Virtual Laboratory Environment

Gauthier Quesnel, Eric Ramat, Jean-Christophe Soulié, David Duvivier, Raphaël Duboz

Virtual Laboratory Environment : un environnement de multimodélisation et de simulation de systèmes complexes

Gauthier Quesnel [1] — Eric Ramat [2] — Jean-Christophe Soulié [3] — David Duvivier [2] — Raphaël Duboz [4]

[1] *INRA, UR875 Biométrie et Intelligence Artificielle, F-31326 Castanet-Tolosan, France*

gauthier.quesnel@toulouse.inra.fr

[2] *Université Lille-Nord de France, EA4491 LISIC-ULCO, BP 719, 62228 Calais Cedex, France*

{ramat,duvivier}@lisic.univ-littoral.fr

[3] *CIRAD, UPR AIVA, Avenue Agropolis, 34398 Montpellier Cedex 5, France*

jean-christophe.soulie@cirad.fr

[4] *CIRAD, UPR AGIRs 22, Campus International de Baillarguet, 34398 Montpellier Cedex 5, France*

raphael.duboz@cirad.fr

RÉSUMÉ. *L'étude des systèmes complexes dynamiques est une problématique de plus en plus importante dans le cadre de l'activité scientifique. L'étude de tels systèmes nécessite fréquemment l'emploi d'un cadre de modélisation et de simulation avec des outils mathématiques et informatiques suffisamment évolués pour prendre en compte la diversité de ces systèmes. En effet, l'une des particularités des systèmes complexes est que leur étude est interdisciplinaire et par conséquent, les paradigmes ou formalismes employés par les différentes communautés sont hétérogènes. Dans cet article, nous présentons une démarche de modélisation basée sur la théorie systémique et sur le formalisme à événements discrets DEVS qui supporte le couplage de modèles hétérogènes. Nous développons les bases théoriques de notre plate-forme VLE (*Virtual Laboratory Environment*) issue des travaux sur DEVS ainsi que ses caractéristiques techniques. Finalement, nous illustrons notre démarche sur deux exemples développés à l'aide de VLE.*

Studia Informatica Universalis., pages 205 à 234

ABSTRACT. *The study of complex dynamic systems is a most important issue in the context of scientific activity. The study of these systems uses generally a modeling and simulation framework with mathematical and computational tools evolved enough to take into account the diversity of these systems. Indeed, one characteristic of complex systems is that their study is interdisciplinary and therefore, paradigms and formalisms used by different communities are heterogeneous. In this paper, we present a modeling approach based on systems theory and the DEVS formalism, a formalism which allows the coupling of heterogeneous models. We will develop the theoretical basis of our platform VLE (*Virtual Laboratory Environment*) based on DEVS and its specifications. Finally, we illustrate our approach with two examples developed with VLE.*

MOTS-CLÉS : *Multimodélisation, couplage de modèles, DEVS*
KEYWORDS: *Multimodeling, model coupling, DEVS*

1. Introduction

La modélisation est au cœur de l'activité scientifique, que ce soit pour comprendre le monde qui nous entoure ou pour concevoir des objets nouveaux. Dans le cadre de l'étude des systèmes complexes dynamiques, la modélisation et la simulation posent des problèmes qui transcendent les disciplines. En ce sens, modélisation et simulation forment une science à part entière. La théorie des systèmes [Moi77, Ber93] et la théorie de la modélisation et de la simulation [Zei76] offrent des réponses à ces besoins et mettent à la disposition des chercheurs modélisateurs des méthodes, des formalismes pour spécifier les systèmes, des algorithmes et des outils pour les simuler. Actuellement, nous sommes de plus en plus confrontés aux problèmes de construction de modèles multiparadigmes, c'est-à-dire à des modèles qui intègrent différents points de vue, différentes disciplines. Ce type de modèle permet des simulations multiéchelles par exemple [DRP03]. Ces modèles sont également qualifiés de multimodèles car ils sont pluriformalisés : ils intègrent différents formalismes dans une même représentation. Cette intégration, tout en augmentant le potentiel de description des modèles [Dub04], pose des questions difficiles relatives au couplage de modèles discrets et continus en temps et en espace par exemple, ou à la sémantique des données échangées entre modèles. Les réponses ne sont pas simples et engendrent un nouveau degré de complexité. Il existe bien sûr des réponses sur certains aspects. Par exemple, en modélisation multiéchelle l'utilisation des techniques mathématiques d'agrégation de variables permet une modélisation de processus évoluant à des échelles de temps très différentes [AdlPP+08]. Des techniques s'inspirant de l'agrégation de variables, mais pluriformalisées, existent également [DRP03]. Néanmoins, il s'agit de réponses qui ne permettent pas une réutilisation simple de la méthode de couplage utilisée sur des cas très différents de ceux sur lesquels ces méthodes ont été appliquées.

L'objectif principal de cet article est de présenter à la fois une approche de multimodélisation et un outil informatique de mise en œuvre pour répondre aux nouveaux besoins identifiés dans le cadre de la modélisation multiparadigme et de la simulation des systèmes complexes.

Nous nous situons au niveau des outils informatiques orientés vers

l'aide à la multimodélisation et la simulation des systèmes dynamiques, spatialisés ou non. Nous entendons par outil informatique, toute méthode, tout formalisme ou tout cadre conceptuel aidant à la conception et à la mise en œuvre du processus de modélisation multiple et de simulation. Pour réduire encore notre champ d'action, nos questions récurrentes sont :

– Comment formaliser le plus universellement possible des modèles de systèmes spatio-temporels ? Existe-t-il un formalisme universel ou fédérateur ?

– Comment coupler plusieurs modèles issus de paradigmes différents ? Comment en gérer la cohérence temporelle, spatiale et sémantique ? Tous les éléments ne seront pas abordés dans cet article.

– Comment disposer d'une formalisation "exécutable" ?

– Comment faire coopérer ou exécuter au sein d'une même plateforme des modèles de simulation conçus séparément ?

– Comment s'affranchir des problèmes techniques de simulation ?

– Comment réutiliser des modèles déjà existants, validés et reconnus robustes ?

L'histoire de VLE (*Virtual Laboratory Environment*) débute en 2003 [RP03] avec une première réflexion sur la modélisation des systèmes complexes orientée agents et objets. À cette époque, les systèmes modélisés sont principalement des écosystèmes marins. Mais rapidement, les besoins de formalisation, de traçabilité des modèles, de couplage, se font ressentir. Nous nous tournons alors vers la communauté DEVS (*Discrete EVent systems Specification* [Zei76, ZKP00]). Dès 2004 un premier noyau de simulation DEVS est développé autour de l'idée de DEVS-Bus et de l'utilisation d'XML pour la représentation des structures de modèles couplés ainsi que des expériences de simulation [Dub04]. Les premiers articles sur la réalisation de multimodèles sur ce simulateur sont publiés [QDR04, QDVR05, VR05] en parallèle au développement des travaux sur les analogies possibles entre les Systèmes Multi-Agents (SMA) et DEVS [DRQ04, Dub04]. De ces expériences est né le besoin d'un environnement de modélisation, de simulation et d'analyses complet. Le développement de cette plate-forme est alors initié en 2006 [Que06] et le fondement de ce projet est de fournir

un environnement mathématiquement et informatiquement fiable pour l'étude des systèmes complexes dynamiques. Une première version du projet, de son architecture, de ses ambitions, est alors présentée à la communauté de la simulation [QDRT07]. VLE permet, à cette date, le développement de modèles suivants quelques formalismes principalement continus et événementiels. En 2007, le projet VLE est choisi comme solution technique pour la modélisation et la simulation par la plate-forme de service RECORD [CGMC+07] initiée par l'INRA (Institut scientifique de recherche agronomique). En 2008, l'environnement évolue en voyant son utilisabilité progresser ainsi que sa stabilité. Les travaux sont également menés pour introduire de nouveaux formalismes principalement pour répondre aux problèmes rencontrés par les modélisateurs agronomes, mais pas uniquement puisque de nouveaux utilisateurs universitaires et issus d'autres organismes se sont liés au développement de VLE. En 2011, VLE est jugé suffisamment mature pour que l'INRA choisisse sa diffusion dans l'ensemble des unités ayant recours à la modélisation et la simulation d'agro-écosystèmes. VLE est alors utilisé comme noyau de simulation pour la plate-forme RECORD. La plate-forme RECORD ajoutant à ce noyau des bibliothèques de modèles réutilisables et des services à l'utilisateur comme des interfaces graphiques, des tutoriels et des formations.

VLE est aujourd'hui une plate-forme de modélisation et de simulation générique avec un champ d'application très vaste. Il permet de modéliser des systèmes dynamiques micro, méso et macroscopiques dans des domaines aussi variés que les modèles d'agronomie, les modèles de plante, les modèles de maintenance industrielle, les modèles d'écologie marine etc. VLE ne peut alors pas être comparé directement à une plate-forme ou un outil tels que *Sol Virtuel* [1] ou *OpenFluid*[2] qui proposent eux-mêmes un noyau de simulation mais avec un cadre et un domaine clairement défini. Pour une meilleure comparaison, il est plus réaliste de comparer ces plate-formes ou outils avec la plate-forme RECORD.

1. http://www.inra.fr/sol_virtuel/le_projet_sol_virtuel : une représentation du sol et de ses interactions physiques, chimiques et biologiques pour améliorer la prévision de l'impact des activités anthropiques et des changements climatiques.
2. http://www.umr-lisah.fr/openfluid/ : décrite comme une plate-forme de modélisation et de simulation pour la modélisation et la simulation couplée de flux spatialisés s'appuyant sur une représentation numérique des paysages.

Cet article se divise en quatre principales parties. Afin de fixer le socle théorique de notre approche, nous présentons le formalisme DEVS et son extension *Parallel DEVS*. Ensuite, en s'appuyant sur la systémique, nous montrons comment une démarche de modélisation est possible et surtout, nous développons le lien entre les notions de système, sous-système, interaction... et une modélisation DEVS. La troisième partie est consacrée à notre plate-forme de multimodélisation et de simulation : *Virtual Laboratory Environment*. Nous terminons par deux exemples typiques et en relation avec l'agronomie au sens large.

2. DEVS comme support formel de la multimodélisation

Lors de la construction de notre solution de multimodélisation et de simulation, nous avons tout d'abord cherché une solution à l'universalité de la couche formalisme. Il existe une multitude de solutions lorsque nous prenons un formalisation particulier et que nous cherchons une solution de couplage *ad hoc*. La communauté des formalismes hybrides (citons par exemple les réseaux de Petri hybrides ou continus [RD05]) œuvre dans cette direction. Nous verrons par la suite que tous ces formalismes (comme DEVS) répondent à notre problématique mais ne sont pas toujours accessibles aux modélisateurs. De plus, le modélisateur utilise déjà des paradigmes qu'il a sélectionné en fonction de ses compétences, en fonction de l'adéquation avec les propriétés de son système. Il est donc important de tendre vers un formalisme unificateur qui donne la possibilité d'encapsuler les autres formalismes. DEVS est un candidat et nous allons montrer comment DEVS spécifie les systèmes.

DEVS [ZKP00] est un formalisme de haut niveau basé sur les événements discrets pour la modélisation de systèmes complexes discrets et continus. Le modèle est vu comme un réseau d'interconnexions entre des modèles atomiques et couplés. Les modèles sont en interaction via l'échange d'événements estampillés. PDEVS [CZ94] étend le formalisme DEVS classique [ZKP00] essentiellement par l'introduction de la notion de "bags" d'entrée pour la fonction de transition externe. Les "bags" regroupent les entrées en terme d'événements qui ont été émis à la même date. Leur traitement produiront un effet dans les "bags" futurs. Ce formalisme offre une solution pour gérer les événements simultanés

tout en garantissant le principe de causalité, ce que DEVS classique ne permet pas simplement. Pour une présentation plus détaillée, vous pouvez lire le chapitre 3 de [ZKP00].

PDEVS définit un modèle atomique comme un ensemble de ports d'entrée et de sortie et un ensemble de fonctions de transitions d'états. La notion de port permet de distinguer sémantiquement les événements mis en entrée d'un modèle ou produits par un modèle.

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

où :

X est l'ensemble des valeurs d'entrée
Y est l'ensemble des valeurs de sortie
S est l'ensemble des états
$ta : S \to \mathbb{R}_0^+$ est la fonction d'avancement du temps
$Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$
    Q est l'ensemble des états totaux
    e est la durée écoulée depuis la dernière transition
$\delta_{int} : S \to S$ est la fonction de transition interne
$\delta_{ext} : Q \times X^b \to S$ est la fonction de transition externe
    X^b est un "bag" d'éléments pris dans X
$\delta_{con} : S \times X^b \to S$ est la fonction de conflit
    sachant que $\delta_{con}(s, \emptyset) = \delta_{int}(s)$
$\lambda : S \to Y$ est la fonction de sortie

Si aucun événement externe ne survient, alors le système reste dans l'état s pour une durée définie par $ta(s)$. Quand $e = ta(s)$ alors le système change d'état conformément à δ_{int}. Si un événement transportant la valeur x est mis en entrée du système alors que ce dernier est dans l'état (s, e), le système change d'état via la fonction $\delta_{ext}(s, e, x)$. Si un événement externe se produit quand $e = ta(s)$, la fonction de conflit $\delta_{con}(s, x)$ est alors utilisée afin de définir le nouvel état. Par défaut, la fonction de conflit δ_{con} est définie comme suit :

$$\delta_{con}(s, x) = \delta_{ext}(\delta_{int}(s), 0, x)$$

Le modélisateur peut préférer l'ordre inverse :

$$\delta_{con}(s, x) = \delta_{int}(\delta_{ext}(s, ta(s), x))$$

Naturellement, cette fonction peut être complètement définie par le modélisateur.

Chaque modèle atomique peut être couplé avec un ou plusieurs autres modèles atomiques afin de former un modèle couplé. Cette opération de couplage peut être répétée dans l'objectif de former une hiérarchie de modèles couplés. L'ensemble des modèles atomiques et couplés et leurs connexions se nomme la *structure du modèle*. Un modèle couplé est défini par :

$$N = \langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\} \rangle$$

où X et Y sont les ports d'entrée et de sortie, D l'ensemble des modèles et :

$\forall d \in D, M_d$ est un modèle PDEVS
$\forall d \in D \cup \{N\}, I_d$ est l'ensemble des "influenceurs" de d :
$\quad I_d \subseteq D \cup \{N\}, d \notin I_d$
$\forall d \in D \cup \{N\},$
$\quad \forall i \in I_d, Z_{i,d}$ est une fonction définissant le type d'interaction de i vers d :
$\quad\quad Z_{i,d} : X \to X_d,$ si $i = N$
$\quad\quad Z_{i,d} : Y_i \to Y,$ si $d = N$
$\quad\quad Z_{i,d} : Y_i \to X_d,$ si $i \neq N$ and $d \neq N$

L'ensemble des "influenceurs" de d est l'ensemble des modèles qui interagissent avec d et $Z_{i,d}$ définit les types de relations entre le modèle couplé ou l'un des modèles couplés i et d. Un modèle i peut générer un événement vers d et donc le perturber. Inversement, le modèle d peut générer un événement à destination d'un autre modèle pour le perturber. Si nous nous plaçons dans une approche hiérarchique, les événements peuvent provenir ou peuvent être à destination de modèles externes au modèle couplé auquel d appartient. Il existe, néanmoins, une contrainte : le modèle d ne peut pas s'auto-influencer.

3. La démarche de modélisation

La démarche de modélisation proposée dans notre approche se base sur la systémique [Moi77]. Dans notre contexte, le formalisme DEVS

est intimement lié à la systémique. L'idée de base est de représenter, de concevoir ou d'analyser les choses du monde réel sous forme de systèmes dans un point de vue global. Le système est décomposé en sous-systèmes et en composants élémentaires, l'ensemble étant en interaction. Cette décomposition forme alors une organisation hiérarchique où les relations entre les éléments sont représentées de manière logique. On parle alors de niveaux d'organisation. Selon les points de vue, les relations peuvent représenter des aspects structurels et/ou fonctionnels.

Les aspects structurels regroupent les éléments constitutifs du système en terme de nature et de nombre, les frontières du système (ce qui est dedans et ce qui est en dehors du système) et les relations entre les éléments. Les relations sont les lieux de transport d'information et d'influence. Chaque élément doit définir ses relations potentielles avec les autres éléments.

Le point de vue fonctionnel définit les fonctions assurées par les éléments et les flux entre les éléments via le réseau de relations. On parle aussi de boucle de rétroaction où l'action de certains éléments sur d'autres impliquent des ajustements de la part de ces premiers. En résumé, l'aspect fonctionnel définit les relations fonctionnelles entre les entrées et les sorties d'un élément du système.

Notre approche de modélisation s'inscrit totalement dans cette vision du monde mais en plus, nous nous intéressons aux systèmes dynamiques où la notion du temps est fondamentale. L'état du système évolue dans le temps sous l'influence de ses composants. Cette évolution respecte deux grands principes :

– la causalité, l'état futur du système est fonction de l'état passé et des événements passés et présents ;

– le déterminisme, à partir d'un état initial, il n'existe qu'un seul état futur.

Cette dernière propriété peut être relaxée dans le cas de systèmes stochastiques. Le temps peut être continu, l'évolution du système est alors continue, ou peut être discrète, l'état du système n'est défini qu'à certaines dates généralement définies par un pas de temps. Au cœur des systèmes qui nous intéressent, des aspects spatiaux peuvent être présents. Les éléments du système évoluent alors dans un espace à une ou

plusieurs dimensions et sont donc localisés par des coordonnées spatiales. Comme pour le temps, deux approches sont possibles : l'espace est vu comme un continuum et chaque élément possède des coordonnées continues ou l'espace est discrétisé (l'exemple classique étant la représentation dans des automates cellulaires). L'espace peut être aussi représenté par un graphe de voisinage quelconque. Les éléments sont alors localisés sur des lieux au sens général et les arcs du graphe représentent les relations spatiales entre les éléments (par exemple, un réseau de villes ou d'exploitations agricoles, les relations étant les liens d'échanges potentiels).

Les bases étant posées, voici les étapes de notre démarche (Une représentation graphique de cette démarche est fournie sur la figure 1). Tout d'abord, il est important de mettre cette démarche en relation avec le formalisme DEVS. De plus, nous allons décrire notre démarche dans une approche descendante, par décomposition du système global en sous-systèmes. L'approche inverse est aussi possible. Nous aborderons cette question en conclusion.

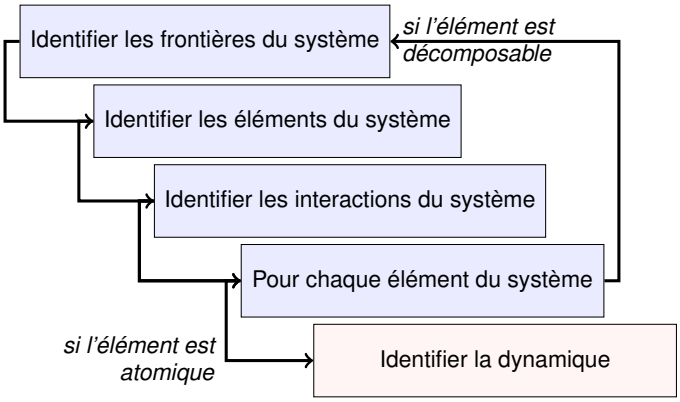

Figure 1 – Démarche de modélisation

La première étape consiste à identifier les frontières du système global. Deux options sont possibles en fonction de la propriété d'ouverture du système. Si le système est fermé, il faut "juste" identifier les éléments constituant le système selon le niveau d'organisation le plus élevé. Si le système est ouvert, il faut en plus identifier les éléments externes

au système qui sont potentiellement en interaction avec le système. Si nous transposons cette première étape en DEVS, le modélisateur doit donc identifier les modèles atomiques ou plus vraisemblablement couplés constituant le modèle couplé global. Si le système est ouvert, il est nécessaire de définir les ports d'entrée/sortie du modèle global. Par exemple, dans un modèle de système naturel, cela peut être les éléments météorologiques qui sont plutôt des facteurs forçants du modèle et probablement totalement indépendant du système.

La deuxième étape a maintenant pour objet la définition des interactions entre les éléments constituant le système. Par définition, nous devons identifier le réseau de communication entre les éléments. Étant dans un contexte DEVS, les communications sont des envois d'événements entre modèles. À une date donnée, un modèle émet en sortie un événement et cet événement est transmis, en entrée, à l'ensemble des modèles connectés. Ces interactions peuvent être vues comme des dépendances fonctionnelles ou des flux informationnels ou des perturbations. Une dépendance fonctionnelle est, par exemple, l'utilisation des valeurs ou des variations d'une variable numérique d'un modèle dans un autre modèle. Typiquement, si les éléments se limitent à des processus décrits par des équations différentielles, les relations seront alors l'utilisation de variables décrites par des équations différentielles dans l'un des modèles du système au sein du schéma de variation dans un autre modèle. Par exemple, si les deux compartiments d'un système proie-prédateur sont décrits dans deux sous-systèmes, les variations de la population de l'un a une influence sur l'autre. Le deuxième type d'interaction, les flux informationnels, consiste à informer de l'état d'une partie du système via une donnée. Cette information peut avoir pour but de prendre une décision en fonction de l'état du système. Les décisions prises pour le pilotage d'un système, par exemple, peuvent se faire via la notion de perturbation. Une perturbation est une modification de la trajectoire courante d'un sous-système. Pour poursuivre sur l'exemple du système proie-prédateur, nous pouvons imaginer qu'une entité externe au système veuille contrôler l'une des deux populations. En fonction d'un seuil observé à une certaine date (flux informationnel), l'entité externe prend la décision d'éliminer une partie d'une des populations, une perturbation est alors envoyée. La contrainte en terme de modélisation est d'être capable de prendre en compte la perturbation. Ce n'est pas

toujours très simple : comment garantir qu'une modification sur une variable en cours d'intégration au sein d'une équation différentielle est correcte ?

L'identification des interactions impose aussi la définition de ports d'entrées-sorties. Deux approches sont encore possibles :

– chaque sous-système (modèle) possède un unique port d'entrée et un unique port de sortie et le modèle est alors capable de traiter les informations reçues ;

– il existe plusieurs ports d'entrée et de sortie et le nommage des ports permet de spécifier la sémantique des interactions.

Il est de loin préférable d'adopter la deuxième stratégie.

L'étape suivante consiste à distinguer les sous-systèmes décomposables et les non-décomposables. Pour les premiers, il suffit de réitérer la démarche d'identification des limites, des éléments constituants et des interactions. Quant aux seconds, le modélisateur doit faire un choix de formalismes pour spécifier la dynamique temporelle du sous-système. Contrairement aux approches classiques où le choix du formalisme est assez limité, notre démarche propose au modélisateur de faire le choix le plus en adéquation avec les éléments à représenter. Si la dynamique se résume à un ensemble de variables continues évoluant en continu dans le temps alors les équations différentielles semblent un bon choix. En revanche, si la dynamique est plutôt un ensemble d'états discrets que l'on peut décrire par un automate à états finis plus ou moins sophistiqué alors un simple automate de type Moore ou un statechart ou encore un Réseau de Petri peut être utilisé. L'idée est séduisante mais dans les faits, tout n'est pas simple. Néanmoins, c'est l'approche que nous préconisons et nous verrons comment DEVS et la plate-forme VLE permettent de supporter ces choix de formalismes.

En résumé, notre démarche de modélisation se base sur l'approche systémique où :

– les limites des systèmes et sous-systèmes sont définies par les notions de modèles atomiques et couplés de DEVS ;

– les interactions sont représentées par les connexions entre modèles via les ports d'entrée et de sortie des modèles ;

– les niveaux hiérarchiques sont supportés par la hiérarchie des modèles couplés.

Les deux idées importantes sont la prise en charge des différents types d'interaction et le multiformalisme. Il est très important de penser ses modèles selon plusieurs dimensions d'interactions : dépendance fonctionnelle (fonctionnement normal du système), flux informationnel (pour la prise de décision) et perturbation. Ce dernier point est sûrement le moins courant et le plus difficile à prendre en compte. La notion de perturbation représente le potentiel d'intégration d'un modèle au sein d'un modèle plus global. Ce point est aussi un point important dans notre démarche : le couplage de modèles. Il est nécessaire qu'un modèle soit ouvert pour qu'il soit potentiellement couplable avec d'autres modèles. C'est une dimension rarement prise en compte lors de la modélisation mais elle permet d'envisager des approches de modélisation par composition (approche ascendante). Si un modèle est ouvert, il peut alors être couplé avec d'autres modèles compatibles et former un modèle couplé et donc définir un nouveau niveau hiérarchique.

Nous allons maintenant voir comment la plate-forme VLE et ses outils mettent en œuvre la démarche.

4. VLE comme plate-forme de multimodélisation

Comme nous l'avons présenté dans la section 2, le formalisme PDEVS est très général et le développement de modèles opérationnels est loin d'être simple. Pourquoi ? Premièrement, il est nécessaire de s'inscrire dans une modélisation à événements discrets or de nombreux modèles sont à base d'équations différentielles (à temps continu) ou d'équations aux différences (à temps discret) et leur reformulation en modèles à événements discrets PDEVS n'est pas triviale et/ou, peut nécessiter un temps non négligeable. Deuxièmement, PDEVS laisse une totale liberté sur l'expression des états, des fonctions de transition, des valeurs transportées par les événements. Nous sommes loin d'un formalisme profondément expressif. PDEVS offre simplement une structuration des modèles et une algorithmique minimaliste. En réalité, cette simplicité est le garant pour définir un socle solide et simple pour le couplage des modèles. Naturellement, il est alors nécessaire de dis-

poser d'une infrastructure informatique pour aider le modélisateur. La plate-forme « *Virtual Laboratory Environment* » (VLE) est une solution modulaire et extensive pour réaliser des modèles multiformalismes par couplage. Nous allons voir comment VLE répond à la fois aux défis conceptuel et technique.

Dans cette partie, après une brève explication de l'architecture logicielle du cadre VLE, nous décrivons le fonctionnement interne de son noyau de simulation et les outils fournis aux modélisateurs pour développer leurs modèles. Enfin, en dernière partie, nous verrons comment utiliser VLE dans d'autres contextes.

4.1. *Architecture logicielle*

L'architecture logicielle de l'environnement VLE suit les dernières tendances du génie logiciel pour le développement de logiciels. L'environnement VLE est ainsi découpé en plusieurs bibliothèques réutilisables et sous licence libre (la figure 2 représente graphiquement l'architecture de VLE). Chacune de ces bibliothèques répond à un besoin ou à un problème particulier et propose une interface de programmation simple et souple à utiliser pour les développeurs. À l'aide de ces bibliothèques, sont développés des logiciels, comme un simulateur, des outils de visualisation de résultats, des interfaces graphiques de modélisation mais également des encapsulations vers des langages de programmation différents afin d'ouvrir la plate-forme à d'autres langages et à d'autres utilisations. De plus, certaines de ces bibliothèques sont extensibles via l'utilisation de greffons ou *plug-ins*. Ainsi, les développeurs et les utilisateurs peuvent étendre la plate-forme sans avoir à développer directement dans les interfaces de VLE.

Au-dessus de VLE, les bibliothèques les plus importantes dont dispose le développeur de modèles ou de programmes sont :

utils : une bibliothèque pour augmenter la portabilité de la plate-forme. Elle fournit des fonctions de chargements dynamiques de bibliothèques, de conversion de données, etc.

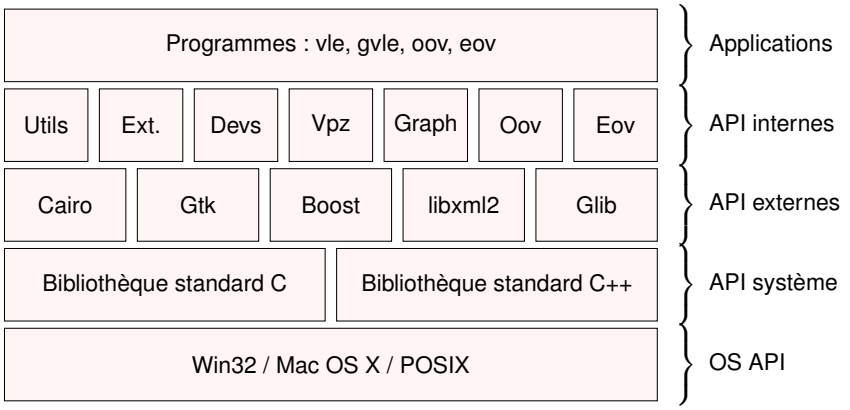

Figure 2 – Diagramme d'architecture de VLE.

graph : cette bibliothèque permet de représenter la structure des modèles DEVS avec les modèles couplés, modèles atomiques, les ports et les connexions. Elle fournit également des fonctions utiles au noyau de simulation pour calculer le chemin des événements de manière efficace.

vpz : une représentation objet des données, structure de modèles, conditions expérimentales et observations, de VLE. Son utilisation principale est l'enregistrement et la lecture des plans d'expériences.

devs : le noyau de simulation qui implémente le noyau de simulation PDEVS. Elle peut être étendue par des greffons pour développer des comportements de modèles atomiques.

extension : l'ensemble des extensions que nous fournissons aux modélisateurs, équations différentielles, équations aux différences, systèmes à événements discrets, etc.

oov, eov : les outils pour l'enregistrement de résultats en cours ou en fin de simulation dans des fichiers de sorties, dans des objets mémoire ou dans des interfaces graphiques.

Dans les deux prochaines sections, nous décrivons plus en détail les bibliothèques *devs* et *extension*. Enfin, en dernière partie, nous développons la partie modélisation avec l'outil GVLE.

4.2. *Simulation*

Dans la démarche de décomposition de l'architecture logicielle de VLE, nous avons développé le noyau de simulation en suivant les algorithmes abstraits PDEVS, présentés dans la section précédente, sur une implémentation DEVS Bus introduite par Kim [KK98]. Ce bus permet de développer rapidement un noyau de simulation DEVS en découpant les tâches d'une simulation : gestion du graphe de modèles, gestion de l'échéancier, etc. La figure 3 représente graphiquement la décomposition des tâches dans un DEVS Bus.

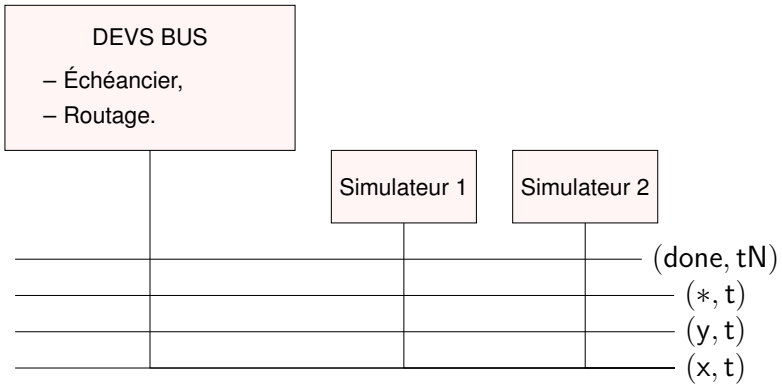

Figure 3 – Diagramme d'architecture en DEVS Bus de VLE.

Le DEVS Bus est un *pattern* classique de bus logiciel en informatique. En DEVS, le bus contient quatre types de connexion pour l'échange de messages entre les modèles et entre les modèles et le coordinateur racine, (x, t), (y, t), $(*, t)$ et $(done, tN)$. Sur ce bus, deux types de composants peuvent se connecter. Le premier, le coordinateur, gère la structure du modèle pour le routage des événements ainsi que l'échéancier. Les composants constitutifs du deuxième type sont les simulateurs des modèles atomiques.

Dans un cadre de simulation basé sur un DEVS Bus, la multimodélisation est créée en s'appuyant sur une des propriétés essentielles de DEVS qu'est l'encapsulation. En effet, B. P. Zeigler a montré que les autres formalismes issus des systèmes dynamiques où le temps est discrétisé, événementiel ou continu, peuvent être représentés ou généralisés en DEVS [ZKP00]. Ainsi, plutôt que de développer des simulateurs spécifiques aux modèles à temps discrets de type DTSS, à temps continu comme DESS, ou hybride comme DEV&DESS ou HFSS [Bar08], l'encapsulation propose de modéliser ces formalismes directement en tant que modèle DEVS. Le schéma 4 montre l'évolution du bus logiciel avec les formalismes.

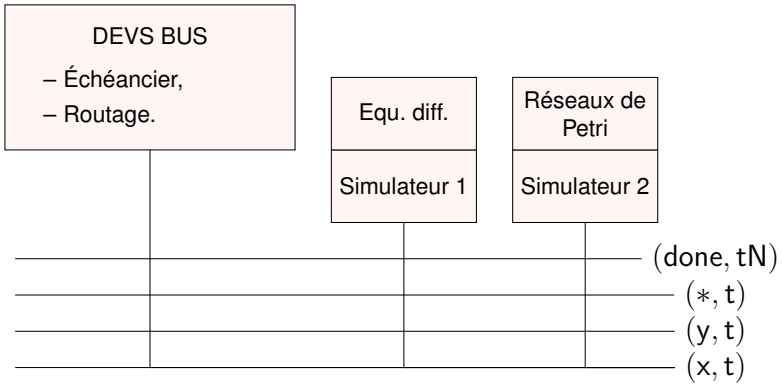

Figure 4 – DEVS Bus et multiformalisme dans VLE.

4.3. *Extensions DEVS*

Dans le but de fournir un simulateur complet et opérationnel, nous proposons aux modélisateurs un ensemble complet de formalismes pour le développement de multimodèles. Nous proposons ainsi une extension spécialisée dans la simulation d'équations aux différences, deux extensions pour la simulation d'équations différentielles ordinaires, un ensemble d'extensions pour les systèmes événementiels et une extension pour la gestion dynamique des modèles.

4.3.1. *Équations aux différences*

Cette extension, nommée *DifferenceEquation* permet de développer des modèles à temps discret qui calculent la valeur d'une variable réelle en t en fonction d'elle-même à $t-\Delta t$, $t-2\Delta t\ldots$ et en fonction d'autres variables réelles en t, $t-\Delta t$, $t-2\Delta t\ldots$ Formellement, cette extension permet de résoudre des systèmes d'équations :

$$\begin{aligned}
&\forall i \in \{1,\ldots,n\},\\
&\qquad X_i(t) &&= f_i(\underline{\mathbf{Z}}_i(t), \underline{\mathbf{W}}_i(t))\\
&\qquad \underline{\mathbf{Z}}_i(t) &&= \big[X_i(t-\Delta t),\ldots,X_i(0)\big]\\
&\qquad \underline{\mathbf{W}}_i(t) &&= \big[\ldots,\underline{\mathbf{W}}_j(t),\ldots\big] \text{ pour } j \in \{1,\ldots,n\} \text{ et } j \neq i\\
&\qquad \underline{\mathbf{W}}_j(t) &&= \big[X_j(t), X_j(t-\Delta t),\ldots,X_j(0)\big]
\end{aligned}$$

4.3.2. *Équations différentielles ordinaires*

L'environnement VLE propose plusieurs méthodes numériques pour la résolution des équations différentielles ordinaires, très employées pour représenter des phénomènes physiques ou biologiques. Nous séparons ces outils en deux catégories. La première regroupe les méthodes classiques comme Euler ou Runge Kutta qui fonctionnent par discrétisation de l'espace du temps dans leurs méthodes. La deuxième, appelée quantification, et disponible sous le nom QSS [KJ01], discrétise l'espace des valeurs des variables. Pour ces deux types de méthodes, nous avons prévu des méthodes pour la perturbation des équations en cours de simulation et également l'observation comme le dépassement de seuil sur la valeur des variables ou sur leurs dérivées dans les méthodes dérivées d'Euler et de manière purement événementielle pour les méthodes quantification.

4.3.3. *Automates à états finis*

Cet ensemble d'extensions appelé FSA permet de définir des modèles sous forme de graphe de transitions entre des états discrets. Avec VLE, nous proposons quatre sortes d'automates :

– Moore : un automate à états dont les valeurs en sortie ne dépendent que de l'état courant.

– Mealy : également un automate à états, mais pour lequel les valeurs des variables de sortie dépendent à la fois de l'état courant et des variables d'entrée.

– FD-DEVS [HZ06] : une restriction de DEVS à un état défini par un ensemble fini d'états discrets.

– Réseaux de Petri généralisés : formalisme prévu pour la modélisation et l'analyse du comportement des systèmes dynamiques à événements discrets. La version proposée par VLE correspond à des Réseaux de Petri T-temporisés, synchronisés, généralisés, avec arcs inhibiteurs ; les interactions avec l'extérieur sont possibles grâce la génération de jetons dans les places ou le franchissement de transitions sur la réception d'événements.

– Statechart d'UML [Har87] : principalement utilisé dans la communauté du développement logiciel pour la modélisation du comportement des objets. VLE propose une implémentation presque complète à l'exception de la hiérarchisation des états et des historiques.

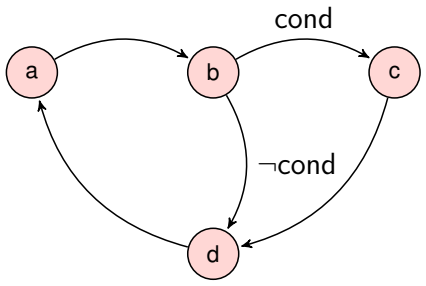

Figure 5 – Un exemple d'automate à quatre états et cinq transitions possibles entre ces états.

4.3.4. *Extension décision*

Cette extension, ou modèle générique, n'implémente pas un formalisme particulier mais un ensemble de formalismes regroupés autour de la prise de décision pendant la simulation. Elle est utilisée lorsque

le modélisateur a besoin d'organiser dans le temps des activités, par exemple l'activité de récolte en agriculture, dans le but d'atteindre des objectifs, par exemple, optimiser les rendements des cultures d'une exploitation agricole. Cette extension regroupe ainsi un planificateur de tâches, un moteur d'inférence d'ordre 1 (logique de prédicats) et un simulateur de plan.

4.3.5. *Gestion dynamique des modèles*

À l'origine, la structure des modèles couplés (modèles et connexions) est purement statique. Cela signifie que le nombre de modèles, le nombre et la nature des connexions ne changent pas au cours de la simulation. Dans [Bar96, Bar97], Barros propose une extension Dynamic Structure DEVS (DS-DEVS) qui offre la possibilité de créer et de faire disparaître les modèles et les connexions en cours de simulation. Cette extension, contrairement aux autres, a une implication sur le noyau de synchronisation afin de garantir la causalité. La conséquence est que les modifications de structure ne sont appliquées qu'en dernier lieu (après l'ensemble des bags).

Au sein de VLE, l'extension DS-DEVS est utilisable à la fois pour modifier la structure en cours de simulation, mais aussi pour construire la structure à l'état initial. Dans ce dernier cas, l'idée est d'offrir la possibilité au modélisateur de construire son modèle selon un algorithme, l'exemple classique étant l'automate cellulaire. Il est nettement plus simple de décrire algorithmiquement les règles de construction d'un automate cellulaire (voisinage…) que de définir "à la main" les connexions. VLE offre dans ce cadre un autre mécanisme : la notion de classes de modèles. Un modèle couplé ou atomique peut être défini comme une classe et être instancié à la volée via des modèles DS-DEVS nommés Executive.

4.4. *Modélisation*

Notre but avec cette plate-forme est de proposer un outil complet pour réaliser le cycle de modélisation et de simulation (*cf.* figure 6). Chacune des activités de ce cycle est prise en charge par un ou plusieurs composants de VLE.

Modélisation à l'aide des outils classiques de modélisation, équations différentielles, équations aux différences, automates à états, systèmes à événements discrets ou DEVS en dernier niveau étape.

Implémentation en code informatique de votre modèle, utilisation des bibliothèques de VLE et des extensions proposées et de l'interface graphique GVLE pour composer vos modèles.

Préparation du plan d'expérience avec l'initialisation des paramètres et des variables, le nombre de réplicats, les modèles à observer, comment les observer et où diriger les données : GVLE.

Exécution des simulations depuis VLE, GVLE, RVLE ou PyVLE sur une machine locale ou une grille de calcul.

Notre but est ainsi d'offrir aux modélisateurs l'accès à DEVS, pour la modélisation de modèles hétérogènes, la simulation sur simple ordinateur portable jusqu'aux clusters et autres grilles de calcul et l'analyse des sorties, et si possible en proposant une intégration dans leurs outils.

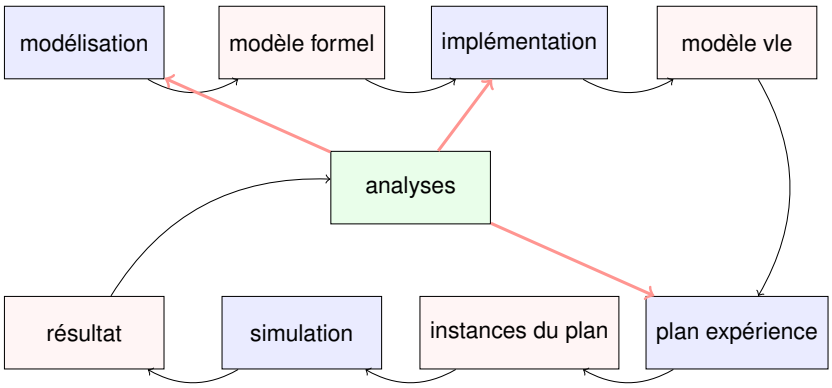

Figure 6 – Le cycle de modélisation et de simulation

4.4.1. *Modélisation*

L'IDE de VLE propose un ensemble d'outils pour aider le modélisateur à débuter, construire, développer et analyser ses modèles. L'IDE

principal de VLE, GVLE, centralise la plupart des éléments du cycle de modélisation et de simulation présenté précédemment (*cf.* figure 6). Il permet :

– l'implémentation du comportement de modèles atomiques avec son éditeur de code source intégré, une gestion de la compilation des modèles.

– l'utilisation de greffons de modélisation pour la saisie simplifiée du comportement des modèles basés sur les extensions proposées par VLE. Nous proposons ainsi un éditeur pour les réseaux de Petri, pour les *Statechars* d'UML ou pour les équations aux différences.

– le développement de la structure des modèles, avec la définition des modèles couplés, des sous-modèles attachés, de leurs ports de connexions.

– l'édition des conditions expérimentales pour la définition de plans d'expériences ainsi que l'association entre les modèles et les outils pour les observer.

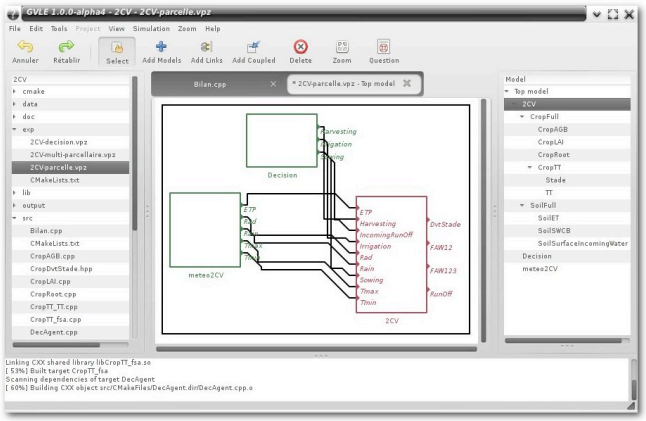

Figure 7 – Capture d'écran de l'IDE de VLE, GVLE.

4.4.2. *Analyse*

La méthodologie et l'infrastructure employées pour le développement de VLE nous ont permis de proposer de nouvelles activités autour

de la simulation. En effet, du fait de l'utilisation de bibliothèques et de greffons pour étendre ces bibliothèques, la réutilisation des composants de VLE nous a grandement simplifié le développement d'applications tierces. Ainsi, nous proposons des applications pour l'exécution de simulations depuis des interfaces graphiques, via des langages de scripts ou depuis des environnements Web. Ces derniers sont basés sur une bibliothèque Python, nommée *pyvle*, qui encapsule une partie des bibliothèques de VLE et permet de paramétrer les entrées et sorties des modèles et d'exécuter des simulations à distance, le tout depuis des sites Internet. De la même manière, nous proposons un paquet pour l'environnement de statistiques R [R D11], projet très utilisé dans les domaines dans lesquels nous travaillons. Les utilisateurs de R peuvent alors employer les algorithmes et fonctions fournis par cet environnement pour la réalisation de plans d'expériences, d'optimisation et d'analyse de sensibilité de leurs modèles.

5. Des exemples concrets

L'objectif de cette troisième partie est de montrer la variété des modèles pris en charge par VLE. Cette diversité est considérée selon plusieurs axes : le couplage de formalismes différents, la nature du temps et de l'espace, l'existence préalable ou non de code...

Le premier modèle que nous allons traiter est un modèle de plante selon l'approche "structure-fonction" : nous représentons explicitement la structure de la plante (feuille, bourgeon, axe, entre nœu...) à laquelle nous attachons des processus. Chaque partie de la plante (structure) joue un rôle au sein du système global. Ecomeristem [LDK$^+$06, LDCV08] est un modèle de type structure-fonction représentant la croissance ecophysiologique de culture. Il a été réalisé pour simuler le riz (modèle de plante pour les céréales) et pour traiter de la morphogénèse de la plante, ainsi que de sa plasticité. Celle-ci dépendant de son potentiel génétique (G) et de sa sensibilité à l'environnement (E) : eau, température, radiation... Initialement, il a été créé pour simuler la morphogénèse végétative de la plante, mais est actuellement étendu pour simuler tout le cycle de la plante. Il formalise le fonctionnement du méristème apical (la régulation GxE de l'initiation, le pré-dimensionnement et l'avortement

des organes). Une plante moyenne est simulée de manière journalière en termes de topologie, taille d'organe et poids de matière sèche. La géométrie de la plante n'est pas considérée et l'acquisition de ressource est, jusqu'à présent, simulée par une approche "big leaf". Le système racinaire est considéré comme un seul et unique compartiment avec une demande en ressource journalière afin de réaliser sa croissance. E interagit avec la plante par le biais de deux variables d'état de la plante : l'eau et le statut carboné (C). Ce dernier est calculé tous les jours en réalisant le ratio entre l'apport de la plante (par le biais de la photosynthèse) et ce dont les organes ont besoin (la somme des demandes des organes nécessaires à leurs croissances journalières). La plante répond à ces variables d'état en fonction de paramètres génétiques définissant : i) la morphogénèse potentielle (vitesse d'apparition des feuilles, taille, tallage) et ii) la sensibilité à E (réponse du tallage au carbone, transpiration de la feuille, réponse de l'expansion de la feuille à la sécheresse...). La principale caractéristique du modèle Ecomeristem est l'évolution dynamique de sa structure. VLE propose une implémentation de DS-DEVS proposée par Barros [Bar96, Bar97]. Elle autorise l'apparition et la disparition de modèles et de connexions au cours de la simulation. Par défaut, PDEVS se base sur une structure statique. Cette fonctionnalité est donc fondamentale et doit être correctement intégrée à l'algorithme des simulateurs abstraits afin de garantir la causalité, en particulier. Pour concevoir Ecomeristem, DS-DEVS est le premier élément et il est accompagné de la notion de classes de modèles. Chaque entité de la plante est définie au sein d'une classe de modèles : axe, phytomère, feuille... Des automates à états modélisent les différentes phases des organes de la plante et lors de certaines phases, des modèles DS-DEVS instancient les classes : un nouveau phytomère ou une nouvelle feuille, par exemple. Les processus élémentaires sont des équations aux différences dont le pas de temps est la journée. L'aspect dynamique de ce modèle impose quelques précautions au niveau de la synchronisation des processus car il existe de multiples interactions entre les processus des organes et des processus plus globaux au niveau de la plante.

Le deuxième modèle est un exemple de couplage de deux modèles existants : fonctionnement biophysique du bassin versant et fonctionnement technique d'exploitations agricoles. D'une part, nous avons un modèle spatial, TNT2 [BDR$^+$02], représentant les processus biophy-

siques de transfert de particules (nitrate, phosphore...) dans un bassin versant ainsi que la croissance des plantes sur les parcelles. D'autre part, Melodie [CRB+07] propose une modélisation d'exploitation agricole intégrant les processus de décision des opérations techniques (pilotage), de démographie et d'ingestion/déjection des cheptels (porcins et bovins) et de transformation biochimiques. La relation entre les deux modèles est multiple : impact des décisions liées aux cultures (itinéraires techniques - semis, apport d'éléments organiques ou non, récolte...), impact des récoltes sur les stocks d'aliments. Contrairement au premier modèle, nous avons deux gros modèles, tous les deux développés en C/C++ mais ayant chacun adopté sa propre stratégie de modélisation. De plus, ce sont tous les deux des simulateurs, cela signifie qu'ils embarquent leur propre boucle de simulation, or VLE doit piloter l'avancement du temps. Nous avons traité les deux modèles différemment. Dans le cas de TNT2, la boucle temporelle était facilement isolable au sein du code. La partie prenant en charge les processus journaliers a été mise sous forme de fonction. Cela a permis de pérenniser l'existant en ne modifiant que très légèrement le code. Le modèle est maintenant sous forme d'une librairie et le travail sous VLE consiste à construire un "wrapper" qui rend le modèle compatible PDEVS. Quatre aspects sont alors à considérer : l'avancement du temps, la fonction de transition interne, la fonction de transition externe et la fonction de sortie. La problématique de l'avancement du temps est très simple puisque le modèle est à pas discret. La fonction ta est donc une constante. La fonction de transition interne est tout aussi simple puisqu'elle consiste à invoquer la librairie de calcul des processus biophysiques. La fonction de transition externe et la fonction de sortie sont plus délicates et sont liées aux interactions définies entre les deux modèles. Concernant Melodie, il a fallu décomposer le modèle en plusieurs sous-modèles afin de n'utiliser dans un premier temps, que la partie décisionnelle générant les itinéraires techniques et les plans d'épandage. Comme première approximation, seules ces informations sont utiles. Nous avons dû adjoindre à cette partie un modèle de changement d'échelle temporelle mais aussi de changement de sémantique. En effet, le modèle est annuel alors que TNT2 est journalier et les représentations des parcelles, des cultures... diffèrent entre les deux modèles.

En conclusion, ces deux exemples ont permis de montrer comment nous avons pu aborder deux systèmes totalement différents et surtout comment nous avons utilisé la puissance de modélisation de VLE : la multimodélisation et le multiformalisme.

6. Perspectives et conclusion

Nous avons montré comment un formalisme de haut niveau (DEVS), une démarche de modélisation centrée systémique et l'outil VLE peuvent répondre à la problématique de la modélisation et de la simulation des systèmes complexes. Nous avons développé une solution qui permet de répondre à la problématique de multimodélisation multiparadigme par couplage de modèles dans un contexte événementiel.

Aujourd'hui, VLE va évoluer en augmentant son champ d'application dans les très gros systèmes où le nombre de modèles est très important. Ces travaux nécessitent le développement d'une nouvelle version du noyau de simulation pour introduire des capacités de parallélisation. Ces travaux, sont aujourd'hui en cours et les premiers résultats sont attendus en 2012. Du point de vue technique, il est fondamental d'augmenter la simplicité d'utilisation des API en particulier pour l'utilisation des formalismes. Il est aussi probable qu'il faille intégrer de nouveaux paradigmes. De plus en plus de demandes émergent dans le domaine de systèmes multiagents. Une formalisation existe mais son intégration au sein de l'actuelle plate-forme n'est pas effective. Ce domaine est un énorme chantier car il nécessite une réflexion sur la sémantique des concepts (classes) qui seront proposés aux modélisateurs.

Les systèmes que nous explorons font appel à des modèles de décision et nécessitent des opérations de planification et d'optimisation. À l'heure actuelle, quelques modules d'optimisation ont été validés en interaction avec VLE mais une intégration plus fine doit être envisagée. Il est probablement intéressant d'étudier l'optimisation dans la perspective des cadres expérimentaux où la partie optimisation est elle-même un modèle DEVS voire un ensemble de modèles en interaction.

Une autre voie est l'intégration d'éléments humains dans la boucle de simulation et l'interaction en temps réel avec un modèle. Dans une

démarche plus collaborative de l'activité de modélisation, il serait intéressant d'offrir une version ouverte et intégrant l'humain comme modèle au sein d'un modèle global. RT-DEVS [CHZ03] tend à répondre à ce besoin.

Références

[AdlPP+08] P. Auger, R. Bravo de la Parra, C. Poggiale, E. Sanchez, and T. Nguyen Huu. Aggregation of variables and applications to population dynamics. *Structured Population Models in Biology and Epidemiology, Springer*, 2008.

[Bar96] F. J. Barros. Dynamic Structure Discret Event System Specification : Formalism, Abstract Simulators and Applications. 13(1) :35–46, 1996.

[Bar97] F. J. Barros. Modeling Formalisms for Dynamic Structure Systems. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 7 :501 – 515, october 1997.

[Bar08] F. J. Barros. Semantics of dynamic structure event-based systems. In *Proceedings of the second international conference on Distributed event-based systems*, DEBS '08, pages 245–252, New York, NY, USA, 2008. ACM.

[BDR+02] V. Beaujouan, P. Durand, L. Ruiz, P. Aurousseau, and G. Cotteret. A hydrological model dedicated to topography-based simulation of nitrogen transfer and transformation : rationale and application to the geomorphology–denitrification relationship. *Hydrol. Process*, 16 :493–507, 2002.

[Ber93] L. Von Bertalanffy. *Théorie générale des systèmes*. Dunod, 1993.

[CGMC+07] P. Chabrier, F. Garcia, R. Martin-Clouaire, G. Quesnel, and H. Raynal. Toward a simulation modeling platform for studying cropping systems management : the record project. In *Proceedings of the International Congress on*

Modelling and Simulation, Christchurch. New Zealand, december 2007. nternational Society for Computer Simulation.

[CHZ03] Y. K. Cho, X. Hu, and B. P. Zeigler. The rtdevs/corba environment for simulation-based design of distributed real-time systems. *Simulation*, 79 :197–210, 2003.

[CRB+07] X. Chardon, C. Rigolot, C. Baratte, A. Le Gall, R. Martin-Clouaire S. Espagnol, J.P. Rellier, C. Raison, J.C. Poupa, and P. Faverdin. Melodie : a whole-farm model to study the dynamics of nutrients in integrated dairy and pig farms. In *MODSIM 2007 : International Congress on Modelling and Simulation*, pages 1638–1645. eds L Oxley and D Kulasiri, 2007.

[CZ94] A.C.H. Chow and B.P. Zeigler. Parallel DEVS : a parallel, hierarchical, modular, modeling formalism. In *Proceedings of the 26th conference on Winter simulation*, pages 716–722, Orlando, Florida, United States, 1994.

[DRP03] R. Duboz, É. Ramat, and P. Preux. Scale transfer modelling : Using emergent computation for coupling an ordinary differential equation system with a reactive agent model. *Systems Analysis Modelling & Simulation*, 43(6) :793 – 814, June 2003.

[DRQ04] R. Duboz, E. Ramat, and G. Quesnel. Systèmes multi-agents et théorie de la modélisation et de la simulation : une analogie opérationnelle. In *Actes des douzièmes Journées Francophones sur les Systèmes Multi-Agents (JFSMA) - Systèmes multi-agents défis scientifiques et nouveaux usages*. Olivier Boissier et Zahia Guessoum eds., 2004.

[Dub04] R. Duboz. *Intégration de modèles hétérogènes pour la modélisation et la simulation de systèmes complexes : Application au transfert d'échelles en écologie marin*. PhD thesis, Université du Littoral Cote d'Opale, ULCO, Calais, France, mars 2004.

[Har87] D. Harel. Statecharts : A visual formalism for complex systems. *Sci. Comput. Programming*, 8 :231–274, 1987.

[HZ06] M. H. Hwang and B. P. Zeigler. A modular verification framework using finite and deterministic devs. In *Proceedings of 2006 DEVS Symposium*, pages 57–65, 2006.

[KJ01] E. Kofman and S. Junco. Quantized State Systems. a DEVS Approach for Continuous Systems Simulation. In *Transactions of SCS.*, volume 18, pages 123–132, 2001.

[KK98] J. Y. Kim and T. G. Kim. A heterogeneous simulation framework based on the DEVS bus and the High Level Architecture. In *Winter Simulation Conference*, Washington, DC, 1998.

[LDCV08] D. Luquet, M. Dingkuhn, and A. Clément-Vidal. Modeling drought effects on rice with ecomeristem : feedbacks of water and carbohydrate relations on phenotypic plasticity. In *In 'Whole plant physiology modelling project. Final meeting*, 2008.

[LDK+06] D. Luquet, M. Dingkuhn, H.K. Kim, L. Tambour, and A. Clément-Vidal. Ecomeristem, a model of morphogenesis and competition among sinks. In *Rice. 1. Concept, Validation and Sensitivity analysis. Functional Plant Biology*, volume 33, pages 309–323, 2006.

[Moi77] J. L. Le Moigne. *La théorie du système général. Théorie de la modélisation*. PUF, 1977.

[QDR04] G. Quesnel, R. Duboz, and É. Ramat. DEVS wrapping : A study case. In *In proceedings of Conceptual Modeling and Simulation*, pages 374–382, Genoa, Italy, october 2004.

[QDRT07] G. Quesnel, R. Duboz, E. Ramat, and M.K. Traoré. VLE : A Multimodeling and Simulation Environment. In *Proceedings of the Summer Simulation Multiconference*

(SummerSim'07), pages 367–374, San Diego, California, USA, july 2007.

[QDVR05] G. Quesnel, R. Duboz, D. Versmisse, and É. Ramat. DEVS coupling of spatial and ordinary differential equations : Vle framework. In *Proceedings of the Open International Conference on Modeling and Simulation - OICMS*, pages 281–294, Clermont-Ferrand, France, 2005.

[Que06] G. Quesnel. *Approche formelle et opérationnelle de la multi-modélisation et de la simulation des systèmes complexes*. PhD thesis, Université du Littoral Cote d'Opale, ULCO, Calais, France, décembre 2006.

[R D11] R Development Core Team. *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[RD05] H. Alla R. David. *Discrete, Continuous, and Hybrid Petri Nets*. Springer-Verlag, 2005.

[RP03] E. Ramat and P. Preux. Virtual laboratory environment (vle) : An software environment oriented agent and object for modeling and simulation of complex systems. *Journal of Simulation Practice and Theory*, 11, 2003.

[VR05] D. Versmisse and E. Ramat. Management of perturbations within a spatialized differential equations system, in the devs framework. In *European Simulation and Modeling Conference (ESM)*, Porto, Portugal, october 2005.

[Zei76] B. P. Zeigler. *Theory Of Modeling and Simulation*. Wiley Interscience, 1976.

[ZKP00] B. P. Zeigler, D. Kim, and H. Praehofer. *Theory of modeling and simulation : Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, 2000.