration.

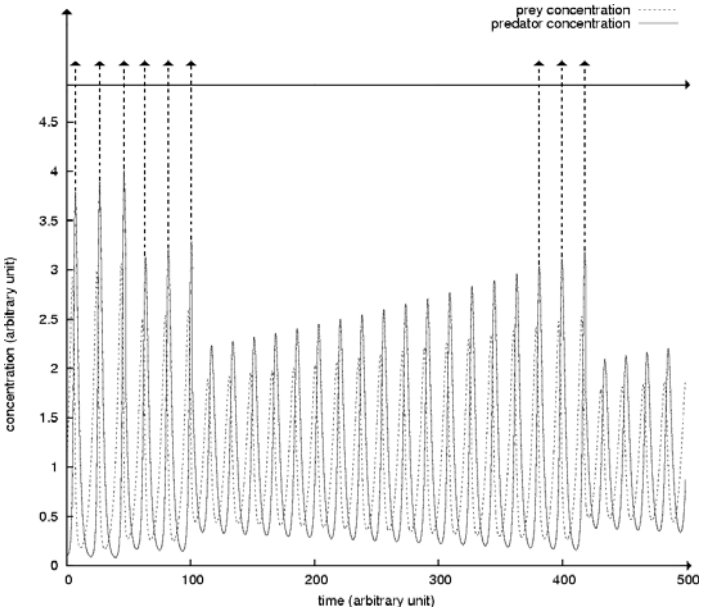


Figure 4: Oscillation of preys and predator concentration over time. Predator are taking off (-1.2 predators) if the concentration reach 3 times the arbitrary units of 3 predators. Taking predator introduce instabilities in the evolution of populations.

2.2. **CellQSS mapping in VLE : Spatial Differential Equations**

Another class of differential equations is partial differential equation. For this class, it's possible to use the previous tool, the ODE wrapper but, in this section, we propose another approach: the mapping technique in VLE framework. We will show that it is possible to generalize the use of DEVS formalism, Cell-DEVS and QSS with the partial differential equations with the derivative.

In the first part, we present the concept of quantification integrator and an extension of DEVS for cellular automaton. The first permits to propose a method to solve an ordinary differential equation. The mainly property of this method is to be express into DEVS framework. The second formalism is an extension of DEVS and proposes formalism for spatial process. In our case, the partial differential equation, we limit ourselves to the partial derivatives according to the time and the space.

2.2.1 Brief presentation of quantification integrators

Parallel to the traditional methods appeared new techniques of numerical resolution of differential equations based on the quantification of the output values rather than on the discretization of time. Thus, Kofman proposes in [Kofman 2001] [Kofman 2002] through QSS1 and QSS2 two methods of numerical resolution of first order differential equations. For the presentation of these two methods, let us consider the following system:

$$S\begin{cases} \dot{x}_i(t) = f_1(x_1(t),\ldots,x_n(t)) \\ \qquad\qquad \vdots \\ \dot{x}_n(t) = f_n(x_1(t),\ldots,x_n(t)) \end{cases}$$

The global idea is to approximate the functions $x_i(t)$ by constant piecewise functions (for QSS1) or of the affine piecewise functions (for QSS2) then to encapsulate each x_i in an atomic DEVS model. At an instant denoted t, each model has like state variable a constant for QSS1 or an affine function for QSS2. Lastly, it's appropriate for the models to compute an internal transition when their states move away too much the theoretical curve to integrate.

2.2.2 Another extension of DEVS: Cell-DEVS

The extension named Cell-DEVS was born from the following observation: many models use discrete spaces and use formalisms such as the cellular automata. Wainer and Giambiasi in [Wainer 01] developed Cell-DEVS. This extension must be able to describe and simulate models as a multidimensional cellular automata and discrete events. The dynamics of the cells is time-lag i.e. that the state of cell will be modified according to the state of its neighbour cells, but it will be known by neighbour cells after a certain time. The basic idea is to provide a simple mechanism of definition of the synchronization of the cells. As for any proposal for an extension, the authors offer at the same time the extension of the formalism which is

summarized with the addition of variables and their semantics, and the abstract simulator. In this paper, we propose a simplified structure according to the coupling with QSS formalisms.

Our Cell-DEVS atomic model is defined as a structure:

$$M = \langle X, Y, I, S, \delta_{\text{int}}, \delta_{ext}, \lambda, ta, \sigma, d, N \rangle$$

Where

X is the set of external input events;
Y is the set of external output events;
I is the interface of cell (list of neighbours);
$N = \{(\eta_i, s_i) / \eta_i \in I\}$ is the state of neighbourhood (it's a simplification compared to the original version);
S is the state variables used in each cell without the state of neighbourhood;
d $\in \boldsymbol{R}_0^+$ is the diffusion delay of the state;
$\delta_{\text{int}} : S \to S$ is the internal transition function;
$\delta_{ext} : Q \times X \to S$ is the external transition function, where Q is the state values defined as:

$$Q = \{(s, e) / s \in S \times d, e \in [O, ta(s)]\}$$

λ: S $\to Y$ is the output function;
ta: S x d $\to \boldsymbol{R}_0^+ \cup \infty$, is the state's duration function.

2.2.3 Integration of QSS into Cell-DEVS

By applying directly the principle of discretization of space and the transformation into linear system of order 1, it is already possible to integrate this kind of equation:

$\frac{\partial^n T}{\partial^n t} = \varphi\left(\frac{\partial^{n-1} T}{\partial t^{n-1}}, \ldots, \frac{\partial T}{\partial t}, T, \frac{\partial T}{\partial x_1}, \frac{\partial^2 T}{\partial x_1 \partial x_2}, \ldots, \frac{\partial^p T}{\partial x_1 \ldots \partial x_p}, \ldots\right)$ if an approximation of $\partial^p T / \partial x_1 \ldots \partial x_p$ is possible.

Let us show this assertion: let $f_i(t)$ the discretization of $\partial^p T / \partial x_1 \ldots \partial x_p$ then $f_i(t)$ will depend on $T(t, x), T(t, x-h), \ldots$ and will not depend on differential. Let us $T_1(t) = \frac{\partial T}{\partial t}, \ldots, T_{n-1}(t) = \frac{\partial^{n-1} T}{\partial t^{n-1}}$. Thus the previous equation becomes:

$$\frac{\partial^n T}{\partial T^n} = \varphi(T_{n-1}(t), \ldots, T_1(t), T(t), f_1(t), f_2(t), \ldots, f_p(t))$$

and it's possible to transform to a system:

$$\begin{cases} \dfrac{\partial T}{\partial t} = T_1 \\ \dfrac{\partial T_1}{\partial t} = T_2 \\ \dfrac{\partial T_2}{\partial t} = T_3 \\ \vdots \\ \dfrac{\partial T_{n-2}}{\partial t} = T_{n-1} \\ \dfrac{\partial T_{n-1}}{\partial t} = \varphi\left(T_{n-1}(t), \ldots, T_1(t), T(t), f_1(t), f_2(t), \ldots, f_p(t)\right) \end{cases}$$

Only ordinary differential equations are defined. Each equation depends on several ODE. In the case of spatial differential equation, if the space is discretized then these equations depend on neighbourhood.

2.2.4 Algorithms

In this part, we describe algorithms written with VLE framework. Before the description of algorithms, we define some variables of DEVS abstract simulator:

- t : the current time
- t_L : the time of last event
- t_N : the time of next event ($t_N = t_L + ta(S)$)
- e : the time elapsed since last transition
- σ : the time remaining in the current state ($\sigma = t - t_N = ta(S) - e$)

Each variable is indexed by the number of differential equation and by the index of cell. In the following, we omit to indicate the index of cell in order to simplify algorithms.

The internal transition function is identical to that of QSS. Indeed, if no event occurs before σ_i then the threshold q_i is reached and the gradient must be revalued.

```
processInternalEvent(internalEvent : Event)
```

Let i the current differential equation (i. e. $\sigma_i = 0$)

If $\dot{x}_i > 0$ Then $q_i = q_i + 1$ Else $q_i = q_i - 1$

$x_i = q_i \Delta q$

$\dot{x}_i = f_i(t)$

$\forall j \neq i, \sigma_j = \sigma_j - \sigma_i$

$\sigma_{\min} = \min_{i \in [1;n]} \sigma_i$

If $\left|\dot{x}_i\right| < \varepsilon$ Then $\sigma_i = +\infty$

Else If $\dot{x}_i > 0$ Then $\sigma_i = \dfrac{(q_i + 1)\Delta q - x_i}{\dot{x}_i}$

Else $\sigma_i = \dfrac{q_i \Delta q - x_i}{\dot{x}_i}$

The advanced time function is very simple. The duration of state without external perturbation is equal to the minimal of time to reach a threshold q_i.

```
Time getTimeAdvance()
   Return σ_min
```

When the gradient of one or more equations is modified by internal event (i.e. the state is modified), the values of variables of the system are posted to neighbour cells from the output function. Each cell has only one output port for the propagation of state of cell.

```
EventList getOutputFunction(currentTime : Time)
   Y = ∅
   If state is modified Then
       V = ∅
       For all differential equation f_i Loop V = V ∪ {X_i} End Loop
       Y = Y ∪ {(out,V)}
   Return Y
```

Two kinds of external event can occurs: one of neighbour cell have changed its state or another model coupled to CellQSS model makes a perturbation. In these two cases, the gradient of each function can be modified. So, one must to compute the state of system when external event occurs and the gradient function is compute too in order to estimate the new σ_i.

```
processExternalEvent(externalEvent : Event)
   If externalEvent is a change of state of neighbour Then
       Let p ∈ I , the name of neighbour
       Let v, the new state of current neighbour
       s_p = v where (p, s_p) ∈ N
   Else processPerturbation
   For all differential equation f_i Loop
       Let e_i = t − t_{L_i}
       If e > 0 Then x_i = x_i + e ẋ_i
       ẋ_i = f_i(t)
       If |ẋ_i| < ε Then σ_i = +∞
       Else If ẋ_i > 0 Then σ_i = ((q_i + 1)Δq − x_i) / ẋ_i
                Else σ_i = (q_i Δq − x_i) / ẋ_i
       T_{L_i} = t
   End Loop
```

In case of external perturbation, the gradient of each function is reset and the quantum is updated.

```
processPerturbation(externalEvent : Event)
   For all differential equation f_i Loop
       Ẋ_i = 0
       q_i = E(Ẋ_i / Δq)
       σ_i = 0
   End Loop
   σ = 0
```

2.2.5 Example

Let us consider the example of the equation of diffusion of heat on a bar [Fletcher 2000] for illustration of QSS and Cell-DEVS:

$$\frac{\partial T(t,x)}{\partial t} = K \frac{\partial^2 T(t,x)}{\partial x^2}$$

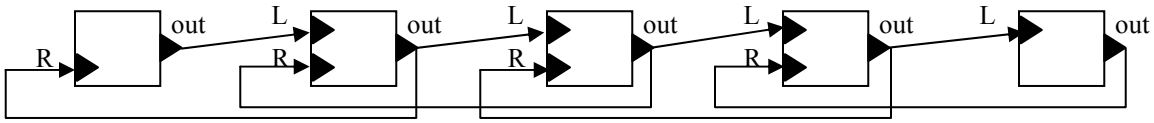
where K is the coefficient of diffusion.

We make an approximation of T(t,x) by its values in various points of the bar after having chosen a space step ∂x. Denoted $T_i(t)$ these values, the previous equation becomes:

$$\frac{\partial T_i(t)}{\partial t} = K \frac{T_{i-1}(t) - 2T_i(t) + T_{i+1}(t)}{(\Delta x)^2}$$

for all i from 1 to N, using, by example, an approximation of the second derivative in three points.

One of the traditional approaches (the explicit method, for instance) would now consist to approximating $\partial T_i(t)/\partial t$ by $(T_i(t+\Delta t) - T_i(t))/\Delta t$. But the equations form at this level a system of first order differential equations in $T_i(t)$. It is thus possible to solve them using a QSS method.



3. Discussion and Perspectives

The first question in this article is how to bring making of VLE wrapper from existing models. Several stages are necessary. The first part, we must find in study model, elements like time, states, events etc. Some models like Petri net can make problems because time does not exist in basic Petri net structure however time is a principal element of DEVS formalism. In this case, time must be recreated in the formalism in order to satisfy DEVS functions [Quesnel 2005].

Second stage is to define state of model. State S is an important element. The state can be accompanied of DEVS formalism elements. After state, we must define an internal transition function δ_{int} who allows changing state. An external transition function δ_{ext} can be defined to manage external events. We must attach each element to its DEVS function. Lastly, we must find in the model how an input event could come to disturb the current state.

This paper is a first step toward the integration of complex model such as multi-paradigm systems. If we consider pre-existing models which are not formalised, Wrapping techniques can be very useful to permits there reusability

In section 2, we defined two DEVS models used in VLE framework to model continuous systems. First, we use wrapping technique to develop a wrapper to an ODE formalism based on Runge Kutta integration method. Second, we develop a model based on mapping method to perform coupling between a cell automata, CellDEVS, and an ODE system, QSS, to build spatialized equation systems.

With these models, we offer to model continuous and discrete models. However, is VLE framework providing tools and models to build hybrid systems?

Hybrid systems are defined as systems using both continuous and discrete behaviour. Differential equations are used for representing continuous system, discrete event for discontinuities. There is some works on this type of systems like the Fluid Stochastic Petri Nets (FSPNs) defined by V.G. Kulkarni and K.S Trivedi [Horton Kulkarni 1998]. In this type of Petri net, some features are different from traditional Petri net.

- The tokens transport continuous variables.

- Two type of transition exist:
    - Stochastic: Transitions are fired after a random timed.
    - Classical but with condition: Transitions are fired fallow some conditions. The Conditions are function of continuous variables on tokens.

In next section, we study the implementation possibilities, in VLE framework, of hybrid system. We explain the Heterogeneous Flow System Specification (HFSS) defined by F.J. Barros [Barros 2002] and its potential integration in VLE.

The HFSS is a paradigm for simulating hybrid systems. HFSS is based on DEVS formalism and provides discrete and continuous input and output ports, a set of states, a transition function which depends on discrete and continuous inputs.

HFFS is defined by:

$$HFSS = \left(X, Y, S, \rho, \tau, q_0, \delta, \Lambda_c, \lambda\right)$$

Where:

$X = X_c \times X_d$ is the of input flow values, X_c the set of continuous input flow values, X_d the set of discrete input flow values.

$Y = Y_c \times Y_d$ is the of output flow values, Y_c the set of continuous output flow values, Y_d the set of discrete output flow values.

S is the set of partial states.

$\rho : S \to \Re_0^+$ is the time to input function.

$\tau : S \to \Re_0^+$ is the time to output function.

$q_0 = \left(s_0, e_0\right) \in Q$ is the initial state, with e is the time to transition function.

$\delta : Q \times \left(X_c \times X_d^{\emptyset}\right) \to S$ is the transition function

HFSS entries are continuous or discrete. Continuous input ports, using sampling, have an influence on continuous system. Discrete inputs can change the state of HFSS model. HFSS uses threshold to build discrete outputs and continuous output ports. The fundamental idea of HFSS paradigm is the discrete entries cause change of state on system and thus cause change between differential functions.

HFSS seems to be a technique to develop efficient hybrid systems in speed building and simulation. HFSS uses optimisation in simulator to increase simulation. However, the main critical remark for us is the incompatibility with DEVS formalism used in VLE framework. In DEVS we need two transition functions: an internal and an external to make models compliant with DEVS abstract simulator.

An alternative to HFSS i.e., to allow creation of hybrid systems and to create an equivalent to HFSS, is to use the GDEVS (Generalized Discrete Event Specification formalism [Giambiasi *et al.* 2000]). It's a compliant DEVS models that uses polynomials of arbitrary degree to represent the piecewise input-output trajectories of a discrete event model instead of DEVS who approximates the input, output, and state trajectories through piecewise constant segments. The input ports can change a coefficient to change state of continuous variables. A coefficient event is considered as an instantaneous event. The GDEVS model provides to VLE framework new methods to develop hybrid systems and a good replacement to HFSS.

REFERENCES

[Barros 2003] Barros F., "Dynamic Structure Multi-Paradigm Modeling and Simulation," A*CM Transactions on Modeling and Computer Simulation*, Vol. 13, No. 3, pp. 259-275, 2003.

[Barros 2002] Barros F., "Modeling and Simulation of Dynamic Structure Heterogeneous Flow Systems." *SIMULATION: Transactions of The Society for Modeling and Simulation International*, Vol. 78, No. 1, 18-27, 2002.

[Branicky 1997] Branicky M. S. and Mattson S. E., Simulation of hybrid systems. In Hybrid Systems IV, P. J. Antsaklis, W. Korn, A. Nerode, and S. Sastry, Eds. Lecture Notes in Computer Science, vol. 1273. Springer-Verlag, New York, 1997

[Ciardo 2003] Ciardo G., Dinol D.M. and Trivedi K.S., Discrete-Event Simulation of Fluid Stochastic Petri Nets, *IEEE Transactions on software engineering*, 25 (2): pages 207-217, 1999.

[D'Abreu 2003] D'Abreu M. C. and Wainer G. A., Models for Continuous and Hybrid System Simulation, Proceedings of the 2003 Winter Simulation Conference, pp. 642-649, 2003.

[Duboz 2003] Duboz R. and Ramat E., Towards the simulation of scale transfert in ecological modelling using computational emergence: Parametrization of classical population model with reactive agent model. *Systems Analysis Modelling Simulation*, volume 43, 793–814, 2003.

[Duboz 2002] Duboz R., Xml for the representation of semantic in model coupling. In F. J. Barros and N. Giambiasi, editors, AIS'2002. AI, *Simulation and Planning in High Autonomy Systems*, 267–270, Lisboa, Portugal, april 7-10, 2002.

[Fishwick 1995] Fishwick, *P.A. Simulation Model Design and Execution*. Number ISBN 0-13-098609-7. Prentice Hall, 1995.

[Fishwick 1992] Fishwick, P.A. and Zeigler B.P. A multi-model methodology for qualitative model engineering. *ACM transaction on Modeling and Simulation*, **2-**1:52–81, 1992.

[Fletcher 2000] C.A.J. Fletcher, Computational Techniques for Fluid Dynamics, Volume 1 and 2, Springer, 2nd Edition, 2000.

[Giambiasi *et al.* 2000] Giambiasi N., Escude B., Ghosh S. GDEVS: A generalized discrete event specification for accurate modeling of dynamic systems, *Transactions of the Society for Computer Simulation International* 17(3): 120-134, 2000.

[Horton Kulkarni 1998] Horton G., Kulkarni V.G. and Trivedi K.S., Fluid Stochastic Petri Nets: Theory, Application and Solution. European Journal of Operational Research, 105: 184-201, 1998.

[Kim 1996] Kim Y.J. and Kim T.G., A heterogeneous distributed simulation framework based on devs formalism. In Sixth Annual Conference On Artificial Intelligence, *Simulation and Planning in High Autonomy Systems*, 116–121, La Jolla, California, USA, 1996.

[Kofman 2001] Kofman, E. and Junco, S. (2001). Quantized state systems. A devs approach for continuous systems simulation. *In Transactions of SCS*, volume 18, 123-132, 2001.

[Kofman 2002] E. Kofman, "A Second Order Approximation for DEVS Simulation of Continuous Systems". *Transactions of SCS*, **78**-2, pp. 76–89, 2002.

[Kulkarni 1993] Kulkarni, V.G. and Trivedi, K.S., FSPNs: Fluid Stochastic Petri Nets. In 14th International Conference on Applications and Theory of Petri Nets, pages 24-31, 1993.

[Quesnel 2005] G. Quesnel, R. Duboz and E. Ramat, Wrapping into DEVS Simulator: A Study Case, International Mediterranean Modeling Multiconference, pp. 374-382, 2005.

[Lotka 1925] Lotka, A.J. *Elements of physical biology*. Baltimore: Williams & Wilkins Co, 1925.

[Tuffin 2001] Tuffin B., Chen D.S. and Trivedi K.S., Comparison of Hybrid Systems and Fluid Stochastic Petri Nets. Discrete Event Dynamic Systems, 11 (1-2), 2001.

[Vangheluwe *et al.* 2002] Vangheluwe, H., Lara, J., and Mosterman, P.J. An introduction to multi-paradigm modelling and simulation. In F.J. Barros and N. Giambiasi, editors, AIS'2002. Simulation and Planning in High Autonomy Systems, 9–20. *Society for Modelling and Simulation International*, Lisbon, Portugal, April 2002.

[Zeigler 1976] Zeigler, B.P. *Theory Of Modeling and Simulation*. Wiley Interscience, 1976.

[Zeigler et al. 2000] Zeigler, B.P., Kim, D. and Praehofer, H. Theory of modeling and simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems. Academic Press, 2000.