3.1 Introduction

Dans notre travail, nous abordons le problème du transfert d'échelles et pour cela nous proposons de coupler deux modèles du même système perçu à deux échelles différentes (nous avons présenté ces modèles au chapitre 2.3). Ces deux modèles sont hétérogènes de plusieurs points de vue :

- paradigme,
- représentation du temps et de l'espace,
- stochastique *versus* déterministe,
- formalisme,
- implémentation.

Une hétérogénéité entre paradigmes implique une différence de points de vue sur le système modélisé. C'est précisément ce qui nous intérresse ici pour la modélisation du transfert d'échelles. En effet, le modèle d'agents réactifs propose une vision centrée sur les individus du système alors que les équations différentielles nous proposent une vision globale de la population. Nous reviendrons en détail sur cet aspect au chapitre 5. Ici, nous nous intéressons aux quatre autres points cités plus haut. Dans les deux modèles, la représentation du temps doit être explicite. Néanmoins, le système d'équations différentielles que nous avons présenté n'a pas de composante spatiale. Le couplage des deux modèles va donc impliquer une transformation de données échangées par les modèles afin de les rendre compatibles. De même, le modèle agent du copépode est stochastique et le système d'équations différentielles déterministes. Là aussi, le couplage devra tenir compte de cette spécificité.
L'hétérogénéité au niveau du formalisme implique que l'implémentation des deux modèles soit également hétérogène. Ceci pose le problème de la compréhension globale du simulateur utilisé pour coupler les deux modèles. Si un système complexe peut être modélisé à l'aide de plusieurs formalismes, différentes approches sont possibles [VLM02] :

- une intégration des formalismes utilisés dans un seul nouveau formalisme. C'est l'idée développée par H. Vangheluwe [Van00] avec le DAE (Differential Algebraic Equations) ou B.P. Zeigler [ZKP00] avec le DEV&DESS. Ces deux formalismes intègrent des modèles à temps continu et discret,
- une spécification des sous-modèles du système dans un formalisme unique. Cette approche est différente de la précédente dans le sens où elle nécessite de trouver un formalisme commun à tous les sous-modèles et une réécriture de l'ensemble des sous-modèles dans ce formalisme,
- une approche de co-simulation, c'est-à-dire que chaque sous-modèle possède son propre simulateur caractéristique du formalisme dans lequel le modèle est spécifié. La difficulté est de coupler ces simulateurs.

Dans ce chapitre, nous proposons d'utiliser la deuxième approche pour la formalisation de notre système couplé. Il s'agit donc de formaliser un système d'agents réactifs situés et un système d'équations différentielles dans le même formalisme. Le but de cette opération est de fournir une description compréhensible, unifiée et non ambiguë de notre modèle couplé. En introduction de cette thèse, nous avons présenté DEVS comme étant un formalisme capable de spécifier un grand nombre de systèmes dynamiques. Nous nous sommes donc orientés vers ce formalisme pour la spécification de notre système couplé.

De nombreux travaux existent sur la formalisation des SMAs. Nous avons donc d'abord orienté nos recherches dans ce sens. Il est possible de trouver des exemples des différents formalismes utilisés pour les SMAs dans le livre de Weiss [Wei99] avec principalement des formalismes lo-

giques pour la spécification du raisonnement dans les SMAs cognitifs. Dans ce travail, nous nous intéressons aux SMAs réactifs dans un contexte de modélisation et simulation de systèmes dynamiques. Là aussi des travaux existent, nous y reviendrons dans la discussion de ce chapitre. La prise en compte de la dynamique implique que le temps est une variable primordiale des systèmes considérés. Nous aurions également pu nous tourner vers des formalismes purement mathématiques. Il est en effet possible de spécifier des SMAs à l'aide d'équations différentielles par exemple. Les physiciens s'y intéressent depuis peu avec des modèles de particules browniennes actives [Sch97][CS02] ou des modèles de collaboration entre robots [LG02]. Néanmoins, cette formalisation peut s'avérer complexe et surtout limitante quant à la spécification des comportements des agents[38]. De plus, nous avons voulu conserver la nature discrète des individus et de leurs interactions dans le milieu naturel en nous intéressant à l'influence du type de distribution des proies sur l'efficacité des prédateurs. Il existe des modèles anciens d'ordre statistique qui tentent de rendre compte d'une telle influence [Har68], mais ils s'avèrent peu expressifs au regard des mécanismes qui sont réellement en jeu.

Le comportement est plus simple à modéliser s'il est représenté par un enchaînement d'états qui caractérisent l'activité de l'individu. À un ensemble d'états peut correspondre un certain type de réponse de l'individu à des *stimuli* externes. De plus, ces *stimuli* ne sont généralement pas des fonctions continues dans le milieu naturel mais plutôt des évènements ponctuels. Dans sa version originale, le formalisme DEVS permet la spécification de changements d'états évènementiels. Nous avons donc choisi ce formalisme pour la spécification de notre système d'agents réactifs.

DEVS a été utilisé la première fois en 1994 pour la spécification de SMAs [UA94]. Depuis, il n'y a pas eu, à notre connaissance, de travaux faisant un rapprochement clair entre les SMAs et DEVS. Il existe des travaux comme ceux de M.F. Hocaoglu *et al.* [HFS02], qui proposent une coopération entre un SMA et un modèle DEVS représentant l'environnement mais ils ne formalisent pas le SMA. Ainsi, dans un premier temps, nous présentons le formalisme DEVS afin de proposer un passage entre le paradigme d'agents réactifs situés vers DEVS sous la forme d'une analogie. Nous poursuivons par la formalisation du modèle agent du copépode en DEVS et son couplage au système d'équations différentielles. Nous terminons ce chapitre par une discussion sur l'utilité d'une telle approche et sur les perspectives qu'elle ouvre.

3.2 Le formalisme DEVS

DEVS définit une syntaxe basée sur la formalisation des systèmes. Cette formalisation a elle-même pour origine les mathématiques discrètes [ZKP00]. Le formalisme DEVS manipule les concepts de structure, d'ensemble et de fonction mettant en relation les différents éléments de ces ensembles. Dans ce formalisme, la représentation du temps est essentielle. En effet, dans un modèle à évènements discrets, ce sont les occurrences des évènements qui déterminent l'avancement du temps. Ainsi, nous pouvons dire que le modèle « construit » son temps au cours de la simulation. Dans ce qui suit, nous présentons les bases du formalisme DEVS.

[38] La notion de choix pour un agent est, par exemple, difficile à formaliser avec des équations différentielles.

3.2.1 DEVS atomique

Un modèle DEVS dit atomique correspond à la structure suivante :

$$DEVS =< X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

où :

$X = \{(p, v)|p \in IPorts, v \in V_X\}$ est l'ensemble des ports et des valeurs d'entrée, avec V_X l'ensemble des valeurs possibles sur les ports d'entrée,
$Y = \{(p, v)|p \in OPorts, v \in V_Y\}$ l'ensemble des ports et des valeurs de sortie, avec V_Y l'ensemble des valeurs possibles sur les ports de sortie,
S l'ensemble des états du système,
δ_{ext} la fonction de transition externe,
δ_{int} la fonction de transition interne,
λ la fonction de sortie,
ta la fonction d'avancement du temps,
$IPorts$ l'ensemble des noms des ports d'entrée,
$OPorts$ l'ensemble des noms des ports de sortie.

Nous reprenons et détaillons ces différents éléments ci-dessous.
Les notions de port d'entrée et de port de sortie permettent de représenter graphiquement une vue externe d'un modèle DEVS par une boîte où figurent ces ports. Les vecteurs d'entrée et de sortie sont l'union de tous les ports du modèle (figure 3.1).

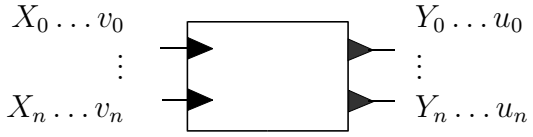

FIG. 3.1 – Représentation graphique d'un modèle DEVS atomique. Les triangles à l'intérieur de la boîte figurent les ports d'entrée. Les triangles à l'extérieur de la boîte figurent les ports de sortie.

La figure 3.1 nous montre que v_i est une valeur prise par un port d'entrée. Cette valeur appartient à l'ensemble des valeurs possibles du port X_i. De même, u_i est une valeur prise par un port de sortie. Cette valeur appartient à l'ensemble des valeurs possibles du port Y_i. Un port d'entrée prend une valeur lors de l'émission d'un évènement attaché à ce port. Un port de sortie prend une valeur lorsque la fonction de sortie prend une valeur pour ce port. Cette représentation graphique nous sera utile tout au long de ce chapitre.

L'ensemble S des états du système est un vecteur d'attributs. Les attributs peuvent être de type très différents (i.e. variables prenant leurs valeurs dans $\mathbb{R}$, ensemble fini de valeurs constantes, etc.). L'ensemble des valeurs prises par les attributs à un instant donné est appelé état du système.

La fonction d'avancement du temps $ta(s)$ définit le temps pendant lequel le modèle restera dans l'état s si aucun évènement externe ne survient. Elle est définie par :

$ta : S \rightarrow \mathbb{R}^+$

Un état $s \in S$ est dit passif si et seulement si $ta(s) = \infty$. De même, un état $s \in S$ est dit transitoire si et seulement si $ta(s) = 0$. La fonction $ta(s)$ définit la base de temps, c'est-à-dire

l'ensemble des valeurs possibles pour le temps. Ici, cette base est réelle, ce qui implique que la simulation à évènements discrets simule un temps continu.

L'ensemble Q des états totaux du système est :

$Q = \{(s, e) | s \in S, 0 < e < ta(s)\}$ où e représente le temps écoulé dans l'état s.

Ce concept d'état total (s, e) permet de mettre en évidence le fait que le temps lui-même fait partie de l'état du système. Il est donc possible de spécifier un état futur en fonction du temps écoulé dans l'état présent. Cet ensemble n'apparaît pas au niveau de la structure du modèle DEVS. Il permet de spécifier la fonction de transition externe. DEVS propose en effet deux fonctions de transition différenciant les évolutions autonomes du modèle de celles dues à des évènements externes au système. Cette dernière s'écrit comme suit :

$\delta_{ext} : Q \times X \to S$

Cette fonction représente la réponse du système aux évènements d'entrée. Le modèle est dans un état s à un instant t. Lorsqu'un évènement externe arrive sur un port d'entrée X_i, alors la fonction δ_{ext} indique le nouvel état du modèle en fonction de Q.

La fonction de transition interne, partie autonome du modèle, est définie par :

$\delta_{int} : S \to S$

Cette fonction spécifie les états futurs des états actifs. Elle est activée si aucun évènement externe ne survient. La durée de vie d'un état s (le temps qui s'écoule entre l'entrée dans s et la sortie de s) est donnée par l'évaluation de $ta(s)$ à l'entrée dans s.
Cette décomposition de la fonction de transition en deux fonctions constitue l'un des points forts du formalisme DEVS. En effet, elle autorise une spécification indépendante des évolutions autonomes du modèle et des perturbations liées aux évènements externes.

La fonction de sortie est une application de l'ensemble des états S dans l'ensemble des ports de sorties Y. Elle est définie par :

$\lambda : S \to Y$

Cette fonction sera activée lorsque le temps écoulé dans un état donné sera égal à sa durée de vie. Par suite, λ n'est définie que pour des états actifs, c'est-à-dire $\forall s | ta(s) \neq \infty$.

Un système peut être autonome et donc ne recevoir aucun évènement extérieur. La dynamique du système est alors le seul fait de la fonction de transition interne. Cette fonction de transition est définie pour spécifier les changements d'état dus exclusivement à l'état interne du système et au temps.
Considérons le système entrant à l'instant t dans l'état s. Si aucun évènement externe ne survient, alors le système changera d'état à $t + ta(s)$. La fonction ta donne la durée pendant laquelle le système sera dans un certain état. La fonction $ta(s)$ est évaluée à l'entrée dans l'état s. Cette fonction est souvent la plus complexe à déterminer car elle définit la date à laquelle le modèle dans un état s passera dans un état s', ce qui implique d'être capable d'anticiper sur les changements d'états.

Illustrons l'évolution d'un modèle DEVS sur un exemple. La figure 3.2 présente un graphe de transitions d'états à partir duquel nous pouvons dérouler un scénario.

À l'état initial, le système est dans l'état s_0 à T_0. La fonction ta nous indique que pour l'état s_0, le système changera d'état à $T_0 + ta(s_0)$ si aucun évènement externe ne survient. À

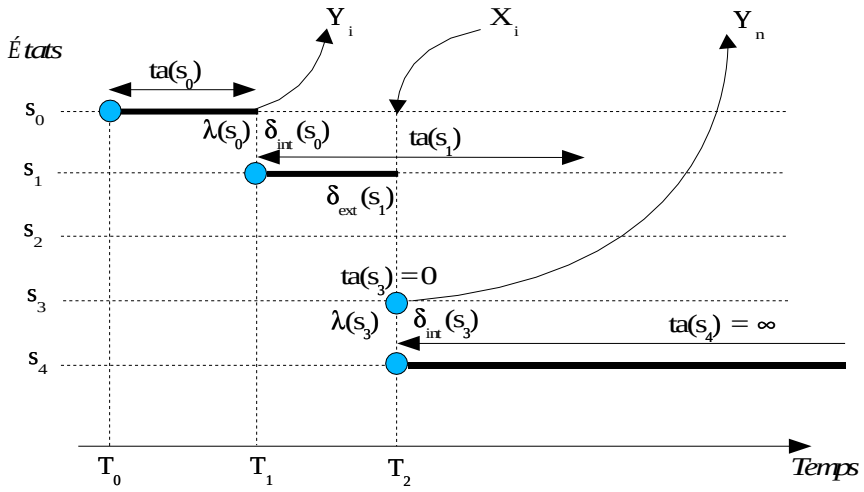

FIG. 3.2 – Exemple de graphe de transitions d'un modèle DEVS atomique. Les lignes pointillées verticales représentent les dates d'occurrences d'évènements. Les cercles pleins représentent l'état courant du système et les lignes pleines horizontales l'avancement du temps. Une transition est marquée par le passage d'un niveau à un autre sur la verticale (voir le texte pour le déroulement du scénario).

$T_1 = T_0 + ta(s_0)$, aucune entrée n'a eu lieu. La fonction de sortie $\lambda(s_0)$ est donc activée et Y_i prend pour valeur la valeur produite par l'évaluation de cette fonction. Après avoir affecté les ports de sortie, la fonction de transition interne δ_{int} est appliquée. Le système passe dans l'état $s_1 = \delta_{int}(s_0)$ et changera d'état à $T_1 + ta(s_1)$. À l'instant $T_2 < T_1 + ta(s_1)$, un évènement externe arrive en entrée sur le port X_i. Il est alors fait appel à la fonction de transition externe pour déterminer le nouvel état. Dans ce cas, la fonction de sortie n'est pas appliquée : elle n'est appliquée que lors d'une transition interne. À l'instant T_2, le système passe dans l'état $s_3 = \delta_{ext}((s_1, e), v)$ avec $e = T_2 - T_1$ et v la valeur de l'évènement attaché au port X_i. Ici, c'est bien la fonction de transition externe qui détermine le changement d'état. On suppose que s_3 est transitoire ($ta(s_3) = 0$). Il y a évaluation immédiate de la fonction de sortie $\lambda(s_3)$ qui donne une valeur au port de sortie Y_n. Cette évaluation est instantanément suivie par celle de $s_4 = \delta_{int}(s_3)$. Le nouvel état s_4 est passif ($ta(s_4) = \infty$).

Cet exemple[39] nous permet également d'introduire une solution adoptée pour permettre l'émission d'un évènement sur un port de sortie au même instant que la réception d'un évènement externe. En effet, comme nous l'avons dit plus haut, la fonction de sortie λ est évaluée uniquement avant une transition interne, ce qui interdit toute émission d'évènement sur ce type de transition. En introduisant un état transitoire, nous pouvons spécifier une transition interne instantanée sur cet état, ce qui permet l'émission d'un évènement. « Tout se passe comme si » l'émission de l'évènement avait lieu sur une transition externe. Nous verrons par la suite l'utilité d'une telle méthode.

Comme l'indique le type des modèles DEVS ainsi spécifiés (atomiques), il est possible de

[39] Au regard de cet exemple, nous pouvons nous demander ce qui se passe si une fonction de transition interne et une fonction de transition externe sont activées exactement au même instant. Pour répondre à cette question, DEVS permet l'expression d'un fonction de gestion des conflits, δ_{con}, qui définit la transition qui sera activée. Nous n'utiliserons pas cette fonction dans notre spécification.

composer des modèles formés d'un ensemble de sous-modèles. C'est la connexion de l'ensemble des ports des modèles atomiques entre eux qui permet cette composition. Ainsi, il est possible de formaliser le couplage de modèles DEVS. C'est ce que nous présentons dans ce qui suit.

3.2.2 DEVS couplé

Un modèle DEVS couplé définit comment coupler un ensemble de modèles DEVS entre eux pour former un nouveau modèle. Le modèle couplé ainsi construit introduit la notion de modularité dans la formalisation DEVS. Il est ainsi possible de modifier les connexions entre modèles composants ou de remplacer un composant par un autre. De plus, le modèle couplé peut lui-même faire partie d'un autre modèle couplé. Ceci permet une construction hiérarchique des modèles. Cette décomposition hiérarchique reflète une vision réductionniste indissociable de l'activité du modélisateur qui doit identifier l'ensemble des composants élémentaires de son système. Néanmoins, c'est un réductionnisme faible [Atl86] dans le sens où le modélisateur doit tenir compte des interactions entre les modèles en définissant les connexions. La figure 3.3 montre un modèle couplé (N) et ses modèles composants (A et B) ainsi que différentes connexions entre ces modèles *via* des ports.

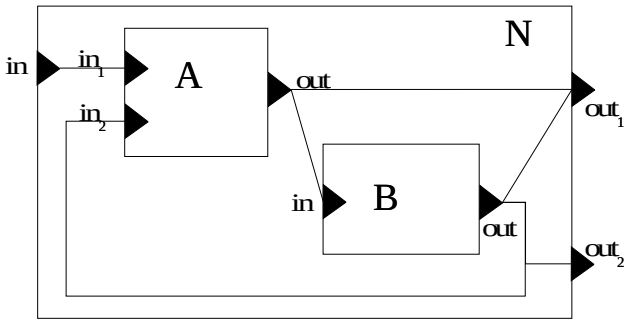

FIG. 3.3 – Représentation graphique d'un modèle DEVS couplé. Le nom des modèles est en lettres capitales. Le nom des ports est en minuscules.

Un modèle couplé comprend les informations suivantes :
- l'ensemble des modèles qui le composent,
- l'ensemble des ports d'entrée qui recevront les évènements externes,
- l'ensemble des ports de sortie qui émettront les évènements,
- les couplages entre ports d'entrée et ports de sortie des modèles composant le modèle couplé.

Un modèle couplé, aussi appelé réseau de modèles, possède la structure suivante :

$$N =< X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC >$$

La définition de X et Y est identique à celle d'un modèle atomique. Les entrées et sorties sont composées de ports, chaque port peut prendre des valeurs ; chaque port possède son propre domaine de valeurs. D est l'ensemble des noms de modèles intervenant dans le modèle couplé. M_d est un modèle DEVS. Les variables représentant les entrées et les sorties des modèles composants sont ici indexées par l'identifiant du modèle pour les différencier de X et Y du modèle couplé.

$$M_d =< X_d, Y_d, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

Les entrées et les sorties du modèle couplé sont connectées à certaines entrées et sorties des modèles composants. Trois types de connexions sont possibles. Les *External Input Connexions, (EIC)* qui s'écrivent :

$$EIC = \{((N, a), (d, b))|a \in IPorts_N,\ b \in IPorts_d,\ d \in D\}$$

avec $IPorts_N$ l'ensemble des ports d'entrée du modèle couplé et $IPorts_d$ l'ensemble des ports d'entrée du modèle M_d. Autrement dit, c'est l'ensemble des ports d'entrée du modèle couplé associés aux ports d'entrée des modèles composants auxquels ils sont connectés.

De même, les *External Output Connections, (EOC)* qui définissent comment l'ensemble des sorties des modèles composants est associé à l'ensemble des sorties du modèle composé, sont définies comme suit :

$$EOC = \{((d, b), (N, a))|a \in OPorts_N,\ b \in OPorts_d,\ d \in D\}$$

avec $OPorts_N$ l'ensemble des ports de sortie du modèle couplé et $OPorts_d$ l'ensemble des ports de sortie du modèle M_d.

À l'intérieur du modèle couplé, les sorties d'un modèle composant peuvent être couplées aux entrées des autres modèles composants. Une sortie d'un modèle ne peut pas être couplée à l'une de ses propres entrées. Les *Internal Connections, (IC)* sont définies comme suit :

$$IC = \{((i, a), (j, b))|i, j \in D,\ i \neq j,\ a \in OPorts_i,\ b \in IPorts_j\}$$

Nous allons maintenant illustrer le formalisme DEVS couplé en nous basant sur la figure 3.3. Soit N un modèle DEVS formalisé à l'aide de la structure suivante :

$$N =< X, Y, D, \{M_A, M_B\}EIC, EOC, IC >$$

avec :

$X = \{(p, v)|p \in IPorts_N, v \in V\}$ l'ensemble des couples port-valeur d'entrée
$Y = \{(p, v)|p \in OPorts_N, v \in V\}$ l'ensemble des couples port-valeur de sortie
$IPorts = \{in\}$ le nom du port d'entrée
$OPorts = \{out_1, out_2\}$ les noms des ports de sortie
$D = \{A, B\}$ les noms des modèles composants
M_A un modèle DEVS atomique
M_B un modèle DEVS atomique
$EIC = \{((N, in), (A, in_1))\}$ l'ensemble des connexions en entrée du modèle couplé
$EOC = \{\ ((B, out), (N, out_1)),\ ((B, out), (N, out_2)),\ ((A, out), (N, out_1))\ \}$l'ensemble des connexions en sortie du modèle couplé
$IC = \{((A, out), (B, in)), ((B, out), (A, in_2))\}$ l'ensemble des connexions internes

À partir de cette spécification, il est possible de définir des ensembles qui décrivent le modèle DEVS couplé de façon plus précise. Ces ensembles sont en fait l'union ou le produit cartésien de certains ensembles des modèles composants. Par exemple, l'ensemble des états séquentiels du modèle couplé est composé du produit cartésien des vecteurs d'états des modèles composants et s'écrit :

$$S_N = \overset{d \in D}{\times}\ S_d$$

La fonction d'avancement du temps ta du modèle couplé s'écrit :

$ta(s_N) = min\{\sigma_d|d \in D\}$ avec $\sigma_d = ta(s_d) - e_d$ le temps restant jusqu'à la prochaine transition s_N, i.e. la date minimale à laquelle l'un des modèles composants effectuera sa prochaine transition.

Nous pouvons également spécifier l'ensemble des valeurs de sortie du modèle couplé en fonction des valeurs de sortie des modèles composants. Ces valeurs sont celles des ports qui lui sont connectés, c'est-à-dire des ports de sortie des modèles composants. Nous pouvons définir l'ensemble des valeurs de sortie du modèle couplé V_N comme suit :

$$V_N = \{\ v \mid (p,v) \in Y_N,\ (p\prime,v) \in Y_d,\ d \in D,\\ p \in OPorts_N,\ p\prime \in OPorts_d,\ \{(d,p\prime),(N,p)\} \in EOC\ \}$$

Nous pourrions faire de même avec les valeurs d'entrée du modèle couplé. Le but ici est de montrer qu'il est possible de formaliser des sous-ensembles dans un réseau de modèles. Nous utiliserons un peu plus loin une telle approche pour établir une correspondance entre DEVS et les SMAs.

3.2.3 Importance de DEVS pour la suite

Nous venons de présenter le formalisme DEVS dans une perspective particulière, celle de spécifier un modèle d'agents réactifs situés afin de coupler formellement ce modèle à un système d'équations différentielles. Nous retiendrons ici les principales caractéristiques de ce formalisme :

1. DEVS est un formalisme abstrait indépendant de l'implémentation,
2. il offre une vision modulaire et hiérarchique des systèmes dynamiques,
3. les évènements attachés aux ports peuvent prendre des valeurs dans divers domaines ($(p,v) \in \mathbb{R},\mathbb{N},\mathbb{C}$, *String* etc...), ce sont les fonctions de transitions externes attachées à ces ports qui manipulent les évènements,
4. les fonctions de transitions internes formalisent le comportement autonome du modèle,
5. la notion de temps est centrale dans DEVS : c'est la fonction d'avancement du temps ta qui détermine la dynamique du système.

DEVS semble très proche du formalisme des Automates à états finis et à Registres[40] (AR). En fait, deux caractéristiques principales différencient DEVS des ARs :

1. la fonction de transition entre états des AR est décomposée en deux. Ainsi, comme nous l'avons dit plus haut, la formalisation du comportement interne du modèle est bien différenciée de celle des évolutions dues aux évènements externes,
2. la notion de couplage hiérarchique, impliquant la modularité et une vision multi-niveaux d'un système.

Comme pour tout formalisme, nous attendons de DEVS des possibilités de simplification d'écriture ou de déduction de propriétés sur les systèmes formalisés. La plus intéressante des propriétés est celle de « fermeture sous couplage ». Cette propriété garantit qu'un modèle DEVS couplé est rigoureusement équivalent à un modèle DEVS atomique appelé « résultant » en terme de comportement dynamique (séquences d'états). Le lecteur intéressé par la démonstration peut se référer à [ZKP00].

3.3 DEVS : une sémantique opérationnelle

DEVS est un formalisme abstrait. Le passage de la formalisation à l'implémentation se fait par l'intermédiaire d'algorithmes mis au point par B.P. Zeigler pour ce qui concerne les modèles

[40] Ce formalisme est décrit dans les ouvrages traitant de l'architecture des ordinateurs.

atomiques ou couplés et pour d'autres formes de DEVS [ZKP00]. Ces algorithmes sont appelés « simulateurs abstraits » et font de DEVS une sémantique opérationnelle, c'est-à-dire implémentable sans ambiguïté sur un ordinateur.

Tel que nous l'avons présenté, le formalisme DEVS ne dit rien sur l'initialisation des modèles. Nous pouvons également noter que DEVS ne dit rien sur ce qui déclenche les transitions internes (les états eux-mêmes ?). Ce sont en fait des caractéristiques de l'algorithme de simulation et c'est donc le simulateur abstrait qui se charge de les implémenter.

Dans un contexte de modèle couplé, le simulateur abstrait d'un modèle DEVS atomique peut être vu comme une boîte (figure 3.4) acceptant en entrée trois types d'évènements et générant un évènement de sortie.

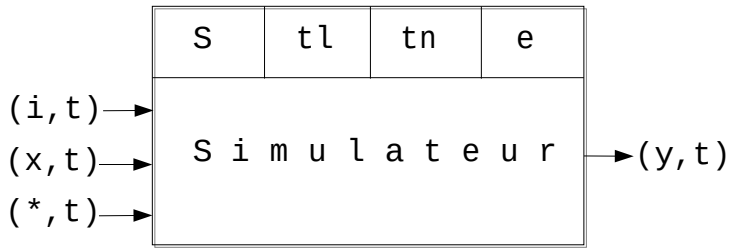

FIG. 3.4 – Le simulateur abstrait d'un modèle atomique. Il correspond à l'algorithme de simulation de la dynamique d'un modèle DEVS atomique. Les évènements d'entrée et de sortie sont décrits dans le texte et l'algorithme est donné en annexe D.1

Les évènements d'entrée-sortie notés sur la figure 3.4 sont les suivants[41] :

- évènement d'initialisation `(i,t)` :
  l'état `s` du modèle est initialisé à l'instant `t`, la date `tn` de la prochaine transition interne est calculée à l'aide de `ta(s)`,
- évènement externe `(x,t)` :
  un autre modèle envoie un évènement à la date `t`, le modèle est dans l'état `s` depuis `tl` (la durée `e` écoulée dans l'état `s` est `e=t-tl`) et devrait changer d'état à `tn` sur une transition interne. Le simulateur abstrait traite l'évènement en calculant le prochain état déterminé par `Trans_ext(s)` (δ_{ext}),
- évènement interne `(*,t)` :
  le modèle a atteint la date de fin d'état courant `s`, le modèle change d'état selon sa fonction de transition interne `Trans_int(s)` (δ_{int}),
- évènement de sortie `(y,t)` :
  le modèle génére un évènement de sortie à la date `t` ayant pour valeur `Fonct_Sortie(s)` ($\lambda(s)$).

Nous donnons en annexe D.1 l'algorithme du simulateur abstrait d'un modèle atomique. Cet algorithme fournit un cadre général. Les fonctions de transition et d'avancement du temps sont caractéristiques du modèle considéré. Le passage de la spécification formelle DEVS à l'algorithme

[41] Nous utilisons les `caractères d'imprimerie` pour différencier les évènements, variables et fonctions manipulés par le simulateur de ces mêmes éléments du modèle formel.

de simulation du modèle passe donc par l'implémentation de ces fonctions. Il apparaît clairement ici que ces fonctions ne sont pas spécifiées dans un formalisme particulier, elles peuvent être analytiques, logiques, voire exprimées sous la forme d'un algorithme. DEVS apparaît comme une « capsule formelle » qui garantit le comportement en terme de dynamique de changement d'état. Toutefois, il n'est pas possible de valider l'ensemble des règles de transitions, elles relèvent de la responsabilité du modélisateur.

Comme nous l'avons dit plus haut, DEVS offre une vision hiérarchique des modèles couplés. Le simulateur abstrait d'un modèle atomique est donc couplé avec d'autres simulateurs. C'est le simulateur abstrait du modèle couplé qui a la responsabilité de gérer l'ensemble des connexions entre modèles et donc le routage des évènements dans le réseau de modèles. Le simulateur abstrait d'un modèle couplé est appelé coordinateur ; il s'inscrit lui-même dans une hiérarchie de coordinateurs couplés. Il est possible de représenter cette hiérachisation sous la forme d'un arbre dont la racine est appelée « coordinateur racine ». La figure 3.5 présente une telle hiérarchie en regard de celle des spécifications formelles DEVS.

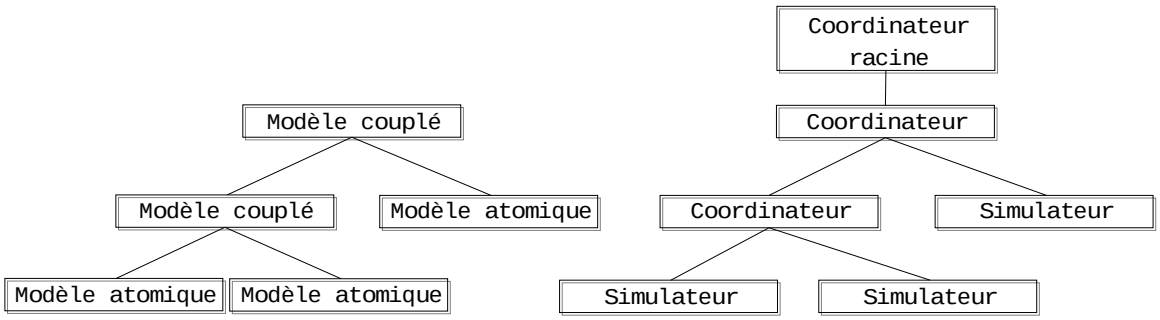

FIG. 3.5 – Traduction d'un modèle DEVS couplé dans sa hiérarchie de simulateurs abstraits. À gauche figure la hiérarchie des modèles atomiques DEVS et à droite la hiérarchie des simulateurs et coordinateurs associés. Chaque coordinateur correspond au simulateur abstrait d'un modèle DEVS couplé. L'algorithme d'un coordinateur est donné en annexe D.2 et celui du coordinateur racine, qui correspond à la boucle générale de simulation, en annexe D.3(D'après [ZKP00])

Les coordinateurs sont qualifiés de « parents » des simulateurs (qui sont donc les « enfants ») par analogie à la représentation en arbre donnée par la figure 3.5. Le coordinateur reçoit et génére les mêmes types d'évènements que le simulateur ; il envoie les évènements à tous ces enfants (simulateurs et coordinateurs) et à son coordinateur parent. Nous pouvons décomposer l'algorithme de simulation d'un coordinateur en quatre phases (il est donné en annexe D.2) :

- l'initialisation :
  le coordinateur transmet l'évènement d'initialisation à tous ses enfants,
- la réception d'un évènement externe en entrée :
  cet évènement est envoyé à tous les simulateurs enfants appartenant à EIC,
- la réception d'un évènement de sortie provenant d'un enfant :
  cet évènement est envoyé soit aux enfants connectés entre eux, et donc appartenant à IC, sous la forme d'un évènement externe, soit au parent conformément à l'ensemble EOC, sous la forme d'un évènement de sortie,
- la réception d'un évènement de transition interne :
  cet évènement est transmis à l'enfant correspondant.

Il existe d'autres simulateurs abstraits correspondant à des modèles DEVS particuliers (comme les modèles parallèles [ZKP00]) et pour des extensions de DEVS (voir par exemple [Bar96]). La présentation de ces deux simulateurs abstraits permet de mieux comprendre la dynamique opérationnelle d'un modèle DEVS. L'implémentation de ces simulateurs abstraits est facilitée par une approche objet. En effet, DEVS se traduit assez naturellement dans un contexte de modélisation orientée objets tel que D. Hill le décrit [Hil96]. Le concept de composition s'illustre par celui de modèle hiérarchique en DEVS. Ainsi, un modèle couplé est composé (au sens des objets) d'autres modèles. Les notions d'héritage et de polymorphisme permettent de généraliser des structures comme les ports de modèles atomiques ou couplés. Il existe de nombreuses implémentations de simulateurs DEVS dans des langages orientés objets, par exemple J-DEVS [FCB02], une implémentation en Java, ou encore des *frameworks* basés sur une approche objet pour la simulation dans l'industrie [NMZ98][42].

Dans ce chapitre, notre objectif est de coupler deux modèles formels hétérogènes. Après avoir présenté le formalisme DEVS, nous allons montrer comment le paradigme d'agents réactifs situés peut être spécifié dans ce formalisme. Nous allons donc commencer par discuter d'une analogie entre DEVS et les agents réactifs. Nous proposons ensuite une formalisation DEVS du couplage des deux modèles décrits au paragraphe 2.3 page 36. Puis, en nous basant sur les simulateurs abstraits que nous venons de présenter, nous discutons de l'implémentation de ce modèle couplé.

3.4 Du paradigme d'agents réactifs situés vers le formalisme DEVS

Nous nous intéressons ici aux systèmes d'agents réactifs situés que nous pouvons généralement considérer comme des systèmes dynamiques. J. Odell [Ode02] le dit même des agents en général : « *... les agents sont dynamiques car ils peuvent exercer un certain degré d'activité* ». Il donc possible d'ordonner les actions ou activités des agents dans le temps, même quand celui-ci n'est pas explicite dans la modélisation. DEVS étant lui-même une formalisation des systèmes dynamiques, il est possible d'utiliser ce formalisme pour la spécification des systèmes d'agents réactifs.
L'idée d'une formalisation à évènements discrets pour les SMAs est pratiquement contemporaine à leur apparition [Fer95] [KB94]. De fait, DEVS est apparu comme un bon candidat pour leur spécification. C'est au milieu des années 90, avec essentiellement les travaux de A.M. Uhrmacher [UA94] [US98], que les premiers articles proposant une formalisation DEVS des SMAs sont apparus. Ces travaux ont commencé par mettre en évidence la relation entre l'individualité d'un agent et la structure d'un modèle DEVS, couplé ou atomique. Ainsi, ils décrivent les fonctions de transitions internes comme formalisant le comportement autonome de l'agent et les fonctions de transitions externes comme formalisant la perception. Naturellement, les fonctions de sortie peuvent alors être vues comme les actions des agents sur l'environnement ou sur les autres agents. Les avancées postérieures permettent également la formalisation d'agents cognitifs. Nous revenons sur ce point dans la discussion de ce chapitre.

Néanmoins, l'utilisation de DEVS pour la formalisation des SMAs est peu ou pas reconnue. Ceci est peut-être lié à la séparation des communautés de ceux qui font de la simulation, notamment

[42]Un grand nombre d'implémentations de simulateurs sont disponibles sur le site `http://www.sce.carleton.ca/faculty/wainer/standard/`

de systèmes artificiels, de celles des modélisateurs utilisant le paradigme agent. Il y a donc un effort de transversalité à effectuer, DEVS pouvant enrichir l'ensemble des techniques et outils déjà disponibles pour la simulation centrée agents. Nous voulons faire ici un premier pas dans cette direction en proposant une formalisation DEVS des notions manipulées par les SMAs. En effet, dans les travaux que nous connaissons, nous n'avons pas trouvé une telle démarche. Nous nous bornons ici aux agents réactifs situés. Nous reviendrons dans la discussion sur de possibles extensions aux autres types d'agents. Nous considérons donc ici les notions essentielles dans la formalisation d'un SMA réactif, à savoir :

- les agents,
- leurs perceptions,
- leurs actions,
- leur autonomie,
- l'environnement,
- le SMA lui-même.

L'action peut être soit proactive soit réactive. Dans le premier cas, elle correspond à une action autonome de l'agent. Dans le deuxième cas, elle correspond à la réponse à un stimulus. Nous définissons donc la proactivité comme les actions associées à un comportement autonome et la réactivité comme les actions en réponse aux perceptions.

3.4.1 Formalisation d'un agent réactif en DEVS

L'agent

L'agent réactif peut être vu comme une entité autonome qui perçoit son environnement et agit sur lui ou sur d'autres agents de façon « réflexe ». De cette « définition » extrêmement minimaliste, nous pouvons tirer deux caractéristiques essentielles :

1. l'agent a une individualité, un comportement propre,
2. il est capable d'interagir.

Nous pouvons déjà considérer un modèle DEVS comme un agent [UA94]. En effet, un modèle DEVS a un nom et une structure particulière qui définissent son identité et ses « frontières avec l'extérieur ». Les ports d'entrée et de sortie représentent respectivement les récepteurs et effecteurs de l'agent. Les fonctions de transitions externes gouvernent les changements d'états de l'agent liés à l'arrivée de stimuli, les fonctions de sorties définissent les actions de l'agent, et les fonctions de transitions internes correspondent aux comportements autonomes. Néanmoins, les agents sont rarement si simples. Aussi, un modèle atomique ne peut pas suffir à leur formalisation. Ainsi nous postulons qu'un agent réactif est, dans la plupart des cas, un modèle DEVS couplé.
Le comportement et les interactions d'un agent peuvent être très complexes. Les principes de modularité et de décomposition hiérarchique énoncés dans notre présentation de DEVS nous permettent d'adopter un niveau de détail suffisant pour exprimer cette complexité. Ainsi, il peut y avoir plusieurs modèles DEVS qui formalisent l'interaction et le comportement. Cela semble évident pour l'interaction qui peut se décomposer en perception et action. Nous donnons à un agent réactif la structure d'un modèle DEVS couplé suivante :

$$Agent =< X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC >$$

Où, comme pour un modèle DEVS « classique » :
Agent désigne le modèle DEVS couplé d'un agent

$X = \{(p, v) | p \in IPorts, v \in X_p\}$ est l'ensemble des récepteurs et des valeurs qu'ils peuvent prendre
$Y = \{(p, v) | p \in OPorts, v \in Y_p\}$ est l'ensemble des effecteurs et des valeurs qu'ils peuvent prendre
D est l'ensemble des noms des modèles atomiques ou couplés composants l'agent
M_D est l'ensemble des modèles atomiques ou couplés composants l'agent (cf. formalisation au paragraphe 3.2.1et 3.2.2).

avec en particulier :

$EIC = \{((Agent, a), (d, b)) | a \in IPorts_{Agent},\ b \in IPorts_d\}$
$EOC = \{((d, b), (Agent, a)) | a \in OPorts_{Agent},\ b \in OPorts_d\}$
$IC = \{((i, a), (j, b)) | i, j \in D,\ i \neq j,\ a \in OPorts_i,\ b \in IPorts_j\}$

L'ensemble EIC définit les ports du modèle d'agent qui sont connectés aux ports des modèles composants qui reçoivent des évènements externes. Cet ensemble permet donc de définir les modèles composants qui participent au système de perception de l'agent. De même, l'ensemble EOC définit les ports du modèle d'agent connectés aux ports des modèles composants qui émettront des évènements. Cet ensemble permet de définir les modèles composants qui participeront aux actions de l'agent. Néanmoins, action et perception impliquent des changements d'états de l'agent, il est donc nécessaire de compléter leur formalisation par les fonctions de transitions gouvernant ces changements d'états. De même, la notion d'autonomie est complexe et nécessite la prise en compte de ces fonctions.

La perception

La perception est le processus par lequel l'agent acquiert des informations sur le monde qui l'entoure. La perception est représentée en DEVS par l'arrivée d'un évènement externe dans un modèle d'agent, entraînant un changement d'état dans un ou plusieurs modèles composants. Soit A un agent formalisé avec un DEVS couplé, la perception P_A de A est définie comme l'ensemble des fonctions de transitions externes tel que :

$P_A = \{\delta_{ext_d} \mid \delta_{ext_d}(S_d, (v, ip_d))\}$
avec :
$(d, ip_d) \in EIC$ où ip_d est un port d'entrée au modèle $d \in D$,
et D est l'ensemble des modèles composants.

Comme nous l'avons vu au paragraphe 3.2.1, tout évènement externe implique un changement d'état. Nous considérons donc la perception comme un changement de l'état interne de l'agent (i.e. $S_A = \overset{d \in D}{\times} Q_d$) dû à un *stimulus* externe. Du fait que le domaine des valeurs prises par les ports d'entrée n'est pas restreint à un domaine particulier, nous pouvons considérer tout type de messages (*i.e.* évènements externes) en entrée du modèle agent comme un stimulus. La perception est vue ici comme la réception passive d'un évènement provenant de l'extérieur de l'agent. Dans certains cas, cette réception est liée à une action antérieure de l'agent qui a « interrogé » son environnement, par exemple dans le cas d'agents situés qui veulent connaître les agents voisins à un instant donné.

L'action, la proaction, la réaction

Dans un SMA réactif, les actions des agents sont des « influences » sur l'environnement ou d'autres agents. Elles peuvent être de différentes natures : déplacement, destruction ou modifica-

tion d'un objet de l'environnement, dépôt d'informations etc. L'action d'un agent peut se définir comme une « influence » sur l'extérieur. Nous pouvons représenter cette influence en DEVS par un évènement généré par l'agent, c'est-à-dire l'émission d'un évènement externe par une fonction de sortie.
Soit A un agent formalisé avec un DEVS couplé, l'action F_A de A est définie comme l'ensemble des fonctions de sortie tel que :

$F_A = \{\lambda_d \mid \lambda_d(S_d) = (v, op_d)\}$
avec :
$(d, op_d) \in EOC$ où op_d est un port de sortie du modèle $d \in D$,
et D est l'ensemble des modèles composant l'agent.

La notion de modèle proactif existe en DEVS. Elle désigne un modèle atomique ou couplé qui a uniquement des ports de sortie (*i.e.* $X = \emptyset$ et $Y \neq \emptyset$). Ainsi, son comportement est totalement contrôlé par les fonctions de transitions internes. La proactivité peut donc être vue dans un modèle DEVS couplé d'agent comme l'ensemble des fonctions de sortie qui ont pour origine un modèle proactif. Soit M_p l'ensemble des modèles proactifs composant un modèle couplé A d'agent, nous pouvons formaliser la proactivité W_A comme suit :

$W_A = \{\lambda_{d_p} \mid \lambda_{d_p}(S_{d_p}) = (v, op_{d_p})\}$
avec :
$(d_p, op_d) \in EOC$ où op_d est un port de sortie du modèle,
et $d_p \in M_p$ et $M_p \in D$ est l'ensemble des modèles proactifs composant l'agent.

Cette définition n'est pas totale (ou complète). En effet, un modèle proactif peut entraîner un changement d'état dans un autre modèle composant l'agent. Une conséquence peut être que ce dernier génère une fonction de sortie liée à l'arrivée d'un évènement externe provenant du modèle proactif. Ainsi, il peut y avoir un nombre de transitions quelconque, dans un nombre également quelconque de modèles composants, entre l'émission d'une valeur de sortie par le modèle proactif et l'émission d'une valeur de sortie par le modèle agent. Il y a même plus embarrassant. Si nous considérons la dynamique du système pour définir la proactivité, une même fonction de sortie peut avoir pour origine un modèle proactif ou non. Par exemple, un déplacement physique d'un agent peut être soit une action proactive, soit réactive : cela dépend du contexte dynamique et environnemental de l'agent. La réactivité se heurte aux mêmes difficultés. Néanmoins, comme pour la proactivité, nous pouvons donner une définition minimale de la réactivité d'un agent. Soit A un modèle d'agent, une partie des actions réactives d'un agent peuvent être formalisées comme étant l'ensemble des fonctions de sortie λ_M associées au modèle couplé de l'agent et aux états transitoires des modèles composants. En effet, comme nous l'avons dit au paragraphe 3.2.1 page 53, les états transitoires permettent de simuler l'émission d'une fonction de sortie sur une transition externe. Cette réaction à un évènement d'entrée peut être assimilée à la réactivité R_A de A telle que :

$R_A = \{\lambda_d \mid \lambda_d(s_d) = (v, op_d),\ ta(s_d) = 0\}$
avec :
$s_d \in S_d \mid d \in D - \{M_p\}$ l'ensemble des états des modèles composants non proactifs,
$(d, op) \in EOC$ où op_d est un port de sortie du modèle $d_p \in D - M_p$,
et D est l'ensemble des modèles composant l'agent.

L'action et la réaction sont considérées ici dans leur définition minimale comme une interface fonctionnelle de l'agent vis-à-vis de l'extérieur.

L'autonomie

Les concepts d'action et réaction sont à mettre en relation avec celui d'autonomie, qui est plus que l'ensemble des transitions internes dans un modèle DEVS. En effet, le comportement proactif d'un agent peut être la conséquence d'une « prise de décision » autonome[43] dans un modèle proactif qui envoie des évènements vers des modèle composants. Ainsi, l'ensemble des fonctions des modèles composants qui ne sont pas reliées aux ports d'entrée ou de sortie du modèle couplé de l'agent définissent l'autonomie de l'agent.
Considérant un agent A comme un DEVS couplé, toutes les fonctions de transition externe ne recevant pas d'évènement lié à un port d'entrée du modèle couplé et toute les fonctions de transition interne des modèles composants le modèle couplé définissent le comportement autonome O de l'agent A. Ainsi :

$$O_A = \{\delta_{int_d} \cup \delta_{ext_d} \mid \delta_{ext_d}(S_d, (v, ip))\}$$

avec :
$(v, ip_d) \notin IOC$ où op_d est un port d'entrée et v sa valeur,
$d \in D$ l'ensemble des modèles composants,
S_d est l'ensemble des états des modèles composants $d \in D$.

La perception, l'action et l'autonomie formalisent le comportement C_A (au sens éthologique du terme[44]) de l'agent réactif A et peut s'écrire :

$$C_A = F_A \cup P_A \cup O_A$$

Cette spécification de comportement revient à dire que toutes les fonctions de transitions définies dans les modèles composant l'agent réactif spécifient son comportement. Seules les fonctions d'avancement du temps (i.e. ta) n'apparaissent pas ici. Ces fonctions définissent la dynamique de l'agent, c'est-à-dire le temps nécessaire à l'agent pour percevoir et agir. Nous ne faisons donc pas usage de la structure du modèle couplé pour définir le comportement de l'agent. Comme nous l'avons dit plus haut, plusieurs modèles composants peuvent intervenir dans la spécification de la perception par exemple. De plus, ces modèles composants ne sont pas oligatoirement couplés pour définir un modèle composé qui encapsulerait toute la spécification de la perception. Ainsi, la structure du modèle de l'agent réactif peut être très variable et son utilisation dans la spécification du comportement spécifique à un agent particulier. Nous pouvons tout de même dire que les trois ensembles IOC, EOC, IC de cette structure formalisent la topologie des interactions ou communications (qui est connecté avec qui ?) pour un agent particulier.
Les messages échangés entre les agents ou entre les agents et l'environnement correspondent aux évènements eux-mêmes. Ainsi, la valeur v prise par les ports des modèles DEVS correspond à la valeur des messages échangés entre modèles couplés. Nous dirons donc que les évènements sont le support des communications dans le système, c'est-à-dire le « véhicule » supportant la valeur des messages.
Un reproche que nous pouvons faire à DEVS est que ce formalisme fixe la structure une fois pour toutes. S'il est nécessaire de représenter des modifications dynamiques de comportement d'un agent A, il doit être possible de modifier l'ensemble C_A (l'aspect fonctionnel de l'agent) et A lui-même (l'aspect structurel de l'agent). DEVS tel que nous l'avons présenté n'offre pas cette possibilité, néanmoins il existe des extensions de DEVS qui permettent de formaliser des chan-

[43] Même si cette prise de décision est très limitée dans un agent réactif.

[44] Ensemble des activités des êtres vivants et de leurs réactions physiologiques aux conditions de leur milieu. Définition du dictionnaire de l'Académie française

gements de structures et de compositions dynamiques [Bar96]. Nous présentons une application des changements de structures pour le couplage du modèle d'agents réactifs du copépode avec un système d'équations différentielles dans la suite de ce chapitre. Nous reviendrons dans la discussion sur l'utilité des changements dynamiques de structures dans les SMAs.

À partir des définitions que nous venons de donner, nous élaborons un tableau d'analogies[45] entre le paradigme d'agent réactif et le formalisme DEVS (tableau 3.1). Nous différencions explicitement la structure et le comportement. Cette approche est celle généralement utilisée pour la formalisation des SMAs (voir discussion). Dans ce tableau, nous introduisons l'ensemble des états comme faisant partie de la structure de l'agent. Comme nous l'avons dit plus haut, ce sont les fonctions de transitions appliquées sur ces états qui définissent le comportement.

3.4.2 Formalisation de l'environnement

L'environnement est une composante essentielle des SMAs[46]. Dans le cas de SMAs situés, nous pouvons lui prêter les caractéristiques minimales suivantes :

- il constitue le référentiel des agents, il possède donc une métrique ;
- il définit un espace possédant généralement des limites ou des frontières ;
- il peut contenir des entités passives (sans comportements) ou des informations ;
- il est le siège des interactions et des communications indirectes.

Le dernier point correspond par exemple aux traces volatiles déposées par des agents au cours de leur déplacement dans l'environnement. Ces traces peuvent alors être perçues par d'autres agents qui modifieront leur comportement en conséquence. Cette représentation indirecte des interactions peut mener à des comportements émergents de l'ensemble des agents [Dro93].
Du fait que les agents ont une perception limitée de l'environnement [Fer95], celui-ci conditionnne les interactions entre agents situés, donc repérés dans cet environnement. Cette représentation de l'espace est une des avancées majeures apportées par les SMAs. En effet, elle rend possible la modélisation d'interactions discrètes, locales et ponctuelles. Ce type d'interactions est très difficile à représenter avec des équations différentielles par exemple.

L'environnement peut être modélisé de différentes façons. Généralement, trois cas sont distingués [Sou01] :

1. l'environnement centralisé,
2. l'environnement « distribué »,
3. l'environnement vu comme un agent.

Dans le premier cas, l'environnement est défini par une seule structure qui représente et contient tous les éléments de l'environnement. Les interactions des agents avec l'environnement sont donc représentées par des liaisons qui permettent aux agents « d'interroger » l'environnement et à ce dernier « de répondre ».
Dans le deuxième cas, l'environnement est représenté par un ensemble de cellules formant une grille. Chaque cellule peut être vue comme un environnement centralisé. Cette approche à l'avantage de pouvoir représenter des environnements très hétérogènes. Son inconvénient majeur est de contraindre la géométrie des interactions. Par exemple, la perception d'un agent sera limitée

[45] Rapport de ressemblance ou de correspondance que l'esprit perçoit entre deux êtres, deux objets ou deux séries. Définition du dictionnaire de l'Académie française.

[46] Mis à part les SMAs purement communiquant, où l'environnement n'est pas du tout représenté [Fer95].

TAB. 3.1 – Passage du paradigme d'agent réactif vers le formalisme DEVS

	Agents réactifs	***Spécification DEVS***
Structure	L'agent	Modèle DEVS couplé : A
	Composants de l'agent	Modèles DEVS atomiques ou couplés : $\{D\}$
	Relations entre les composants de l'agent et l'extérieur	Connexions externes d'entrée et de sortie : EIC et EOC
	Relation entre composants	Connexions internes : IC
	Récepteurs	Ports d'entrées du modèle couplé : $Iports$
	Effecteurs	Ports de sorties du modèle couplé : $Oports$
	Ensemble des états	Ensemble des états du modèle couplé : $S_N = \overset{d \in D}{\times} S_d$
	Supports de communication	Évènements : v, une valeur dans un domaine quelconque
Comportement	Actions	Fonctions de sortie : $\lambda \in F_A = W_A \cup R_A$
	Proaction	Fonctions de sortie : $\lambda \in W_A$
	Réaction	Fonctions de sortie : $\lambda \in R_A$
	Perception	Fonctions de transitions externes : $\delta_{ext} \in P_A$
	Autonomie	Fonctions de transitions internes et externes : $\delta_{ext} \cup \delta_{int} \in O_A$
	Dynamique	Fonctions d'avancement du temps : ta

à un nombre de cases, la zone perçue aura donc implicitement la forme de l'assemblage des cases. Cette représentation est celle adoptée par la majorité des SMAs [Sou01] et des plateformes de simulations orientées agents comme CORMAS [BBPP98] par exemple.
Dans le troisième cas, toutes les entités de l'environnement sont considérées commes des agents ainsi que l'entité qui leur sert de support. Cette approche a été initiée dans des plateformes de modélisation et simulation agents comme Swarm [Hie94] ou plus tard dans MadKit [GF98].

Dans un SMA, l'environnement peut avoir une dynamique propre, c'est-à-dire qu'il peut modifier ses propriétés, son état ou l'état des entités qu'il contient au cours du temps. Cette dynamique peut-être représentée sous forme d'automates à états finis dans un environnement centralisé, ou sous la forme d'automates cellulaires dans un environnement « distribué » par exemple.
La formalisation d'un environnement dynamique centralisé ou « distribué » est possible en DEVS en considérant chaque case de la grille comme un modèle atomique, l'environnement devenant alors un modèle couplé. En plus d'offrir les possibilités attendues de repérage dans l'espace, cette méthode permet d'exprimer des dynamiques propres à chaque cellule. Le lecteur intérressé par une telle formalisation peut se référer aux travaux de G. A. Wainer et N. Giambiasi [WG01] qui définissent CELL-DEVS comme une sémantique opérationnelle pour la modélisation de modèles cellulaires dynamiques.
Il est également possible de formaliser les trois types d'environnements cités plus haut en DEVS. Dans ce qui suit, nous donnons la formalisation DEVS d'un environnement centralisé. La plateforme JAMES développée par A.M. Uhrmacher [US98] implémente un environnement distribué formalisé avec DEVS. Dans cette architecture, les agents sont eux aussi des modèles DEVS. Le déplacement des agents sur la grille se fait par création/destruction des connexions entre modèles. Comme nous l'avons dit plus haut, la formalisation de tels changements de structure est possible avec une extension de DEVS que nous présentons plus loin.

Une autre approche concernant l'environnement a été proposée par J.C. Soulié : c'est l'approche multi-environnements [Sou01]. Cette technique consiste en l'utilisation de plusieurs environnements pour la représentation des différentes situations dans lesquelles un agent peut se situer, soit simultanément, soit successivement. Cette proposition est proche des représentations adoptées dans les Systèmes d'Informations Géographiques (SIG) qui représentent les données de terrains sur des cartes différentes selon la nature des données (couvertures végétales, urbanisation, hydrométrie etc.). Cette technique impose une gestion de l'intégrité des données et de la synchronisation lorsque l'agent est plongé dans plusieurs environnements au même moment. Néanmoins, elle revient à utiliser l'approche des environnements « distribués » citée plus haut. Là aussi, nous pensons que DEVS peut permettre la formalisation de tels environnements[47].

Dans ce qui suit, nous présentons la formalisation DEVS d'un environnement centralisé. Ce type d'environnement est celui que nous utilisons dans notre modèle du copépode. Il définit un espace continu dans lequel les agents se déplacent et agissent. Nous considérons l'environnement comme non proactif. Il doit cependant être capable de répondre à des questions posées par les agents comme « où suis-je ? » ou « qui sont mes voisins ? » etc. Il doit également être capable de stocker des données dynamiques comme la position des agents ou des entités présentes dans l'environnement. Une telle structure peut être formalisée par un modèle DEVS atomique comme

[47] J.C. Soulié est actuellement au Laboratoire d'Informatique du Littoral, comme l'auteur de cette thèse. Des interactions sont à venir...

suit :

$$Environnement =< X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

où :

X et Y sont respectivement l'ensemble des ports d'entrées et de sorties,

$S = \{(phase, E, M)\}$ est l'ensemble des états,

avec :

phase pouvant prendre la valeur $Idle$ ou $S_{X_i} \mid i = 1...n$, n étant le nombre de ports d'entrée du modèle,

$Idle$ est l'état passif. Chaque port d'entrée X_i correspond à une question possible pour l'environnement.

E est l'ensemble des entités passives du système (l'ensemble des coordonnées spatiales des objets),

M est l'ensemble des données concernant la métrique de l'espace (origine des axes, limites de l'espace s'il est borné),

$\delta_{ext} : (Idle, E, M) \times X_i \rightarrow (S_i, E, M)$ correspondent à la réception de la question par l'environnement,

$\lambda(S_i, E, M)) : (S_i, E, M) \rightarrow Y_i$ sont les réponses de l'environnement aux requêtes externes,

$\delta_{int} : (S_i, E, M) \rightarrow (Idle, E, M)$ est le retour à l'état passif après réponse à une question,

$ta(Idle, E, M) = \infty$, l'environnement est toujours en attente d'une question,

$ta(S_i, E, M) = 0$, les réponses aux questions sont instantanées.

Du fait que $ta(S_i, E, M) = 0$, il n'y a qu'une fonction de transition externe appliquée à l'état passif $(Idle, E, M)$. Ceci permet la formalisation d'interactions instantanées (de type question-réponse) entre les agents et leur environnement. Comme nous l'avons dit en introduction de ce chapitre, DEVS ne permet pas l'écriture de message asynchrone bloquant comme dans un diagramme de séquence UML par exemple. Nous devons le rendre explicite par l'introduction d'un état passif, toujours en attente d'un évènement externe. Cette décomposition apparaîtra de nouveau dans la formalisation de l'agent copépode.
L'état de l'environnement permet de stocker les entités E présentes dans l'environnement. Le choix de ce que représente E dépend du système formalisé. Dans le cas d'agents situés, E peut contenir la liste des objets de l'environnement avec leur position. E peut également contenir la position de tous les agents.

3.4.3 Formalisation d'un SMA orienté simulation

Un SMA peut se définir comme un ensemble d'agents plongés dans un ou plusieurs environnements. Les agents interagissent entre eux et avec l'environnement. Nous pouvons considérer un SMA comme un système composé de sous-systèmes en inter-relation. Comme pour un agent, nous posons qu'un SMA est forcément un modèle DEVS couplé qui contient l'environnement et les agents eux-mêmes. Un Système d'Agents Réactifs situés (SAR) avec environnement centralisé possède la structure suivante :

$$SAR =< D, \{M_d | d \in D\}, IC >$$

où :
$D = \{Agent_{1...n}, Environnement\}$ où n est le nombre d'agents,
M_d est l'ensemble des modèles DEVS des agents et le modèle de l'environnement,
IC est l'ensemble des connexions internes entre modèles composants du SAR ; elles représentent les relations entre agents et entre l'environnement et les agents,
$X = \emptyset$,
$Y = \emptyset$,
$EOC = \emptyset$,
$EIC = \emptyset$.

Le fait que $X = Y = EOC = EIC = \emptyset$ indique que le système est fermé. L'évolution d'un SAR est considérée ici comme totalement autonome. DEVS spécifie le comportement et la structure d'un modèle. Ce formalisme ne s'attache pas à la définition formelle de l'initialisation du modèle ou de variables observables en cours de simulation[48]. Ces deux aspects font parties du niveau opérationnel. Comme nous l'avons souligné plus haut, DEVS ne permet pas de modifier les relations entre modèles couplés, donc entre les agents et entre les agents et l'environnement. La structure présentée ici correspond donc à un SMA où les relations sont permanentes. Pour donner une vision plus globale des possibilités de DEVS, nous allons maintenant en présenter une extension que nous utilisons dans la formalisation de notre système couplé (paragraphe 3.6.2 page 83).

3.4.4 Formalisation du changement dynamique de structure d'un modèle

En 1985, G. Klir fut le premier à introduire la notion de structure dynamique dans un cadre de modélisation et simulation [Kli85]. Récemment, des approches formelles ont été proposées par A.M. Uhurmacher [Uhr01]. Mais c'est F. Barros, en introduisant le Dynamic Structure DEVS (DS-DEVS) [Bar96] qui donne la formalisation la plus satisfaisante pour les modèles à structure dynamique. Dans son travail, il montre que ce formalisme conserve toutes les propriétés de DEVS (notamment la modularité et la décomposition hiérarchique et la fermeture sous couplage). Il montre également qu'un DS-DEVS est équivalent à un DEVS couplé. Seulement, dans la pratique, il serait très difficile, voire impossible d'écrire ce modèle équivalent. Cette propriété garantit la possibilité de coupler un modèle DS-DEVS avec un modèle DEVS. F. Barros dit à propos des systèmes dynamiques en général :

> « ... de tels changements structuraux peuvent mieux reflèter les dynamiques réelles des systèmes modélisés dans lesquels des changements drastiques et évènementiels de comportements peuvent être observés. »

Cette affirmation est particulièrement pertinente dans le contexte des SMAs où les relations entre agents sont susceptibles de changer dynamiquement. Ainsi, DS-DEVS semble bien approprié à la formalisation des SMAs dans le contexte de la simulation.

DS-DEVS définit l'ensemble des connexions d'un modèle couplé comme étant un état du système. Ainsi, par transition interne ou externe, les connexions peuvent changer au cours du temps. De même, il est possible d'ajouter ou de faire disparaître des modèles DEVS. Ainsi F. Barros nous offre une perspective très intéressante pour la formulation des évolutions dynamiques des interactions dans les SMAs. Un modèle DS-DEVS définit donc un réseaux de modèles tel que :

[48] Cette formalisation est possible.

$$DSDEVN_{\Delta} =< X_{\Delta}, Y_{\Delta}, \chi, M_{\chi} >$$

avec :

Δ le nom du réseau de modèles que définit DS-DEVS,

X_{Δ} l'ensemble des couples ports-évènements d'entrée sur le réseau de modèles,

Y_{Δ} l'ensemble des couples ports-évènements de sortie sur le réseau de modèles,

χ le nom du modèle exécutif, c'est-à-dire le modèle qui définit la structure dynamique du réseau,

M_{χ} le modèle éxécutif lui-même.

X_{Δ} et Y_{Δ} ne sont pas obligatoirement des ports actifs. C'est le modèle exécutif M_{χ} qui spécifie les ports actifs à un instant donné. Ce modèle est un DEVS atomique tel que :

$$M_{\chi} =< X_{\chi}, Y_{\chi}, S_{\chi}, \delta_{ext_{\chi}}, \delta_{int_{\chi}}, \lambda_{\chi}, ta_{\chi} >$$

S_{χ} est un n-uplet particulier définissant l'ensemble des états du système. Tout changement de ce n-uplet correspond à un changement de structure du modèle exécutif. L'ensemble des états S_{χ} du modèle éxécutif est défini par :

$$S_{\chi} = (X_{\Delta}^{\chi}, Y_{\Delta}^{\chi}, D^{\chi}, \{M_i^{\chi}\}, \{I_i^{\chi}\}, \{Z_{i,j}^{\chi}\}, V^{\chi})$$

avec :

X_{Δ}^{χ} l'ensemble des évènements d'entrée du DS-DEVS,

Y_{Δ}^{χ} l'ensemble des évènements de sortie du DS-DEVS,

D^{χ} l'ensemble des noms des modèles composant le réseau,

M_i^{χ} les modèle DEVS qui composent le réseau $\forall\ i \in D^{\chi}$,

I_i^{χ} l'ensemble des influences de i $\forall i \in D^{\chi} \cup \{\chi, \Delta\}$, c'est-à-dire l'ensemble des modèles susceptibles de recevoir un évènement en provenance du modèle i, y compris le modèle DS-DEVS lui-même,

$Z_{i,j}^{\chi}$ l'ensemble des connexions du DS-DEVS allant du modèle $i \rightarrow j$ $\forall j \in I_i^{\chi}$,

V^{χ} l'ensemble des autres variables d'états nécessaire pour calculer les fonctions de transitions.

$Z_{i,j}^{\chi}$ représente les connexions entre modèles qui sont effectives à un instant donné. Cet ensemble formalise la topologie des relations qui existent dans le réseau du modèle. Un changement de cet ensemble modifie la topologie du réseau d'interaction et donc la structure du modèle exécutif.
La définition de I_i^{χ} comme un ensemble « d'influences » est celle adoptée par F. Barros dans sa définition d'un modèle DS-DEVS. Elle est à mettre en relation avec le même terme utilisé dans les SMAs basé sur le principe « d'influences-réactions » [FM96]. Dans ce type de modèle, un agent est une entité qui perçoit et produit des influences sur son environnement et non des actions au sens de modifications d'états. La différence est notoire, les influences se combinent sans rien modifier directement, ce sont les actions propremement dites qui modifient l'état du système. Dans un modèle DS-DEVS, les influences représentent l'ensemble des relations possibles des modèles composants. Nous conservons ici cette définition.

Du fait que les changements de structure ont lieu sur des transitions internes ou externes du modèle exécutif, nous pouvons distinguer les changements autonomes (sur les transitions internes) des changements provoqués par un autre modèle extérieur au DS-DEVS. Comme pour

DEVS, il existe des simulateurs abstraits pour DS-DEVS, le lecteur intéressé peut se référer à l'article de présentation du formalisme [Bar96].

Dans ce qui suit, nous allons formaliser le système d'agents réactifs situés couplé avec le système d'équations différentielles que nous avons présenté au paragraphe 2.3. La première partie de ce couplage formel nécessitait l'expression d'un système d'agent réactif en DEVS. Dans ce paragraphe, nous avons établi une analogie entre les agents réactifs situés et DEVS qui nous permet de formaliser le modèle des copépodes se nourrissant de phytoplancton dans un espace 3D. Nous allons donc poursuivre notre exposé par la présentation de cette formalisation. Dans un deuxième temps, nous proposons l'équivalent DEVS d'un algorithme de résolution numérique d'équations différentielles, ce qui nous permet de coupler les deux modèles.

3.5 Formalisation des agents copépodes dans leur environnement

Dans ce paragraphe, nous formalisons le modèle du copépode présenté au paragraphe 2.3 page 36 vu comme un agent réactif. Nous avons discuté de l'utilisation de DEVS pour la formalisation de ce type d'agents dans la section précédente. Nous rappelons que le système est composé de copépodes se nourrissant de cellules de phytoplancton dans un espace continu en 3D.

3.5.1 Le système d'agents réactifs situés

Nous définissons notre système d'agents réactifs situés (SAR) comme un modèle DEVS couplé, composé des agents copépodes et de l'environnement de la manière suivante[49] :

$$SAR =< D, \{M_d | d \in D\}, IC >$$

où :
$D = \{Copepode_i, Environnement\}$ avec $i = 1 \dots n$ où n est le nombre total de copépodes dans le système,
$M_{Copepode_i} \mid i = 1 \dots n$ est l'ensemble des modèles DEVS des copépodes,
$M_{Environnement}$ est le modèle de l'environnement,
$IC = \{\ ((M_{Copepode_i}, outcop_1), (Environnement, inenv_1)),$
$((M_{Copepode_i}, outcop_2), (Environnement, inenv_2)),$
$((M_{Copepode_i}, outcop_3), (Environnement, inenv_3)),$
$((Environnement, outenv_1), (M_{Copepode_i}, incop_1)),$
$((Environnement, outenv_2), (M_{Copepode_i}, incop_2))\ \}$

Le système est fermé, il n'y a pas de ports d'entrée ou de sortie propres au système. Ce modèle DEVS spécifie la structure des relations entre les agents et l'environnement. Comme le montre l'ensemble des connexions internes IC, nous ne considérons pas de relation « inter-copépodes » dans notre système. Les agents copépodes disposent de trois effecteurs sur l'environnement et de deux récepteurs. Comme les cellules de phytoplancton sont considérées comme statiques, elles n'apparaissent pas à ce niveau de la spécification mais comme des objets de l'environnement. La

[49] Les noms donnés aux ports d'entrée et de sortie sont arbitraires. Néanmoins, le nom des ports d'entrée commence par « in » et celui des ports de sortie par « out ».

construction hiérarchique des modèles DEVS donne une vision synoptique du système d'agents réactifs représentée par la figure 3.6.

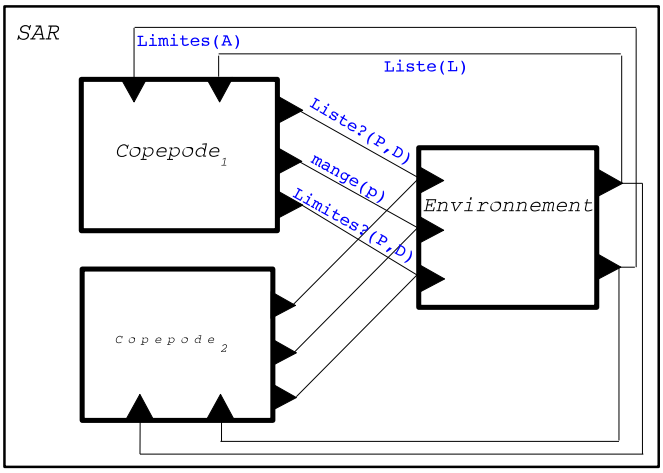

FIG. 3.6 – Représentation de la structure des connexions internes du système d'agent réactif en considérant deux agents copépodes. Les noms sur les connexions correspondent au nom des évènements.

Les noms des évènements représentés sur la figure 3.6 correspondent aux noms attachés aux valeurs v des ensembles X et Y d'un modèle DEVS atomique ou couplé. Dans le tableau 3.1 de la page 65, nous avons identifié les évènements comme étant le support des communications entre agents et entre les agents et l'environnement. Dans toute notre spécification, nous utilisons la notation suivante pour les couples port-valeur (p, v) (« $\Leftrightarrow$ » signifie équivalent) :

$$(p, v) \Leftrightarrow (p, nom(v)) \mid p \in IPorts, v \in X_p \ \vee \ p \in OPorts, v \in Y_p$$

avec *nom* l'identifiant de l'évènement qui porte la valeur v ; v pouvant être un n-uplet de valeurs séparées par des virgules.
Cette notation permet d'ajouter du sens pour le lecteur en identifiant le nom de l'évènement à son action sur le modèle affecté par son arrivé. Par exemple sur la figure 3.6, le point d'interrogation à la fin de « `Liste?(P,D)` » permet d'informer le lecteur que cet évènement correspond à la demande d'une liste à l'environnement et que pour répondre à cette demande, le modèle receveur a besoin de connaître les valeurs de `P` et `D`. L'évènement « `Liste(L)` » en retour correspond à l'envoi effectif de cette liste `L` à l'agent demandeur. Il en est de même pour les évènements « `Limites?(P,D)` » et « `Limites(A)` ». L'évènement « `Mange(p)` » indique à l'environnement que le copépode consomme une cellule de phytoplancton. Nous allons décrire ces évènements dans ce qui suit en spécifiant le modèle du copépode, puis celui de l'environnement, pour comprendre la dynamique des interactions entre ces modèles.

3.5.2 Le modèle des agents copépodes

Le modèle DEVS couplé du copépode est divisé en cinq modèles DEVS atomiques :

1. activité :

ce modèle spécifie dans quelle phase se trouve le copépode. Ces phases correspondent notamment à celles présentées à la section 2.3.2 page 40 (recherche de proie, perception, chasse et capture). Ce modèle joue un rôle central dans la formalisation du comportement.

2. perception :
   ce modèle rend compte de la perception d'une proie par le copépode.
3. gestion des rebonds :
   ce modèle rend compte de la perception des limites spatiales de l'environnement.
4. gestion de l'énergie :
   ce modèle gère les ressources energétiques du copépode. Il intègre un sous-modèle physiologique d'*A. tonsa* décrit en annexe B et C.
5. direction aléatoire :
   ce modèle permet au copépode de changer de direction aléatoirement.

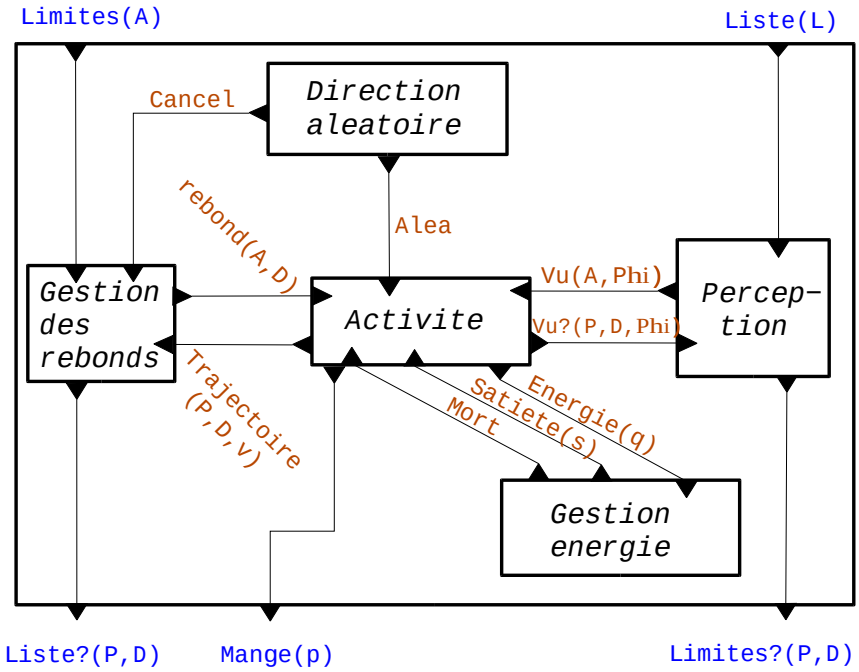

FIG. 3.7 – Représentation graphique du modèle couplé de l'agent copépode. La formalisation est donnée dans le texte en ce qui concerne le modèle « activité » et en annexe E.1 pour les autres. En bleu figure les évènements externes attachés aux ports d'entrée ou de sortie du modèle couplé. Les évènements véhiculés par les connexions internes figurent en rouge. Nous voyons le rôle central joué par le modèle d'activité (voir sa formalisation dans le texte).

Le modèle DEVS couplé peut être représenté graphiquement (figure 3.7). Cette représentation met en évidence les évènements qui « circulent » à l'intérieur du modèle couplé de l'agent et les évènements échangés avec le modèle de l'environnement. Cette figure met en évidence le rôle central joué par le modèle d'activité. C'est lui qui gère le déplacement du copépode par l'intermédiaire du modèle de perception et de gestion des rebonds. Le modèle d'activité possède un unique port connecté sur l'extérieur de l'agent. Il correspond à la seule action du copépode sur

l'environnement, à savoir l'ingestion d'une particule. Nous donnons la formalisation du modèle couplé de l'agent en annexe E.1. Dans ce qui suit, nous présentons les modèles DEVS atomiques en interaction dans le modèle de l'agent. Nous commençons par la formalisation du modèle d'activité du copépode. Ce modèle est affecté, directement ou indirectement, par les autres modèles composant l'agent. Nous décrirons ces derniers de manière informelle (nous donnons néanmoins leur spécification formelle en annexe E).

Le modèle d'activité

Le modèle d'activité est un modèle DEVS atomique qui possède la structure suivante :

$$activite =< X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

où :

$X = \{ (inact_1, vu(A, \Phi)), (inact_2, rebond(A, \vec{D})),$
$(inact_3, mort), (inact_4, satiete(s)), (inact_5, alea) \}$
$Y = \{ (outact_1, vu?(P, \vec{D}, \Phi)), (outact_2, trajectoire(P, \vec{D}, v)),$
$(outact_3, energie(q)), (outact_4, mange(p)) \}$

Les évènements et les valeurs qu'ils attachent aux ports d'entrée et de sortie (par exemple l'évènement $vu(A, \Phi)$ affecte au port $inact_1$ les valeurs A et Φ) sont décrits dans ce qui suit. Nous commençons par décrire l'ensemble des états du modèle d'activité, représenté par le n-uplet suivant :

$$S = (\Phi, P, \vec{D})$$

Le premier attribut Φ représente l'état d'activité du copépode : la phase. Il peut prendre une des valeurs suivantes :

init : le copépode est initialisé,
percoit? : état transitoire,
cherche : le copépode recherche sa nourriture,
rien : le copépode ne perçoit aucune proie,
chasse : le copépode perçoit une cellule de phytoplancton et se dirige vers elle,
capture : la nourriture est suffisamment proche, le copépode la capture,
manipule : le copépode manipule, ingère la proie et sédimente (il tombe),
chute : le copépode sédimente,
marche aleatoire : le copépode n'a pas faim, il se déplace aléatoirement,
rebond? : état transitoire,
rebond : le copépode effectue un rebond,
mort : plus aucune activité du copépode.

La plupart des phases sont liées au déplacement du copépode. Elles conditionnent principalement son comportement, c'est-à-dire les séquences d'activation des fonctions de transitions.

Les phases « *percoit*? » et « *rebond*? » représentent des états passifs du copépode. Elles permettent de modéliser des interactions de type question-réponse entre le modèle de perception et le modèle d'activité, comme nous l'avons vu pour le modèle de l'environnement(paragraphe 3.4.2 page 64).
Le calcul de la nouvelle position P' du copépode fait intervenir l'attribut P, qui a pour valeur

les coordonnées spatiales $\{x, y, z\}$ du copépode. Nous avons donc $P' = P + (\vec{D} \times v \times e)$, v étant la vitesse du copépode et e le temps passé dans l'état.

En fin de phases « *init* », « *recherche* », « *chasse* » ou « *chute* », le copépode termine un déplacement. Le modèle d'activité génère un évènement de sortie adressé au modèle de perception afin de connaître la position de la prochaine proie. C'est le rôle des fonctions de sorties λ suivantes[50] :

$\lambda(init, P, \vec{D}) = vu?(P, \vec{D}, init)$
$\lambda(cherche, P, \vec{D}) = vu?(P, \vec{D}, cherche)$
$\lambda(chasse, P, \vec{D}) = vu?(P, \vec{D}, chasse)$
$\lambda(chute, P, \vec{D}) = vu?(P, \vec{D}, chute)$

Après avoir généré la fonction de sortie, le modèle d'activité passe dans un état transitoire *percoit*? dans l'attente de récupérer la réponse venue du modèle de perception. Les fonctions de transitions internes suivantes formalisent ce passage[51].

$\delta_{int}((init, P, \vec{D})) = (percoit?, P, \vec{D})$
$\delta_{int}((cherche, P, \vec{D})) = (percoit?, P' = P + v(ta(s))\vec{D}, \vec{D})$
$\delta_{int}((chasse, P, \vec{D})) = (percoit?, P' = P + v(ta(s))\vec{D}, \vec{D})$
$\delta_{int}((chute, P, \vec{D})) = (percoit?, P' = P + v(ta(s))\vec{D}, \vec{D})$

La phase *percoit*? définit un état passif, d'où : $ta(percois?, P, \vec{D}) = \infty$.

À la réception de la réponse, le modèle d'activité transite vers un nouvel état déterminé par la valeur de la phase Φ elle-même déterminée par le modèle de perception. Les fonctions de transitions externes suivantes formalisent ces changements d'états[52] :

$\delta_{ext}((percoit?, P, \vec{D}), (vu(A, cherche))) = (cherche, P, \vec{D})$ où A est l'ensemble des coordonnées de la cellule perçue par le modèle de perception.
$\delta_{ext}((percoit?, P, \vec{D}), (vu(A, chasse))) = (chasse, P, \vec{D}' = \overrightarrow{PA})$ où $\vec{D}'$ est la nouvelle direction du copépode.
$\delta_{ext}((percoit?, P, \vec{D}), (vu(A, capture))) = (capture, P, \vec{D})$
$\delta_{ext}((percoit?, P, \vec{D}), (vu(nil, rien))) = (rien, P, \vec{D})$

Pour ces états, les fonctions d'avancement du temps sont :

$ta(cherche, P, \vec{D}) = \|\overrightarrow{PA}\|/v$ avec v la vitesse du copépode.
$ta(chasse, P, \vec{D}) = \|\overrightarrow{PA}\|/v$
$ta(capture, P, \vec{D}) = 0$
$ta(rien, P, \vec{D}) = 0$

Les deux premières fonctions d'avancement du temps ci-dessus prennent des valeurs qui

[50] Les fonctions de sortie $\lambda : S \to Y$ sont définies par $\lambda(s) = nom\_evenement(v_1, ..., v_n)$ avec $s \in S$ et n le nombre de valeurs portées par l'évènement.

[51] Les fonctions de transitions internes $\delta_{int} : S \to S$ sont définies par $\delta_{int}(s) = (s')$ où $s \in S$ et $s' \in S$ définissent respectivement l'état initial et l'état final.

[52] Les fonctions de transitions externes $\delta_{ext} : Q \times X \to S$ sont définies par $\delta_{ext}((s), (v_1, ..., v_n), e) = (s')$ avec $s \in S$, $s' \in S$ et n le nombre de valeurs portées par l'évènement. Si e n'intervient pas dans le calcul du nouvel état, alors il n'apparaît pas dans la définition de δ_{ext}. Dans ce cas $\delta_{ext} : S \times X \to S$

dépendent du modèle de perception. C'est lui qui joue le rôle d'interface avec l'environnement. Le modèle d'activité a pour principale fonction de calculer les coordonnées du copépode à la fin de chaque déplacement et de réorienter son vecteur de direction $\vec{D}$.

Si le copépode ne perçoit aucune proie, il passe dans la phase *rien* de durée nulle ce qui lui permet, par transition instantanée vers un état passif, d'interroger le modèle de gestion des rebonds qui lui communiquera ses nouvelles coordonnées et sa nouvelle direction aux limites de l'espace le cas échéant. Si la phase est égale à *rien* alors nous avons la fonction de sortie suivante :

$\lambda(rien, P, \vec{D}) = trajectoire(P, \vec{D}, v)$ avec v la vitesse du copépode.
$ta(rien, P, \vec{D}) = 0$

La séquence suivante formalise la reprise de la chasse dès que le copépode a atteint les limites de l'espace.

$\delta_{int}(rien, P, \vec{D}) = (rebond?, P, \vec{D})$
$ta(rebond?, P, \vec{D}) = \infty$
$\delta_{ext}((rebond?, P, \vec{D}), (rebond(A, \vec{D}'))) = (rebond, A, \vec{D}')$
$ta(rebond, P, \vec{D}) = 0$
$\lambda(rebond, P, \vec{D}) = vu?(P, \vec{D})$
$\delta_{int}(rebond, P, \vec{D}) = (vu?, P, \vec{D})$

Dans sa phase de recherche de proie ou d'atteinte d'une limite de l'espace, le copépode ne perçoit pas de cellule. Il doit pouvoir changer sa direction avant d'atteindre une nouvelle cible. C'est le rôle de l'évènement d'entrée *alea* provenant du modèle de direction aléatoire. Dans la phase de chasse, il perçoit sa proie et se dirige vers elle sans hésiter. Le copépode peut donc « décider » de changer de direction sous l'influence du modèle de changement de direction aléatoire. C'est seulement dans les phases *cherche* et *rebond*? que cela se produit sous l'influence d'un évènement externe comme suit :

$\delta_{ext}((cherche, P, \vec{D}), alea) = (percoit?, P, \vec{D}')$ avec $\vec{D}'$ généré aléatoirement[53]
$\delta_{ext}((rebond?, P, \vec{D}), alea) = (percoit?, P, \vec{D}')$ avec $\vec{D}'$ généré aléatoirement
$\delta_{ext}((S - \{cherche, rebond\}), alea) = (S - \{cherche, rebond\})$

La dernière transition externe spécifie le fait que dans toutes les autres phases, l'évènement *alea* n'a aucune conséquence sur le modèle d'activité. Ce modèle a deux fonctions de sorties spécifiques pour la capture. Elles sont adressées au modèle de gestion de l'énergie et au modèle de l'environnement et s'écrivent comme suit :

$\lambda(capture, P, \vec{D}) = mange(p)$ indique à l'environnement la proie qui est consommée.
$\delta_{int}(capture, P, \vec{D}) = (manipule, P, \vec{D}')$ avec $\vec{D}'$ orienté vers le bas.
$ta(manipule, P, \vec{D}) = h$, le temps de manipulation d'une proie.
$\lambda(manipule, P, \vec{D}) = energie(q)$ avec q la quantité d'énergie absorbée.
$\delta_{int}(manipule, P, \vec{D}) = (chute, P, \vec{D})$
$ta(chute, P, \vec{D}) = tc$ avec tc, le temps de chute.

À tout moment, le copépode peut mourir si son énergie passe sous un seuil critique. Cet évènement est déclenché sur une transition externe générée par le modèle de gestion de l'énergie. Les fonctions de transitions externes suivantes formalisent ce changement d'état.

[53]Nous revenons sur les politiques de distribution des cellules de phytoplancton dans la partie 5.

$\delta_{ext}(s, mort) = (mort, t, P, \vec{D}),\ \forall s$
$ta(mort, date, P, \vec{D}) = \infty$

La phase *mort* conduit à l'arrêt total du modèle d'activité.
La satiété (le fait d'avoir faim ou pas) est contrôlée par le modèle de gestion de l'énergie. Chaque fois qu'une cellule est mangée, un évènement externe est généré. Le modèle de gestion de l'énergie peut réagir en envoyant un évènement de satiété au modèle d'activité. Alors, le copépode entre dans une phase *marche aleatoire* où ses mouvements sont générés de façon aléatoire. La séquence suivante formalise ce passage :

$\delta_{ext}((chute, P, \vec{D}), satiete(true)) = (marche\ aleatoire, P, \vec{D'})$ où $\vec{D'}$ est généré aléatoirement.
$ta(marche\ aleatoire, P, \vec{D}) = random()$, ce qui correspond à un tirage aléatoire uniforme tel que $0 < ta(s) < c_{max}$ où c_{max} est la durée maximale avant un changement de direction.
$\delta_{int}(marche\ aleatoire, P, \vec{D}) = (marche\ aleatoire, P, \vec{D'})$ où $\vec{D'}$ est généré aléatoirement.
$\delta_{ext}((marche\ aleatoire, P, \vec{D}), satiete(faux)) = (vu?, P, \vec{D})$

La phase *marche aleatoire* permet d'éviter l'interrogation du modèle de perception en cas de satiété. Ainsi, le déplacement du copépode est aléatoire tant que le modèle d'activité n'est pas soumis à une transition externe liée à un évènement $satiete(faux)$.

Le modèle de perception

Comme le montre la figure 3.7 page 72, le modèle de perception joue le rôle d'interface entre l'agent et l'environnement. À chaque instant, le copépode est localisé dans l'espace et se déplace selon une certaine direction. Le modèle d'activité demande (*i.e.* envoie un évènement) au modèle de perception afin de savoir si une cellule de phytoplancton se trouve sur sa trajectoire courante : c'est le rôle de l'évènement $vu?(P, \vec{D}, \Phi)$.
Cet évènement correspond à l'arrivée d'un évènement externe pour le modèle de perception. Celui-ci passe alors dans un état transitoire lui permettant d'interroger instantanément le modèle de l'environnement. Ce dernier lui communique la liste L des cellules de phytoplancton susceptibles d'être perçues par le copépode. Le modèle de perception peut alors déterminer la prochaine cellule cible (aux coordonnées A) du copépode par un calcul géométrique effectué sur une transition interne. La figure 3.8 donne une représentation simplifiée de ce calcul (voir annexe E.2 pour les détails).

Après ce calcul, le modèle de perception génère la fonction de sortie $vu(A, \Phi)$. Ainsi, le modèle d'activité sera en mesure d'effectuer le déplacement nécessaire pour entrer dans une nouvelle phase. La valeur de la fonction de sortie dépend de l'attribut Φ donné par l'évènement de transition externe en provenance du modèle d'activité et de la géométrie du volume de perception du copépode. La formalisation complète du modèle de perception est donnée en annexe E.2.

Le modèle de gestion des rebonds

Le modèle de gestion des rebonds s'attache à définir le comportement du copépode lorsqu'il arrive aux limites de l'espace. Si le modèle de perception envoie un évènement $Vu(A, \Phi)|A = nil$, cela signifie qu'aucune cellule ne se trouve sur la trajectoire courante du copépode. Alors le modèle d'activité demande au modèle de gestion des rebonds de lui communiquer les coordonnées du point d'intersection entre la droite qui porte le vecteur directeur $\vec{P}$ du copépode et le plan

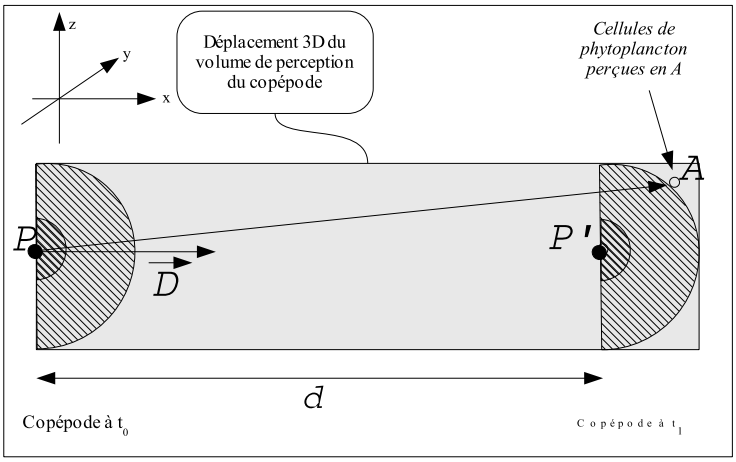

FIG. 3.8 – Représentation simplifiée du calcul de la nouvelle position P' du copépode au temps t_1 à partir de sa position initiale P en fonction de la position A de la cellule cible et du vecteur directeur $\vec{D}$ du copépode. Ici, le copépode est supposé être en phase de recherche. La distance d calculée permet alors de connaître le temps nécessaire pour que le copépode entre en phase de chasse.

définissant la limite de l'espace devant lui. Le calcul est du même type que celui illustré par la figure 3.8. La formalisation du modèle de gestion des rebonds est donnée en annexe E.3.

Le modèle de changement de direction aléatoire

Comme nous l'avons vu pour le modèle d'activité avec l'état *marche aleatoire*, il est possible de donner des valeurs stochastiques à $ta(s)$ pour simuler la génération non déterministe d'évènements. Il est en fait possible d'introduire de l'aléatoire dans un modèle DEVS à plusieurs niveaux :

- les fonctions de sortie peuvent émettre des évènements portant des valeurs aléatoires dans un intervalle,
- les fonctions d'avancement du temps peuvent être aléatoires,
- les fonctions transitions internes ou externes peuvent entraîner des changements d'états aléatoires.

Pour les modèles manipulant des intervalles flous (dont la valeur des bornes est aléatoire), il existe une sémantique décrite comme une extension de DEVS (Fuzzy DEVS [KPJK96]). Ici, nous sommes intéressés par un modèle pouvant générer des évènements à des dates aléatoires. Ces évènements indiquent au modèle d'activité de modifier le vecteur directeur $\vec{D}$ du copépode. Pour cela, seule la fonction d'avancement du temps qui permet, à échéance, de déclencher la fonction de sortie, doit être stochastique. Ainsi, le modèle de direction aléatoire émet une fonction de sortie à des dates tirées aléatoirement dans un certain intervalle. Nous pouvons alors formaliser un tel système comme un modèle DEVS atomique (annexe E.4).
Ce modèle ne possède pas de port d'entrée ni de fonction de transition externe. Il est caractéristique des modèles de type générateur [ZKP00]. C'est le seul sous-modèle purement proactif interne au modèle du copépode. Il déclenche un comportement qui n'est pas déterminé par l'environnement. La « prise de décision » proactive se réduit ici à une fonction aléatoire. En effet, nous n'avons pas à ce jour d'information sur ce qui fait changer de direction le copépode lorsqu'il ne perçoit pas de proie.

Le modèle de gestion de l'énergie

Le modèle de gestion de l'énergie représente l'activité physiologique du copépode. En entrée, le modèle d'activité informe le modèle de gestion de l'énergie qu'une proie a été capturée. En retour, le modèle de gestion de l'énergie informe le modèle d'activité qu'il a suffisamment consommé de proies en lui envoyant un évènement de satiété $satiete(s)|s = vrai$ ou qu'il a faim de nouveau $satiete(s)|s = faux$. Ce modèle peut également indiquer au modèle d'activité qu'il a atteint un seuil d'énergie critique et qu'il doit cesser son activité (évènement $mort$).

Le modèle de gestion de l'énergie illustre un aspect particulier de DEVS. Comme nous l'avons dit plus haut, DEVS n'impose pas de formalisme particulier pour les fonctions de transition ou d'avancement du temps. Ici, nous allons montrer qu'il est possible d'intégrer un sous-modèle basé sur des équations différentielles en considérant ces dernières comme des fonctions de transitions. Pour cela, il est nécessaire d'interpréter les événements d'entrée comme des conditions initiales pour le système d'équations différentielles et les valeurs calculées comme des évènements de sortie.
P. Caparroy et F. Carlotti [CC96] définissent un système d'équations différentielles pour modéliser le budget énergétique du copépode (équations décrites en annexe B). Ces équations considèrent l'ingestion comme continue. Dans notre approche, l'ingestion est discrète (évènement $energie(q)$ provenant du modèle d'activité). Nous pouvons choisir entre deux approches pour calculer l'évolution de budget énergétique du copépode entre deux évènements d'ingestion :

1. à chaque réception de l'évènement $energie(q)$, nous résolvons le système d'équations à l'aide d'un schéma numérique d'intégration. Ceci nous oblige à discrétiser le temps entre deux évènements d'ingestion pour calculer l'évolution du contenu de l'estomac du copépode. Cette méthode est très coûteuse en temps de calcul.
2. à chaque réception de l'évènement $energie(q)$, nous calculons les valeurs de sortie en utilisant les solutions analytiques du système d'équations différentielles. Cette approche est moins coûteuse en temps de calcul puisqu'il ne s'agit plus de faire des itérations pour trouver une solution approchée du système.

La deuxième approche consiste à étudier le système d'équations afin de trouver une solution analytique exacte ou approchée, ce qui permet de spécifier le modèle en terme d'évènements correspondant au calcul du temps nécessaire pour que le système atteigne certains seuils. Les seuils discrétisent l'espace de variation de variables du système. Nous considérons ici deux seuils, un seuil de satiété au-dessus duquel le copépode n'a pas faim, un seuil d'énergie minimale endessous duquel le copépode meurt.

En observant le système d'équations de l'annexe B page 175, nous nous apercevons que la première équation ne fait pas intervenir les autres équations du système si la variable I (correspondant à l'ingestion) est supprimée du système. La conséquence est que nous pouvons alors calculer une solution analytique exacte du système (le calcul est donné en annexe C). La solution analytique est calculée à partir d'une nouvelle condition initiale déterminée par la valeur de $X1$ (la concentration des proies dans l'estomac du copépode) plus la quantité d'énergie apportée par une cellule de phytoplancton.
Pour résumer, l'évènement externe $energie(q)$ perturbe le système d'équations en définissant une nouvelle condition initiale sans modifier la forme générale de la solution. Les évènements de sortie $\{satiete(s)|s = vrai \ \vee \ s = faux\}$ sont générés à partir de valeurs seuils.

3.5.3 Le modèle de l'environnement

Dans ce qui suit, nous présentons la formalisation DEVS de l'environnement du copépode. Il s'agit d'un volume d'espace continu dans lequel sont distribuées des cellules de phytoplancton qui constituent les objets inactifs de cet espace. C'est un environnement centralisé, il suit donc la formalisation définie au paragraphe 3.4.2 page 67. Nous considérons l'environnement comme non proactif, *i.e.* sans dynamique autonome. Une fonction $ta(Idle) = \infty$ formalise le fait que le modèle de l'environnement est toujours en attente d'un évènement externe. Ce type de modèle est appelé « passif » [ZKP00]. Un modèle passif peut agir comme un enregistreur, *i.e.* il peut enregistrer les actions des agents et les informer, lorsqu'ils le demandent, de son état.

L'environnement est interrogé par le modèle du copépode qui lui envoie trois évènements externes : $liste?(P, \vec{D})$, $limites?(P, \vec{D})$ et $mange(p)$ (voir figure 3.6 page 71). Le modèle de l'environnement peut donc subir trois transitions externes qui vont modifier son état. Dans tous les cas, la réponse de l'environnement est instantanée. Pour cela, toutes les fonctions de transitions externes sont associées à des états internes transitoires. Ces états transitoires permettent le calcul instantané (au sens de la dynamique du modèle) des fonctions de sortie. Ces fonctions formalisent les réponses de l'environement aux différentes questions qui lui sont posées.

Pour la construction de la liste L des cellules susceptibles d'être perçues par le copépode, nous utilisons une technique classique [HE88]. Cette technique est basée sur un choix de structure particulier pour l'enregistrement des données (les coordonnées de cellules). Nous élaborons une grille qui discrétise l'espace. Cette grille permet d'associer chaque cellule de phytoplancton à une case en fonction de sa position. Cette structure peut être implémentée très facilement sous la forme d'une liste chaînée. En considérant la position P et la direction $\vec{D}$ du copépode (donnée en entrée au modèle de l'environnement par le modèle du copépode), il est possible de déterminer l'ensemble des cases se trouvant sur sa trajectoire. Ainsi, nous pouvons dresser la liste L des cellules appartenant à l'ensemble des cases coupées par sa trajectoire (la taille des cases est fonction de la distance de perception du copépode). La figure 3.9 illustre la méthode.

Lorsque l'environnement reçoit un évènement externe $limites?(P, \vec{D})$, il renvoie simplement les coordonnées de ses limites spatiales avant la transition interne d'un état transitoire. Lorsqu'il reçoit l'évènement $mange(p)$, l'environnement supprime la cellule de phytoplancton considérée. La formalisation de l'environnement sous la forme DEVS atomique est donnée en annexe E.6.

Dans ce paragraphe, nous avons formalisé notre système d'agents réactifs situés du copépode en utilisant DEVS. Nous disposons ainsi d'un cadre formel d'intégration de notre modèle avec d'autres paradigmes de modélisation. Dans ce qui suit, nous voulons coupler un modèle d'équations différentielles ordinaires, représentant notre modèle prédateurs-proies à un niveau d'abstraction supérieur. La première étape est de formaliser le système d'équations différentielles de façon à le rendre « compatible DEVS », c'est-à-dire de l'exprimer dans un langage qui manipule les évènements, base fondamentale de DEVS. De cette façon, nous aurons deux modèles exprimés dans le même formalisme. En utilisant le concept de modèle couplé décrit dans DEVS, il est alors possible de coupler ces deux modèles de façon formelle.

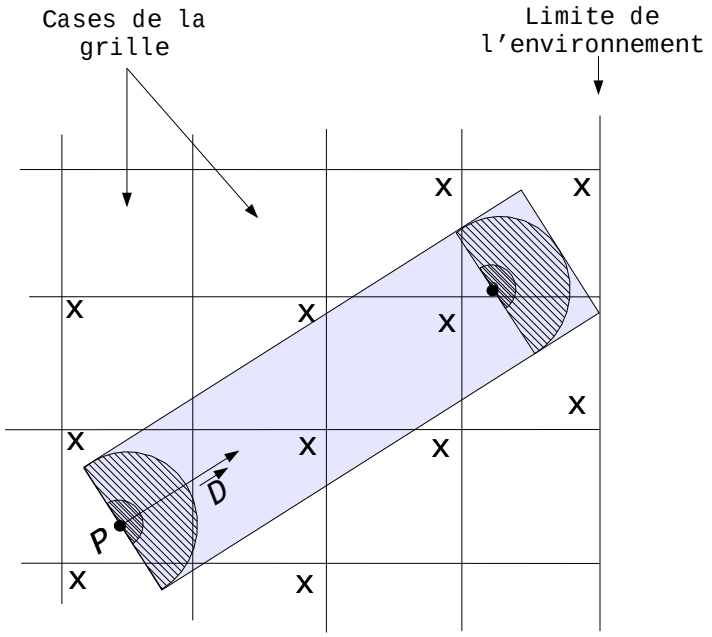

FIG. 3.9 – Représentation 2D de la discrétisation de l'espace pour le stockage des coordonnées des cellules de phytoplancton. Chaque case en contient un certain nombre. Toutes les cases marquées d'un « x » contiennent les celulles susceptibles d'être vues par le copépode avant qu'il n'atteigne une limite de l'environnement.

3.6 Couplage d'un système d'équations différentielles avec un modèle DEVS d'agents réactifs

Dans le paragraphe précédent, nous avons formalisé un système d'agents réactifs en DEVS. Nous voulons coupler ce système avec un système d'équations différentielles du premier ordre. L'objectif de cette partie est de montrer que DEVS offre un cadre formel pour l'intégration des agents réactifs avec les modèles mathématiques classiquement utilisés pour la modélisation des systèmes dynamiques : les équations différentielles. Nous ne décrivons pas ce couplage pour tous les types d'équations différentielles mais pour le type « équations différentielles ordinaires du premier ordre ». Pour formaliser le couplage, la première étape est de spécifier ce type de système dans un formalisme à évènements discrets.

Une des possibilités pour obtenir une simulation à évènements discrets d'un système continu est de remplacer la discrétisation du temps classiquement utilisée pour les équations différentielles, par une quantification de l'espace continu défini par le domaine de variation des variables. Cette technique est dite des systèmes quantifiés(*Quantized Systems*) [ZL98]. Cette méthode implique une transformation des modèles continus en modèles à évènements discrets. Un des problèmes majeurs de cette représentation est la présence d'un nombre infini de transitions d'états des variables continues dans un intervalle de temps fini. Cette difficulté a été résolue par E. Kofman [Kof02] en introduisant la notion de systèmes à états quantifiés (*Quantized State Systems*). Cette technique s'avère efficace et peu coûteuse en temps de calcul pour la résolution d'un système d'équations différentielles du premier ordre. Néanmoins, cette méthode ne nous est pas apparue la plus simple dans notre approche de couplage, c'est pourquoi nous ne l'utilisons

pas.
Une autre possibilité de coupler formellement les deux modèles est de les représenter dans un formalisme unificateur tel que le « *discrete event and differential equation specified system* » (DEV&DESS [ZKP00]). Ce formalisme, principalement basé sur les travaux de H. Praehofer [Pra91], permet d'écrire un modèle DEVS et un système d'équations différentielles avec une même sémantique.

Il existe une méthode plus simple basée sur le fait que la résolution numérique des équations différentielles nécessite une réécriture du système sous la forme d'équations discrètes. Ainsi, le simulateur des équations différentielles est très généralement un système à temps discret. Ces systèmes peuvent être formalisés par des modèles DEVS [ZKP00]. C'est cette dernière approche que nous allons utiliser.

3.6.1 Présentation du système d'équations différentielles comme un système à temps discret

Nous considérons le système d'équations différentielles ordinaires du premier ordre que nous avons décrit en introduction (paragraphe 2.3.3 page 44). Nous rappelons ce système ici :

$$\frac{dN}{dt} = r(N)N - G(N,P)P \tag{3.1}$$
$$\frac{dP}{dt} = G(N,P)P - mP \tag{3.2}$$

où N est une concentration en proies et P une concentration en prédateurs.

Les équations 3.1 et 3.2 ci-dessus possèdent des solutions analytiques simples si les fonctions $r(N)$ et $G(N,P)$ sont ramenées à des constantes. Seulement, ce n'est très généralement pas le cas dans un modèle d'interactions proie-prédateur (cf. chapitre 5). Aussi nous devons avoir recours à des méthodes numériques pour calculer la dynamique de tels systèmes. Lorsque l'évolution des variables est unidimensionnelle (c'est le cas ici où la seule dimension est le temps), nous pouvons utiliser la méthode de Runge-Kutta d'ordre 4 [Ant02]. L'algorithme de cette méthode est le suivant. Nous posons :

$$f_1(t,N,P) = r(N)N_t - G(N,P)P_t \tag{3.3}$$
$$f_2(t,N,P) = G(N,P)P_t - mP_t \tag{3.4}$$

et :

$$N_{t+h} = N_t + \left\{\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\right\} \tag{3.5}$$
$$P_{t+h} = P_t + \left\{\frac{1}{6}(s_1 + 2s_2 + 2s_3 + s_4)\right\} \tag{3.6}$$

avec :

$$\begin{aligned} k_1 &= hf_1(t, N_i, P_i) \\ k_2 &= hf_1(t + \tfrac{1}{2}h, N_t + \tfrac{1}{2}k_1, P_t) \\ k_3 &= hf_1(t + \tfrac{1}{2}h, N_t + \tfrac{1}{2}k_2, P_t) \\ k_4 &= hf_1(t + \tfrac{1}{2}h, N_t + k_3, P_t) \\ s_1 &= hf_2(t, N_i, P_i) \\ s_2 &= hf_2(t + \tfrac{1}{2}h, N_t, P_t + \tfrac{1}{2}s_1) \\ s_3 &= hf_2(t + \tfrac{1}{2}h, N_t, P_t + \tfrac{1}{2}s_2) \\ s_4 &= hf_2(t + \tfrac{1}{2}h, N_t, P_t + s_3) \end{aligned}$$

où h est le pas d'intégration et t le temps.

Les valeurs N_{t+h} et P_{t+h} au temps $t + h$ dépendent uniquement des valeurs N_t et P_t au temps t et du pas de temps constant h. Nous sommes donc en présence d'un système à temps discret. Connaissant les valeurs N_0 et P_0, nous avons un système autonome avec un avancement du temps constant h. Un tel système peut s'écrire sous la forme d'une structure telle que :

$$NP =< S, \delta, h > \tag{3.7}$$

où :

$S = \{(N, P)\}$ avec $(N, P) \in \mathbb{R} \times \mathbb{R}$ est l'ensemble des états.

$\delta(N, P) = (N_{t+h}, P_{t+h})$ est la fonction de changement d'état avec N_{t+h} donné par l'équation 3.5 et P_{t+h} par l'équation 3.6.

$h = constante$ est l'avancement du temps.

Un tel système peut être formalisé par un modèle DEVS. Pour cela, il est nécessaire d'introduire les notions d'évènement et de port. Dans ce qui suit, nous considérerons le système 3.7 comme un intégrateur qui fournit les valeurs de N et P à chaque pas d'intégration. La contrainte que nous nous imposons ici est que l'équation $G(N, P)$ n'est pas connue. Par conséquent, la valeur de $G(N, P)$ au temps t des équations 3.3 et 3.4 n'est pas connue. Nous appelons g cette valeur. Notre couplage repose sur le fait que g peut être donné à tout instant par un modèle extérieur. En d'autres termes, le système d'équations différentielles peut être perturbé à tout moment de sa résolution numérique par l'arrivée de la valeur g. Nous considérons l'arrivée de cette perturbation comme un évènement externe pour le modèle NP[54]. Ainsi, nous pouvons réécrire la structure 3.7 sous la forme d'un modèle DEVS comme suit :

$$NP =< X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta >$$

avec :

$X = \{(inNP, value(g'))\}$

$Y = \{(outNP, compute(N, P))\}$

$S = (N, P, g, \sigma)$ avec σ une durée.

$\delta_{ext}(S, value(g'), e) = (N, P, g', \sigma \leftarrow \sigma + e)$ avec e la durée passée dans l'état.

$\lambda(S) = (N_{t+h}, P_{t+h})$ avec h le pas d'intégration.

$\delta_{int}(S) = (N_{t+h}, P_{t+h}, g, \sigma = 0)$

$ta(S) = h - \sigma$

[54] Nous détaillerons les aspects conceptuels et pratiques de cette méthode dans notre chapitre applicatif 5

S'il n'y a pas de transition externe, alors $ta(S) = h$, sinon $ta(S)$ correspond toujours à la durée telle que $\sum_{i=1}^{n} e_i + ta(S) = h$ avec n le nombre d'évènements externes. En d'autres termes, la transition interne a toujours lieu après une durée constante égale à h (ce qui préserve une résolution avec un pas de temps au maximum égal à h de l'équation différentielle). S'il y a plusieurs transitions externes avant la transition interne, c'est la dernière valeur de g qui est prise en compte dans le calcul de λ.

3.6.2 Couplage formel des deux modèles

Nous avons formalisé deux systèmes *a priori* hétérogènes dans un formalisme unique. Comme nous l'avons vu en introduction de ce chapitre (paragraphe 3.2.2) et avec le modèle du copépode (paragraphe 3.5.2), il est possible de coupler des modèles DEVS et donc de coupler formellement le modèle d'agent réactif et le modèle d'équations différentielles. Néanmoins, notre système couplé possède quelques particularités qui nous obligent à pousser un peu plus loin la formalisation.

Reformalisation du SMA comme un tout

Dans un modèle DEVS, la base de temps des différents modèles atomiques doit être identique. C'est le cas dans nos deux modèles puisque celle-ci est réelle ($ta(S) \in \mathbb{R}^+$). Néanmoins, ces deux modèles peuvent décrire une dynamique à des échelles de temps différentes[55]. Nous considérons ici qu'il y un modèle lent (le système d'équations différentielles) et un modèle rapide (le modèle d'agents)[56]. Le modèle lent émet un évènement en direction du modèle rapide, celui-ci s'exécute et renvoie un évènement au modèle lent. Ici, nous désirons séparer les échelles de temps, c'est-à-dire que le temps simulé par le modèle rapide soit très inférieur au temps simulé par le modèle lent[57]. Formellement, nous pouvons résoudre ce problème en introduisant un délai de réponse dans le modèle lent (voir [WG01] pour l'application d'une telle méthode dans le cas d'automates cellulaires temporisés formalisés avec DEVS). Le concept d'évènement permet effectivement d'effectuer des sauts dans le temps sans aucune charge de calcul. Ici, le délai de réponse est introduit par l'avancement du temps $ta(S)$ du modèle NP décrit au paragraphe précédent. Le modèle NP génère un évènement de sortie à chaque pas d'intégration vers le modèle d'agents réactifs situé SAR décrit au paragraphe 3.5.1. Cet évènement déclenche une transition externe du SAR qui va alors simuler $g(N, P)$ et envoyer une valeur g' au modèle NP. Comme le temps simulé Δt pour calculer g est très inférieur à h, le modèle SAR se trouve dans un état d'inactivité pendant $h - \Delta t$, jusqu'au prochain évènement externe envoyé par le modèle NP. Nous couplons ainsi un modèle lent et un modèle rapide. Nous verrons au chapitre 5 qu'une telle technique permet la simulation d'un transfert d'échelle et quel en est l'intérêt.

Nous avons donc choisi ici de considérer que le modèle d'agents réactifs peut perturber la résolution numérique du système d'équations différentielles en lui envoyant la valeur de g sous la forme d'un évènement externe. En retour, le système d'équations différentielles indique au modèle d'agents le nombre de cellules de phytoplancton et de copépodes, N et P respectivement, présents dans son environnement. Les cellules de phytoplancton font partie de l'état de

[55] Nous reviendrons en détail sur ce fait au chapitre 5

[56] Lent et rapide réfèrent ici au temps simulé, non pas au temps d'exécution

[57] La justification d'une telle méthode, appelée méthode de séparation des échelles de temps, est donnée au paragraphe 2.2.4 page 33

l'environnement. Une modification de ce nombre entraîne donc un changement d'état de l'environnement. Par contre, le nombre de copépodes est caractéristique du SMA lui-même, donc du modèle couplé qui le formalise. Nous sommes donc devant un modèle DEVS couplé dont la composition change dynamiquement. Comme nous l'avons vu, DEVS dans sa version initiale ne permet pas de formaliser un tel changement. Pour répondre à ce problème, F. Barros a introduit une extension de DEVS, le Dynamic Structure DEVS (DS-DEVS) [Bar96] que nous avons présenté au paragraphe 3.4.4 page 68.

Ainsi, le modèle DEVS du système d'agents réactifs situés formalisé au paragraphe 3.5.1 de la page 70 devient un DS-DEVS (i.e. $\Delta = RAS$) où :

$$M_i^{\chi} = \{Environnement, Copepode_n \mid n = 1, \ldots, \infty\}$$

Nous considérons un nombre infini de modèles d'agents potentiellement présents dans le réseau. Il n'est pas possible de connaitre à l'avance le nombre maximum de copépodes, ce nombre étant calculé par le modèle NP. L'ensemble des influences s'écrit comme suit :

$$I_{RAS}^{\chi} = \{Environnement\}$$

Ce qui implique que le modèle de l'environnement recevra l'évènement $calcul(N, P)$ en provenance du modèle NP. Cet évènement indique au RAS qu'il doit calculer la valeur g, il prend deux paramètres qui sont respectivement la quantité de proies et de prédateurs,

$$I_{Environnement}^{\chi} = \{RAS, Copepode_n\}$$

C'est l'environnement qui enverra l'évènement $value(g)$ au modèle NP.

$I_{Copepode_n}^{\chi} = \{Environnement\}$ tous les agents influencent l'environnement.

L'ensemble des connexions du modèle exécutif est défini comme suit :

$$Z_{RAS,Environnement}^{\chi} = \{(RAS, in) \rightarrow (Environnement, inenv4)\}$$
$$Z_{Environment,RAS}^{\chi} = \{(Environment, outenv3) \rightarrow (RAS, out)\}$$
$$Z_{Environnement,Copepode_n}^{\chi} = \{\ ((Environnement, outenv_1) \rightarrow (Copepode_n, incop_1)),\ ((Environnement, outenv_2) \rightarrow (Copepode_n, incop_2))\ \}$$
$$Z_{Copepode_n,Environnement}^{\chi} = \{\ ((Copepode_n, outcop_2) \rightarrow (Environment, inenv_2)),\ ((Copepode_n, outcop_3) \rightarrow (Environnement, inenv_3))\ \}$$

L'ensemble des connexions du modèle exécutif définit le nombre de copépodes effectivement connectés à l'environnement. Ainsi, nous posons que V (la variable d'état propre au DS-DEVS) est : $V^{\chi} = \theta$, avec θ le nombre de copépodes connectés. Ce qui implique que tout modèle $Copepode_n$ avec $1 \leq n \leq \theta$ fait partie du réseau actif défini par le modèle exécutif. Si θ est modifié, alors l'ensemble Z^{χ} est modifié également, ce qui fait varier le nombre de copépodes actifs dans le SMA. C'est ce qui se passe sous l'influence d'une transition externe du modèle exécutif à l'arrivée de l'évènement $calcule(N, P)$. D'où la formule suivante :

$\delta_{ext}^{\chi}(S_{\chi}, calcul(N, P)) = (S)$ avec $\theta = P$
$\delta_{int}^{\chi}(S_{\chi}) = \emptyset$ la fonction de transition interne n'est pas définie.
$ta^{\chi}(S_{\chi}) = \infty$

La structure du modèle RAS ainsi formalisé n'est pas très différente de la précédente. Le

système n'est plus isolé et les connexions sont maintenant dynamiques. La figure 3.10 montre la structure du modèle RAS en mettant en évidence les nouvelles connexions nécessaires pour le couplage avec le système d'équations différentielles (nous expliquons le rôle de l'évènement *init* dans ce qui suit).

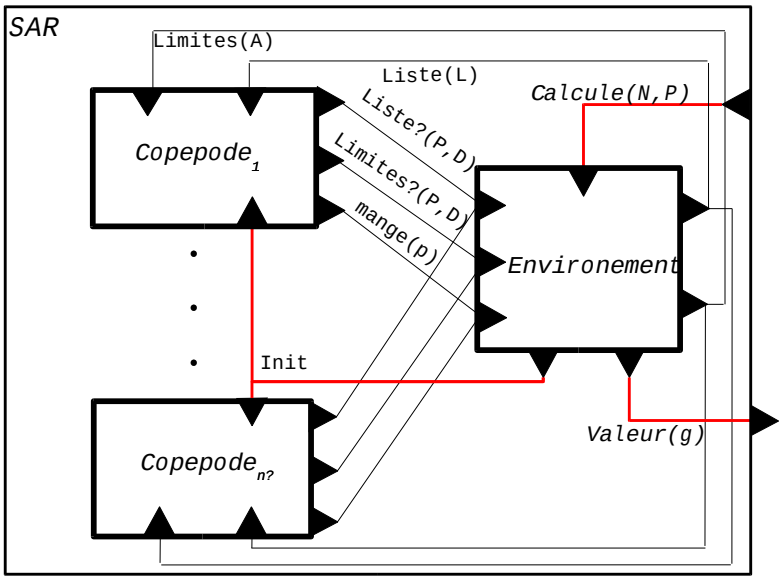

FIG. 3.10 – Représentation de la structure des connexions du model RAS. Les liens en rouge représentent les nouvelles connexions apportées pour le couplage avec le modèle NP. Les trois points indiquent que le nombre de modèles *Copepodes* est maintenant quelconque.

Adaptation des modèles composants le SAR

Le modèle SAR n'étant plus fermé, il reçoit et émet des évènements vers l'extérieur. Il doit maintenant être capable de fournir une valeur de g au système d'équations différentielles. Dans ce système, $G(N, P)$ mesure le nombre de proies qui sont consommées par unité de temps et par copépode (nous reviendrons sur cette fonction en détail au paragraphe 5.2.1). La valeur de $G(N, P)$ au temps t est disponible à tout moment dans le modèle de l'environnement des agents. En effet, il suffit pour cela de compter le nombre de cellules de phytoplancton qui ont été consommées et de le diviser par le temps simulé par le modèle d'agents et par le nombre d'agents copépodes. Pour cela, il est nécessaire que le modèle SAR connaisse le nombre d'agents dans le milieu. De plus, le nombre de cellules de phytoplancton et de copépodes peut varier sous l'influence du système d'équations différentielles. Nous devons donc adapter le modèle SAR pour qu'il puisse calculer $g = G(N, P)$ et générer une fonction de sortie afin de communiquer sa valeur au modèle NP. Cette adaptation consiste à ajouter dans le modèle SAR des états et les fonctions de transitions associées. Cette opération de reformalisation est présentée en annexe E.7.

Comme nous l'avons dit au début de cette section à la page 83, le modèle SAR doit simuler une certaine durée T pour évaluer g. Cette durée est très inférieure au pas de temps d'intégration h du modèle numérique. Ainsi, entre chaque pas de temps d'intégration du modèle NP, le modèle SAR reste dans un état passif en attendant le prochain évènement lui demandant d'évaluer g en

provenance du modèle NP. Pour passer dans cet état passif, à chaque fois qu'un modèle *Copepode* interroge l'environnement, celui-ci vérifie d'abord que le temps simulé n'est pas supérieur au temps nécessaire pour évaluer g. Si le temps de simulation est suffisant, alors le modèle de l'environnement émet une fonction de sortie qui porte la valeur g vers le modèle NP et transite vers l'état passif. La figure 3.11 illustre l'interaction entre le modèle NP et le modèle SAR.

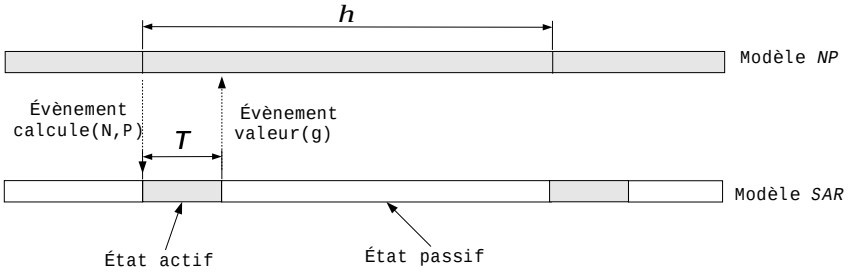

FIG. 3.11 – Représentation de l'échange d'évènements entre le modèle NP et le modèle SAR. Le temps est croissant de la gauche vers la droite. Le modèle NP demande au modèle SAR de simuler une valeur de g à chaque début de pas d'intégration. Le modèle SAR évalue g durant $T << h$, h étant le pas d'intégration du modèle NP. L'état passif du modèle SAR représente l'attente d'un nouvel évènement de demande d'évaluation de g.

Formellement, nous exprimons l'envoi d'autant de valeurs de g qu'il y a de copépodes dans le modèle SAR (voir annexe E.7). Nous avons vu que le modèle NP ne prendra en compte que la dernière valeur de g dans le calcul de sa transition interne.
Ainsi, une fois la dernière valeur de g envoyée, tous les copépodes se trouvent dans un état passif, en attente d'une réponse de l'environnement (lui-même dans un état passif) ; seule l'arrivée d'un évènement demandant le calcul de g peut réactiver le modèle SAR. C'est le rôle de l'évènement *init* envoyé à tous les copépodes par l'environnement lorsqu'il reçoit un évènement externe de demande d'évaluation de g.

Nous venons de spécifier un système d'équations différentielles et un modèle d'agents réactifs de façon à ce qu'ils puissent échanger des données dynamiquement. Un autre point important est la nature de ces données. En effet, elles sont continues pour le premier et discrètes pour le second. De plus, nous avons un modèle mathématique déterministe qui communique avec un SMA de nature stochastique. Il est donc nécessaire, pour finaliser ce couplage, d'introduire un modèle pivot qui gère ces deux aspects. C'est ce que nous allons faire.

Structure du système couplé

Nous allons considérer un modèle pivot entre le modèle RAS et le modèle NP. Nous avons dans le premier modèle une valeur g portée par l'évènement $value(g)$ qui est le résultat d'une simulation stochastique. Dans le deuxième modèle, cette valeur est censée être déterministe, nous allons considérer la moyenne de g donnée par plusieurs simulations de RAS comme un bonne estimation de $G(N, P)$ dans le modèle déterministe (nous justifions ce choix au chapitre 5 page 147). Le modèle pivot est chargé de faire cette moyenne. Ce dernier convertit également les valeurs N et P continues du modèle déterministe en valeurs discrètes pour le modèle SAR. En connaissant les masses m et m' des individus représentées par les quantités N et P continues (respectivement les concentrations en phytoplancton et copépodes), nous pouvons convertir ces

concentrations continues en concentrations discrètes (N/m). Nous donnons ici la structure totale du système couplé à l'aide de la figure 3.12.

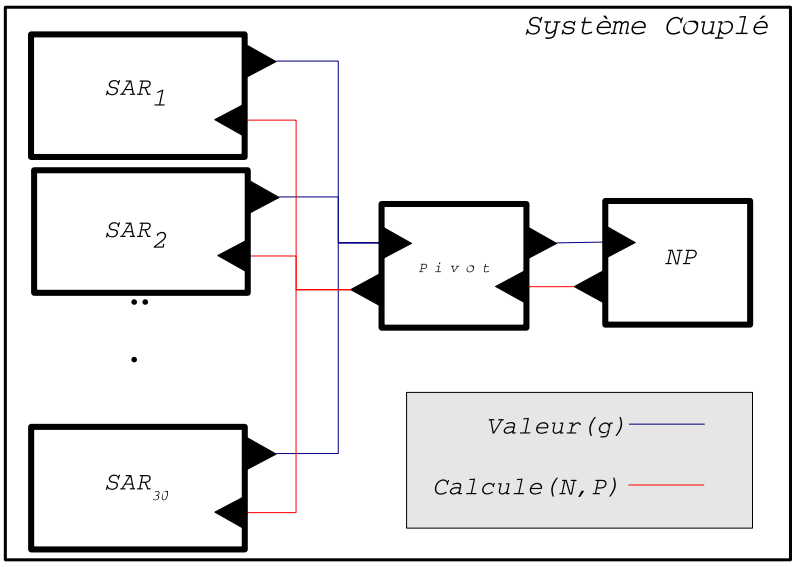

FIG. 3.12 – Structure du modèle de simulation du sytème couplé. Le modèle pivot joue le rôle d'intégrateur des résultats de simulations des SARs. Nous considérons 30 modèles SARs pour le calcul d'une valeur moyenne de g dans notre application (voir partie 5).

La figure 3.12 est un modèle DEVS couplé et fermé. L'ensemble des modèles SARs qui le composent sont des modèles DS-DEVS. Le modèle couplé représente le modèle de simulation tel que nous l'utilisons dans notre application (partie 5). Chaque modèle SAR est simulé dans les même conditions de concentration en proies et en prédateurs. Ces concentrations changent à chaque pas de temps de résolution du système d'équations différentielles.

Nous avons vu à la section 3.3 que DEVS était une sémantique opérationnelle offrant des algorithmes de simulation. Maintenant que nous avons présenté la formalisation du couplage d'un système d'équations différentielles avec un système d'agents réactifs en DEVS, nous discutons brièvement de son implémentation.

3.6.3 Un mot sur l'implémentation

Des simulateurs abstraits ont été développés pour les modèles DEVS et DS-DEVS [ZKP00] [Bar96]. Nous présentons ceux des modèles atomiques et couplés en annexe D.1 et D.2. Pour notre implémentation du modèle couplé présenté dans le paragraphe précédent, nous avons réimplémenté les algorithmes des simulateurs abstraits en les adaptant aux particularités de notre modèle. En effet, dans notre modélisation des interactions entre l'environnement et le copépode par exemple, nous utilisons un enchaînement d'états transitoires et d'états passifs pour simuler la « réponse » instantanée d'un modèle à une « question » posée par un autre (la figure 3.13 illustre cet enchaînement). Ce type d'interaction entre tout à fait dans le cadre des simulateurs abstraits. Il est néanmoins beaucoup plus efficace d'implémenter cette interaction sous la forme d'un simple appel de fonction.

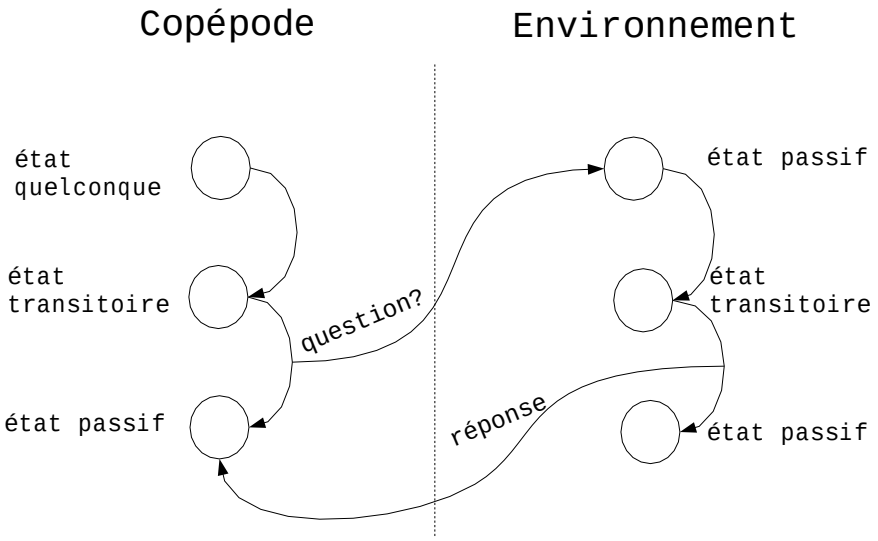

FIG. 3.13 – Interaction de type question-réponse dans la spécification du modèle d'agent réactif. L'implémentation ne considère pas les états transitoires et passifs du copépode, ni l'état transitoire de l'environnement mais relève d'un envoi de message (au sens de le programmation objet), donc d'un appel de fonction.

Comme nous l'avons dit au paragraphe 3.3, l'implémentation de modèles DEVS est facilitée par l'utilisation d'un langage orienté objets. Notre système couplé est donc codé en C++. Ce langage permet notamment de gagner en temps d'exécution par rapport à des langages interprétés comme Java ou Smalltalk. Ainsi, les modèles de copépodes et de l'environnement sont implémentés comme des objets par exemple (et non des agents). L'implémentation de l'interaction question-réponse illustrée figure 3.13 s'effectue simplement avec un envoi de message (au sens de la programmation orientée objet). Dans ce type d'enchaînement, les états transitoires et passifs n'existent en fait que formellement. Il est important de noter que cet « artifice » d'implémentation ne contredit pas la spécification formelle. Ainsi, notre implémentation garantit le comportement décrit par les modèles que nous avons spécifiés dans cette partie. Nous allons maintenant discuter de cette formalisation et de son intérêt.

3.7 Discussion

Dans cette partie, nous avons formalisé un système couplé formé de deux sous-systèmes principaux, à savoir un système d'agents réactifs et un système d'équations diférentielles. Le but de cette intégration formelle est de fournir une écriture unifiée de ce système couplé. Pour cela, nous avons commencé par établir un parallèle entre le paradigme d'agents réactifs et le formalisme DEVS. Ansi, nous avons pu formaliser notre système d'agents réactifs. Nous avons également spécifié en DEVS le système d'équations diférentielles afin de formaliser totalement notre modèle couplé. De plus, les algorithmes des simulateurs abstraits nous permettent de simuler parfaitement les modèles ainsi spécifiés. Pour commencer cette discussion, nous revenons sur le choix de DEVS pour bénéficier d'une sémantique opérationnelle.

Dans la plupart des spécifications de SMAs, nous pouvons distinguer deux types de formalismes : un formalisme pour la structure et un autre pour le comportement. Par exemple, une des propositions de J. Ferber était le formalisme BRIC [Fer95] basé sur une approche modulaire et hiérarchique pour la structure et les réseaux de Petri colorés pour le comportement. Cette représentation est très proche de DEVS. En effet, les réseaux de Petri sont considérés comme une sous-classe de DEVS et la structure modulaire et hiérarchique est un principe fondateur de ce formalisme. Le reproche majeur fait à BRIC (par l'auteur lui-même) est qu'il ne permet pas de formaliser des évolutions de la structure du système (*i.e.* des connexions entre agents, donc des interactions). Au début de ce chapitre, nous avons présenté DS-DEVS, une extension de DEVS qui permet de formaliser les changements dynamiques de structure. Ainsi, ce problème n'existe plus dans les formalisations basées sur DS-DEVS. C'est en posant certains problèmes de la modélisation d'écosystèmes, où les changements de structures spatiales par exemple sont primordiaux [Uhr95], qu'A.M Uhrmacher propose l'utilisation de structures dynamiques pour la formalisation des SMAs [Uhr01].

Nous aurions pu adopter une approche de type génie logiciel pour la spécification de notre modèle. Il existe en effet une extension de UML pour les agents, AUML [OPB00], qui explicite des notions telles que l'autonomie ou la proactivité. À notre connaissance, il existe encore très peu de travaux basés sur ce formalisme [Hug02] alors que DEVS est très largement employé en modélisation et simulation. De plus, AUML apparaît peu adapté au multiformalisme, le comportement interne des agents n'étant pas vraiment formalisé puisqu'il s'agit de méthodes au sens des objets.
Récemment, J. Ferber et O. Guntknecht ont proposé un formalisme basé sur le π-calcul et la *Chemical Abstract Machine* [FG00]. Ce travail fait une classification des SMAs et propose une formalisation pour l'ensemble. Néanmoins, la notion même de dynamique n'est pas très claire. Le π-calcul permet d'exprimer des processus parallèles avec envoi de messages mais le temps n'apparaît pas de façon explicite. De même que pour AUML, nous ne voyons pas clairement comment ce formalisme peut s'intégrer avec d'autres dans la perspective de la spécification de systèmes hybrides complexes.

D'autres formalismes existent et le champ de recherche et d'applications des SMAs est un domaine très vaste. Aussi nous ne prétendons pas trouver le "$\vec{f} = m\vec{\gamma}$" des SMAs[58]. Nous voulons montrer que DEVS est un formalisme qui permet la spécification de SMAs considérés dans un système plus large nécessitant un couplage avec d'autres formalismes. DEVS offre également une approche originale de la gestion du temps dans les SMAs, où le temps est généralement discrétisé.

3.7.1 Sur l'usage de DEVS pour la formalisation des SMAs

Représentation du temps

Dans la grande majorité des SMAs, le temps est discret. Ce choix implique que toutes les actions des agents se font de manière simultanée. Le choix de l'ordre des actions (*sheduling*) est alors souvent confié à un générateur de nombres pseudo-aléatoires. Une autre solution est de faire évoluer l'état de tous les agents au temps $t+1$ en fonction de l'état du système au temps t. Cette technique implique la gestion de conflits entre les agents voulant accéder à une même ressource exclusive par exemple. Quoi qu'il en soit, c'est le « top » de l'horloge globale du système qui

[58] Expression employée par un participant à la conférence JFIADSMA à Saint-Étienne en 2000.

décide quand auront lieu les changements d'états. Les points de synchronisation entre actions sont donc fonction du pas de temps considéré. Cette approche peut être formalisée en DEVS, les systèmes à temps discret pouvant être simulés sans erreur par un modèle DEVS [ZKP00]
Dans une approche à évènements discrets, c'est le système lui-même qui « construit » son temps. En effet, l'avancement du temps est calculé par chaque modèle en fonction de son état. Cette technique ne supprime pas le problème de la simultanéité des actions, mais le système contrôlant entièrement sa dynamique, la simultanéité est uniquement liée aux actions des agents, à la dynamique du système et non pas au « top » d'horloge. Comme nous l'avons vu, la base de temps est réelle ($t \in \mathbb{R}$) dans un modèle à évènements discrets, le temps est considéré comme continu. Cette représentation permet d'effectuer des sauts temporels, ce qui représente une économie de calcul. En effet, c'est uniquement sur un changement d'état que des opérations sont effectuées. Ainsi, nous évitons tout calcul inutile lié au temps discret pour tout agent à qui il n'arrive rien au « top » d'horloge. Cette technique impose une modélisation particulière des interactions puisqu'elle ne se limite plus à une interrogation du voisinage au temps t mais à la mise au point de calcul permettant de prédire ces interactions au temps $t + \Delta t$, avec Δt la durée dans l'état courant. En fait cela revient à consulter l'état du système dans le futur.

Cette notion d'évènement permet également de coupler des modèles qui simulent des systèmes à des échelles de temps très différentes. Comme nous l'avons vu avec le couplage du système d'équations différentielles et du modèle agent, si l'activité du système à dynamique rapide peut être considérée comme constante entre deux évènements du modèle lent, alors il suffit de simuler le modèle rapide sur un temps très court, puis de la faire attendre dans un état passif. Cet état ne nécessite aucun calcul, il est donc possible de faire un saut dans le temps jusqu'à la prochaine transition externe en provenance du modèle lent. Ainsi, les sauts dans le temps sont possibles dans un modèle à évènements discrets, ce qui rend possible un transfert d'échelle. Nous abordons l'utilité de tels tranferts dans notre chapitre applicatif (chapitre 5 page 125).

Représentation de l'espace

Comme nous l'avons dit au début de cette discussion, l'une des difficultés majeures imposées par une spécification à évènements discrets des interactions spatiales entre agents est que ces interactions doivent être prévues à l'avance. En effet, un agent doit être capable de savoir avec qui il entrera en interaction la prochaine fois[59]. Des algorithmes tels que le calcul du point de rencontre d'objets mobiles peuvent être extrêmement coûteux dans une simulation. En effet, beaucoup de calculs inutiles peuvent être effectués pour rien si les objets changent fréquemment de direction. C'est là une limitation à la conception de systèmes d'agents réactifs situés en évènements discrets. Il serait intéressant de savoir à partir de quand une telle représentation ne devient plus possible. Néanmoins, cet argument ne permet pas de rejeter DEVS pour la spécification des agent réactifs situés. En effet, comme nous l'avons dit plus haut, les systèmes à temps discret, où les interactions spatiales se limitent à une consultation du voisinage à chaque pas de temps, peuvent être formalisés en DEVS. Ce formalisme offre même le possibilité de définir des réseaux de cellules complexes avec une dynamique particulière pour chaque cellule.

[59] Ce qui oblige à utiliser des algorithmes proches de ceux utilisés en informatique graphique tel que le lancer de rayon, par exemple

Extension possible aux agents cognitifs

Bien entendu, DEVS peut être utilisé pour la spécification de certains types d'agents cognitifs. L'analogie la plus simple est de considérer les états des modèles DEVS comme support de la mémoire et des états mentaux et les fonctions de transitions comme le support de la cognition, c'est-à-dire des règles agissant sur cette mémoire et ces états mentaux. C'est l'approche adoptée par B. Schattenberg et A.M Uhrmacher [SU01] pour la spécification d'agents planificateurs (qui élaborent des plans d'actions).
Dans un modèle proactif de type générateur, il est possible d'intégrer des règles complexes de comportement pouvant simuler la prise de décision de l'agent, ces règles pouvant être formalisées dans la logique propositionnelle par exemple. De la même façon que les fonctions de transitions peuvent être analytiques, elles peuvent être logiques. DEVS offre un cadre de spécification dynamique qui n'est pas limitant dans la description des fonctions de transition. Ainsi, un formalisme comme celui de P. Cohen et H. Levesque [CL90] pourrait être « embarqué » dans DEVS. Les auteurs ont défini une logique de l'action rationnelle qui a eu beaucoup de retentissement dans le monde des SMAs [Fer95]. Ce formalisme repose sur une logique modale augmentée d'un certain nombre d'opérateurs représentant des attitudes propositionnelles ou, ce qui nous intéresse ici, des séquences d'évènements. L'action est considérée comme un évènement qui rend possible la satisfaction d'une proposition. Une des critiques à l'encontre de ce formalisme est qu'il ne rend pas compte des modifications apportées par les actions sur les agents ou l'environnement. En incorporant ce type de formalisme à DEVS, une telle carence pourrait être comblée.

Symbolic-DEVS [ZC92] permet d'intégrer des symboles dans le calcul de la fonction d'avancement du temps. Ces symboles peuvent être remplacés par leur valeur, ce qui ramène Symbolic-DEVS à un DEVS classique. Ils peuvent également rester des symboles, les valeurs de sortie devenant des fonctions de l'avancement du temps. Le résultat de la simulation sera alors une famille de polynômes, ce qui permet *a posteriori* de tester plusieurs valeurs pour les symboles. Ainsi, en une seule simulation, il est possible d'avoir une représentation symbolique de l'évolution du système. Cette technique peut également être appliquée pour la simulation du raisonnement automatique [Wos88], la famille de polynômes pouvant faire l'objet d'une recherche de satisfaction de contraintes pour la détermination de la valeur des symboles.

Néanmoins, des aspects importants des SMAs, comme l'émergence, ne sont pas du ressort de la formalisation. En effet, cet aspect n'inclut pas seulement une description de la structure des relations entre agents, mais également l'observation de la dynamique du système. Il serait possible de spécifier un agent observateur de structure émergente (voir [Ser00]), mais pas l'émergence elle-même.

3.7.2 Sur l'importance de l'intégration au niveau formel

Un système formel se définit habituellement par une syntaxe (ou langage), définissant la notion de formule bien formée, des axiomes et des règles de dérivation. Les formules bien formées de DEVS correspondent aux modèles atomiques ou couplés, qui doivent répondre à une sémantique précise. Si un modèle est bien formé, il est possible de prouver qu'un modèle couplé est équivalent à un modèle atomique. Ainsi, DEVS garantit la décomposition hiérarchique et la modularité des systèmes. Ces notions sont fondamentales pour une conception multi-échelles. En effet, elles permettent de factoriser des modèles composites en garantissant le comportement global. Il est donc possible d'avoir des formulations du système à différents niveaux d'abstrac-

tion.
Le formalisme DEVS dérive de la théorie des systèmes exprimée par les mathématiques discrètes. On trouve dans le livre référence de Zeigler [ZKP00] la preuve que DEVS spécifie un système formel. C'est sur cette base qu'il devient possible d'y intégrer d'autres formalismes dérivant eux-mêmes de la théorie des systèmes (comme les équations différentielles). Cette intégration formelle permet de construire des modèles complexes dont il est possible de garantir le comportement. En d'autres termes, nous sommes capables de savoir exactement ce qui se passe dans le modèle. Ce dernier point est très important dans des perspectives d'études de scénarios ou d'hypothèses faites sur un système réel à l'aide d'un modèle de simulation. Nous pensons que c'est un point essentiel pour que les modèles complexes tels que les SMAs accèdent au rang d'outils scientifiques reconnus par de nombreuses disciplines. De plus, la formalisation de multi-modèles (hétérogènes) dans un système d'écriture unique améliore la communicabilité, donc l'échange de modèles. Le processus de vérification des résultats par des tiers, élément important de l'activité scientifique, est amélioré.

Lorsque nous formalisons un système, une des premières attentes est de pouvoir inférer son comportement dynamique (le résultat de la simulation) sans même simuler, mais en étudiant le système formel. On sait que, même avec les mathématiques classiques, c'est très généralement impossible aussitôt que le système devient un tant soit peu réaliste [Pav94]. Il en est de même pour DEVS. Il est possible de connaître localement le comportement d'un modèle atomique en dessinant un graphe de transitions d'états par exemple, qui illustre bien la dynamique du modèle. Néanmoins, si le modèle est plus complexe, alors cette représentation (possible dans l'absolu) devient vite incompréhensible et ne donne aucune information sur l'ensemble des états possibles du système. C'est pourquoi la simulation est, assez généralement, le seul recours pour l'étude de la dynamique des systèmes complexes.

Nous observons, en accord avec H. Atlan [Atl86], que les différents niveaux d'organisation du vivant s'articulent autour de disciplines différentes. En d'autres termes, chaque discipline offre sa batterie de concepts et de formalismes pour décrire le réel à un certain niveau d'abstraction. Dans ce cadre, le changement d'échelles peut être vu comme un changement de discours, ou encore de paradigme. Il est vrai que les différents formalismes ne sont pas attachés à un niveau d'organisation particulier, mais sont choisis en fonction de la pertinence de leur utilisation dans un contexte donné. Toutefois, comme nous l'avons vu, le fait de pouvoir coupler différents formalismes permet une représentation unifiée de plusieurs points de vues sur un système (macroscopique avec les équations différentielles et microscopique avec le SMA). Qui dit plusieurs points de vue peut dire plusieurs niveaux d'organisation. Ceci est à mettre en relation avec le principe de décomposition hiérarchique. La décomposition structurelle des systèmes (plus de détails) entraîne souvent une descente dans les échelles spatiales et temporelles des dynamiques considérées (voir figure 1.1 page 4). Ainsi, dans un contexte de multi-formalisation, DEVS offre un cadre multi-niveaux et multi-points de vue.

3.8 Conclusion

Dans cette partie, nous avons proposé le couplage formel d'un système d'agents réactifs situés avec un système d'équations différentielles. Cette formalisation a d'abord nécessité de rendre compte du paradigme d'agents réactifs dans un formalisme le plus général possible (*i.e.*

qui permet de spécifier un grand nombre de systèmes). C'est ce que nous avons fait en proposant une analogie entre les agents, leur environnement et DEVS. Ce formalisme a été choisi à la fois pour sa capacité d'intégration de modèles hétérogènes et pour ses notions de couplage et de décomposition hiérarchique qui autorise une vue multi-niveaux. Nous avons ensuite formalisé le simulateur d'un système d'équations différentielles dans ce même formalisme, ce qui permet de coupler formellement les deux modèles *a priori* hétérogènes. DEVS possède également des extensions, comme le DS-DEVS que nous avons présenté et utilisé dans ce chapitre. Nous pensons que ces extensions sont nécessaires pour la formalisation d'autres types d'agents que les agents réactifs.
En accord avec A.M Uhrmacher, P.A. Fishwick et B. Zeigler [UFZ01], nous pensons que l'adoption de DEVS pour la formalisation SMAs, dans un contexte de modélisation et simulation, peut être un moyen de favoriser les échanges entre les communautés des modélisateurs au sens large et celle des concepteurs de SMAs, par l'adoption d'un langage commun. Comme tout échange de connaissances, il est sûrement bénéfique. Dans un premier temps, l'adoption de DEVS et des très nombreux travaux qu'il a suscités ouvre les portes à trente ans de réflexions sur la modélisation des systèmes. Ces réflexions ont débouché sur des concepts (comme le multi-formalisme), des outils et des algorithmes (comme les simulateurs abstraits) de simulation qui peuvent venir enrichir le champ de recherche des SMAs. Nous donnons quelques exemples d'enrichissements possibles dans le chapitre suivant en présentant par exemple le concept de DEVS-BUS.

DEVS est un formalisme qui, utilisé rigoureusement, peut devenir relativement « verbeux ». Nous en avons eu un exemple dans cette partie (voir également l'annexe E). La rigueur et l'expressivité de DEVS pourraient alors devenir un obstacle pour l'échange et la compréhension des modèles, en obligeant les modélisateurs à adopter une sémantique relativement complexe. Ceci semble en contradiction avec les arguments présentés dans la discussion de ce chapitre. Néanmoins, il semble que ce soit « le prix à payer » pour apporter une formalisation claire des systèmes complexes. Cette complexité, c'est une tautologie, ne peut pas être abordée de manière simple. Il y a donc un besoin très important de formalisation des SMAs pour que ceux-ci deviennent des outils reconnus par les sciences biologiques notamment. Dans ces dernières, la modélisation est souvent un moyen de tester des hypothèses. Une connaissance fine et *a priori* du modèle est donc nécessaire.
Le côté prolixe de DEVS est diminué si l'on considère le principe de décomposition hiérarchique. Il est possible de considérer un modèle DEVS, couplé ou non, seulement du point du vue de ses ports et des évènements qui leur sont attachés. Il y a ainsi une factorisation possible qui facilite la réutilisabilité et la diffusion d'un modèle prédéfini.

DEVS offre également un cadre de spécification qui permet de coupler des modèles hétérogènes. La notion de couplage est inhérente à ce formalisme. Ainsi, l'écriture de modèles dans des formalismes compatibles DEVS permet de définir proprement les interfaces d'échanges de données entre modèles couplés. Néanmoins, il laisse à la charge du modélisateur de faire communiquer des modèles « qui puissent communiquer entre eux ». En d'autres termes, les données échangées sont admissibles et traitables par les modèles en interaction. Dans une perspective de couplages dynamiques et automatisés des modèles, la seule description des structures de couplage et de la synchronisation ne suffit plus. Il est nécessaire d'avoir des descriptions de haut niveau qui permettent de décider si une donnée envoyée par un modèle est admissible par l'autre, et ceci avant d'effectuer une connexion opérationnelle. De plus, la quasi totalité des modèles de simulation implémentés sous l'appellation « SMA » ne sont pas des modèles DEVS. Il apparaît néanmoins

très intéressant de pouvoir coupler ces modèles. Nous proposons un début de réponse à ces deux questions dans le chapitre suivant.