Université du Littoral
Côte d'Opale

École doctorale de l'Université du Littoral - Côte d'Opale

Intégration de modèles hétérogènes pour la modélisation et la simulation de systèmes complexes

Application à la modélisation multi-échelles en écologie marine

Thèse

présentée et soutenue publiquement le 30 mars 2004

pour l'obtention du grade de

Docteur

(spécialité informatique)

par

Raphaël Duboz

Composition du jury

Président :	N. Giambiasi	Professeur (IUSPIM – Marseille)
Rapporteurs :	A. Drogoul	Professeur (Université de Paris VI)
	D.R.C. Hill	Professeur (ISIMA – Clermont-Ferrand)
Examinateurs :	Y. Lagadeuc	Professeur (Université de Rennes)
	P. Preux	Professeur - directeur de thèse (Université de Lille III)
	É. Ramat	Maître de conférence - HdR (Université du Littoral - Côte d'Opale)

Laboratoire d'Informatique du Littoral — UPRES JE-2335

Mis en page avec la classe thloria.

Remerciements

Il est d'usage de commencer ses remerciements par une dédicace à son directeur de thèse. Cette primeur n'est pas ici une simple politesse. J'ai découvert une relation particulière qui lie un directeur de thèse à son thésard, et à travers elle, un homme ouvert et curieux, un scientifique éclairé, qui a su m'encourager et me mener au bout d'une belle aventure humaine. C'est donc avec le plus grand respect que je remercie ici le Professeur Philippe Preux, pour son encadrement et ses conseils.
J'aurais également pu commencer par essayer d'expliquer en quoi Éric Ramat a été la pierre angulaire de mon travail. Il a accepté de se lancer dans le vide avec moi. Son soutient quasi quotidien, nos discussions agitées et toujours constructives, son envie de partager ses connaissances et la grande liberté qu'il m'a laissée, m'ont permis de réaliser ce travail. Il n'y a pas de mot qui puisse exprimer ce que je dois à Éric. C'est en toute amitié que je le remercie ici.

Je remercie très sincèrement le Professeur Alexis Drogoul de l'honneur qu'il m'a fait d'accepter de rapporter cette thèse. Sa lecture à la fois positive et critique m'a permis de prendre plus de recul sur mon travail. Je tiens également à exprimer toute ma gratitude au Professeur David Hill. Ses remarques et ses conseils seront toujours un enrichissement pour moi. Au delà de sa valeur scientifique, c'est aussi un homme généreux que je salue.
Je remercie messieurs les Professeurs Norbert Giambiasi et Yvan Lagadeuc d'avoir bien voulu examiner cette thèse. Ils ont été, l'un comme l'autre, à l'origine scientifique d'une grande partie de mon travail.

En écrivant ces remerciements, je pense évidemment aux membres du LIL, permanents et non permanents, qui m'ont accueilli et soutenu. La « salle des thésards » est encore pour moi autant un lieu de vie qu'un lieu de travail. Merci à Mourad pour sa gentillesse et sa culture, à Laurent pour ses accompagnements nocturnes au laboratoire, à Dominique pour sa disponibilité, à Denis pour nos discussions scientifiques et politiques. Je n'oublie pas tous les autres, même « ceux d'en face » comme on dit avec humour. Sincèrement merci.
Même si il n'est plus au LIL, je ne peux m'empêcher de citer ici Nordine Melab. Il m'a beaucoup donné dans les deux premières années de ma thèse, aussi bien scientifiquement qu'humainement. Un grand merci.

Des personnes étrangères au laboratoire ont également contribué à mon évolution scientifique. Je tiens à remercier particulièrement Frédéric Amblard, Guillaume Deffuant et tous les membres du LISC du Cemagref de Clermont-Ferrand. J'ai là encore une pensée pour David Hill, Norbert Giambiasi et plus récemment Jean-Pierre Müller. Je suis bien sûr reconnaissant envers tous ceux qui ont apporté les briques nécessaires à la construction de ce manuscrit.

Je veux également remercier Romain Réchou pour les dernières corrections orthographiques ainsi que Sébastien Duvivier pour son aide précieuse dans la partie purement mathématique de ma thèse. Sans son aide, mon modèle ne serait pas tel qu'il est.

J'ai une pensée un peu particulière pour Ovide Arino, décédé récemment. Les quelques échanges scientifiques que j'ai eus avec Ovide étaient prometteurs d'une longue coopération. Je termine mes remerciements par une affectueuse pensée pour tous mes proches, famille et amis. Je ne les citerai pas ici, bien qu'ils aient une part plus qu'importante dans mon parcours. Ces remerciements concernent directement ma thèse. Je les remercie pour le reste de ma vie.

À Jémila
à ma mère, ma famille, mes amis

à l'humour sérieux de la recherche...

Table des matières

Table des figures

Liste des tableaux

Avant-Propos

1 Contexte de travail

Le travail présenté ici a été réalisé au sein du Laboratoire d'Informatique du Littoral (LIL), à Calais. Il s'inscrit dans une des thématiques de l'équipe Modélisation, Évolution et Simulation des Systèmes Complexes (MESC). L'axe MESC étudie les systèmes complexes adaptatifs selon deux directions. La première concerne les algorithmes et la programmation génétiques, la structure des paysages adaptatifs et leur utilisation pour l'optimisation combinatoire. La deuxième direction concerne la modélisation et la simulation des systèmes complexes naturels, plus particulièrement marins, en collaboration avec des océanologues.

C'est dans la deuxième « branche » de MESC que s'inscrit cette thèse. Elle est la continuité d'une thématique de recherche qui est née de collaborations avec la station marine de Wimereux. Les premiers résultats de cette collaboration ont été publiés en 1998 [RPSL98]. L'interaction entre informaticiens et écologues fut très féconde en terme de problématiques soulevées par les deux disciplines. La question principale des écologues est de savoir si la turbulence influence la capture des proies par des prédateurs aquatiques microscopiques. Cette question est en apparence très éloignée des préoccupations du chercheur en informatique. Pourtant, il s'est avéré que pour répondre à cette question, une des solutions possibles est la modélisation du comportement alimentaire de l'animal. Cette modélisation peut être effectuée à l'aide des mathématiques mais également de représentations plus « informatiques », comme les systèmes multi-agents. Une autre question des écologues est de savoir si le comportement alimentaire individuel influence le comportement global d'une population. Cette question, dite du « transfert d'échelle », pose des problèmes concrets de formalisation et d'implémentation de méthodes et d'outils informatiques.

Nous avons donc décidé d'utiliser le questionnement des écologues comme support d'une réflexion plus générale sur la modélisation en informatique, et plus particulièrement sur l'intégration de modèles hétérogènes. La modélisation est devenue une méthode largement répandue dans la pratique scientifique et l'informatique apparaît aussi comme la science de la modélisation. Ainsi, nous voyons émerger des « laboratoires virtuels », comme VLE (Virtual Laboratory Environment), développé au sein de l'équipe MESC [RP03]. Cette thèse reprend les réflexions menées dans le cadre de ce développement.

Les problématiques liées à la modélisation et la simulation trouvent leurs places dans différents groupes de travail du CNRS auxquels nous participons, comme MMS (Modélisa-

tion Mulitple et Simulation) du GDR Macs, l'action spécifique VERSIM (VERs une théorie de la SIMulation) et MIMOSA (Méthodes Informatiques de MOdélisation des Systèmes à base d'Agents) du GDR I3.

Dans cette thèse, nous voulons également montrer que la recherche en informatique peut apporter des concepts et des outils utiles aux autres sciences. Nous avons donc choisi de poursuivre et d'étendre nos collaborations avec les écologues. La partie application de notre travail se situe dans le cadre du Programme National sur l'Environnement Côtier (PNEC)[1] et plus particulièrement dans l'Action de Recherche Thématique (ART2), dynamique de populations : structures hydrodynamiques et cycles biologiques. Cette ART est actuellement en cours de restructuration et le LIL, en la personne d'Éric Ramat, devrait y tenir une place encore plus importante. Nous poursuivons donc nos échanges avec le centre d'océanologie de Marseille, plus particulièrement avec J.C. Poggiale, mathématicien, F. Carlotti, écologue, et l'université de Rennes, avec Y. Lagadeuc, lui aussi écologue. Ensemble, nous travaillons sur des aspects plus théoriques et exploratoires de l'apport de la modélisation à base d'agents en écologie marine. Nous en donnons une bonne illustration dans cette thèse.

C'est avec un esprit neuf pour la recherche en informatique et une culture naissante de modélisateur des systèmes vivants que nous avons entamé cette thèse. Nous avons voulu apporter cette culture et celle issue de la théorie des systèmes à l'élaboration d'une thèse centrée sur la modélisation informatique. Ce travail est donc pluridisciplinaire par beaucoup d'aspects. Pour cela, nous essayons de présenter le plus clairement possible l'ensemble des concepts nécessaires à la lecture de ce document. La présentation de la structure du document peut aider le lecteur à mieux cerner notre discours.

2 Structure du document

Premier chapitre : introduction générale

Cette thèse fait appel à un ensemble de références issues de plusieurs branches de l'informatique, mais également de la théorie des systèmes, ou encore d'axes divers de l'écologie. Dans ce contexte, nous avons choisi un plan un peu particulier. En effet, nous ne présentons pas de partie « état de l'art » à proprement parler. Nous nous sommes bien entendu largement inspirés des travaux proches de nos réflexions pour l'élaboration de ce travail, et nous ne manquons pas de les citer. Nous avons simplement préféré introduire les outils et concepts clés là où ils sont utilisés. Néanmoins, nous commençons notre exposé par une introduction générale qui rappelle au lecteur l'ensemble des concepts essentiels à la lecture de ce travail. C'est le premier chapitre de cette thèse.

[1] Information sur le PNEC `http://www.cnrs.fr/cw/dossiers/doseau/recher/program/pnec.html`

Deuxième chapitre : présentation de la problématique

Si nous reprenons le titre de cette thèse (intégration de modèles hétérogènes pour l'intégration de systèmes complexes), plusieurs questions viennent à l'esprit. De quel type d'hétérogénéité parlons-nous ? De quel type de modèles et de systèmes ? Ou encore, pourquoi vouloir intégrer des modèles hétérogènes ? Nous sommes face à une problématique relativement large. Pour répondre à ces questions, nous proposons d'orienter notre travail vers les problèmes formels et opérationnels du couplage de modèles mathématiques avec des modèles informatiques. Dans le deuxième chapitre, nous précisons notre problématique et introduisons une méthode de couplage qui permet de simuler un transfert d'échelles[2]. Nous présentons ensuite le système écologique que nous considérons comme cadre d'étude. Dans le troisième et le cinquième chapitre, nous appliquons la méthode développée ici au système écologique en question. Le deuxième chapitre joue donc le rôle de pivot entre l'introduction générale et les développements plus précis qui suivent.

Troisième chapitre : couplage formel d'un SMA avec un système d'équations différentielles

Ainsi, à partir des modèles et de la méthode de couplage présentés de manière informelle dans le chapitre précédent, nous abordons la question de l'intégration au niveau formel dans le troisième chapitre. Ceci nous amène à proposer un formalisme pour la spécification des modèles d'agents réactifs et pour la spécification d'un système dynamique où cohabitent plusieurs niveaux d'organisations. Le troisième chapitre donne la formalisation complète du modèle utilisé pour notre application au cinquième chapitre.

Quatrième chapitre : intégration opérationnelle

Nous voulons également proposer des solutions opérationnelles pour l'intégration de modèles hétérogènes. Cette question rejoint des problématiques classiques en génie logiciel, où l'hétérogénéité des applications engendre des besoins d'intégrations. Nous abordons ce quatrième chapitre sous un angle différent, celui de l'intégration de simulateurs. Le but ici n'est pas le développement total d'une plateforme logicielle mais la détermination d'un cadre conceptuel et d'outils permettant le couplage effectif des modèles.

Cinquième chapitre : application proie prédateur en écologie marine

Le cinquième chapitre illustre l'utilité de l'intégration de modèles hétérogènes pour la compréhension de la dynamique des systèmes écologiques. Ce chapitre est le plus interdisciplinaire de ce travail. Il fait appel à des notions d'écologie, d'informatique et de mathématique. Nous avons voulu adopter un discours progressif pour permettre au lecteur de mieux cerner les questions purement écologiques. Ce chapitre permet notamment d'utiliser notre méthode de transfert d'échelles pour étudier le système présenté au deuxième chapitre et formalisé au troisième.

[2]Le passage des propriétés existantes à un niveau d'organisation vers un autre niveau d'organisation.

3 Remarque générale avant lecture

Ce document est une intégration à bien des niveaux. Comme nous l'avons vu, il y a le sujet lui-même qui traite de cette question, mais l'intégration va plus loin. Notre formation, à la fois d'écologue et d'informaticien, nous pousse à réaliser un rapprochement des deux disciplines. La difficulté principale est alors de trouver un cadre intégrateur qui supporte des intérêts de chacune. Nous avons trouvé ce cadre dans l'équipe MESC du LIL et le sujet développé ici. Bien entendu, cette thèse reste une thèse d'informatique et nous privilégions les développements dans cette discipline.
Comme tout travail de recherche, il n'est pas le produit d'une évolution linéaire et constante. Les rencontres, les remises en question et l'apprentissage permanent nous font évoluer chaque jour. Ainsi, face à un sujet aussi vaste qu'intéressant, nous proposons un état des lieux précis de l'ensemble de nos réflexions et réalisations. Nous vous souhaitons une bonne lecture.

1

Introduction générale

Résumé

Le travail présenté dans cette thèse, bien que largement centré sur l'informatique, est pluridisciplinaire par bien des aspects. Nous présentons donc un cadre intégrateur de disciplines, la systémique. Ce cadre nous permet d'introduire la notion de hiérarchie des systèmes. Cette hiérarchisation, ainsi que la notion de composition, nous amène naturellement à la complexité. Nous expliquons alors en quoi la complexité n'est pas étudiée ici pour elle-même, mais vue comme une caractéristique des systèmes considérés, les écosystèmes. Nous introduisons également le problème du transfert d'échelle entre niveaux d'organisation.
Un deuxième aspect de notre travail concerne l'hétérogénéité des représentations. Cette hétérogénéité nous amène à vouloir intégrer des modèles différents dans une même simulation, nous parlons alors de multi-modélisation. Cette intégration se traduit le plus souvent par des couplages formels et/ou opérationnels de modèles. Nous présentons une avancée majeure dans ce domaine, HLA, ce qui permet de décrire le cadre technique de cette problématique.

Sommaire

1.1 Définitions

Le travail présenté dans cette thèse concerne la modélisation et la simulation informatique des systèmes dynamiques. Notre réflexion et les développements associés se réfèrent aux notions de système, modèle et modélisation, ainsi que de simulation. De plus, nous avons choisi la modélisation du transfert d'échelle en écologie marine comme cadre d'application, ce qui illustre ces notions dans un contexte pluridisciplinaire. Dans ce premier chapitre, nous donnons un ensemble de définitions et décrivons un ensemble de concepts qui permettent de définir le cadre général de cette thèse.

1.1.1 Systémique

Notre étude s'inscrit dans un courant de pensée qui propose une certaine vision du monde, la systémique. De cette vision découle une pratique particulière de l'activité scientifique. La partie proprement opératoire de l'analyse systémique est née des travaux de V. Bertalanffy [Ber68], N. Wiener [Wei48] ou plus récemment J.W Forrester [For80]. Elle met l'accent sur la notion du système comme faisant partie d'un tout. Jusque là, l'approche scientifique était résolument réductionniste, c'est-à-dire qu'elle procédait par décomposition du réel pour en isoler une partie qui devenait l'objet d'étude. Cette approche a trouvé ses limites dans des perspectives d'explications plus globales où les mécanismes décrits au niveau individuel ne suffisent pas à expliquer le comportement de l'ensemble. Précisons d'abord la notion de système. P.A. Fishwick [Fis95] nous donne la définition suivante :

> « Un système est une partie de réalité où opèrent le temps et l'espace et des relations causales entre les différentes parties de ce système. Nous posons nécessairement des frontières en fabriquant un monde fermé et en identifiant clairement les éléments qui font partie du système et ceux qui l'affectent de l'extérieur. »

P.A. Fishwick parle ici de systèmes physiques, biologiques ou sociaux (différents de systèmes de pensée par exemple). Ces types de systèmes sont effectivement ceux qui relèvent de la systémique. Nous voyons ici que la systémique se veut transdisciplinaire (ou pluridisciplinaire) en essayant de trouver des lois générales indépendantes des contextes d'applications. Cette transdisciplinarité nous intéresse particulièrement dans notre travail.
La définition récente citée plus haut fait intervenir le temps mais surtout l'espace comme opérant sur l'évolution du système. Ainsi, les systèmes sont vus comme dynamiques *et* spatialisés, à la différence d'une vision originale essentiellement dynamique. Dans notre application présentée au paragraphe 2.3, nous nous intéressons principalement au rôle de l'espace sur la dynamique d'un système particulier. Nous retenons donc ici la définition de « système » de P.A. Fishwick.

La notion d'interaction était déjà présente chez les fondateurs de la systémique. Néanmoins cette première vision insistait principalement sur les flux entre les parties (transfert de matière, d'énergie ou d'informations) et les boucles de rétroactions (le système agissant sur lui-même). Dans cette perspective, l'interaction est considérée comme un échange, comme le passage d'une quantité d'un élément du système à un autre ayant pour conséquence une transformation de ce dernier. Une évolution du courant systémique est apparue avec des travaux comme ceux d'E. Morin [Mor77], F. Verla [Ver89] ou I. Prigogine [PS79]. Ces travaux mettent l'accent sur la notion d'auto-organisation des systèmes comme le reflet des interactions des parties. Cette évolution a donné naissance au concept d'autopoïèse (un système auto-organisé tend à garder constante son

organisation[3]). Cette vision rénovée de la systémique a elle-même donné naissance à une néo-systémique, la kénétique [Fer95], qui centre son objet d'étude sur les interactions entre entités et l'organisation qui en résulte dans les systèmes. Dans notre application, nous nous inscrivons plutôt dans le deuxième courant de la systémique en centrant notre étude sur les interactions entre les composants d'un système et ses conséquences sur les propriétés d'un système.

La notion de partie, élément ou entité d'un système est centrale en systémique. Un système étant lui-même une partie du monde, il se pose alors la question de la granularité ou niveau de description que l'on choisit pour représenter un système. De plus, la relation d'inclusion des parties dans d'autres parties amène à une hiérarchisation des systèmes [ZKP00]. Cette notion implique de pouvoir observer un système à différents niveaux d'abstraction ; elle sera centrale dans notre travail.

1.1.2 Échelles – Niveaux d'organisation – Niveaux de description

Échelles, niveaux d'organisation et niveaux de description sont des notions très proches liées à la théorie de la hiérarchie. Cette théorie est apparue à la fin des années 1960, dans la continuité de la pensée systémique [Pat73]. Le mot hiérarchie n'a pas ici le sens d'une structure verticale d'autorité mais plutôt celui d'emboîtement, à l'image de poupées russes. Dans cette théorie[4], un système peut être décrit à différents niveaux qui correspondent à une échelle d'espace et une échelle de temps particulière. Une échelle est un intervalle ou une fenêtre de temps et/ou d'espace caractéristique des éléments et des processus qui constituent un système à un niveau de description donné. Ainsi, la théorie de la hiérarchie a pour but de trouver les points communs et les différences dans la description par niveaux d'organisations dans les systèmes. Cette théorie s'applique également à définir les interactions et les contrôles entre niveaux. Dans cette thèse, nous ne développons pas la théorie de la hiérarchie à proprement parler mais nous contribuons à la réflexion sur les interactions entre niveaux d'organisation. Précisons maintenant ce que sont ces niveaux dans les sciences du vivant.

Niveaux d'organisation et sciences du vivant

E. Laszlo, dans la préface de la réédition de la théorie générale des systèmes de V. Bertalanffy [Ber68], nous dit :

> « Sa conception [à V. Bertalanffy] organiciste se réfère à l'organisme vivant en tant que système organisé et définit la tâche fondamentale de la biologie comme la découverte des lois applicables aux systèmes biologiques à leurs différents niveaux d'organisation. »

[3] Une organisation peut être définie comme un agencement des relations entre composants ou individus qui produit une unité – ou un système – dotée de qualités inconnues au niveau des composants ou individus. L'organisation lie de façon inter-relationnelle des éléments ou événements ou individus divers qui dès lors deviennent les composants d'un tout. Elle assure solidarité et solidité relative, donc assure au système une certaine possibilité de durée en dépit de perturbations aléatoires [Mor77].

[4] Le mot « théorie » est ici usurpé puisqu'il ne s'agit pas d'une certaine explication d'une partie du monde mais plutôt d'une vision particulière qui amène ses propres interrogations. Nous conservons néanmoins ce terme.

Cette vision hiérarchique est reflétée par la décomposition de la biologie au sens large (*i.e.* étude du vivant) en plusieurs disciplines correspondant chacune à un niveau d'organisation donné. De la biochimie (niveau moléculaire), en passant par la biologie cellulaire (niveau des cellules), la physiologie (niveau des organes ou des individus), jusqu'à l'écologie (niveau des grands ensembles d'individus), nous parcourons les différents niveaux d'organisations du vivant. La figure 1.1 présente les échelles de temps et d'espace caractéristiques des différents niveaux d'organisation.

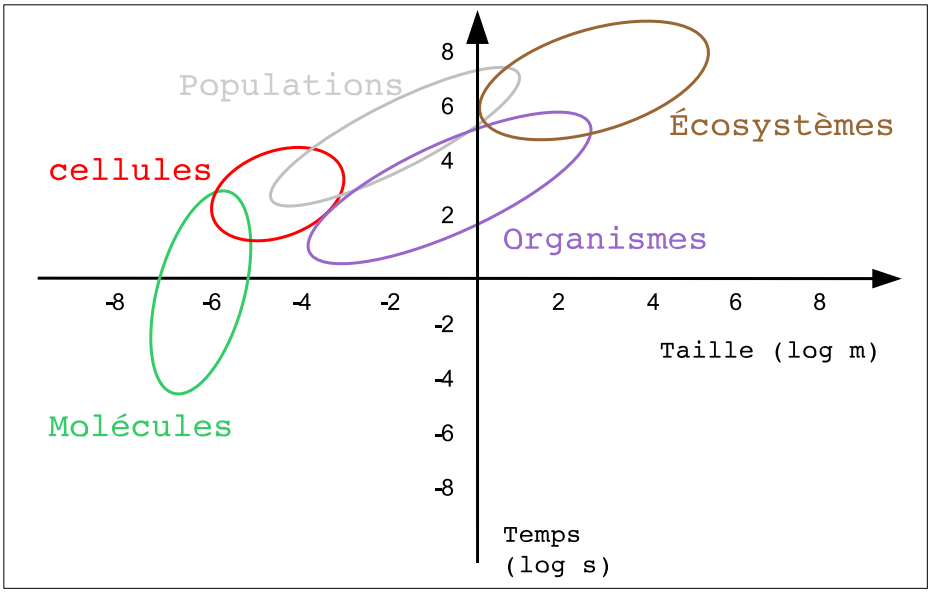


FIG. 1.1 – Échelles spatio-temporelles caractéristiques des systèmes vivants. Nous remarquons que l'augmentation des dimensions selon une échelle entraîne l'augmentation des dimensions sur l'autre. D'après A. Pavé [Pav94].

La figure 1.1 nous montre que, lorsque nous décrivons un système, l'augmentation des dimensions en considérant une échelle entraîne l'augmentation des dimensions pour l'autre. Le temps et l'espace apparaissent comme des grandeurs caractéristiques importantes des systèmes biologiques. En effet, à un niveau d'organisation donné, le temps et l'espace permettent de caractériser les processus et les organisations. S. Frontier va plus loin. Il dit que temps et espace sont structurants pour les systèmes vivants [FPV95], c'est-à-dire qu'ils imposent des contraintes qui ont des conséquences sur la dynamique des systèmes vivants. Nous en verrons un exemple dans le travail présenté ici.

En considérant différents niveaux d'organisation ou de description des systèmes, nous pouvons nous poser la question de la nature des limites entre niveaux. Le découpage effectué par les différentes disciplines reflète un changement dans les méthodes d'études et de discours (de vocabulaire) pour la description des systèmes aux différents niveaux d'organisation. Ainsi le concept de reproduction sexuée entre individus n'a-t-il rien à voir avec celui d'affinité enzymatique au niveau moléculaire qui lui même ne peut pas être décrit par les équations de la physique des particules [Atl86]. Est-ce à dire que les niveaux d'organisation sont totalement étanches les uns aux autres ? Si nous considérons un assemblage de cellules qui composent un organe, nous sommes bien obligés d'admettre que cet assemblage est à l'origine des propriétés de l'organe et

de son rôle dans l'organisme de l'individu. Ainsi, la nature semble « Une » et pourtant, nous ne pouvons pas la décrire comme un tout dans un seul vocabulaire, avec une seule méthode. Cette multiplicité des moyens de description dans un contexte multi-échelles est une question qui va également nous intéresser dans cette thèse.

Transfert d'échelles et émergence

Depuis la fin du 19^{e} siècle, les biologistes portent un grand intérêt à ce qu'ils nomment les lois d'échelles (*scaling laws*) [BWE00]. Ils ont remarqué que, des bactéries jusqu'aux éléphants, les organismes suivent des lois d'échelles remarquablement simples. Ces lois, élaborées empiriquement, relient l'évolution de quantités comme le taux métabolique (la variation de la consommation d'énergie par l'organisme), la durée de vie ou la fréquence de battement du cœur, à des longueurs, des surfaces ou des volumes ; nous pouvons prendre comme exemple la longueur de l'artère aorte ou la taille d'un individu. Ces lois sont qualifiées d'allométriques. Elles ont très généralement la forme suivante :

$$Y = Y_0 M^b$$

où Y est une variable biologique observable, Y_0 une constante, M la masse et b le facteur d'échelle.

Les lois allométriques sont des lois de puissance (*power laws*) et sont très utilisées en biologie. Par exemple en 1930, M. Kleiber et S. Brody montrent que le taux métabolique cité plus haut varie en fonction de la masse des organismes à la puissance 3/4 . En fait, il est surprenant de noter que la plupart des lois allométriques relatives à la masse d'un individu ont un facteur d'échelle b multiple de 1/4. Le caractère universel de ces fonctions nous dit quelque chose sur la façon dont s'organisent les êtres vivants et sur les contraintes qui les font évoluer [BWE00]. Le problème de ces lois est qu'elles sont empiriques et ne disent rien des mécanismes qui contrôlent les différentes variables descriptives entre les différents niveaux d'organisation. Néanmoins, les lois d'allométries montrent que les différents niveaux d'organisation sont interdépendants et ont des propriétés communes. Il peut être intéressant d'aller un peu plus loin dans la compréhension des interactions entre niveaux d'organisation.

Comme nous l'avons dit plus haut, la théorie des hiérarchies étudie les relations entre les différents niveaux d'organisation. Dans cette théorie, chaque niveau est soumis à un contrôle hiérarchique, c'est-à-dire qu'un niveau d'organisation donné est partiellement contrôlé par les niveaux inférieurs et supérieurs. Cette idée implique que les différents niveaux « agissent » les uns sur les autres. Nous appelons cette action inter-niveaux organisationnels un « transfert d'échelles » car elle implique le passage de contraintes entre niveaux. Plus loin dans ce travail, nous proposons une méthode pour exprimer ces contraintes.

Le transfert d'échelle est à mettre en relation avec un concept développé par la pensée systémique. Il s'agit de l'émergence. Ce concept, souvent illustré par la phrase « le tout est plus que la simple somme des parties » peut être défini simplement. Un phénomène est dit émergent si (définition inspirée de [Mül]) :

- la dynamique des entités interagissantes dans un système à un niveau d'organisation donné produit un phénomène global observable à niveau d'organisation hiérarchiquement supérieur,

– ce phénomène global est observé et décrit dans un vocabulaire ou une théorie distincte de la dynamique sous-jacente.

En d'autres termes, quand la seule description des entités et des phénomènes qui les affectent ne suffit pas à expliquer les dynamiques ou les fonctionnalités nouvelles observées à l'échelle de l'ensemble des entités, nous parlons d'émergence. Même dans le cas où nous avons affaire à des machines déterministes (ou à des modèles déterministes de systèmes naturels), dès que leur fonctionnement ne peut être décrit que par des systèmes d'équations ou tout autre formalisme assez complexe, nous ne pouvons pas savoir à l'avance quel sera le résultat du calcul de ces équations. Les solutions peuvent être « étonnantes », parce que contre-intuitives. Souvent, ce calcul n'est faisable qu'au coup par coup, sur un ordinateur, de telle sorte que le résultat n'est plus très différent de celui d'une expérience. Nous reviendrons plus tard sur ce point. Nous voulons noter ici que l'émergence et un concept fortement lié à l'observateur et au langage qu'il utilise pour décrire un phénomène à un niveau donné.
Même dans le cas de l'ordinateur, la mise en relation de son état physique et de la fonction logique d'un programme en cours d'exécution est pratiquement impossible en raison des multiples niveaux d'organisation qui séparent le logiciel des composants électroniques. En accord avec H. Atlan [Atl86], nous comprenons pourquoi la philosophie réductionniste est battue en brèche non pas par l'existence de mystères ou de difficultés non résolues, mais au contraire par le succès de l'outil informatique appliqué à l'analyse et la synthèse d'organisations artificielles complexes où des comportements nouveaux émergent des interactions entre les entités composant le système.

La hiérarchie en niveaux d'organisation des système naturels et artificiels nous amène à la question du nombre de ces niveaux. Chaque niveau est abordé par des langages de description et des méthodes d'investigation particuliers. La frontière entre deux niveaux peut être séparée en deux par l'apparition d'une nouvelle discipline. L'exemple le plus spectaculaire est celui de la biochimie qui fait le pont entre chimie et biologie. Ainsi, le nombre de niveaux peut sembler infini puisque que la séparation en deux d'un niveau particulier peut potentiellement avoir lieu. Nous nous retrouvons dans la situation énoncée par les fameux paradoxes de Zénon d'Élée (v$^{\text{ème}}$ siècle. avant J.C.) qui nous disent qu'il est toujours possible de séparer un segment en deux et donc que le segment n'a pas une longueur finie. Nous voyons que certaines limites de la science ne se situent pas seulement dans l'infiniment grand ou dans l'infiniment petit, mais également dans les frontières ou articulations entre niveaux d'organisation.
La compréhension des systèmes est également rendue difficile par le nombre des composants et des interactions entre ces composants. La notion de complexité exprime cette difficulté, nous la présentons maintenant.

1.1.3 Systèmes complexes

« Complexité » fait partie des mots à la mode aujourd'hui. Nous ne pouvons l'évoquer en feignant d'ignorer l'extrême confusion qui l'affecte. Il n'existe pas réellement de théorie de la complexité à proprement parler, bien qu'il existe une riche pensée de la complexité [Mor77] et une abondante activité intellectuelle sur les systèmes complexes[5]. La théorie des systèmes complexes reste centrée sur le système et la complexité n'apparaît pas comme une propriété étudiée pour elle-même. Toutefois, il existe des définitions de la complexité qui cherchent à en donner

[5]La collection du *Santa Fe Institute Studies in the Sciences of Complexity* est un des exemples concrets de cette activité.

des mesures. Nous en donnons ici un exemple.

Mesures de la complexité

La théorie de la calculabilité a permis de formuler des mesures de complexité des objets finis. Ainsi, la complexité algorithmique donne une mesure de la variation du temps de calcul en fonction de la taille des données en entrée d'un algorithme. Cette mesure est effective est permet de comparer l'efficacité de deux algorithmes pour la résolution d'un même problème par exemple. Néanmoins, cette mesure devient difficile à calculer pour des algorithmes un tant soit peu compliqués (*i.e.* longs à décrire).

En 1965, Kolmogorov donne une mesure de la complexité [LV97]. Cette mesure s'applique à une chaîne de symboles binaires. Nous en donnons ici la définition.
Soit U un algorithme de décompression, alors la complexité $K_U(x)$ d'une chaîne binaire x en considérant U est :

$$K_U(x) = min\{|y| \mid U(y) = x\}$$

où $|y|$ désigne la longueur d'une chaîne binaire y.

En d'autres termes, la complexité de x est définie par la plus petite chaîne y décrivant x en utilisant l'algorithme U. Cette mesure est souvent associée à la taille du plus petit algorithme décrivant l'objet (ici la chaîne de symboles). Cette mesure a été étendue par C. Bennett avec sa mesure de la complexité organisée ou en profondeur. Elle rend compte du temps de calcul de l'algorithme minimal décrivant une suite de symboles. Elle ajoute la notion de temps de calcul à la complexité de Kolmogorov. Nous pouvons également citer l'entropie thermodynamique comme mesure du désordre, de la non-similarité et donc de la complexité. Nous ne traitons pas de ce type de complexité dans notre travail.

Complexité descriptive

Il existe plusieurs formes de complexité. Nous pouvons ajouter aux notions décrites précédemment d'autres notions non formelles comme l'imprédictabilité, l'incertitude (imprévisibilité probabiliste ou statistique), l'indécomposabilité ou l'incontrôlabilité d'un système qui peuvent entrer dans un discours sur la complexité [CPM94]. L'abondance des définitions nous fait penser que la complexité n'est pas une propriété intrinsèque des objets observés. Elle apparaît plus comme une propriété de la relation entre l'observateur et l'objet observé. Un exemple simple est celui de la suite de nombres pseudo-aléatoires générée par un algorithme particulier. Pour un observateur humain, cette suite apparaît « complexe » (sans ordre apparent). Néanmoins, elle est « simple » au sens de Kolmogorov puisque déterministe et générée par un algorithme qui peut être de petite taille. La complexité apparaît donc comme une notion relative et le besoin d'une théorie de la complexité reste entier.

Dans notre travail, nous sommes proches de la définition de Von Neumann et Burks [NB66]. Nous qualifions un système de complexe si :

> l'ensemble des comportements possibles du système considéré est très difficile (ou impossible) à caractériser à partir de ses règles de fonctionnement.

Nous qualifions cette définition de descriptive. Pour être concret, considérons un système de 100 composants pouvant être dans 2 états différents. Un tel système peut se trouver dans 2^{100} ou 10^{30} configurations différentes. Si nous voulons décrire un tel système comme une fonction de ses composants, nous devons étudier ses 10^{30} configurations possibles sachant qu'environ 10^{19} secondes se sont écoulées depuis le début de l'univers ! Cet exemple, emprunté à R. Levins [Lev73], nous fait comprendre que la complexité caractérise une limite à notre compréhension. Ainsi, les solutions adoptées pour décrire les systèmes qualifiés de complexes sont souvent holistes, c'est-à-dire qu'elles considèrent le système comme un tout à un certain niveau d'abstraction. Même les approches réductionnistes sont obligées de s'arrêter à un certain niveau hiérarchique élémentaire pour adopter une description holiste des éléments constituant le système considéré.
Dans notre travail, nous participons à l'étude d'un archétype de système complexe, l'écosystème. Dans ce qui suit, nous donnons quelques généralités sur la relation entre l'étude des écosystèmes et la notion de complexité.

Écosystèmes et complexité

L'écologie moderne a pour objet l'étude des êtres vivants dans leur milieu (y compris l'Homme) et pour méthodes non seulement la méthode expérimentale mais aussi les méthodes de nombreuses autres sciences avec lesquelles elle travaille : mathématiques, chimie, paléontologie, géographie, économie, informatique... Elle a une dimension pratique qui en fait aussi une science appliquée avec une dimension politique et sociale. L'écologie peut être définie comme la science des écosystèmes. S. Frontier définit un écosystème comme étant composé d'un environnement physico-chimique (le biotope) et d'un ensemble de communautés. Une communauté est l'ensemble des populations, c'est-à-dire des individus d'une même espèce. Cet ensemble de communauté s'appelle une biocœnose. Ainsi, un écosystème se définit de la façon suivante [FPV95] :

Écosystème = biotope + biocœnose

L'écosystème est une unité fonctionnelle en écologie, c'est-à-dire une entité qui évolue en permanence et de manière autonome sous l'influence des flux d'énergie et de matière qui la traverse. Cette unité peut être décomposée en plusieurs niveaux hiérarchiques d'organisation. Nous pouvons considérer que le niveau le plus bas en écologie est l'individu et le niveau le plus haut l'écosystème. Dans les faits, un écologue qui travaille au niveau individuel ne peut ignorer totalement les mécanismes et constituants biologiques qui constituent l'individu aux niveaux inférieurs (organes, cellules, etc...). L'écologue travaille à de nombreux niveaux d'abstraction pour représenter et comprendre la dynamique d'un écosystème. De plus, chaque niveau est constitué par un très grand nombre d'entités et de relations entre ces entités qui font des écosystèmes un archétype de système hiérarchique complexe. Le lecteur intéressé par une discussion précise sur les écosystèmes dans le contexte des systèmes complexes peut se référer notamment à un article synthétique de J.H. Brown [Bro94].
Les relations les plus étudiées par les écologues sont les relations trophiques (qui se rapportent à la nutrition). C'est ce type de relations, qui est principalement à l'origine des flux de matière et d'énergie dans les écosystèmes, que nous allons considérer dans la partie applicative de notre travail.

La complexité des écosystèmes et les grandes échelles de temps et d'espace les caractérisant sont des limites à l'utilisation d'une approche purement expérimentale en écologie[6]. Pour

[6]L'approche expérimentale est nécessaire en écologie, elle se traduit souvent par des campagnes d'échantillonnages sur le terrain.

appréhender cette complexité, il est nécessaire de construire des abstractions qui reflètent le fonctionnement de ces systèmes et nous permettent de mieux les comprendre.

1.1.4 Modélisation et simulation

Démocrite avec son modèle de l'atome ou Platon avec son « monde des idées » faisaient déjà de la modélisation. La compréhension du réel passe évidemment par le filtre de notre pensée qui construit des abstractions du monde sensible. Il y a même des thèses qui avancent que l'intelligence elle-même procède par modélisation et/ou simulation, c'est-à-dire par construction d'une image stylisée et dynamique du réel, puis création de scénarios sur la base de ce modèle pour se plonger dans l'avenir et prendre des décisions [Dup94]. Cette thèse a l'intérêt majeur de nous faire percevoir le lien entre notre propre mode de fonctionnement intellectuel, d'ordre intuitif, au moins partiellement, et l'activité scientifique, rigoureuse et démonstrative, qui finalement engendre des modèles qui se veulent surtout communicables. Le but n'est pas ici d'entrer dans une discussion propre à l'épistémologie, mais de dire que finalement, la notion de modèle fait partie intégrante des sciences. Seulement, selon la discipline, voire le contexte historique, la définition de modèle et de modélisation peut varier notablement. Pour avoir une réflexion sur ces définitions, il est intéressant de considérer les travaux récents de F. Varenne. Il discute ces notions dans le contexte actuel de « l'informatisation des sciences ». F. Varenne rappelle que, dès 1979, A.A.B. Fritsker répertorie 21 définitions de simulation, confirmant la relation forte entre la définition et le contexte d'utilisation des termes [Var01].
C'est pourquoi, comme dans tous travaux traitant de modélisation et simulation, nous devons donner les définitions de « modèle » et de « simulation » propres à notre discours. Nous commençons par en citer trois, contemporaines et proches de nos activités.

1. Pour A. Pavé [Pav94] : « Un modèle est une représentation symbolique de certains aspects d'un objet ou d'un phénomène du monde réel. »
2. Pour J. Ferber [Fer95] : « Un modèle, en science, est une image stylisée et abstraite d'une portion de réalité. »
3. Pour P.A. Fishwick [Fis95] : « Modéliser c'est décrire la réalité sous la forme d'un système dynamique, à l'aide d'un langage de description, à un certain niveau d'abstraction. »

Les deux premières définitions nous rappellent qu'un modèle représente le monde réel à l'aide d'une symbolique, c'est-à-dire d'un ensemble de signes. La troisième définition introduit la notion de niveau d'abstraction ou hiérarchique dont nous avons parlé précédemment. Nous complétons ces définitions par celle de Minski, datant de 1965, et souvent citée dans les mémoires de thèse de doctorat ou les habilitations à diriger des recherches en modélisation [Mil00] [Ser00] [Hil00].

> « *To an observer B, an object A^* is a model of an object A to the extent that B can use A^* to answer questions that interest him about A* »

Comme le remarque très justement D.R.C Hill, cette définition met l'accent sur le fait que le modèle doit nous permettre d'apprendre quelque chose sur le système modélisé [Hil00]. De plus, lorsqu'il est formalisé, le modèle doit pouvoir servir de vecteur de communication et d'échange ou encore de cadre d'étude pour un système particulier.
Dans la construction du modèle, nous apprenons des choses sur le système et nous identifions les limites de notre connaissance du système. Ensuite, c'est la simulation qui permet de répondre aux questions sur le fonctionnement et la dynamique du système. En informatique, la simulation correspond à la résolution pas à pas d'équations mathématiques dont on ne connaît pas la

solution ou de moteurs d'inférences à base de règles (réalisation d'automates). Ainsi, la simulation correspond au modèle plongé dans le temps et/ou dans l'espace [CH97].

1.2 Paradigmes de modélisation

À l'origine, « paradigme » désigne un mot type donné comme modèle pour une conjugaison, une déclinaison[7]. Cette définition, qui renvoie à la notion de modèle, a évolué dans le contexte de l'activité scientifique pour devenir « le référentiel d'un système de pensée » ; c'est-à-dire une certaine conception du monde, des enjeux et des méthodes, d'une discipline scientifique considérés comme valables par ceux qui en sont les praticiens [Kuh72]. Nous pouvons alors parler d'un paradigme de modélisation comme étant l'ensemble des définitions et formalismes, des méthodes, des outils et techniques qui caractérisent une activité de modélisation. Par exemple, nous parlons du paradigme objet, caractérisé par les notions d'encapsulation, d'héritage et de polymorphisme. Ce paradigme est formalisé par l'U*nified* M*odelling* L*anguage* (UML) et implémenté à l'aide de langages de type objet comme Java, C++ ou Smalltalk. Ce paradigme se prête très bien à une modélisation discrète des entités d'un système et des scénarios d'interactions entre ces entités [Hil96].

Il existe beaucoup de paradigmes de modélisation. Parmi les plus connus, nous pouvons citer les modèles stochastiques, engendrant des simulations dites de « Monte Carlo » et initiées par V. Neumann. Ce type de modèles a engendré les modèles de type automates cellulaires (comme le jeu de la vie de Conway par exemple). Nous pouvons également citer les algorithmes évolutionnaires ou les techniques d'apprentissage comme les réseaux de neurones ou le Q-learning. Même si ces techniques n'entrent pas parfaitement dans notre définition de « modèle » (elles n'essaient pas de représenter les processus et les entités du système, mais son activité), elles peuvent permettre de simuler effectivement le fonctionnement de systèmes réels.
Dans ce qui suit, nous présentons seulement les paradigmes que nous allons considérer dans cette thèse, les équations différentielles et les systèmes multi-agents, considérés comme des modèles individus-centrés.

1.2.1 Équations différentielles

Une équation différentielle met en jeu une fonction inconnue y, ses dérivés jusqu'à un ordre n donné, et, explicitement ou non, une variable, par exemple x. La forme la plus générale que l'on puisse donner est :

$$F(x, y, y', y'', ..., y^{(n)}) = 0$$

La solution, ou intégrale, de l'équation correspond à toute fonction $y(x)$ qui satisfait la relation identiquement, c'est-à-dire pour toute valeur de la variable x. Nous n'allons pas décrire ici les différents types d'équations différentielles. Nous voulons seulement insister sur certains des aspects de ces systèmes qui nous intéressent.

Dans le contexte de la modélisation des systèmes dynamiques, les équations différentielles permettent de représenter l'évolution d'une quantité au cours du temps. Cette quantité représente une caractéristique globale du système considéré, comme une température, le nombre

[7] Le nouveau petit Robert, 1995

d'individus d'une population, une vitesse d'ensemble, etc. Le plus souvent, les équations différentielles correspondent à un « modèle agrégé », c'est-à-dire un modèle qui représente l'ensemble des mêmes entités composant le système par une grandeur moyenne, correspondant à une modélisation descendante (ou *top down* dans la terminologie anglo-saxonne).
Les équations différentielles représentent l'outil privilégié de nombreuses disciplines scientifiques pour la modélisation de systèmes dynamiques, notamment en écologie [Pav94]. D'un point de vue informatique, ce type d'outil a plusieurs avantages :

- il présente un intérêt certain depuis de nombreuses années pour les mathématiciens et donc bénéficie de grandes avancées méthodologiques. Ainsi, dans le cas fréquent où il n'est pas possible de résoudre analytiquement les équations, il est possible d'utiliser une des nombreuses techniques de résolution numérique mise au point,
- les équations différentielles manipulent des scalaires (grandeurs continues) qui caractérisent le système modélisé. Ainsi, elles peuvent représenter un système d'un point de vue phénoménologique, c'est-à-dire de ses manifestations externes et mesurables,
- cet outil permet donc d'augmenter le nombre d'entités du système considéré sans aucune charge de calcul supplémentaire.

Les équations différentielles s'adressent à un grand nombre de systèmes. Seulement, comme tout paradigme, elles imposent un point de vue sur les systèmes. En effet, les grandeurs représentent souvent les ensembles comme des agrégats avec des caractéristiques communes. Nous parlons dans ce cas de variables agrégées. Cela peut poser des problèmes, notamment en ce qui concerne les interactions entre composants du système, considérées comme continues et réparties de façon homogène dans l'espace. Cette dernière hypothèse de modélisation s'adapte mal aux systèmes de haut niveau hiérarchique comme les écosystèmes où les systèmes sociaux. Il est par exemple difficile de prendre en compte les évènements ou les perturbations ponctuels, ou alors seulement par une réinitialisation des variables et/ou paramètres du système.

Un des intérêts majeurs des systèmes d'équations différentielles est de pouvoir mettre en relation l'évolution dynamique de plusieurs grandeurs. Ces évolutions peuvent être étudiées théoriquement, nous revenons plus loin sur cet aspect. Ces études sont à la base des travaux en écologie théorique. Elles ont permis de mieux comprendre les équilibres dynamiques comme la stabilité quantitative ou la stratégie de résilience (permanence du réseau d'interaction) qui sont à la base de la permanence des écosystèmes [FPV95]. Néanmoins, ces études théoriques sont le plus souvent impossibles en raison du très grand nombre d'interactions et d'espèces [Pav94]. Alors, la résolution numérique des systèmes d'équations différentielles, c'est-à-dire leur simulation, devient le seul moyen d'explorer leurs dynamiques. Ainsi, la simulation est souvent le seul moyen de vérifier les hypothèses de fonctionnement émises sur un système complexe.

Avec l'augmentation de la puissance de traitement de l'information *via* les ordinateurs, il est maintenant possible d'adopter une autre posture de modélisation. Si nous considérons les entités composant le système comme l'élément de base de la modélisation, alors nous avons une vision ascendante sur le système (ou *bottom up*). En écologie, cette nouvelle approche a donné naissance aux modèles individus-centrés (ou IBM pour Individual Based Models).

1.2.2 Modèles individus-centrés et écologie théorique

Comme son nom l'indique, un modèle individus-centré représente explicitement les individus d'un système donné. Ce terme nous vient de l'écologie numérique qui dès 1970 a centré l'objet de ses modèles sur l'entité de base des écosystèmes, les individus. C'est essentiellement en raison

de l'augmentation de la puissance de calcul des ordinateurs que ce type de modèle est très utilisé en écologie depuis une quinzaine d'années [HDP88] [Gri99]. Dans cette discipline, ces modèles sont encore très généralement formalisés par des systèmes d'équations différentielles.
Il est difficile de définir exactement où commence et où finit l'ensemble des modèles individus centrés. Il y a un *continuum* entre des modèles qui s'intéressent à la représentation du fonctionnement biologique d'un individu moyen par exemple, et d'autres représentant l'ensemble des individus d'une population ou d'une communauté (*i.e.* un ensemble de populations) [CGW00]. Le lecteur intéressé peut se référer au volume 115 de la revue *Ecological Modelling* parue en 1999, qui concentre un grand nombre d'articles de réflexions de fond sur les IBMs (dont le célèbre article de V. Grimm [Gri99]).
Une des principales utilisations des IBMs est l'étude de la variabilté individuelle. Cette étude peut avoir deux motivations différentes, une pragmatique et une « paradigmatique » [Gri99]. La première considère simplement les IBMs comme un outil permettant d'étudier des phénomènes qu'il serait impossible d'étudier avec un modèle à variables d'états agrégées. La deuxième motivation reflète que les approches classiques, qui ont donné les notions d'équilibres dynamiques ou de résilience par exemple, sont jugées insuffisantes, et que la variabilité individuelle est suspectée de jouer un rôle prépondérant dans la dynamique des systèmes écologiques [GWAU99].
Des études ont été menées sur la variabilité individuelle [Uch99]. Elles ont montré que pour un système donné, il peut exister un très grand nombre de dynamiques globales fonction des variabilités individuelles. Ceci nous apprend que la simulation individus-centrée ne peut vraisemblablement pas être prédictive. En revanche, elle peut nous offrir un ensemble de possibles qui constituent une base de tests très intéressante pour l'étude d'hypothèses posées *a priori* sur les systèmes.

Ainsi, la modélisation individus-centrée apparaît actuellement comme un paradigme de modélisation de premier plan en écologie théorique. De plus, les IBMs trouvent une expression « plus naturelle » dans des travaux relativement récents en informatique, comme la programmation orientée objets ou les systèmes multi-agents que nous allons présenter. Ainsi, les représentations des dynamiques individuelles peuvent s'écarter d'un formalisme différentiel pur.

1.2.3 Systèmes multi-agents

Les Systèmes Multi-Agents (SMAs) s'inscrivent dans un courant rénové de la systémique dont nous avons parlé plus haut, la kénétique [Fer95]. Ce courant met particulièrement l'accent sur le rôle des interactions et surtout des modes d'organisations dans la description et la compréhension de la dynamique des systèmes auto-organisés. En fait, les acteurs de ce champ de recherche s'attachent surtout à une description fine de ces interactions et à trouver des méthodologies de conception de modèles auto-organisés comme l'étho-modélisation par exemple [Dro93]. Ce type de méthode utilise l'émergence (i.e. l'apparition) de dynamiques nouvelles du système au niveau global à partir d'une modélisation effectuée à un niveau hiérarchiquement inférieur. Cette approche fait partie de l'Intelligence Artificielle Distribuée (IAD). Ainsi, nous pouvons trouver des méthodes de résolution de problèmes par émergence (par exemple [Mül]) qui proposent généralement une analyse réductionniste et une modélisation/simulation de cette réduction pour retrouver le phénomène global.

Au regard de la diversité des applications utilisant les SMAs, il est encore difficile aujourd'hui de donner une définition unique et unanime des agents. Néanmoins, les auteurs s'accordent sur

les principales caractéristiques de tels systèmes[8]. Nous partons de la définition générale d'un agent donnée par G. Weiss [Wei99] :

> « Les agent sont des entités autonomes qui peuvent être vues comme percevant leur environnement à l'aide de récepteurs et qui agissent sur l'environnement à l'aide d'effecteurs. »

Les aspects importants ici sont les notions d'environnement et d'interactions. Pour compléter la définition précédente, M.N. Huhngs et M.P. Singh [HS98] donnent les caractéristiques majeures suivantes pour les SMAs :

- chaque agent a une connaissance incomplète du système, il est limité dans ses capacités d'actions,
- le contrôle du système est distribué,
- les données sont décentralisées,
- les calculs sont asynchrones.

Un des apports les plus importants du paradigme d'agent est la modélisation explicite des interactions en deux catégories bien différenciées [Fer95] :

- les interactions directes : les agents interagissent ou communiquent par échanges de messages ;
- les interaction indirectes : les agents interagissent ou communiquent *via* l'environnement.

Il est intéressant de noter que ces définitions ne sont pas formelles (nous discutons de la formalisation des agents à la section 3.4 page 59). Elles décrivent en fait les SMAs comme des ensembles d'entités discrètes interagissantes.
Dans une perspective plus informatique, nous pouvons dire que le paradigme objet se prête bien à l'implémentation opérationnelle d'un SMA. Il faut tout de même noter que des notions comme la coopération ou l'autonomie ne font pas partie du paradigme objet. De ce fait, lorsque nous employons l'expression « paradigme agent » dans ce document, nous nous référons aux définitions citées plus haut.
Comme tout paradigme, celui d'agent reflète un point de vue sur un système. Nous pouvons donc nous poser la question de savoir à quel type de sytème ce paradigme s'adresse plus particulièrement. Au regard des définitions que nous avons données, nous pensons que la modélisation par agent est bien adaptée aux systèmes qui prennent en compte explicitement des entités hétérogènes situées (dans un référentiel quelconque) ayant des interactions difficiles à décrire. Les systèmes physiques perturbés, les écosystèmes et les sociétés humaines observées à méso-échelle (familles, réseaux sociaux, etc.) sont des exemples de tels systèmes.

Les domaines d'études et d'applications des SMAs sont très vastes. Le lecteur intéressé peut se référer à deux ouvrages complémentaires, l'un de J. Ferber [Fer95] et l'autre de G. Weiss [Wei99]. Le premier est plutôt le reflet de l'école « réactive ». Cette école considère que les agents ont des capacités réduites de perception, d'action et de cognition. Le second reflète plus l'école « cognitive » , où les agents sont capable de « résonner ». Dans les faits, nous observons un *continuum* entre les agents réactifs et les agents cognitifs, sans être en mesure de faire une dichotomie précise.

[8] Nous pouvons critiquer l'ensemble de ces définitions qui sont « générales » et donc relativement vagues. Néanmoins, elles fixent un ensemble de termes et permettent de comprendre le « paradigme agent ».

1.2.4 SMA et écologie

Présentés ainsi, les systèmes multi-agents apparaissent comme une métaphore efficace pour décrire des composants en interaction dans un système, dès lors que l'on cherche des modèles explicatifs pour comprendre des phénomènes globaux (le lecteur peut se référer à ces deux articles pour une illustration de l'utilisation des SMAs en écologie, [CL96] et [DH01]). C'est pourquoi nous voyons une utilisation croissante de ce type de modèles en écologie où la complexité est une caractéristique essentielle qui doit être appréhendée pour comprendre la dynamique des écosystèmes. Nous pouvons dire que les SMAs sont l'antithèse des modèles phénoménologiques classiques en faisant de la complexité leur moteur de construction.

Pour les écologues, les SMAs sont perçus comme une technique pour implémenter des IBMs. Pour un chercheur en informatique, les IBMs sont vus comme une utilisation possible des SMAs dans des contextes où l'individu est l'unité de base du système. C'est bien sur ce point que ce rejoignent IBMs et SMAs. En fait, ces deux paradigmes sont très proches. Il est averé que les disciplines des sciences du vivant[9] comme l'éthologie sont des sources d'inspirations majeures pour la conception de SMAs [Dro93].
Nous avons donc d'un côté l'écologie théorique, qui (notamment *via* les IBMs) cherche à comprendre comment, et dans quelle mesure, les systèmes naturels sont persistants et évolutifs, auto-organisés, résilients et autonomes, et de l'autre côté, une approche informatique qui cherche à doter des systèmes artificiels de ces mêmes propriétés. Malgré une différence dans les objectifs de ces deux domaines de recherche, il y a un besoin commun de représentation et de compréhension des mécanismes et des questions fondamentales communes. Une anecdote est par exemple le fait que les concepteurs de SMAs se posent la question du paradoxe thermodynamique [PB01] : alors que l'entropie (une mesure possible du désordre) des systèmes thermodynamiques est croissante, les systèmes vivants (et maintenant artificiels) évoluent vers une organisation croissante (donc une diminution de l'entropie). La biologie théorique peut ici apporter des réponses.

Ainsi, nous pensons qu'un rapprochement étroit entre l'écologie théorique et la kénétique peut être très enrichissant pour l'une comme l'autre des disciplines. En effet, la première s'appuyait principalement sur l'histoire ancienne des Mathématiques, avec ses outils théoriques puissants pour la formalisation et l'étude de la dynamique des systèmes. Elle commence maintenant à se tourner vers des outils informatiques. L'autre s'appuie sur l'histoire récente de l'informatique et de l'automatique, qui a également développé de manière rapide et efficace toute une batterie d'outils théoriques et pratiques pour la description des systèmes [ZKP00]. Les écologues théoriciens se sont attachés à construire des modèles plutôt physiques, en introduisant les notions d'équilibres stables et instables, de résilience et de permanence en écologie. De même, les notions de compétitions pour une ressource, de coopération ou d'entraide dans les SMAs trouvent un écho en écologie (symbiose, parasitisme, commensalisme, etc.). Des notions comme le climax (composition stable et pérenne d'un écosystème) ou la richesse spécifique sont des exemples de notions écologiques pouvant être utiles à la kénétique. Il en est de même de la théorie de l'information ou encore des réseaux trophiques. Dans notre travail, nous nous intéressons à ce rapprochement dans un cadre très précis que nous exposerons un peu plus loin.

Nous allons maintenant présenter de façon assez générale les concepts utiles au lecteur pour la compréhension des questions concernant l'intégration de modèles hétérogènes.

[9] Nous incluons ici la sociologie.

1.3 Hétérogénéité et intégration de modèles

Nous commençons par rappeller la définition de l'hétérogénéité[10]comme étant le caractère de ce qui est hétérogène (*i.e.* composé d'éléments de natures différentes). En informatique, l'hétérogénéité concerne tous les niveaux d'abstractions, depuis le matériel jusqu'au niveau conceptuel en passant par les applications ou logiciels. Cette hétérogénéité est renforcée dans un contexte distribué où les applications doivent communiquer au travers d'un réseau. Dans ce contexte, pour que des systèmes informatiques d'architectures différentes puissent communiquer, il faut qu'ils « parlent » au moins un langage commun. Le modèle à sept couches OSI (*Open System Interconnection*) par exemple, répond à ce besoin en proposant une architecture abstraite et une terminologie précise pour décrire et faire communiquer des ordinateurs *via* un réseau informatique.

À tous les niveaux, c'est la diversité qui entraîne l'hétérogénéité. Cette diversité est le reflet de toutes les solutions imaginées et conçues par les informaticiens. Comme l'uniformisation est illusoire (et certainement non souhaitable), c'est la volonté d'intégration (≈ assemblage) qui domine. Nous pouvons résumer la problématique comme suit (où « → » signifie « implique ») :

hétérogénéité → intégration → interopérabilité

Dans notre travail, nous considérons cette problématique dans le cadre de la modélisation et de la simulation. Dans ce cadre, Zeigler et Sarjoughian [ZS00] ont défini une hiérarchie conceptuelle. Cette hiérarchie propose six couches interdépendantes qui représentent chacune une problématique précise. La figure 1.2 présente ce cadre conceptuel hiérarchique.

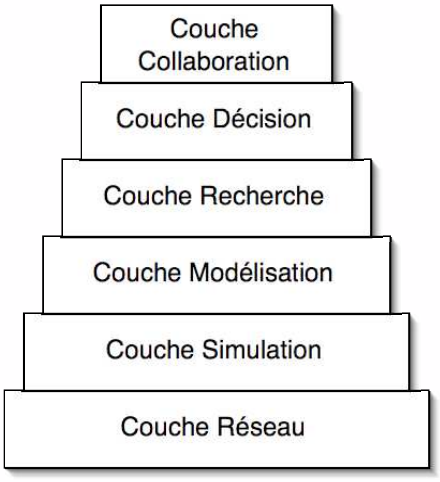


FIG. 1.2 – Cadre conceptuel de la modélisation et de la simulation par Zeigler & Sarjoughian. Ce cadre définit six couches interdépendantes qui correspondent à des problématiques précises (voir le texte pour les détails).

La couche « réseau » contient tous les éléments physiques de calcul (stations de travail, serveur...), de connexion et d'interconnexion et les logiciels et protocoles liés aux réseaux. La couche « simulation » est une couche logicielle qui a pour objectif d'exécuter les modèles. Elle intègre

[10]Le nouveau petit Robert, 1995

les protocoles nécessaires pour les bases de la simulation distribuée, la gestion des accès aux bases de données, le contrôle du cycle d'exécution de la simulation, la visualisation et l'animation des comportements simulés. La couche « modélisation » supporte le développement des modèles dans des formalismes qui doivent être indépendants de l'implémentation de la couche « simulation ». Les trois couches suivantes sont des couches de haut niveau qui concernent les acteurs humains de la modélisation et de la simulation. La couche « recherche » qui met au point les modèles, la couche « décision » qui comprend l'exploration du comportement des modèles et la couche « collaboration », qui comprend les phases de concertation entre les participants à la construction du modèle. Nous situons notre travail au niveau des couches « modélisation » et « simulation » en traitant de l'hétérogénéité des paradigmes et des formalismes pour la modélisation et la simulation des systèmes dynamiques.

1.3.1 Hétérogénéité, paradigmes et formalismes

Que signifie interopérabilité des paradigmes et des formalismes ? Pourquoi vouloir intégrer des modèles hétérogènes ? Ces questions trouvent leur origine dans le contexte actuel de l'informatisation des sciences. L'augmentation de la puissance de traitement des ordinateurs nous autorise à être plus complet dans nos représentations du réel. La question n'est pas de savoir si, par exemple, le réel est discret ou continu mais plutôt de faire cohabiter ces deux représentations au sein d'un même modèle. Il semble assez évident que la diversité du réel soit mieux appréhendée par une diversité des modèles.

L'hétérogénéité des modèles nous amène donc à la construction de multi-modèles. Un multi-modèle rassemble plusieurs paradigmes et/ou formalismes dans sa réalisation (nous parlons alors de multi-modélisation). Ce terme a été introduit par T.I. Ören en 1989 [Ö89] et s'est fait connaître par les travaux de P.A. Fishwick et B.P. Zeigler [FZ92]. Il existe aussi le terme de modélisation multi-paradigme. H. Vangheluwe [VLM02] le définit comme s'adressant à trois axes de recherches orthogonaux :

1. à la modélisation liée au multi-formalisme (donc à la multi-modélisation), c'est-à-dire au couplage de modèles spécifiés dans différents formalismes ;
2. au problème de changement de niveau d'abstraction dans les modèles ;
3. à la méta-modélisation, c'est-à-dire la construction de modèles de modèles.

L'hétérogénéité des paradigmes n'entraîne pas forcément une hétérogénéité des formalismes. Prenons par exemple les IBMs : ils peuvent être formalisés par des systèmes d'équations différentielles, tout comme les modèles de dynamique de population d'individus, qui sont des modèles agrégés du même système. De façon symétrique, deux modèles du même paradigme peuvent être formalisés ou spécifiés différemment. Ainsi, un SMA peut être spécifié en UML (un langage graphique pour la modélisation orientée objets) ou formalisé avec des équations différentielles. Ce qui est important de noter est que le formalisme utilisé impose un point de vue sur le système considéré et certaines contraintes d'implémentation informatique. Nous y reviendrons de façon plus précise dans ce document.

Pour intégrer différents paradigmes, nous considérons les travaux effectués par Zeigler depuis le début des années 1970 [Zei76]. Ces travaux se basent sur les Mathématiques discrètes et plus particulièrement sur les Mathématiques des systèmes. D'une façon générale et indépendante du formalisme, Zeigler introduit une hiérarchie de spécifications des systèmes, en d'autres mots des

niveaux basés sur les connaissances que nous avons d'un système donné. Cette hiérarchie est présentée tableau 1.1.

TAB. 1.1 – Hiérarchie de spécification des systèmes.

Numéro du niveau	Nom du niveau de spécification	Ce que nous savons à ce niveau
0	Cadre d'observation	Comment « stimuler » le système par ses entrées ; quelles variables mesurer et comment les observer au cours du temps
1	Comportement d'entrée/sortie	Données indexées par le temps et collectées depuis un système source. Ce sont des paires entrées/sorties
2	Fonctions d'entrées/sorties	Connaissance de l'état initial. Étant donné un état initial, toute entrée produit une sortie unique
3	Transition d'état	Comment les états internes d'un système sont affectés par les entrées, comment évoluent les états internes et quelles sortie sont produites par quels états
4	Couplage de composants	Comment sont couplés les composants d'un système. Les composants peuvent être spécifiés à un niveau inférieur ou au même niveau, ce qui introduit la notion de structure hiérarchique *via* les composants.

Si deux systèmes sont spécifiés au même niveau d'abstraction, il est alors possible de les comparer afin de définir s'ils sont équivalents en terme de comportement. Cette comparaison est appelée « morphisme ». Cette notion est largement développée dans le livre référence de Zeigler [ZKP00]. Elle permet notamment de définir des équivalences intéressantes entre modèles et simulateurs. Elle permet également de définir la transformation d'un modèle formalisé par A dans un autre formalisme B en définissant des équivalences. Nous parlons alors de *mapping* de A vers B.

Ainsi, dans le contexte de la théorie des systèmes, des travaux formels ont été menés pour développer les fondements théoriques de la modélisation et de la simulation des systèmes dynamiques [Zei76]. Ces travaux ont notamment donné naissance au formalisme DEVS (Discrete

EVent System specification) pour la spécification des systèmes à évènements discrets. DEVS s'abstrait totalement de la mise en œuvre des simulateurs même si, comme nous le verrons plus loin, des algorithmes existent pour une implémentation effective des modèles. De plus, des travaux récents montrent que DEVS peut « encapsuler » de nombreux formalismes tels que les équations différentielles [GEG00] [ZKP00] ou les réseaux de Petri [JW02]. Ce formalisme est donc bien adapté à la spécification des multi-modèles lorsque les modèles composants peuvent être exprimés en DEVS. Il existe également des extensions de DEVS adaptées à des types précis de modèles, nous y revenons plus loin dans cette thèse. La figure 1.3 montre les correspondances entre les spécifications à temps discret et à temps continu dans les systèmes dynamiques et quelles sont leurs relations avec DEVS.

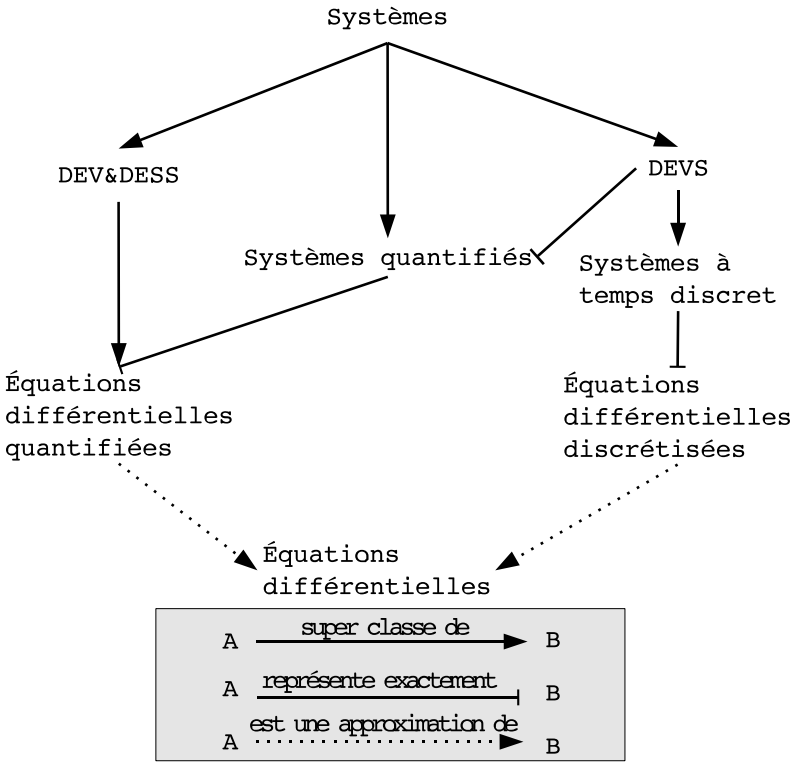


FIG. 1.3 – Correspondance entre les spécifications à temps dicret et à temps continu (d'après [ZKP00]). Les systèmes quantifiés représentent une alternative à la discrétisation classique du temps pour la résolution des équations différentielles [Kof02]. DEV&DESS est un formalisme qui combine le temps discret et le temps continu. Cette figure montre que DEVS peut être considéré comme un cadre formel pour l'intégration de modèle hétérogène.

La figure 1.3 illustre le fait que DEVS est un cadre formel d'intégration de modèles hétérogènes, en tout cas en ce qui concerne des modèles représentant le temps différemment. Ce cadre d'intégration formel doit s'acompagner d'un cadre d'intégration opérationnel, c'est-à-dire d'une possibilité pratique de coupler les modèles. Ici, c'est l'ordinateur qui sert de « machine d'intégration » de formalismes hétérogènes en étant le support d'exécution des algorithmes représentant les spécifications formelles.

1.3.2 Hétérogénéité et couplage de modèles

Dans la pratique, intégrer des modèles hétérogènes revient à coupler des simulateurs. Dans la littérature, le terme « intégration » désigne l'adaptation de simulateurs pour qu'ils puissent fonctionner les uns avec les autres [Fia01]. Ce type de couplage peut aussi correspondre à la réécriture d'un simulateur unique à partir de plusieurs autres. Il existe de nombreux exemples de couplages de modèles dans la littérature. Un des premiers concernant la communauté francophone dans le domaine des SMA, fut le travail de Cambier [Cam94] qui couple un modèle « socio-économique » de pêcheurs avec un modèle « classique » de dynamique de population de poissons. Néanmoins, les techniques mises en jeu sont spécifiques à la problématique et aux modèles.

Dans le cadre de l'intégration de logiciels préexistants, les problèmes techniques liés à l'interopérabilité ont fait l'objet de nombreuses recherches et de développements récents. Nous allons présenter ici une synthèse des principales avancées. Nous présentons l'intégration dans un contexte distribué. En effet, ce contexte englobe tous les problèmes d'interopérabilités. Notre discours concerne ici la couche « réseau » telle que la définit la figure 1.2 page 15.

La couche « réseau » est probablement la plus étudiée et la plus prolifique en terme de technologies mais aussi la couche la plus hétérogène. Aussi, afin de mieux comprendre le problème, nous décomposons cette couche en trois sous-couches relativement indépendantes :

- la sous-couche réseau à proprement parler,
- la sous-couche d'interface d'accès distant,
- la sous-couche langage.

La première sous-couche est probablement la plus stabilisée depuis la domination d'Internet sur le monde des réseaux WAN et LAN. Sous le terme d'Internet, nous intégrons toutes les technologies réseaux et protocoles mis en œuvre pour le transport d'informations d'un processeur [11] à un autre.
La deuxième couche offre des mécanismes de haut niveau pour l'accès distant à des objets ou des interfaces fonctionnelles. Faire communiquer des simulateurs passe évidemment par des services d'invocation de méthodes distantes. Cette deuxième couche est beaucoup plus problématique. En effet, on voit apparaître avec la programmation distribuée et le développement des services Web plusieurs technologies : SOAP[12], Corba[13], RMI[14], DCOM[15], RPC[16], MPI[17], sockets, ... Toutes ces technologies sont dirigées vers un objectif unique (faire communiquer à distance des composants logiciels), mais leurs niveaux d'intervention sont différents. Par exemple, SOAP est un protocole d'invocation de méthodes sur des services distants où les messages et les objets attachés aux messages sont spécifiés en XML[18], un langage de marquage à balise dont nous reparlerons plus loin. SOAP repose dans la plupart de ses mises en œuvre sur le protocole HTTP. L'une de ses caractéristiques est d'être indépendant du langage de programmation (contrairement à RMI qui propose sensiblement les mêmes services mais uniquement en Java).
Avec SOAP, Corba, RMI et DCOM, nous avons à notre disposition des technologies principale-

[11] Nous désignons par processeur tout élément susceptible d'exécuter un programme et de communiquer.
[12] Simple Object Access Protocol
[13] Common Object Request Broker Architecture
[14] Remote Method Invocation
[15] Distributed Component Object Model
[16] Remote Procedure Call
[17] Message Passing Interface
[18] eXtensible Markup Language

ment orientées service Web et orientées objets. RPC, MPI et les sockets sont des bibliothèques de fonctions très proches du niveau réseau. Par exemple, MPI est utilisé dans les clusters[19] de PC pour la programmation parallèle. Ces bibliothèques sont de bas niveau et pas toujours simples à mettre en œuvre.

La dernière sous-couche, la sous-couche langage, est tout aussi délicate. En faisant rapidement le bilan des langages utilisés en programmation de simulateurs, nous nous apercevons que le langage numéro un est le Fortran pour tous les modèles de type numérique. Pour les simulateurs orientés automates cellulaires, les IBMs ou les SMAs, les langages que l'on rencontre sont plutôt Java, C/C++ et Smalltalk. Cette dernière couche peut être vue autrement : au lieu de parler de langage, nous pourrions parler de plate-formes de simulation. Le problème ici n'est pas seulement le langage de programmation mais l'outil complet. Il est alors nécessaire de se poser la question de son intégration dans un environnement, distribué ou non, de simulation. Une réponse relativement complète existe, qui intègre les trois sous-couches décrites précédemment, il s'agît d'HLA.

HLA est une architecture et une interface de spécification pour la simulation répartie. Cette architecture se positionne dans le cadre de l'interopérabilité et la réutilisabilité pour la simulation. Elle s'intéresse à la fois à l'interconnexion de plateformes de simulation et à des aspects de plus haut niveau tel que l'échange d'évènements et la synchronisation. HLA a été développé sous le contrôle de l'Office pour la Modélisation et la Simulation de la Défense américaine (Defense Modeling and Simulation Office - DMSO) afin de proposer des solutions de réutilisabilité et d'interopérabilité des nombreux types différents de simulations développées par le département de la défense américaine. HLA a été adopté par l'OMG (Object Management Group) et approuvé par IEEE (Institute of Electrical and Electronical Engineers) sous le standard IEEE 1516. HLA est issu d'un effort de standardisation des architectures de simulation telles que SIMNET (Simulation Network [20][Kan90]) et DIS (Distributed Interactive Simulation).
HLA est une architecture et non un environnement logiciel. Elle propose un cadre de travail et un vocabulaire commun. L'utilisation d'un RTI (RunTime Infrastructure) est nécessaire afin d'implémenter les opérations d'exécution et de coordination des simulateurs. Le RTI fournit un ensemble de services utilisés par les simulateurs (ou fédérés[21]). La définition des services est indépendante des plateformes et des langages. Nous retrouvons ici des implémentations utilisant les technologies Corba. Un fédéré peut ne pas être un simulateur mais un objet du monde réel. Les fédérés sont regroupés en fédération [22]. Ces services ont pour objectif de coordonner les opérations et les échanges de données durant l'exécution de la simulation globale. L'accès à ces services est défini par la spécification de l'interface HLA. Cette interface est une interface fonctionnelle entre le fédéré (la simulation) et le RTI (voir figure 1.4). La connexion au RTI est assurée par des *ambassadors*[23].

Tout objet d'une simulation est documenté ou spécifié à l'aide de l'OMT (Object Model Template). La spécification regroupe les informations concernant les données échangées entre les fédérés ainsi que les informations concernant la coordination des simulations. Il existe trois

[19]Un cluster de PC est un ensemble de machines reliées entre elles par un réseau rapide (FastEthernet ou Myrinet). Une machine appelée frontal se charge de l'allocation des processeurs en fonction des demandes et de la nature des programmes parallèles à exécuter.

[20]http ://www.stricom.army.mil/PRODUCTS/SIMNET/

[21]Un fédéré est une simulation compatible HLA (HLA-*compliant simulation*). Un fédéré appartient à une fédération.

[22]Une fédération est un ensemble de simulations ou fédérés interopérants.

[23]Les *ambassadors* sont des modules qui permettent d'encapsuler le fédéré pour le rendre HLA-*compliant*

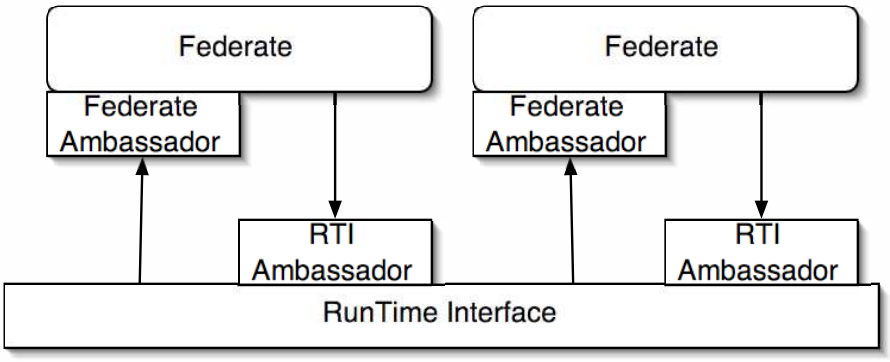


FIG. 1.4 – RunTime Interface

types de modèle :

- le SOM (Simulation Object Model) pour les simulateurs,
- le FOM (Federation Object Model) pour les fédérations,
- le MOM (Management Object Model) pour la gestion de l'exécution.

Le SOM spécifie les caractéristiques des simulateurs utiles aux autres simulateurs en définissant les objets et les interactions qui peuvent être utilisés à l'extérieur.
Le FOM spécifie les échanges de données entre fédérés et les données partagées. Il est directement en relation avec le RTI qui se charge de l'aspect opérationnel. Le FOM garantit l'interopératibilité des simulateurs développés par des entités différentes et la réutilisabilité de simulateurs existants.
Le MOM réunit les informations et mécanismes nécessaires à la gestion de l'exécution d'une fédération de fédérés. Parmi ces informations et mécanismes, on peut citer : la gestion des objets (création, destruction, envoi de messages, ...), la gestion du temps, la gestion de la transmission des données entre fédérés. La plupart de ces éléments sont relatifs à la gestion des données (transport, stockage, définition, relation, ...) sauf la gestion du temps qui est un point fondamental en simulation. La gestion du temps consiste à coordonner l'avancement du temps dans les différentes simulations de la fédération en garantissant la causalité. Plusieurs techniques sont supportées parmies lesquelles nous pouvons citer : les techniques classiques (approches synchrones fortes, approches asynchrones faibles ou fortes -TimeWrap-, ...), le temps discret et le temps continu. La gestion du temps par la fédération consiste alors à offrir des services d'autorisation d'avancement du temps et de gestion de files d'attente d'événements estampillés. Un fédéré gère sa propre horloge locale à condition que la fédération lui ait donné l'autorisation d'avancer. Cette autorisation est naturellement en adéquation avec la technique adoptée. L'ensemble de ces modèles (SOM,FOM et MOM) font l'objet de standardisation IEEE [24]. Il existe de multiples implémentations du RTI dans divers langages (C++, Java, ...) et pour différentes technologies distribuées (Corba, DCOM, ...).

[24] IEEE Std 1516-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules
IEEE Std 1516.1-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification
IEEE Std 1516.2-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification

En conclusion, nous pouvons dire que HLA est un excellent *framework* pour les couches externes des simulateurs, pour l'interopérabilité des simulateurs, le contrôle de la cohérence des données échangées et de l'avancement du temps. Ici, le simulateur est considéré comme une boîte noire. Il demande l'autorisation d'avancer son horloge à la fédération et non le contraire. Les objets internes aux simulateurs, ainsi que leur dynamique, ne sont pas pris en charge dans ce *framework*. Néanmoins, HLA est une réponse satisfaisante à de nombreux problèmes posés par la couche « réseau ».

1.4 Conclusion

Dans ce chapitre, nous avons présenté le cadre général de cette thèse, à savoir la modélisation et la simulation de systèmes complexes. Nous nous sommes attachés à décrire les relations entre la complexité descriptive et les écosystèmes qui sont notre cadre d'applications. Dans ce contexte, nous avons introduit la notion de système hiérarchisé en niveaux d'organisation, ce qui nous a permis de présenter le transfert d'échelles comme une question importante pour la représentation des systèmes. Nous avons également introduit différents paradigmes de modélisation : les systèmes multi-agents, les modèles individus-centrés ou encore les équations différentielles. Ces paradigmes vont nous servir d'exemples pour illustrer une problématique centrale de cette thèse, l'intégration de modèles hétérogènes. Dans ce contexte, nous avons pu voir que HLA offre un cadre opérationnel normalisé pour l'intéropérabilité des modèles au niveau « réseau » tel que nous l'avons défini.

La multi-modélisation apparaît comme une voie prometteuse pour la construction de simulateurs complexes qui reflètent mieux les systèmes réels et notamment les écosystèmes. Seulement, qui dit complexité des modèles dit complexité de mise en œuvre, difficulté pour faire des tests ou valider les modèles ou même simplement pour les communiquer. L'approche des SMA est particulièrement intéressante pour la modélisation des comportements individuels en écologie, mais ce paradigme offre-t-il un cadre formel de spécification suffisant pour s'intégrer avec des modèles « classiques » de type équations différentielles ?

2

Présentation de la problématique

Résumé

La problématique générale de notre travail est l'intégration de modèles hétérogènes. Nous avons choisi trois axes pour traiter de cette problématique :

- intégration de modèles au niveau formel,
- définition d'un cadre le plus général possible pour une intégration opérationnelle,
- illustration de l'intégration de modèles hétérogènes.

Les deux premiers axes sont étroitement liés et concernent les questions sur l'intégration à proprement parler (formelle et opérationnelle). Le troisième axe concerne l'utilisation d'un modèle couplé.

Nous définissons dans ce chapitre une méthode originale pour simuler un transfert d'échelle entre niveaux d'organisation. La présentation de cette méthode est une parfaite illustration d'intégration de deux modèles hétérogènes. Ensuite, nous présentons le système naturel que nous avons choisi de modéliser afin d'appliquer notre méthode de transfert d'échelle. Il s'agit d'un système proies-prédateurs. Nous adoptons une modélisation orginale (agents situés dans un espace continu) et une autre « classique » (équations différentielles) de ce système. Ce système nous sert d'exemple dans le prochain chapitre qui concerne l'intégration formelle. Nous voulons également montrer tout l'intérêt de notre méthode de couplage pour la modélisation multi-échelles en écologie marine. Aussi, nous posons ici des questions importantes pour l'écologie. Nous revenons sur ces questions au dernier chapitre de cette thèse.

Sommaire

2.1 Les trois points abordés

Récemment, V. Grimm a fait part de « la crise de la communication » qui affecte la modélisation individus-centrée [Gri02]. Cette crise existe également dans le domaine de la modélisation à base d'agents. Avec l'augmentation de la capacité de calcul des ordinateurs, les modèles développés décrivent un très grand nombre d'entités en interaction. Ainsi, ils deviennent de plus en plus complexes et de moins en moins communicables car ils ne sont plus seulement basés sur un et un seul formalisme. Quatre types de propositions sont faites pour résoudre cette crise de la communication des modèles :

- des langages informatiques assez généraux pour la modélisation,
- des librairies spécifiques écrites dans des langages de programmation usuels comme C++,
- des plateformes de modélisation « génériques »,
- développer un protocole général pour la description de modèles de simulation.

La première solution a vu le développement de plusieurs langages [MC97] [FF01]. Néanmoins, ils restent plutôt utilisés par leur créateurs.
Dans le domaine de l'écologie, la deuxième solution a été développée notamment par Lorek et Sonnenschein [LS99]. Seulement, quelle que soit la bibliothèque utilisée, elle ne répond que partiellement au problème de communication des modèles centrés-individus. En effet, les programmes développés à l'aide de ces bibliothèques doivent être complétés de développements propres. Ainsi, le problème de la communication est réduit aux développements spécifiques pour un problème donné, mais subsiste toujours.
La troisième proposition se réfère au développement de plateformes, c'est-à-dire de logiciels plus ou moins génériques. Dans le domaine des SMAs, de nombreuses plateformes ont été proposées (le lecteur intéressé peut se référer à [GD02] pour une comparaison récente de plusieurs d'entre elles). Là encore, ces plateformes résolvent le problème uniquement pour la communauté qui les utilise, mais pas de façon générale.
La dernière proposition nous intéresse particulièrement ici. Nous voulons donner des éléments de réponses pour aller dans la direction d'une meilleure communication des modèles de simulation dans le domaine des SMAs et de l'écologie numérique utilisant des IBMs. Devant la diversité des modèles, cette dernière proposition ne peut pas se résumer à l'adoption d'un langage commun universel de type mathématique. Cette diversité des modèles pose également la question de l'utilité d'une intégration qui a pour effet de complexifier des modèles déjà complexes ! Nous abordons ces questions dans un contexte de multi-modélisation selon trois axes orthogonaux :

- la formalisation des modèles couplés,
- le couplage opérationnel de modèles,
- l'apport des multi-modèles en écologie théorique.

2.1.1 Premier point : intégration formelle

Comme nous l'avons dit dans le chapitre précédent, DEVS est un formalisme abstrait pour la spécification de modèles de simulation. Ainsi, DEVS peut être considéré comme une bonne solution pour aider à résoudre le problème de la communication des modèles de simulation *bottom-up* (agents ou individus-centrés). De plus, ce type de modèles peut entrer dans la construction de multi-modélisation qui augmentent leur complexité. Comme nous l'avons dit, DEVS intègre un grand nombre de formalismes des systèmes dynamiques. Il est donc bien placé pour spécifier les multi-modèles.
Le premier point nous amène donc à rapprocher la théorie de la modélisation et de la simulation telle qu'elle est présentée par Zeigler [ZKP00] avec les paradigmes et formalismes développés

dans le contexte de la modélisation par SMA. En effet, les acteurs de la communauté SMAs ont des difficultés à formaliser leur domaine[25]. Les SMAs apparaissent plus comme des réalisations logicielles liées à un paradigme méthodologique que comme un outil rigoureux de modélisation. Nous reviendrons sur ce point au chapitre 3.

Dans notre travail, nous nous concentrons sur l'utilisation de DEVS pour l'intégration des modèles d'agents réactifs situés et des équations différentielles dans un contexte de simulation multi-échelles. Cette intégration nécessite la spécification d'un modèle d'agents réactifs en DEVS. Nous qualifions cette opération de « *mapping* ». C'est le premier point abordé.

2.1.2 Deuxième point : intégration opérationnelle – représentation des modèles et des expériences

Le deuxième point nous amène à développer un *framework*[26] qui permette d'intégrer des modèles hétérogènes. Comme nous l'avons vu en introduction avec HLA, il existe des solutions techniques opérationnelles pour l'intégration de modèles. Seulement cette solution est relativement « opaque » en ce qui concerne les modèles qui sont couplés. En effet, HLA n'offre pas de description des modèles. Ainsi, nous pensons que cette solution ne suffit pas dans le contexte de l'échange, de la diffusion et de la réutilisation de modèles existants dans un contexte scientifique de questionnement où les utilisateurs ont besoin d'avoir des descriptions suffisantes pour connaître le fonctionnement des modèles. Néanmoins, HLA répond déjà à un ensemble de questions, nous ne pouvons l'ignorer. De même, les solutions techniques des applications distribuées (Corba, Java RMI ou encore SOAP) offrent un ensemble de possibilités opérationnelles.

Nous ne situons donc pas notre réflexion à ce niveau mais au niveau des couches modélisation et simulation présentées figure 1.2 page 15. Dans ce cadre, nous allons utiliser certaines avancées effectuées autour de DEVS en proposant d'ajouter des aspects sémantiques pour la description des modèles. Ce deuxième point concerne également les techniques de simulations de modèles couplés lorsque ceux-ci sont fortement hétérogènes au niveau des formalismes et des paradigmes. Nous proposons alors l'utilisation du « *wrapping* » (une encapsulation des modèles).

Comme nous nous situons dans un cadre où la modélisation et la simulation sont sensées nous apprendre quelque chose, nous faisons des expériences avec les modèles. À ce propos, F. Varenne qualifie la simulation « d'expérience du second genre » [Var03] (entre intuition et expérience directe). Cette définition implique qu'il y ait un aller-retour permanent entre le modélisateur et son modèle, à l'image d'une coévolution du modèle et des questionnements du modélisateur. De plus, devant la complexité croissante des systèmes modélisés, nous sommes de plus en plus enclins à adopter la position de l'expérimentateur étudiant un système « réel ». Nos modèles sont exécutés plusieurs fois, avec des jeux de paramètres différents pour une analyse de sensibilité ou une étude particulière.
Afin de rationaliser cette pratique, nous adoptons la notion de laboratoire virtuel. Il existe plusieurs définitions des laboratoire virtuels. Ils peuvent être des outils de simulation et d'expériences très précis pour la formation des étudiants d'une discipline particulière comme la chimie [SYLL02], la biologie [Rai01] ou la thermodynamique [FWE+99] par exemple. Ils restent dans

[25] Il est plus juste de dire qu'il existe un grand nombre de formalisme, nous développons notre pensée au chapitre 3.

[26] Une traduction possible de *framework* est cadriciel. Nous conservons néanmoins le terme anglosaxon.

ce cas évidemment très liés à la discipline. Il existe également des tentatives de mise au point de laboratoires virtuels génériques [AKB+01]. Pour l'instant, ce type de logiciels posent plus de questions qu'ils n'en résolvent, notamment en ce qui concerne leur validation. Notre point de vue est qu'il n'est pas possible d'être générique dans le domaine de la simulation en terme d'outils. Par contre, nous pensons qu'il est possible de faire communiquer des outils (ou plateformes de simulations) *via* une description commune des modèles et des expériences.

Ainsi, le deuxième point abordé va nous amener à proposer une méthode opérationnelle pour l'intégration de modèles hétérogènes. Nous proposons à la fois une syntaxe pour la description des modèles et des expériences et une architecture pour le couplage et l'expérimentation à proprement parler.

2.1.3 Troisième point : intégration pour la simulation multi-échelles

L'intégration de modèles hétérogènes peut avoir une utilité pour les disciplines utilisant la modélisation comme une méthode de questionnement. C'est le cas de l'écologie théorique dont nous avons déjà parlé en introduction. Ce troisième point va nous permettre d'illustrer le potentiel des multi-modèles. En effet, une des questions importantes en écologie est de savoir comment et dans quelle mesure l'environnement des individus et leurs interactions influencent la dynamique du système global, la population [GWAU99]. Plus généralement, cette question peut se poser pour tout système hiérarchisé.
Un des objectifs de la théorie de la hiérarchie est de relier les différents modèles qui décrivent les niveaux d'organisation de la matière et ainsi de contribuer à une meilleure connaissance des mécanismes et les lois qui « imbriquent » les différents niveaux d'organisation.
En essayant de définir des règles d'intégration, de couplage ou de communications entre différents modèles décrivant différents niveaux d'abstraction, nous participons à l'enrichissement des moyens d'investigations pour appréhender un tel objectif.
Dans ce qui suit (section 2.2), nous proposons une méthode de simulation multi-échelles. C'est au travers de cette méthode que nous allons illustrer le couplage formel au chapitre 3. Nous montrerons l'intérêt de son application en écologie au chapitre 5. La dernière partie de ce chapitre (section 2.3) est la présentation du système écologique considéré, un système proie-prédateur.

2.2 Proposition d'une méthode de simulation multi-échelles

Comme nous l'avons dit en introduction de cette thèse, pour décrire un système physique ou biologique, les équations différentielles sont l'outil fondamental des scientifiques. Cette approche tend à décrire un système constitué de nombreux éléments par des variables agrégées (des concentrations, des densités de populations) et s'intéresse à la dynamique de ces variables. Il s'ensuit une représentation très compacte et très générale de l'évolution du système. Cependant, ces modèles font intervenir des paramètres agrégés dont l'identification peut s'avérer délicate.

Les modèles individus-centrés considérés dans ce paragraphe ne sont pas nécessairement des modèles multi-agents. Nous utilisons donc le terme individus-centrés dès lors que la modélisation se focalise sur les entités composant le système, par opposition à une approche considérant des

variables agrégées représentant l'ensemble des entités.

Ces deux types de modèles peuvent être vus comme des représentations du même système à deux niveaux d'abstraction différents. Dans ce paragraphe, nous tentons de démontrer l'intérêt de développer les deux niveaux d'abstraction ensemble. Notre argument principal est que ces deux niveaux peuvent s'épauler mutuellement, c'est-à-dire s'utiliser l'un l'autre. Dans notre proposition, le modèle individus-centré et le modèle agrégé concernent le même phénomène, le même système, et ne sont pas seulement utilisés chacun pour une partie du problème. Nous sommes proches de l'idée d'abstraction décroissante de Lindenberg [Lin92] qui préconise le développement de modèles de plus en plus fins, dont la compréhension est nourrie de celle des modèles de granularité plus grande, éventuellement dans des formalismes différents. Le concept de multi-modélisation présenté dans le premier chapitre de cette thèse est également apparenté à notre démarche [Fis95]. On y retrouve cette idée de décrire un phénomène à l'aide de plusieurs modèles, à des niveaux d'abstraction ou de raffinement différents.

Pour simuler un transfert d'échelles, nous proposons un type particulier de coopération entre deux niveaux d'abstraction. Cette coopération peut être vue comme l'utilisation du modèle individus-centré comme un laboratoire virtuel, permettant de faire des expérimentations et des mesures, afin d'identifier les paramètres d'un modèle agrégé.
Pour illustrer concrètement et plus précisément cette idée, nous considérons d'abord un exemple simple : la diffusion dans un fluide de particules animées d'un mouvement brownien. Le modèle agrégé se réduit à l'équation d'évolution de la concentration des particules dans l'espace. Le modèle individus-centré représente l'ensemble des particules et leurs mouvements aléatoires. Nous considérons différentes variantes de ce phénomène dans lesquelles le modèle individus-centré peut être appelé « à la volée » par le modèle agrégé pour identifier certains paramètres. Nous montrons que ces opérations peuvent, dans certaines conditions, mener à des changements d'échelles de temps et d'espace. À la fin de ce paragraphe, nous proposons une méthode plus générale pour la modélisation du transfert d'échelles dans les systèmes complexes avant de l'appliquer sur un cas de modélisation en écologie marine.

2.2.1 Modèles agrégé et individus-centré de la diffusion de particules

La diffusion de particules est un exemple particulièrement simple de phénomène pour lequel la description physique microscopique est reliée mathématiquement à la description macroscopique du système par la mécanique statistique. Il est donc aisé d'élaborer un modèle agrégé et un modèle individus-centré de ce phénomène et d'établir un lien entre eux. Ce phénomène nous est donc apparu à la fois suffisamment simple pour être facile à exposer et suffisamment riche pour illustrer notre démarche.

Modèle agrégé

L'étude du phénomène de diffusion par la théorie cinétique a établi l'équation de Fick (équation 2.1). Cette équation formalise le modèle agrégé de la diffusion brownienne [Fic55] dans [Col97]. On s'intéresse ici uniquement à la composante le long de l'axe des abscisses (x). L'équation de Fick s'écrit :

$$\frac{\partial C(x,t)}{\partial t} = D\frac{\partial^2 C(x,t)}{\partial x^2} \tag{2.1}$$

où $C(x,t)$ est la concentration en particules, D le coefficient de diffusion et x l'axe sur lequel a lieu la diffusion.

Deux méthodes existent pour résoudre une équation différentielle. La première consiste à intégrer l'équation analytiquement. La deuxième passe par une résolution numérique utilisant un schéma d'intégration approprié. Lorsque l'on connaît la concentration initiale et les limites de l'espace, il est possible de calculer la solution de l'équation 2.1 analytiquement. Nous obtenons :

$$C(x,t) = \frac{C_0 e^{-x^2/4Dt}}{2\sqrt{\pi Dt}} \tag{2.2}$$

où : C_0 est la concentration initiale, $C(x,t)$ la concentration au temps t à la position x et D le coefficient de diffusion.

La forme de cette solution est une gaussienne représentant la concentration d'un produit diffusant le long de l'axe des x à un temps t donné (*cf.* courbe en pointillés de la figure 2.2). Cependant, si nous supposons que le coefficient de diffusion est variable dans l'espace, la solution analytique n'est pas toujours calculable. Il faut alors utiliser une résolution numérique de l'équation à coefficient variable et à pas constant (équation 2.3) [DL84], en discrétisant l'espace et le temps :

$$\frac{1}{\Delta t}(C_i^{t+1} - C_i^t) - \frac{1}{\Delta x}\left[D_{i+1}\frac{C_{i+1}^t - C_i^t}{\Delta x} - D_{i-1}\frac{C_i^t - C_{i-1}^t}{\Delta x}\right] = 0 \tag{2.3}$$

avec :
C_i^t la concentration en particules au pas d'espace i et au pas de temps t,
C_i^0 la concentration initiale (fixée),
Δx le pas d'espace,
Δt le pas de temps,
D_i le coefficient de diffusion au pas d'espace i.

Nous disposons donc maintenant de deux modèles agrégés du phénomène de diffusion, un modèle analytique et un modèle numérique. Construisons à présent le modèle individus-centré correspondant.

Modèle individus-centré correspondant

Le modèle individus-centré simulant le phénomène de diffusion est particulièrement simple dans un premier temps, mais nous verrons par la suite que celui-ci peut être complexifié à loisir pour éventuellement répondre à des problématiques plus précises.
Nous supposons que les particules évoluent dans un cube dans un repère (x, y, z). Les particules sont initialement distribuées aléatoirement sur le plan défini par $x = 0$, séparant le cube en deux moitiés de même volume. La simulation consiste à donner une direction aléatoire (entre 0 et 2π *rad*) et une vitesse aléatoire (entre 0 et v_{max}) à chaque particule pour chaque itération. La condition aux limites du cube est l'impossibilité pour toute particule d'en sortir. Nous retirons donc aléatoirement une direction et une vitesse pour toute particule qui sort du cube. Ce modèle

apparaît principalement stochastique. Il est donc nécessaire de simuler un très grand nombre de particules pour que l'observation du comportement du modèle ne soit pas un évènement particulier mais le reflet d'un comportement moyen de l'ensemble des particules.
Ce modèle donne la position de toutes les particules à chaque instant. Il permet donc de calculer la concentration en particules en toute partie de l'espace, mais aussi éventuellement d'autres grandeurs (la distance moyenne parcourue par les particules par exemple).

2.2.2 Laboratoire virtuel pour la détermination de paramètres

Le modèle individus-centré peut ainsi nous donner une estimation du paramètre de diffusion D du modèle agrégé, correspondant à une vitesse maximale des particules.
Pour cela, nous considérons le modèle individus-centré comme un laboratoire virtuel sur lequel nous pratiquons des expérimentations et des mesures, de la même façon que pour une expérience classique [Leg97] [Gri99]. Ainsi, en partant d'une distribution aléatoire des particules dans le plan $x = 0$, nous mesurons à chaque instant t l'écart quadratique moyen $\overline{\varrho}$ de l'ensemble des particules à l'aide de l'équation suivante :

$$\overline{\varrho} = \sqrt{\sum_{i=1}^{n} x_i^2 / n}$$

c'est-à-dire la racine carrée de la somme des carrés des distances x_i entre les particules et l'origine de l'axe des x divisée par n, le nombre total de particules (i.e. la moyenne des distances euclidiennes entre chaque particule et le plan d'origine).

Il existe une relation entre $\overline{\varrho}$ et D : $\overline{\varrho} = \sqrt{2Dt}$. Connaissant $\overline{\varrho}$ à un instant donné, nous pouvons en déduire une constante D instantanée :

$$D = \frac{1}{2t}\overline{\varrho}^2 \tag{2.4}$$

La figure 2.1 montre l'évolution de la quantité $\frac{1}{2}\overline{\varrho}^2$ en fonction du temps. C'est en calculant la pente de cette droite que nous pouvons obtenir le coefficient de diffusion D.

On peut ensuite vérifier que le modèle microscopique des particules a le même comportement que l'équation de Fick (modèle macroscopique). La figure 2.2 permet de comparer visuellement les deux modèles.

La figure 2.2 montre les résultats d'une simulation de la diffusion réalisée avec 10^5 particules pour une durée simulée de $30s$ dans un cube d'un volume de $1cm^3$. La vitesse des particules est déterminée par tirage aléatoire suivant une loi uniforme entre 0 et $vmax = 0.1\,cm.s^{-1}$. L'équation 2.4 donne un coefficient de diffusion D d'une valeur de $5.10^{-4}\,s^{-2}$. La simulation individus-centrée s'ajuste très bien avec l'équation de Fick. Un test du χ^2 indique que les deux distributions ne sont pas significativement différentes : $p(\chi^2 = 24,76) > 0,5$ avec 30 degrés de liberté. La méthode utilisée ici est en fait la reproduction de l'expérience réalisée en laboratoire par le physicien J.B. Perrin et ses élèves pour mesurer le coefficient de diffusion [Col97].

La première utilisation que nous faisons ici de la multi-modélisation permet de déterminer des valeurs particulières de paramètres ou de fonctions de ces paramètres. Le modèle individus-centré est alors considéré comme un laboratoire virtuel dans lequel nous conduisons des expériences sur le système pour déterminer des paramètres du modèle agrégé, ici D, que nous pouvons par la suite intégrer à l'équation différentielle (équation 2.2) pour une résolution analytique.

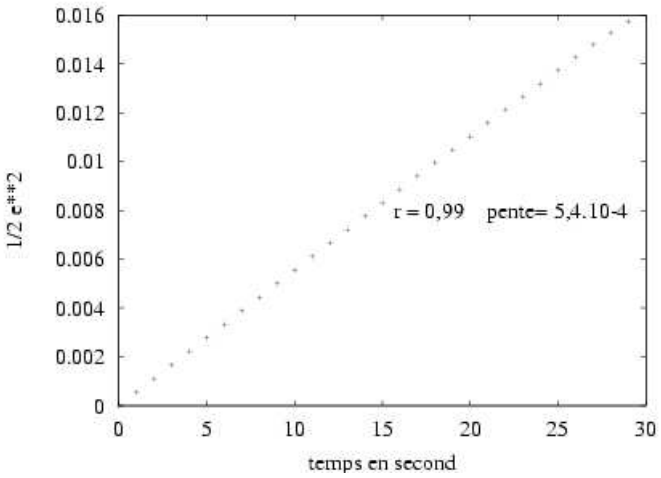


FIG. 2.1 – Équart quadratique moyen $\frac{1}{2}\overline{\varrho}^2$ en fonction du temps. La pente de cette droite correspond au coefficient de diffusion D.

2.2.3 Couplage des deux modèles

Dans le paragraphe précédent, nous avons montré qu'il était possible de paramétrer une équation analytique à l'aide d'un modèle individus-centré. Nous allons montrer ici qu'il est également possible de paramétrer un schéma numérique en cours de résolution.
Dans la plupart des cas, nous ne disposons pas d'une expression de D en fonction de $vmax$ qui pourrait être directement utilisée dans l'équation 2.2. Nous proposons donc d'identifier chaque D_i du modèle numérique (équation 2.3) à l'aide de la méthode présentée au paragraphe précédent. Pour cela, nous utilisons un modèle individus-centré sur 31 individus seulement, pour limiter le temps de calcul. Nous avons vérifié que les résultats sont équivalents à ceux obtenus en simulant 10^5 individus.

Nous simulons donc la diffusion à l'aide du modèle individus-centré pour déterminer, à chaque itération, et pour chaque tranche du cube la valeur des D_i. Nous réintroduisons ces valeurs dans l'équation du modèle numérique pour ensuite déterminer par résolution du modèle numérique les valeurs des différentes concentrations dans chacune des tranches d'espace. La figure 2.3 illustre la dynamique du couplage à l'aide d'un diagramme de séquence UML.

Même si les conditions de simulation restent identiques, la grande stochasticité du modèle individus-centré conduit à des valeurs de D_i variables dans un petit intervalle. C'est pourquoi nous avons choisi un schéma numérique qui considère D variable (équation 2.3). Pour la résolution d'un tel schéma, il est important de connaître le domaine de variation de D_i afin de respecter les conditions de stabilité, définies par l'équation 2.5.

$$\frac{\Delta t}{\Delta x^2}(D_{i+1} + D_{i-1}) < 1 \tag{2.5}$$

Dans notre simulation, le pas de temps et le pas d'espace sont égaux respectivement à $\Delta t = 1$ et $\Delta x = 10^{-2}$, ce qui suppose une valeur maximale du coefficient de diffusion vérifiant la condition suivante : $2D_{max} < 10^{-2}$.

On peut se demander quel est l'intérêt d'un tel couplage. En effet, nous disposons d'un mo-

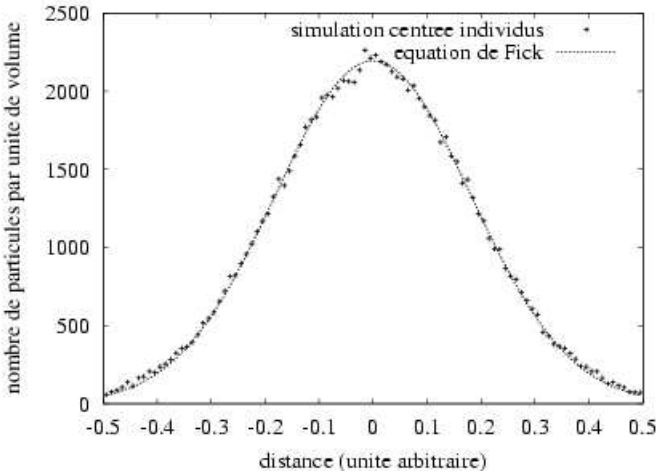


FIG. 2.2 – Simulation centrée individus du phénomène de diffusion pour 10^5 particules (croix) et équation de Fick (ligne pointillée) à $t = 30s$. La simulation et la courbe correspondent presque parfaitement. On considère les deux modèles comme équivalents par leur trace d'exécution.

dèle numérique et d'un modèle IBM qui simule la même chose, aux même échelles d'espace et de temps. Néanmoins, nous pouvons déjà rappeler que nous ne possédons pas de modèle agrégé qui relie la vitesse individuelle des particules à la dynamique globale du système. Le modèle IBM semble d'ores-et-déjà plus expressif que le modèle agrégé. Seulement, les simulations particulaires sont beaucoup plus longues que la résolution d'un schéma numérique. Nous allons montrer qu'il est possible de tirer avantage de deux modèles en donnant un exemple.
Le couplage des deux modèles peut permettre de simuler les effets spatiaux de la variation du coefficient de diffusion sur la dynamique du système. Nous allons illustrer cet exemple en considérant que la concentration des particules a un effet sur leur vitesse, par exemple en considérant une réaction de fuite des particules les unes par rapport aux autres. Cette réaction à la concentration peut être modélisée par une équation de type Monod qui fait augmenter la vitesse en fonction de la concentration jusqu'à un maximum fixe (équation 2.6).

$$v = v_{max} - v_{max} e^{-bC(x,t)} \tag{2.6}$$

où v est la vitesse des particules, v_{max} la vitesse maximum et $C(x,t)$ la concentration courante.

Si nous voulions simuler ce système avec le seul modèle individus-centré, il serait alors nécessaire de développer des algorithmes performants de recherche des plus proches voisins pour déterminer les concentrations locales à chaque particule. Ceci alourdirait grandement les calculs. Nous proposons donc d'utiliser un couplage entre les deux modèles pour simuler le système. Le modèle est le même que précédemment avec, pour l'expression de la vitesse des particules, $v_{max} = 0.5cm.s^{-1}$ et $b = 10^{-3}$. Le coefficient de diffusion maximum est mesurable sur notre système et est égal à $D_{max} = 2.10^{-3}$, ce qui vérifie les conditions de stabilité énoncées par l'équation 2.5. Nous obtenons les résultats illustrés par la figure 2.4. La différence avec le cas d'un simple paramétrage est que le modèle individus-centré a besoin de la valeur de la concentration locale à chaque instant, donnée par le modèle agrégé, pour calculer un nouveau coefficient de diffusion.

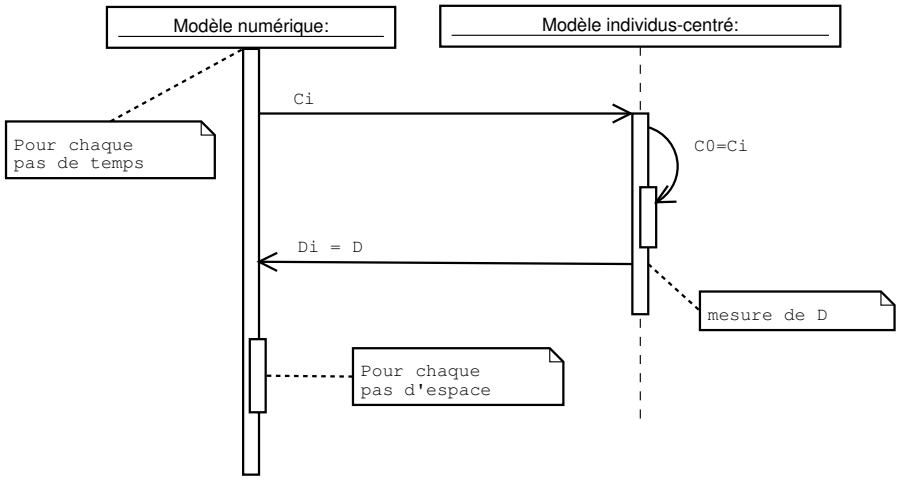


FIG. 2.3 – Diagramme de séquences UML pour l'illustration du couplage entre modèle numérique et modèle individus-centré. Pour chaque pas d'espace Δi, le modèle numérique utilise un modèle individus-centré initialisé avec la concentration correspondante pour déterminer le coefficient de diffusion local.

Avec une telle hypothèse de modélisation (vitesse dépendante de la concentration locale), la figure 2.4 montre que les particules n'atteignent pas le bord du cube pour une même durée de simulation et que la forme en cloche de l'équation de Fick n'est pas conservée pour les petites concentrations. Il n'est pas question ici de faire une étude précise de ce phénomène mais de montrer qu'un tel couplage peut constituer une méthode d'investigation de l'influence des caractéristiques individuelles sur une dynamique globale. Nous voyons également comment le modèle agrégé contraint le modèle individus-centré en lui fournissant une concentration locale. Lorsque l'on considère les deux modèles couplés comme un seul modèle, nous faisons figurer deux niveaux d'abstraction différents d'un même système dans un unique modèle. Peut-on déjà parler ici de transfert d'échelles ? En un sens, oui. Même si les deux modèles simulent une même échelle de temps et d'espace, ils échangent des données de niveaux d'abstraction différents. Néanmoins nous aimerions aller un peu plus loin et montrer qu'il est possible de prendre en considération l'espace et le temps.

2.2.4 Méthode pour le transfert d'échelles

Dans les deux exemples donnés précédemment (paramétrage et couplage), nous avons simulé une même durée dans un même espace pour une même concentration. Les deux modèles considéraient les même échelles de temps et d'espace. Cependant, les mêmes méthodes peuvent être appliquées à des échelles différentes. Dans notre exemple, le changement d'échelle ne modifierait pas *a priori* les dynamiques donc la méthode ne présente pas d'intérêt dans ce cas précis. Néanmoins, cet exemple nous a permis d'introduire l'utilisation du couplage entre un modèle individus-centré et un modèle mathématique agrégé comme une technique possible pour faire coexister deux niveaux d'abstraction différents dans une même simulation. Ainsi, nous allons poursuivre la démarche en séparant les échelles de temps caractéristiques des deux modèles.

Nous nous inspirons ici d'une méthode connue en mathématique sous le nom de « méthode

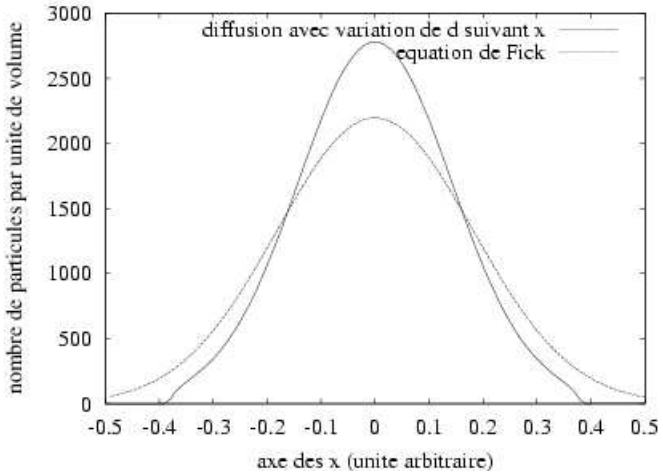


FIG. 2.4 – Résultat de la simulation du phénomène de diffusion pendant $30s$ en considérant que la vitesse des particules diminue avec la concentration (ligne pleine) par comparaison à l'équation de Fick où la vitesse est supposée constante.

d'agrégation de variables » ou de « réduction[27] » [Pog94]. Cette méthode est basée sur un constat simple. Si plusieurs échelles de temps sont en jeu, certaines variables sont lentes (caractéristiques des grandes échelles) relativement à d'autres qui sont rapides (caractéristiques des petites échelles). En schématisant, nous pouvons dire que les variables lentes peuvent être considérées comme des constantes par rapport aux variables rapides. Réciproquement, les solutions stationnaires des équations attachées aux variables rapides (état d'équilibre) sont considérées comme des constantes sur le domaine de variation des variables lentes. Ceci suppose que les équations des variables rapides vont très vite à l'équilibre pour toute variation des variables lentes. H. Haken [Hak91] a mis au point cette technique pour des systèmes d'équations différentielles. Les travaux de L. Fashe *et. al.* sont également à la base de notre travail [FWG98]. Ils définissent un protocole pour identifier les paramètres d'un modèle de dynamique de population à partir d'un IBM particulier. Sans le dire explicitement, ils utilisent leur modèle comme un laboratoire virtuel sur lequel ils font des expériences pour déterminer les paramètres d'un modèle agrégé. C'est la démarche que nous avons suivie au paragraphe précédent. Ils constatent que les paramètres identifiés émergent de la simulation de l'IBM et y voient un nouveau type d'investigation pour la compréhension des modèles agrégés en écologie théorique. Des travaux plus mathématiques existent également sur le paramétrage d'équations différentielles à l'aide d'un IBM [WJ99]. Nous voulons montrer qu'une généralisation de la méthode est possible et applicable au couplage de modèles à des niveaux d'abstraction et des échelles de temps différents.

Dans la communauté des SMAs, les résultats de simulations (ou traces d'exécution) sont considérés comme un épiphénomène, c'est-à-dire une manifestation extérieure et une conséquence de la simulation [Mrj97]. Cet épiphénomène peut à son tour être modélisé, nous parlons alors de méta-modélisation. De plus, si ce méta-modèle est une fonction mathématique, alors nous sommes dans le cas d'un calcul émergent [For90]. En d'autres termes, la fonction mathématique et le modèle de départ sont équivalents au regard de leurs traces de simulation.
En utilisant le couplage entre deux modèles à deux niveaux d'abstraction différents (paragraphe

[27] Cette méthode est également appelée « méthode de variation de la constante ».

précédent), le principe de séparation des échelles de temps des processus et le calcul émergent, nous proposons une méthode générale de couplage pour la simulation du transfert d'échelle. Cette méthode se résume en cinq points :

1. Modéliser un même système à plusieurs niveaux d'abstraction différents.
   Pour tous les niveaux d'abstraction, identifier les mêmes entités du système et leur représentation dans un paradigme particulier du niveau d'abstraction (par exemple des scalaires pour un modèle agrégé).
2. Séparer les échelles de temps et d'espace entre niveaux.
   C'est-à-dire, représenter les processus spatio-temporels caractéristiques du niveau d'abstraction considéré.
3. Considérer un modèle de bas niveau comme un laboratoire virtuel pour un modèle de haut niveau.
   Ce qui revient à faire des expériences virtuelles pour identifier « expérimentalement » des paramètres ou des fonctions qui émergent du modèle de bas niveau et qui sont présents dans le modèle de haut niveau.
4. Paramétrer le modèle de haut niveau ou intégrer le modèle de bas niveau dans le modèle de haut niveau.
   Dans le deuxième cas, le modèle de bas niveau est équivalent à un composant du modèle de haut niveau.
5. Définir les conditions de simulation du modèle de bas niveau à l'aide du modèle de haut niveau.
   Ce qui revient à dire que l'environnement global (conditions initiales, conditions aux limites) du modèle de bas niveau est calculé par le modèle de haut niveau.

Le point 3 a été illustré au paragraphe 2.2.2, les points 4 et 5 au paragraphe 2.2.3. Néanmoins, nous n'avons pas illustré le transfert entre deux échelles de temps. De plus, nous pensons que l'expérimentation virtuelle peut être un moyen de construire un modèle agrégé d'un phénomène difficile à appréhender expérimentalement. Dans ce qui suit, nous allons illustrer ces deux points sur notre modèle agent du copépode, montrant du même coup, sur un modèle plus complexe, les potentialités de notre démarche. La figure 2.5 représente l'approche méthodologique citée plus haut dans le cas de l'écologie.

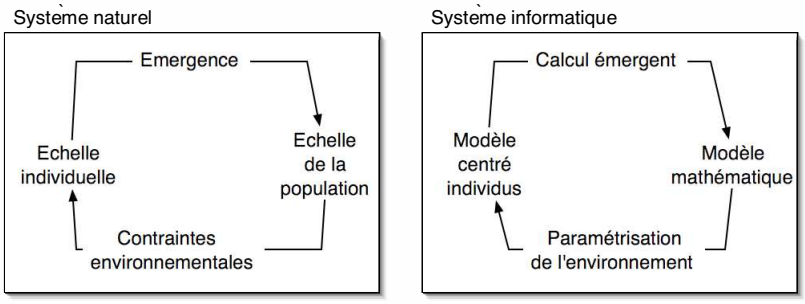


FIG. 2.5 – Approche conceptuelle pour la modélisation du transfert d'échelle entre deux niveaux d'organisation dans les systèmes naturels. La définition de calcul émergent et le principe d'expériences virtuelles sont à la base de cette approche. Les flèches illustrent à la fois le flot de données échangées par les deux modèles (côté informatique) et la boucle de rétroaction d'un niveau d'organisation sur l'autre.

2.3 Présentation du système étudié

Nous allons décrire ici le système considéré comme cas d'étude et d'application dans les chapitres 3 et 5. Il s'agit d'un système proies-prédateurs. Ce type de système est très classique en écologie. Comme son nom l'indique, il illustre un type d'interaction majeur dans les écosystèmes : la prédation de proies par des prédateurs. Ce système est étudié de façon théorique depuis près de 80 ans ! Lotka [Lot25] et Volterra [Vol26] sont les premiers à avoir mis en équation cette relation de prédation et leur modèle est à la base de tous les modèles d'équations différentielles construits en biologie et en écologie.
Ce système, bien connu, va nous permettre d'illustrer l'utilité de l'intégration de deux modèles hétérogènes pour modéliser un système que nous pouvons qualifier de complexe. En effet, les proies et les prédateurs sont des individus qui ont un comportement, une physiologie[28], une histoire, etc, qui rendent leur étude et *a fortiori* leur modélisation, complexes. Si nous considérons une espèce[29] de proies et une espèce de prédateur, nous pouvons identifier deux types d'interaction :

1. interaction directe : les prédateurs capturent les proies,
2. interactions indirectes : par exemple, la quantité ou la distribution des proies a une influence sur l'efficacité des prédateurs (*i.e.* la rapidité avec laquelle les prédateurs consomment les proies), ou encore l'effet de compétition quand plusieurs prédateurs se disputent la même ressource.

Chaque espèce possède ses propres caractéristiques physiques qui lui confèrent certaines aptitudes. Par exemple, le système de perception et de locomotion des prédateurs conditionne leur mode de prédation. Il en est de même en ce qui concerne la fuite des proies. Toutes ces caractéristiques conditionnent la dynamique d'un système proies-prédateurs.
Dans notre travail, nous allons utiliser une nouvelle approche pour décrire ce type de système en le considérant à deux niveaux d'abstraction différents dans le même modèle. Le premier niveau concerne celui de l'individu. Nous modélisons les proies et les prédateurs de façon individuelle en utilisant le paradigme des agents réactifs situés. Le second niveau est celui de la population, où les proies et les prédateurs sont modélisés par des variables réelles manipulées par des équations différentielles. Dans ce qui suit, nous présentons d'abord le système réel, puis les deux modèles.

2.3.1 Système réel et question posée

Nous nous intéressons à un groupe majeur du zooplancton marin[30], les copépodes. La figure 2.6 nous montre des spécimens adultes.

Nous ne présentons ici que les caractéristiques individuelles qui nous intéressent particulièrement. Elles seront abordées au fur et à mesure dans le texte. Le lecteur intéressé par ce groupe fascinant pourra se tourner vers l'un des très nombreux ouvrages dédiés à l'étude du zooplancton. Un manuel est paru récemment sur l'étude du zooplancton qui donne une vue générale des méthodes d'études et modèles existants [CGW00]. Les proies du système sont des

[28] Ensemble des mécanismes physico-chimiques qui permettent à un organisme de se déplacer, se nourrir, se reproduire etc.

[29] Ensemble des individus capables de se reproduire entre eux.

[30] Ensemble des animaux aquatiques qui n'effectuent que des déplacements très petits relativement à celui des masses d'eau.

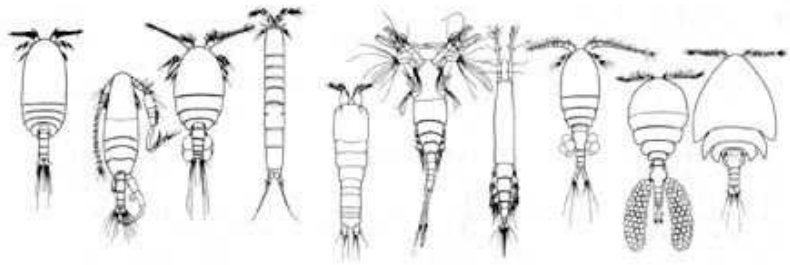


FIG. 2.6 – Vue dorsale de quelques copépodes adultes. Les « antennes » au niveau de la tête sont des organes importants pour la perception des proies. La taille des copépodes adultes est très variable (de 0,5mm à 1cm environ).

cellules de phytoplancton[31](figure 2.7). Elles constituent une des sources de nourritures privilégiées des copépodes[32]. Le phytoplancton est autotrophe (il fabrique sa matière organique à partir de l'inorganique) et constitue une base de l'écosystème marin, appelée production primaire. Les copépodes sont eux hétérotrophes et se situent juste au-dessus du phytoplancton dans le réseau trophique (ancienne pyramide alimentaire). Ainsi ils constituent l'un des groupes principaux de la production secondaire. Les copépodes sont la source de nourriture de la plupart des larves de poissons, ainsi que de certains adultes. Ils jouent donc le rôle de pivot entre la production primaire et de nombreuses espèces de poissons commerciaux. C'est une des raisons pour laquelle ils sont très étudiés en écologie marine. D'un point de vue scientifique, ils permettent de comprendre le flux de matière et d'énergie dans l'écosystème marin et jouent donc un rôle important dans la compréhension du cycle du carbone. Les grands programmes internationaux comme JGOFS[33] ou GLOBEC[34] développent des modèles de réseaux trophiques dans lesquels le compartiment zooplancton est représenté par les copépodes. Trouver des moyens de représentation efficaces et pertinents de la relation trophique entre zooplancton et phytoplancton est donc une question majeure de ce type de programme.

Notre travail s'inscrit dans le cadre du Programme National sur l'Environnement Côtier (PNEC)[35] et plus particulièrement dans l'Action de Recherche Thématique (ART) 2 : « Dynamique de populations : structures hydrodynamiques et cycles biologiques ». En 1998, notre équipe a collaboré avec la station marine de Wimereux. Cette collaboration entre informaticiens et écologues a eu pour but de construire un modèle du copépode à base d'agents réactifs. L'approche adoptée dans ce premier modèle est classique et relativement simple. L'espace est discret et réduit à deux dimensions bien que l'environnement, une masse d'eau, soit clairement à trois dimensions. Cette approche a permis de mettre en évidence de façon mécaniste, une nage orien-

[31]Ensemble des organismes végétaux unicellulaires aquatiques qui n'effectuent que des déplacements très petits relativement à celui des masses d'eaux.

[32]Le régime alimentaire des copépodes est directement lié à l'espèce considérée. Ils sont majoritairement omnivores et même cannibales...

[33]Site JGOFS : `http://www.uib.no/jgofs/jgofs.html`

[34]Site GLOBEC : `http://www.pml.ac.uk/globec/main.htm`

[35]Information sur le PNEC : `http://www.cnrs.fr/cw/dossiers/doseau/recher/program/pnec.html`

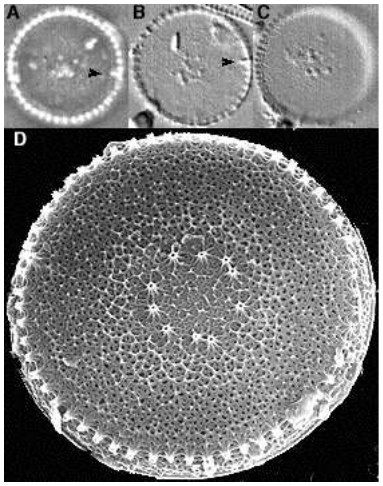


FIG. 2.7 – Photographie au microscope électronique d'une coupe transversale de la micro-algue *Thalassiosira weissflogii*. Cette espèce fait partie du phytoplancton marin. Elle mesure entre 10 et 20 μm de long. Sa couleur varie du brun au vert, en passant par le jaune, en fonction de sa teneur en chlorophylle.

tée du copépode vers les cellules du phytoplancton [RPSL98]. Depuis, nous avons continué à utiliser ce type d'approche avec un nouveau modèle en trois dimensions dans un espace continu et en temps continu que nous présentons par la suite. Actuellement, en collaboration avec le Centre d'Océanologie de Marseille, plus particulièrement avec J.C. Poggiale, mathématicien, la station marine d'Arcachon avec F. Carlotti, écologue spécialiste des copépodes, et l'université de Rennes, avec Y. Lagadeuc, lui aussi écologue et spécialiste des copépodes, nous travaillons sur les aspects plus théoriques de l'apport de la modélisation à base d'agents en écologie marine. Nous voulons illustrer ces apports en essayant d'apporter une méthode permettant de répondre à la question suivante :

> « Comment l'hétérogénéité de la distribution des proies influence-t-elle la prédation des copépodes et a-t-elle une influence sur la dynamique de population ? »

Cette question fait intervenir des échelles de temps et d'espace différentes. En effet, le processus de prédation *stricto sensu* a lieu à l'échelle de la seconde pour les copépodes se nourrissant de phytoplancton, alors que la dynamique de population des copépodes est de l'ordre du mois et celle du phytoplancton de l'ordre de la journée[36]. Nous sommes donc amenés à développer une approche de modélisation qui permette l'intégration de ces différentes échelles de temps.
Beaucoup de travaux concernent l'étude de la relation entre la turbulence et le taux de rencontres (le nombre de proies rencontrées par unité de temps et par prédateur) [CC96] [YOS91] [Osb96] [CPC98]. Ils mettent en évidence que la turbulence influence ce taux de rencontre et donc l'efficacité de capture des prédateurs. En mesurant la quantité d'azote absorbée lors de la nutrition, les observations montrent que le rendement du comportement du copépode (énergie dépensée/énergie ingérée) varie en fonction du type de distribution de la nourriture [BGCS93]. Toutefois, ces travaux ne mettent pas en évidence l'influence de la turbulence à micro-échelle sur la dynamique de population. Ceci est dû à la différence entre les échelles de temps et d'espace considérées.

[36] Ce sont en fait les durées de vie des deux entités en présence.

Il existe également beaucoup de modèles basés sur des équations différentielles qui décrivent la dynamique de population des copépodes et du phytoplancton ainsi que celle de leur interaction trophique [CGW00]. Ces modèles sont plus ou moins précis dans la description du cycle de vie du copépode. Les équations différentielles ont le grand avantage de manipuler des grandeurs scalaires et donc de pouvoir prendre en compte un grand nombre d'individus. La contrepartie du nombre est la difficulté, voire l'impossibilité d'exprimer des mécanismes individuels précis et discrets dans le temps et l'espace. Une possibilité pour prendre en compte ses mécanismes et le paradigme d'agent réactif. Dans ce cas, les possibilités pour exprimer des comportements et des interactions sont très grandes. La contrepartie est qu'une simulation ne peut se faire qu'en considérant un nombre limité d'agents, fonction à la fois du degré de finesse dans l'implémentation des processus et de la puissance de calcul disponible. Nous voyons ici que les deux approches, continue et discrète, apparaissent complémentaires. Pour tenter de répondre à la question posée plus haut, nous allons donc proposer une méthode d'intégration de modèles hétérogènes (un modèle d'agents réactifs et un système d'équations différentielles) pour simuler un transfert d'échelles entre ces deux vues du système proies-prédateurs en question.

L. Seuront a montré que la distribution du phytoplancton peut se caractériser par des lois de distribution multifractale [SSL$^+$99][SSL$^+$96] et que cette distribution est induite par la turbulence. Actuellement et à notre connaissance, il n'existe pas de relation mathématique entre la turbulence et les paramètres de ces lois. Ce qui est intéressant de retenir ici, c'est que la distribution du phytoplancton est hétérogène sous l'effet de la turbulence. L'intuition voudrait nous faire croire que la turbulence homogénéise les distributions, mais la présence d'intermittences de la turbulence induit des « *pulses* » qui fractionnent les distributions [SSL01].
Devant cet état de fait, il semble possible de représenter l'effet de la turbulence sur l'efficacité de capture des copépodes en représentant des distributions hétérogènes de cellules de façon discrète. Pour cela, nous émettons une hypothèse forte : la nature de la distribution ne change pas au cours du temps, ce qui implique que la turbulence est considérée comme constante. À l'heure actuelle, nous ne disposons pas de modèle reliant les variations de la turbulence à une distribution discrète du phytoplancton. Dans les travaux cités plus haut, la modélisation est purement mathématique et les modèles développés ne prennent en compte le comportement que de manière très indirecte. Nous voulons montrer qu'une approche discrète peut permettre de rendre compte de l'effet de l'hétérogénéité de la distribution des proies sur les copépodes.

Le système réel considéré ici se résume donc à deux espèces marines en interaction de type proie-prédateur dans un certain volume d'eau. Nous allons donc être amenés à faire des choix pour la représentation des entités du système que forment les copépodes, le phytoplancton et le volume d'eau.
Dans la partie application de cette thèse, nous essayons de montrer comment la multi-modélisation peut être une méthode originale pour prendre en compte le comportement alimentaire du copépode dans des modèles de dynamique de population. Nous nous intéressons plus particulièrement à une espèce de copépode, *Acartia Tonsa* et à une espèce de phytoplancton, *Thalassiosira weissflogii*(figure 2.7). Notre modélisation est basée sur les travaux de P. Caparroy [CC96] qui a développé un modèle mathématique du comportement alimentaire de ce copépode se nourrissant sur cette algue. Nous allons encapsuler ce modèle dans un agent réactif.

2.3.2 Le modèle à petite échelle : un IBM conçu comme un système d'agents réactifs

Dans ce paragraphe, nous présentons le modèle de bas niveau de façon informelle. Nous allons donc décrire les entités et les processus que nous allons considérer. Nous choisissons une description des mécanismes de la prédation au niveau individuel en modélisant le déplacement des copépodes, leur perception et l'ingestion de proie. Nous construisons pour cela un modèle basé sur le paradigme d'agents réactifs. L'environnement des individus est constitué par une masse d'eau que nous ne représentons pas explicitement. Nous considérons un espace continu cubique dans lequel évoluent les prédateurs.

La première entité, l'algue *T. weissflogii*, n'a pas de déplacement propre. Dans le milieu naturel, cette algue est affectée par la turbulence. Nous ne représentons pas la turbulence directement, mais seulement son effet sur la distribution des proies. C'est donc le type de distribution qui reflétera le niveau de turbulence du fluide.
Devant le rapport de taille entre *A. Tonsa* et *T. weissflogii* (approximativement de 1 pour 1000) nous avons choisi de modéliser les cellules de phytoplancton sous forme de points, localisés par leurs trois coordonnées (x,y,z). Nous considérons que cette entité est un objet passif de l'environnement.

La deuxième entité considérée est le copépode. C'est l'agent réactif de notre système. Il existe des travaux récents qui modélisent le copépode de façon individuelle par une géométrie approchée du corps de l'animal [HOM02b] [HMO02]. Ces travaux concernent une étude des écoulements du fluide autour du copépode et son rôle dans la perception des proies. Dans notre modèle, nous ne considérons le fluide qu'indirectement au travers du coût énergétique de la nage (voir annexe B). Nous ne modélisons donc pas le volume corporel du copépode ; celui-ci est simplement représenté par ses coordonnées dans le repère attaché au cube dans lequel il évolue. C'est le volume de perception qui nous intéresse particulièrement et que nous représentons car c'est lui qui intervient dans le processus de prédation. Ce processus peut être décomposé en une série chronologique d'évènements aboutissant à l'ingestion de la proie. Bien que la structure précise du cycle de la prédation puisse varier entre les groupes taxonomiques de prédateurs zooplanctoniques, la modélisation du processus de prédation chez un copépode nécessite la représentation minimale des phases de recherche, perception, poursuite ou chasse, capture et ingestion [PP86]. Nous ne nous intéressons pas aux phases de reproduction ou de croissance du copépode. Nous construisons donc un modèle purement comportemental. Voyons maintenant les différentes phases modélisées.

Phase de recherche de nourriture : déplacement autonome de l'agent

La phase de recherche de nourriture se caractérise par une exploration de l'espace par le copépode. De nombreux modèles mathématiques ont été proposés dans le but de représenter des déplacements simples, avec des nages de type rectiligne [GS77] ou un mouvement de type brownien [YOS91]. Le modèle retenu par P. Caparroy est celui de Saiz et Kiørboe [Sr95] mis au point en 1995. Ce modèle permet de rendre compte de deux types de nage du copépode, une nage lente et une chute passive, et de les mettre en relation (mathématique) avec le taux de rencontre. Dans la réalité, le déplacement d'un copépode peut être extrêmement complexe, avec une alternance de mouvements rectilignes, hélicoïdaux et/ou de chutes passives, voire de « sauts ». Les travaux précédents de notre équipe ont montré qu'une modélisation où le copépode s'oriente vers ses proies lorsqu'il les perçoit, et adopte une marche aléatoire sinon, reproduit de

façon assez satisfaisante les distributions de phytoplancton observées [RPSL98][37]. P. Caparroy considère que le copépode passe 95% de son temps à nager. Nous considérons dans notre modèle qu'il passe 100% de son temps à nager en ignorant les phases de sauts. Nous ne connaissons pas le déterminisme lié au changement de direction du copépode en l'absence de nourriture. Dans notre modèle, le copépode en phase de recherche se déplace donc aléatoirement dans les trois directions de l'espace.

Phase de perception : réponse à un stimulus externe

Les preuves que les copépodes sont capables de détecter la présence et la position de proies distantes se sont accumulées depuis les années 80 avec l'apparition de la cinématographie à haute résolution [APS80]. Ces observations directes ont montré que le copépode attaque ses proies par un déplacement actif. Les organes utiles au processus sont situés sur la tête du copépode (voir figure 2.6) ; ce sont les divers antennes ou appendices céphaliques de l'animal, où se situent des zones sensibles à deux types de signaux. Ces signaux sont à l'origine de deux types de perceptions :

– la chémoréception,
– la mécanoréception.

La chémoréception est un processus chimique qui fait intervenir la détection de molécules organiques d'origine phytoplanctonique par le copépode. Cette détection est liée à la diffusion des molécules dans le fluide mais également à la génération d'un courant alimentaire par un mouvement coordonné des appendices céphaliques du copépode [HOM02a]. Ce courant alimentaire détermine un volume de perception dans lequel les proies peuvent être localisées.
La mécanoréception est un processus mécanique qui fait intervenir la détection des déformations des lignes de courant du fluide environnant le copépode lorsque celui-ci s'approche d'une proie immobile, ou lorsqu'une proie mobile passe à proximité. Il a été montré que ce processus pouvait être totalement indépendant de la chémoréception [BGVS98]. Une étude récente montre que lors de phases de chute passive du copépode, c'est le mécanisme de mécanoréception qui prédomine [Sr00]. Néanmoins, les deux mécanismes sont présents et agissent en synergie pour la plupart des copépodes dont *A. Tonsa* [Paf98]. Dans les deux cas, la proximité de la proie (sa position) est prépondérante [HOM02a] et se fait à partir d'une certaine distance dont l'origine est la tête de l'animal. Comme nous ne représentons pas le fluide explicitement, nous intégrons la chémoréception et la mécanoréception au sein d'un même volume de perception. L'entrée d'une cellule dans ce volume déclenche l'entrée dans la phase de chasse.

Phase de chasse : déplacement en réponse à un stimulus externe

Une fois perçue par le copépode, la proie peut être redirigée vers les appendices buccaux par le courant alimentaire [HOM02a]. *A. Tonsa* ne semble pas capable de rediriger activement ses proies [CC96]. De nombreuses espèces de copépodes se réorientent vers leur proie si elle est susceptible d'éviter le volume de capture [HOM02a]. Nous modéliserons donc la phase de chasse par une réorientation et un déplacement du copépode en direction de sa proie afin qu'elle entre dans son volume de capture.

Phase de capture : réponse à un stimulus externe

[37]Il a été établi que cette représentation était valide statistiquement, en utilisant des méthodes d'analyses de fréquences basées sur les multifractales

Le processus de capture apparaît comme l'une des étapes du cycle de la prédation la plus complexe à modéliser [Cap96] ou paramétrer de par l'absence d'étude quantitative complète chez les copépodes. Seules des observations visuelles par cinématographie haute fréquence donnent des descriptions qualitatives. Elles ont par exemple montré que la décision d'ingérer ou non une particule était le plus souvent prise après la capture de celle-ci [BGVS98]. Ce comportement reflète la possibilité pour le copépode de choisir d'ingérer ou non une proie en fonction de ses qualités nutritionnelles. Dans notre modèle, nous ne nous intéressons qu'à une espèce de phytoplancton ; ce type de comportement ne sera donc pas pris en compte.
A. Tonsa utilise sa seconde paire de maxilles pour capturer ses proies. Les données disponibles sont le temps de manipulation de l'algue *T. weissflogii* et le rayon du volume de capture d'*A. Tonsa* modélisé par une demi-sphère [CC96]. De plus, la phase de capture se traduit par un arrêt de l'activité de nage du copépode et une chute passive, conséquence de cet arrêt.

Ingestion et processus métaboliques : états internes de l'agent

Il existe de nombreux modèles analytiques qui ont pour ambition de représenter les processus biologiques et physiologiques qui affectent les copépodes [CGW00]. P. Caparroy [CC96] propose un modèle synthétisant les différents modèles développés jusqu'à présent. Il résume, à l'aide de cinq équations différentielles interdépendantes, l'activité de capture et de digestion. Nous avons choisi d'utiliser ce modèle en nous limitant à la digestion, l'activité de capture étant le fruit du déplacement du copépode dans le volume d'eau et de sa perception des proies. Une description détaillée du modèle d'ingestion utilisé est donnée en annexe B.
Revenons sur le processus d'ingestion des proies. Le copépode capture une proie. Après un temps de manipulation, celle-ci est stockée dans l'estomac et entre dans le processus de digestion. L'estomac transforme son contenu soit en énergie utilisable (proies assimilées), soit en déchets (pelotes fécales). Cette transformation est continue. L'énergie utilisable est soit mise à disposition du métabolisme (digestion, nage, etc.) soit stockée (pour la production d'œufs chez les femelles, par exemple). Quant aux déchets, ils sont évacués. La figure 2.8 présente le modèle conceptuel utilisé.

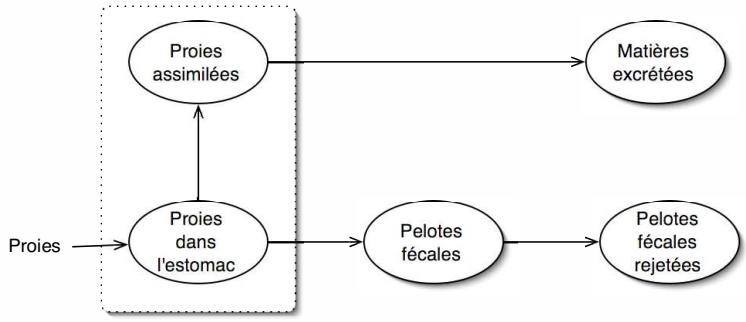


FIG. 2.8 – Modèle conceptuel du processus d'ingestion élaboré par P. Caparroy [CC96]. Les pointillés entourent les deux compartiments qui nous intéressent plus particulièrement. Le compartiment proies dans l'estomac permet de quantifier l'appétit du copépode (effet de satiété). Le compartiment proies assimilées permet de quantifier l'énergie disponible pour le copépode.

Ce modèle prend en compte trois phénomènes essentiels : la satiété (*i.e.* le niveau d'appétit), la vidange de l'estomac et l'excrétion. Les deux derniers phénomènes conditionnent le premier.

La satiété est fonction du contenu de l'estomac du copépode (cf. annexe B) et donc du temps nécessaire pour la digestion. Tous ces phénomènes jouent donc un rôle important dans « la prise de décision » du copépode d'ingérer ou non une cellule de phytoplancton.
Il est important pour nous de représenter la satiété. Nous voulons étudier l'influence de la distribution des particules sur l'ingestion du copépode. Aussi, lorsque les cellules sont regroupées en paquets, le copépode n'a pas à se déplacer pour capturer ses proies. C'est dans ces conditions que l'effet de satiété devient important sur son activité alimentaire. La fonction utilisée est donnée au début de l'annexe C. Dans cette annexe, nous donnons la formulation de l'entrée et de la sortie de l'état de satiété du copépode. Les valeurs de paramètres sont données par le tableau B.2 de l'annexe B. Ces formulations sont basées sur une valeur seuil du contenu en proie de l'estomac du copépode. Au-dessus de cette valeur, nous disons que le copépode est en état de satiété (il ne s'alimente plus) ; en-dessous, il reprend son alimentation.

Pour résumer, notre modélisation de l'activité alimentaire du copépode intègre un modèle physiologique basé sur des équations différentielles et un modèle géométrique de déplacement et de perception du copépode en 3D. La figure 2.9 présente une vue schématique de notre modélisation.

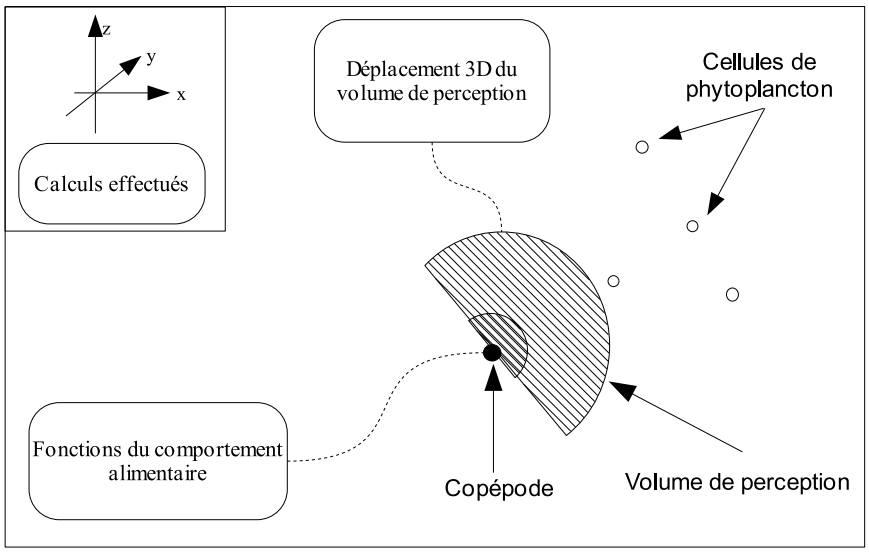


FIG. 2.9 – Résumé schématique du modèle d'agents réactifs du copépode. Le comportement alimentaire est contrôlé à la fois par des fonctions mathématiques (simulant l'activité métabolique de l'animal) et par le déplacement en 3D du volume de perception (demi-cercle hachuré) et de capture (demi-cercle doublement hachuré) dans un espace continu. Les cellules de phytoplancton sont fixes. Le dessin n'est pas à l'échelle.

Le modèle d'agents réactifs décrit ici intègre donc deux types de modélisations : une modélisation continue des processus physiologiques et une modélisation spatiale discrète des agents dans leur environnement. Cette intégration correspond à une vision « classique » de multi-modèles, où un modèle continu pilote, au moins partiellement, un modèle discret [Fis95]. Cette intégration constitue la différence majeure entre notre approche et celle de P. Caparroy où le comportement des copépodes se résume à deux paramètres d'une fonction mathématique permettant de relier

le niveau de turbulence au taux de rencontre.

2.3.3 Le modèle à plus grande échelle : un système d'équations différentielles

Le passage à l'échelle de la population implique de prendre en compte les processus affectant les individus tout au long de leur vie (naissance, croissance et développement, mortalité). Comme pour tous les êtres vivants, ces processus sont très nombreux et nous n'en n'avons qu'une connaissance partielle. Sans entrer dans des détails propres à la biologie du développement, les copépodes, comme tous crustacés, passent par différents stades de développement séparés par des mues (11 stades de l'œuf à l'adulte). Là encore, il existe de nombreux modèles mathématiques du développement [CN92].
Dans un cadre purement biologique, il serait très important de modéliser de façon précise les stades de développement. Dans notre travail, nous voulons montrer que les caractéristiques discrètes et individuelles influencent la dynamique globale. Il n'est pas question ici de faire un modèle quantitatif mais un modèle qualitativement pertinent, offrant un nouvel outil original d'étude en écologie théorique basé sur le couplage d'un modèle d'agents réactifs avec un système d'équations différentielles.
Le système d'équations différentielles doit décrire la dynamique des populations de phytoplancton et de copépodes en interaction. Ce système doit tenir compte des processus de croissance, de mortalité et d'interaction proies-prédateurs. Nous ne considérons pas la reproduction. Nous avons choisi un modèle classique de l'interaction proies-prédateurs appelé modèle de Holling-Tanner [Hol59][Tan75][BR93]. Ce modèle est une extension du modèle de Lotka [Lot25] et Volterra[Vol26]. Il est formé de deux équations différentielles ordinaires (équations 2.7 et 2.8).

Considérons d'abord l'équation de la dynamique des proies qui se décompose en deux parties. Une première partie ($rN(1 - \frac{N}{K})$) correspondant à la croissance de la population de la proie est modélisée par une fonction logistique, ce qui signifie que la croissance est limitée par la disponibilité de la ressource nutritionnelle pour les proies. Une deuxième partie ($G(N, P)P$), qui correspond à la pression de prédation exercée par le prédateur, est modélisée par une fonction particulière. Le signe négatif indique que cette partie est assimilée à la mortalité des proies. Cette mortalité est ici uniquement due à la prédation.

$$\frac{dN}{dt} = rN(1 - \frac{N}{K}) - G(N, P)P \tag{2.7}$$

avec :
t le temps,
N le nombre de proies,
K la capacité de charge du milieu,
P le nombre de prédateurs,
r le coefficient de croissance de proies,
$G(N, P)$ la fonction de l'intensité de prédation sur les proies.

L'équation décrivant la dynamique des prédateurs (équation 2.8) se décompose également en deux parties. Une partie correspond à la croissance des prédateurs par consommation de proies ($eG(N, P)$). L'autre partie modélise la mortalité naturelle des prédateurs (mP).

$$\frac{dP}{dt} = eG(N,P) - mP \tag{2.8}$$

avec :
t, N, P, $G(N,P)$ définis précédemment,
e un coefficient de transformation des proies en prédateurs,
m le coefficient de mortalité des prédateurs.

Si l'expression de $G(N,P)$ est simple, un tel système peut être étudié analytiquement [Pav94]. Il apparaît alors des équilibres dans l'évolution temporelle des variables N et P sous la forme de cycles limites stables pour certaines valeurs de paramètres [BR93]. Il en est de même pour d'autres formes de $G(N,P)$ [Jos98]. Ces études théoriques permettent notamment de déterminer les valeurs des paramètres des équations du système pour lesquelles nous observons des dynamiques particulières. Si nous sommes capables de relier les dynamiques individuelles à ce type de dynamique globale, alors nous pouvons simuler les effets des caractéristiques micro-échelles sur des échelles plus grandes. Ce sera le sujet développé au chapitre 5 dans lequel nous reviendrons précisément sur cette équation et son couplage avec le modèle d'agents réactifs (paragraphe 5.3.1 page 148).

Après avoir introduit la problématique de cette thèse et précisé le cas d'étude, nous allons commencer par la présentation de l'intégration formelle. Nous avons décrit de façon informelle notre modélisation agent du copépode et de façon formelle un modèle de dynamique de population. L'approche d'intégration proposée est basée sur DEVS. Comme nous l'avons dit en introduction de cette thèse, DEVS a l'ambition d'encapsuler la plupart des formalismes dédiés à la spécification des systèmes dynamiques.

3

Intégration formelle : couplage entre un système d'agents réactifs et un système d'équations différentielles

Résumé

Nous considérons deux modèles qui décrivent un même système proies-prédateurs où des copépodes se nourrissent de phytoplancton. Le premier modèle s'intéresse à la dynamique au niveau individuel en modélisant les phases de chasse, de capture et de manipulation des proies. Le second s'intéresse à une dynamique globale au niveau de la population des proies et des prédateurs. En intégrant formellement ces deux approches, nous fournissons une vision unifiée et non ambiguë de notre système couplé. Pour cela, nous utilisons le formalisme DEVS qui offre la possibilité d'intégrer plusieurs formalismes. Pour effectuer ce couplage formel, nous devons d'abord spécifier notre système d'agents réactifs en DEVS, ce qui nous amène à discuter d'une analogie entre ce formalisme et le paradigme des systèmes d'agents réactifs. Ensuite, nous exprimons le système d'équations différentielles comme un système à temps discret, ce qui nous permet de le considérer comme un cas particulier d'un système à évènements discrets, donc spécifiable en DEVS. Ce travail permet de mettre en lumière toute la dynamique du système. De plus, l'existence de simulateurs basés sur DEVS nous offre les algorithmes opérationnels pour notre implémentation. Durant la conception du modèle, nous avons intégré quelques potentialités de DEVS pouvant être utiles pour la simulation à base d'agents ou pour la modélisation des systèmes complexes plus généralement. DEVS nous apparaît comme une sémantique opérationnelle applicable à plusieurs types d'agents et, comme nous le verrons au chapitre 4, un cadre d'intégration solide pour la multi-modélisation.

Sommaire

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).

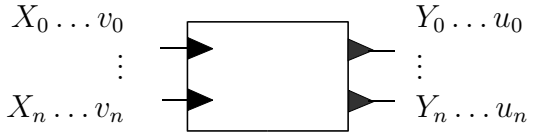


FIG. 3.1 – Représentation graphique d'un modèle DEVS atomique. Les triangles à l'intérieur de la boîte figurent les ports d'entrée. Les triangles à l'extérieur de la boîte figurent les ports de sortie.

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

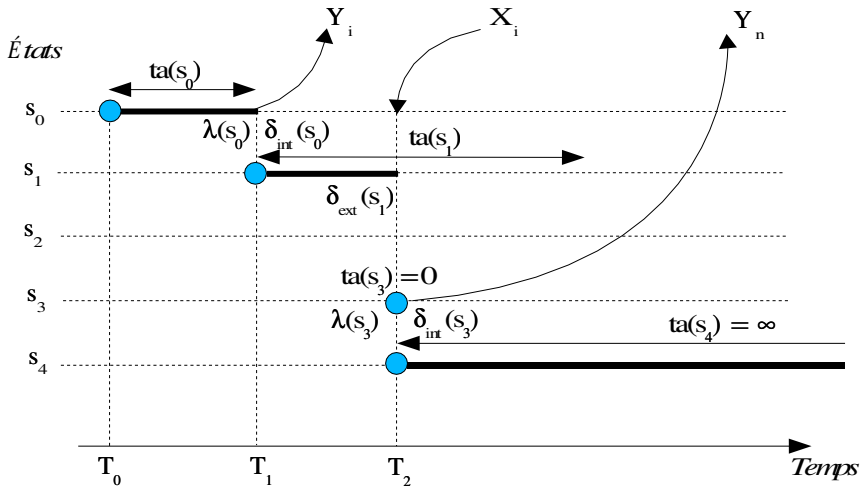


FIG. 3.2 – Exemple de graphe de transitions d'un modèle DEVS atomique. Les lignes pointillées verticales représentent les dates d'occurrences d'évènements. Les cercles pleins représentent l'état courant du système et les lignes pleines horizontales l'avancement du temps. Une transition est marquée par 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.

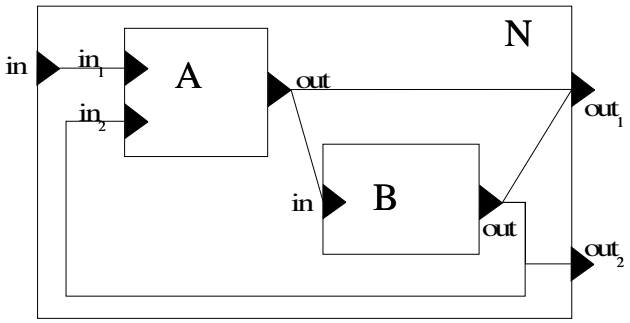


FIG. 3.3 – Représentation graphique d'un modèle DEVS couplé. Le nom des modèles est en lettres capitales. Le nom des ports est en minuscules.

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.

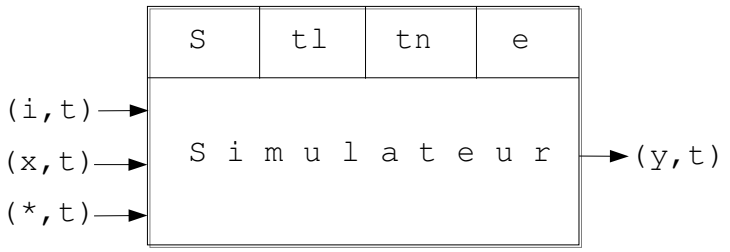


FIG. 3.4 – Le simulateur abstrait d'un modèle atomique. Il correspond à l'algorithme de simulation de la dynamique d'un modèle DEVS atomique. Les évènements d'entrée et de sortie sont décrits dans le texte et l'algorithme est donné en annexe D.1

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.

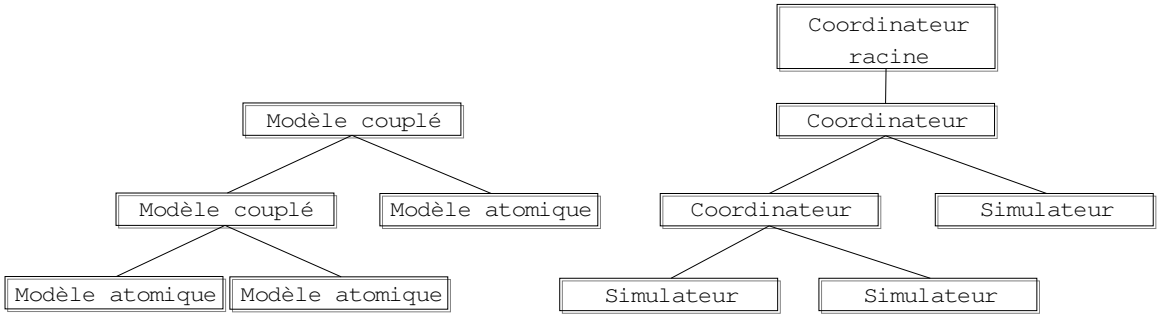


FIG. 3.5 – Traduction d'un modèle DEVS couplé dans sa hiérarchie de simulateurs abstraits. À gauche figure la hiérarchie des modèles atomiques DEVS et à droite la hiérarchie des simulateurs et coordinateurs associés. Chaque coordinateur correspond au simulateur abstrait d'un modèle DEVS couplé. L'algorithme 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 refléter 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.

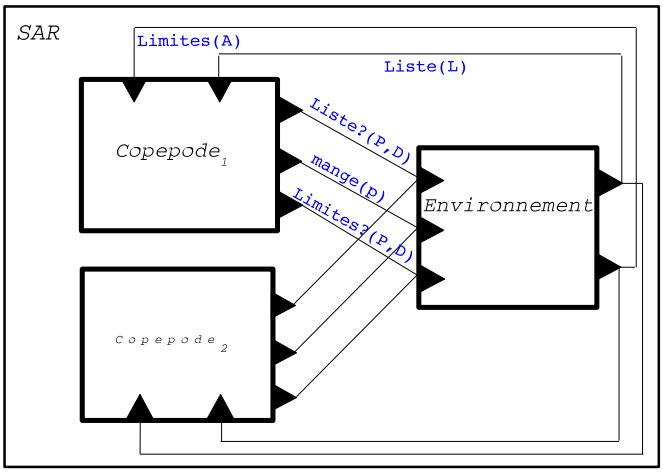


FIG. 3.6 – Représentation de la structure des connexions internes du système d'agent réactif en considérant deux agents copépodes. Les noms sur les connexions correspondent au nom des évènements.

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.

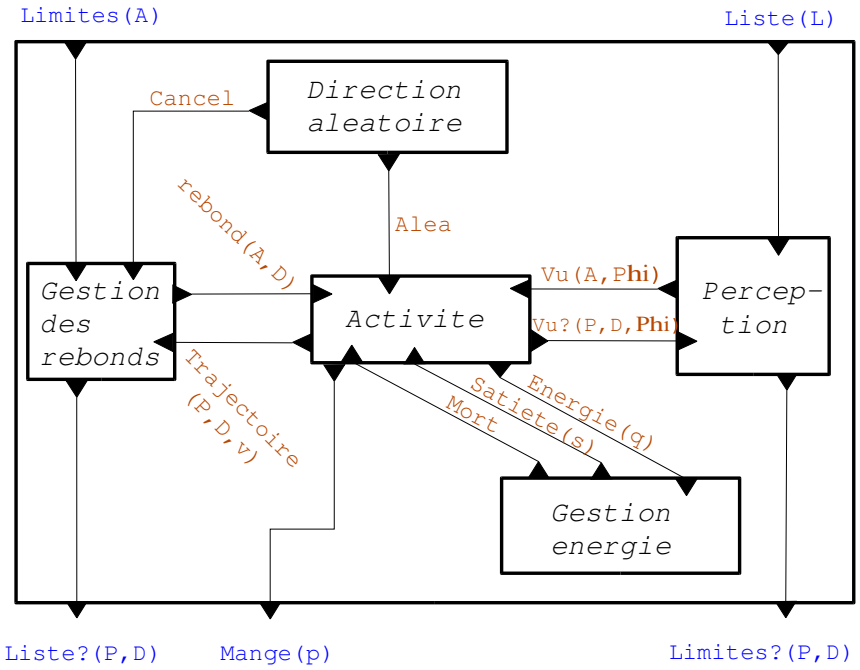


FIG. 3.7 – Représentation graphique du modèle couplé de l'agent copépode. La formalisation est donnée dans le texte en ce qui concerne le modèle « activité » et en annexe E.1 pour les autres. En bleu figure les évènements externes attachés aux ports d'entrée ou de sortie du modèle couplé. Les évènements 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

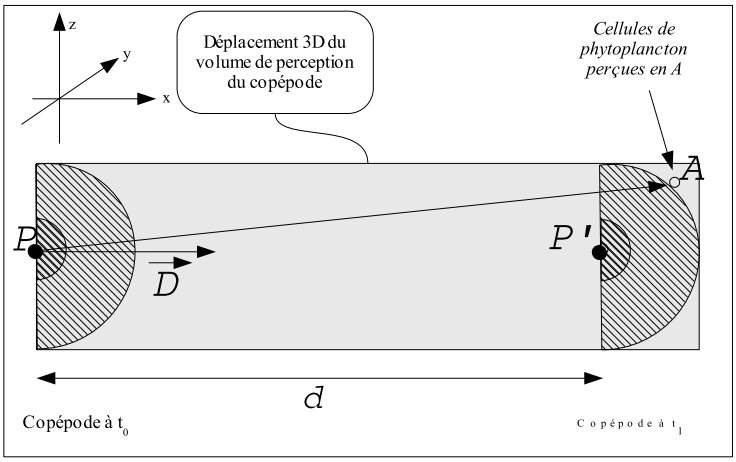


FIG. 3.8 – Représentation simplifiée du calcul de la nouvelle position P' du copépode au temps t_1 à partir de sa position initiale P en fonction de la position A de la cellule cible et du vecteur directeur $\vec{D}$ du copépode. Ici, le copépode est supposé être en phase de recherche. La distance d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

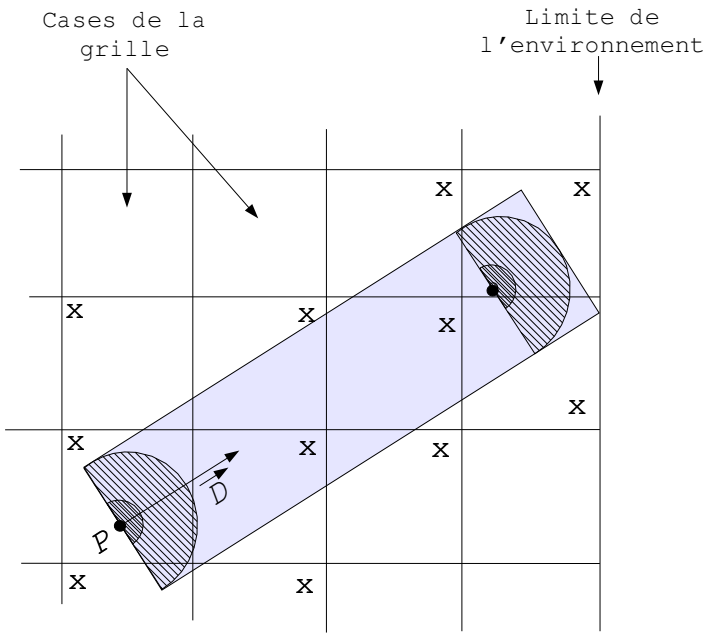


FIG. 3.9 – Représentation 2D de la discrétisation de l'espace pour le stockage des coordonnées des cellules de phytoplancton. Chaque case en contient un certain nombre. Toutes les cases marquées d'un « x » contiennent les celulles susceptibles d'être vues par le copépode avant qu'il n'atteigne une limite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

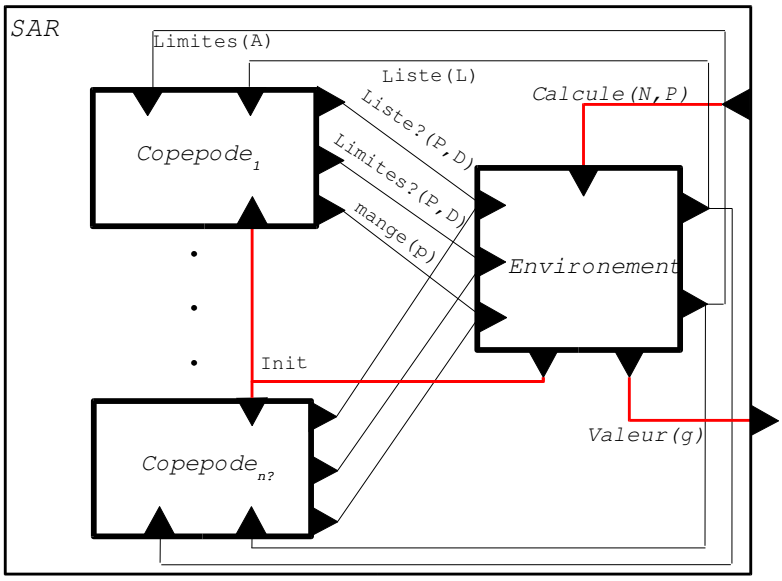


FIG. 3.10 – Représentation de la structure des connexions du model RAS. Les liens en rouge représentent les nouvelles connexions apportées pour le couplage avec le modèle NP. Les trois points indiquent que le nombre de modèles *Copepodes* est maintenant quelconque.

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.

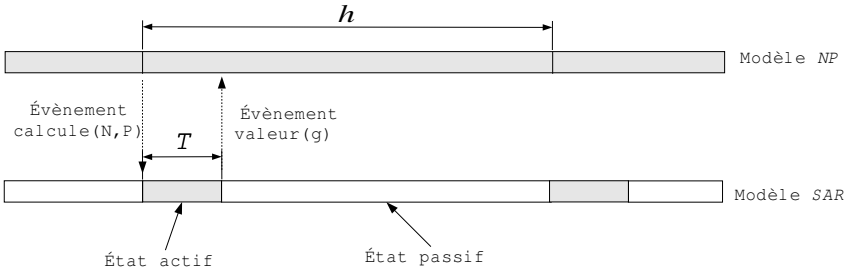


FIG. 3.11 – Représentation de l'échange d'évènements entre le modèle NP et le modèle SAR. Le temps est croissant de la gauche vers la droite. Le modèle NP demande au modèle SAR de simuler une valeur de g à chaque début de pas d'intégration. Le modèle SAR évalue g durant $T << h$, h étant le pas d'intégration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

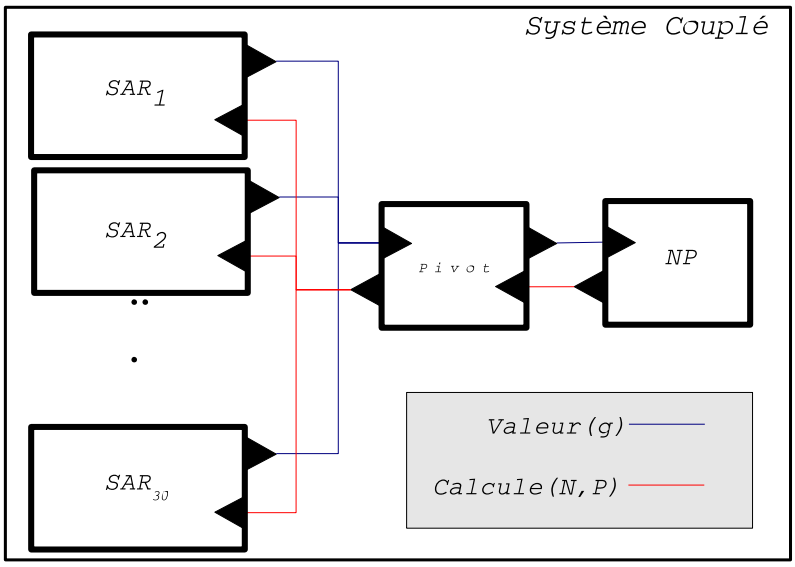


FIG. 3.12 – Structure du modèle de simulation du sytème couplé. Le modèle pivot joue le rôle d'intégrateur des résultats de simulations des SARs. Nous considérons 30 modèles SARs pour le calcul d'une valeur moyenne de g dans notre application (voir partie 5).

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.

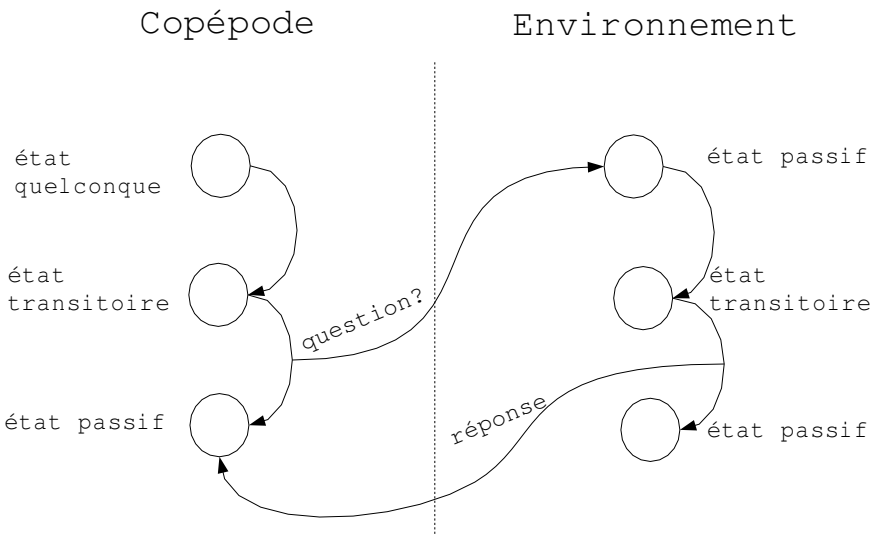


FIG. 3.13 – Interaction de type question-réponse dans la spécification du modèle d'agent réactif. L'implémentation ne considère pas les états transitoires et passifs du copépode, ni l'état transitoire de l'environnement mais relève d'un envoi de message (au sens de le programmation objet), donc d'un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

4

Intégration opérationnelle

Résumé

Nous avons noté en introduction que l'intégration de modèles hétérogènes implique l'interopérabilité des modèles. Cette dernière existe de fait pour les modèles issus d'un même formalisme et pour lesquels des simulateurs compatibles existent. Les choses deviennent plus compliquées lorsqu'il s'agit d'intégrer des formalismes ou des paradigmes hétérogènes.
Dans cette partie, nous proposons un cadre logique et logiciel, un *framework*, qui prend en compte les avancées de ces dernières années en terme de couplage de modèles hétérogènes. Notre *framework* s'articule autour de quatre couches qui définissent différents niveaux pour appréhender l'hétérogénéité. Nous proposons des solutions d'intégration pour chaque couche. De plus, nous proposons de mieux définir le concept d'expériences virtuelles pour la simulation des systèmes complexes. Dans ce cadre, nous proposons notamment deux applications XML pour le couplage de modèles et la définition de plans d'expériences.

Sommaire

4.1 Introduction

Nous reprenons ici le problème de l'intégration de modèles hétérogènes au niveau opérationnel. Nous commençons par énumérer les différentes questions que nous nous sommes posées. Ces questions présentent le cadre général de ce chapitre.

- comment coupler deux modèles issus de paradigmes différents ? Comment en gérer la cohérence ?
- dispose-t-on d'un formalisme opérationnel ?
- comment faire communiquer des modèles de simulation conçus séparément ?
- peut-on s'affranchir des problèmes techniques liés à la simulation (donc à l'exécution de modèles) ?
- comment réutiliser et intégrer des modèles déjà existants ?

Comme nous l'avons dit en introduction de cette thèse, l'intégration de modèles hétérogènes est en fait un problème d'interopérabilité aux différents niveaux d'abstraction décrits par la figure 1.2 page 15. Nous donnons notre propre cadre conceptuel et opérationnel (que nous appelons *framework*) à l'intérieur duquel nous faisons différentes propositions pour résoudre les problèmes d'interopérabilité.

En génie logiciel, plusieurs types de *frameworks* existent : les *frameworks* projet (il faut cadrer le projet dans ses grandes lignes et se référer à des projets-types), les *frameworks* de conception et les *frameworks* de développement. Les *frameworks* de conception et de développement sont plus proches de ce que nous cherchons à mettre en place dans le domaine de la modélisation et de la simulation de systèmes complexes. Un *framework* de conception est une sorte de livre de recettes. La première chose à mettre en place dans un *framework* de conception est un vocabulaire commun, c'est-à-dire une ontologie servant à décrire les choses (en génie logiciel, on parlera de briques logicielles ou matérielles). Ce point est essentiel. Le deuxième aspect des *frameworks* de conception est regroupé sous le terme de *design patterns*. Face à des problèmes récurrents, des *design patterns* sont définis afin de répondre plus rapidement aux problèmes. Le *design pattern Model-View-Controler* (MVC) est un exemple. Un ensemble de données est encapsulé dans un *Model*, le *Controler* permet d'y accéder et la vue est une représentation du *Model*. Pour un même *Model*, plusieurs vues différentes peuvent être définies sans toucher à la partie *Model*. Ce *design pattern* est très utilisé (par exemple, dans la notion d'interface multi-documents - MDI). La troisième forme de *framework* (de développement) est également une source d'inspiration. En effet, ce dernier doit structurer le développement, et non pas simplement offrir des bibliothèques de classes ou de fonctions. Ainsi, un *framework* de développement doit guider le développeur et homogénéiser (en partie) la façon de coder les modèles. En nous inspirant des définitions de *frameworks* de conception et de développement, nous donnons une définition liée à notre approche.

> Un *framework* pour l'intégration de modèles hétérogènes consiste à définir un cadre logiciel et des règles minimales de construction de modèles afin de permettre l'interopérabilité des modèles.

Le cadre logiciel définit l'ensemble des applications et des techniques nécessaires pour la mise en œuvre pratique de l'intégration. Les règles minimales de construction de modèles définissent l'ensemble des règles à suivre pour qu'un modèle soit compatible avec le *framework*.

Nous allons commençer par présenter notre *framework* d'intégration et proposer une syntaxe pour la description des modèles. De plus, l'utilisation des modèles dans le cadre d'expériences

nous a amené à définir une autre syntaxe pour la description des expériences de simulation. Nous concluons par une discussion à propos de notre approche.

4.2 Un *framework* pour l'intégration de modèles hétérogènes

Nous entrons dans cette section avec le dilemme suivant : conserver la diversité des modèles et en permettre l'unification, c'est-à-dire le couplage. Nous abordons cette question sous un angle opérationnel. La création de modèles et de simulateurs est une activité complexe que nous n'abordons pas ici ; nous nous focalisons sur l'intégration de modèles préexistants.

Le couplage de modèles dans un objectif de simulations réparties et comunicantes nous a conduits à organiser notre *framework* selon quatre niveaux d'abstraction (voir figure 4.1) : le niveau opérationnel, le niveau simulation, le niveau modèle et le niveau sémantique.

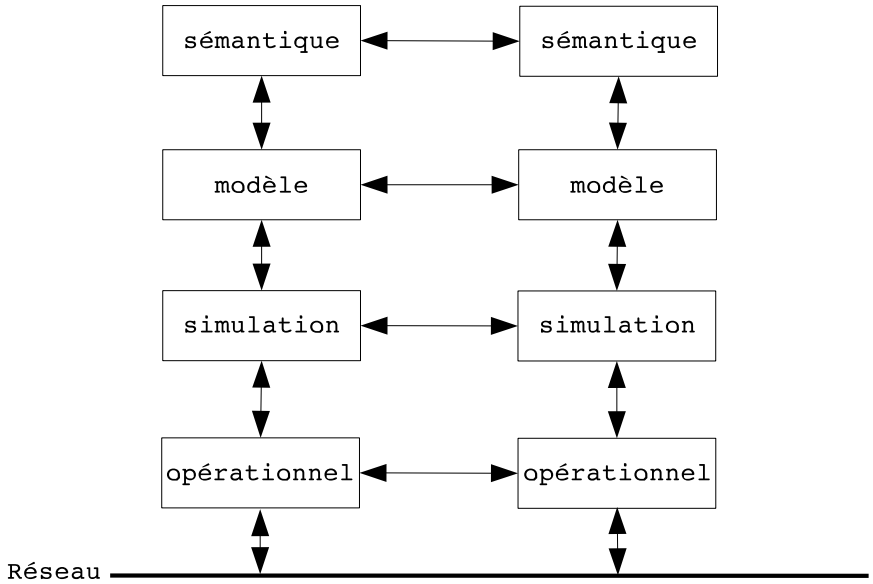


FIG. 4.1 – Hiérarchisation en quatre niveaux d'abstraction (ou couches) de notre *framework* pour l'intégration de modèles hétérogènes. Nous supposons ici deux modèles qui échangent des informations *via* le réseau. La communication entre couches (flèches horizontales) passe forcément par les niveaux inférieurs (flèches verticales).

Nous ne prétendons pas répondre totalement aux différents problèmes qui se posent pour chaque niveau (en particulier les deux derniers), mais allons néanmoins proposer un ensemble de solutions techniques et conceptuelles qui rendent opérationnel notre framework. Pour cela, le niveau opérationnel doit assurer la mise en œuvre de l'échange d'informations entre les éléments de la simulation dans un environnement réparti. Peu d'hypothèses doivent être faites sur la nature de cet environnement réparti. Nous nous sommes donc basés sur un environnement hétérogène au sens matériel, système d'exploitation, protocole et langage de programmation. Le niveau simulation a pour objectif de proposer une base algorithmique pour le couplage de simulateurs

sans spécificité propre à un type de simulateur. La couche modèle doit assurer la spécification des modèles quels que soient le formalisme et le paradigme. Une fois encore, cette couche doit autant que possible s'abstraire des spécificités propres aux formalismes et aux paradigmes afin d'offrir une notation unifiée des modèles. La dernière couche, la couche sémantique, doit permettre la spécification des éléments sémantiques d'un modèle (nature discrète ou continue de l'espace ou du temps manipulé dans un modèle par exemple).

L'idée retenue consiste à définir des bus de communication pour chaque couche. Un bus est un canal à travers lequel les niveaux identiques de notre *framework* communiquent (les flèches horizontales de la figure 4.1). La notion de communication est vue ici du point de vue général. Par exemple, pour la couche modèle, deux modèles communiquent dès lors qu'ils doivent échanger des données. Chaque bus est indépendant l'un de l'autre. Le choix d'un protocole de communication à un certain niveau ne doit pas imposer de contraintes aux niveaux inférieurs et supérieurs. Néanmoins, chaque niveau est en interaction avec ses niveaux voisins (flèches verticales de la figure 4.1). Les interactions entre niveaux peuvent être des traductions ou des encapsulations à l'image du modèle OSI. Conceptuellement, le *framework* se structure donc autour de quatre bus (un bus par couche) : le bus opérationnel, le bus de simulation, le bus des modèles et le bus sémantique. Nous allons maintenant passer en revue les différentes couches en exposant les solutions possibles pour la mise en œuvre de chaque bus.

4.2.1 La couche opérationnelle

Nous l'avons dit en introduction de cette thèse, cette couche est très étudiée et très prolifique en matière de solutions adoptées pour régler le problème de l'hétérogénéité. Les problèmes à ce niveau sont principalement techniques. Nous nous contentons donc ici de présenter les différentes possibilités qui s'inscrivent dans notre *framework*.
Notre couche opérationnelle se présente sous la forme d'un bus où viennent se connecter les éléments de la couche supérieure c'est-à-dire les simulateurs. Le bus opérationnel doit offrir les services liés à la communication, à la distribution des simulateurs indépendamment des protocoles et des langages. Nous pouvons schématiser la couche opérationnelle par la figure 4.2.

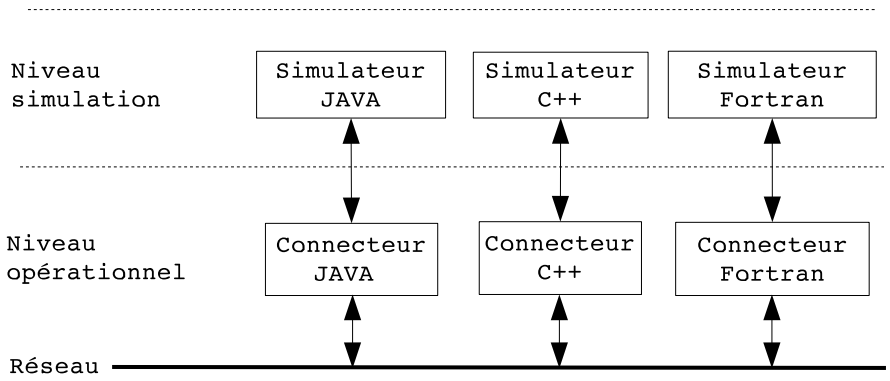


FIG. 4.2 – Couche opérationnelle de notre *framework*. Elle se situe entre le réseau (qui symbolise la couche matérielle) et le niveau simulation.

L'infrastructure repose principalement sur la notion de connecteur. Un connecteur est un

composant logiciel encapsulant l'accès au réseau, le transport d'informations *via* le réseau et l'indépendance par rapport au langage de programmation. Les connecteurs font le lien entre les simulateurs et l'infrastructure physique de communication. Comme nous l'avons vu en introduction de cette thèse avec HLA, les connecteurs doivent proposer une interface universelle (le RTI). Il existe plusieurs technologies possibles pour l'implémentation de ces connecteurs (voir figure 4.3).

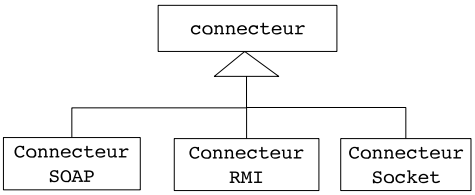


FIG. 4.3 – Exemples de connecteurs possibles pour la couche opérationnelle. Les connecteurs sont liés au langage utilisé pour la couche simulation.

Ci-dessous, nous présentons brièvement quatre cas de figure ayant été étudiés dans le cadre de projets de fin d'années de DESS et du stage de DEA de G. Quesnel [Que03] [QDNR03] :

- tous les simulateurs sont implémentés par des langages différents qui proposent une API[60] SOAP[61],
- certains simulateurs ne disposent pas de l'API SOAP mais proposent les sockets,
- tous les simulateurs sont programmés dans le même langage,
- tous les simulateurs sont en C/C++ et sont exécutés sur un cluster Linux.

Dans tous les cas, le connecteur a pour rôle de transformer les invocations de méthodes et/ou les objets SOAP en invocations et objets (ou structures) correspondants dans les langages utilisés par les simulateurs. L'idée est la même dans le deuxième cas de figure avec les sockets. Ce connecteur est mis à disposition dès lors que le langage utilisé par le simulateur ne dispose pas de l'API SOAP. De plus, SOAP n'est pas obligatoirement la technologie à utiliser. En effet, si la fréquence des échanges entre les simulateurs devient importante, le temps passé à la communication devient prohibitif par rapport au temps de simulation. Il est alors intéressant d'opter pour des moyens de communication plus rapides mais plus difficiles à mettre en place. Avec SOAP, les objets peuvent être transmis *via* le média de communication (après une phase de traduction en XML assurée par SOAP) ce qui n'est pas le cas avec les sockets : il faut que le programmeur prenne en charge les phases de traduction. Le troisième cas de figure est le plus simple. Si les simulateurs sont homogènes en terme de langage et en particulier en Java, l'échange de données et l'invocation de méthodes distantes peuvent être pris en charge par une implémentation Java de la technologie RPC[62] et en particulier RMI[63]. Nous avons pris pour exemple Java mais la remarque reste vraie pour les autres langages (ONC/RPC[64][65] pour C++, par exemple).

Les trois premiers cas de figures utilisent des technologies orientées vers les réseaux de type

[60] Application Programming Interface
[61] Il existe une alternative à SOAP avec XML/RPC disponible en Java, C++...
[62] Remote Procedure Call
[63] Remote Method Invocation
[64] Il existe aussi une version pour Java
[65] http ://www.plt.rwth-aachen.de/ks/english/oncrpc.html

LAN[66] et WAN[67] connectant un ensemble de machines « classiques » sous Linux, Windows, MacOS X, ... Le dernier cas de figure est particulier en spécialisant les technologies vers un type d'architecture précis. Celle-ci consiste en un ensemble de processeurs munis de mémoire non partagée. Les processeurs sont aussi accompagnés d'une interface réseau afin d'être connectés entre eux. Le type de réseau est LAN avec des débits plus élevés que dans un LAN « classique ». Les débits fluctuent entre la centaine de MegaBits et le GigaBit. Le système d'exploitation qui est embarqué est spécifique (très souvent une distribution de Linux légèrement adaptée) et il est augmenté de bibliothèques de communications spécifiques ; dans notre cas il s'agit de MPI[68]. Cette bibliothèque est très proche des sockets et permet de mettre en œuvre les algorithmes nécessaires à la communication entre simulateurs. Un type de connecteurs doit donc être disponible pour ce type d'architectures. Il faut noter que le langage de prédilection est le langage C ce qui n'exclut pas les autres langages mais interdit leur couplage avec MPI [69].

En conclusion, la couche opérationnelle permet à la couche simulation d'être indépendante de l'infrastructure matérielle et logicielle pour la communication entre processus. Elle n'a pas pour rôle de garantir la bonne synchronisation des processus et ne connaît rien des simulateurs. Ceci est confié à la couche simulation.

4.2.2 La couche simulation

La couche simulation a pour objectif d'assurer l'exécution correcte[70] des modèles couplés. L'algorithmique des simulateurs est, dans la majorité des cas, spécifique au formalisme adopté dans la phase de modélisation. Ce constat peut s'avérer être un problème. En effet, il faut alors répondre à la question :

« comment coupler deux simulateurs dont l'algorithmique est très différente ? »

Considérons, par exemple, une partie de système modélisée à l'aide d'un système d'équations différentielles et une autre partie par un réseau de Petri. Nous choisissons pour les équations différentielles une simulation programmée à l'aide d'un algorithme d'intégration numérique de type Runge-Kutta. Pour le réseau de Petri, nous utilisons tout simplement l'algorithme d'évolution du marquage synchrone[71]. Il paraît assez évident que le couplage de deux simulateurs aussi différents n'est pas simple.
Pour résoudre ce type de problèmes de couplage de simulateurs, nous nous sommes orientés vers les travaux de Zeigler [Zei76] [ZKP00] et le formalisme DEVS. Nous avons déjà présenté DEVS en introduction et dans le chapitre précédent. Rappelons simplement ici que DEVS est un formalisme abstrait pour la modélisation à évènements discrets et que ce formalisme a la prétention d'offrir à la fois l'encapsulation d'autres formalismes mais aussi les simulateurs associés. Nous donnons les algorithmes de base des simulateurs abstraits DEVS en annexe D[72]. D'autres algo-

[66] Local Area Network
[67] Wide Area Network
[68] Message Passing Interface
[69] Il existe quelques implémentations Java
[70] Le lecteur intéressé peut se reporter au livre de Zeigler *et al.* [ZKP00] pour une introduction au concept de morphisme qui permet de mettre en relation les modèles formels et leurs simulateurs, et ainsi de vérifier si une simulation est « correcte ».
[71] Toutes les transitions franchissables sont validées en même temps. Si des conflits existent un tirage aléatoire est effectué.
[72] Nous conseillons au lecteur n'ayant pas lu le chapitre précédent de se référer à la partie 3.2 page 50 pour une présentation générale de DEVS.

rithmes peuvent être trouvés dans [ZKP00] notamment pour les simulations distribuées. DEVS contient de façon intrinsèque les notions de hiérarchisation et de couplage. Ainsi, les algorithmes des simulateurs abstraits permettent la simulation de tout modèle DEVS ou compatible avec DEVS (nous approfondissons cette notion un peu plus loin). Le formalisme DEVS nous offre non seulement un cadre formel de spécification de modèles mais aussi des mécanismes opérationnels de simulation. C'est ce point qui nous a fait choisir DEVS pour la couche simulation.

Si nous voulons utiliser les simulateurs abstraits pour la couche simulation, quelles sont les contraintes pour un concepteur de simulateurs ? Les algorithmes des simulateurs abstraits reposent pleinement sur une spécification DEVS, ce qui implique *a priori* qu'il soit nécessaire de réaliser une spécification complète du modèle en DEVS. Deux approches sont en fait possibles : le *mapping* DEVS ou le *wrapping* DEVS.
Le *mapping* consiste à spécifier totalement le modèle en DEVS et ce quel que soit le formalisme utilisé pour la couche modèle. C'est ce que nous avons fait dans le chapitre précédent en spécifiant un modèle d'agents réactifs situés totalement en DEVS. Les travaux de Jacques et Wainer sont un autre exemple de *mapping* [JW02]. Ces travaux s'intéressent à la modélisation à base de réseaux de Petri et Jacques et Wainer proposent une spécification DEVS du formalisme. Il est alors possible de transformer les modèles à base de réseaux de Petri en modèles DEVS. Cette approche permet de garantir une solution DEVS pour la couche simulation et pour la couche modèle.
L'autre approche (le *wrapping*) est un compromis entre la volonté de disposer d'un noyau unifié de simulation qui couple des simulateurs éventuellement non DEVS. Cette idée est initialement apparue au cours de nos réunions de travail avec les membres du LISC[73] de Clermont-Ferrand. Le simulateur doit mettre en œuvre une liste de fonctions qui entrent dans la logique algorithmique des simulateurs abstraits. Le *wrapper* est donc une sorte d'interface au simulateur adapté au formalisme utilisé (voir figure 4.4). Le travail du concepteur de simulateurs « compatible DEVS » se résume alors à la construction de ce *wrapper*[74] ou à l'utilisation du *wrapper* existant pour le formalisme qu'il a choisi.

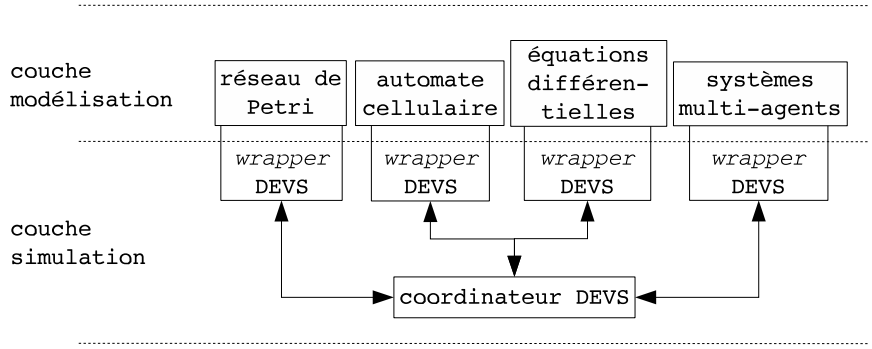


FIG. 4.4 – Schématisation des *wrappers* DEVS. Ce sont des interfaces fonctionnelles basées sur l'algorithmique des simulateur abstraits (voir le texte pour les détails).

L'étude des algorithmes des simulateurs abstraits permet d'identifier six fonctions utiles (voir figure 4.5). L'idée repose sur le fait que DEVS n'impose pas de formalisme pour les fonctions de

[73]Laboratoire d'Igénieurie des Systèmes Complexes.
[74]Nous parlons aussi d'interface DEVS

transition ou de sortie. Ainsi, nous pouvons « cacher » derrière ces fonctions un simulateur écrit pour un formalisme quelconque.

```
public abstract EventList getOutputFunction(Time p_currentTime);
public abstract Time      getTimeAdvance();
public abstract void      init();
public abstract void      processInternalEvent(InternalEvent p_event);
public abstract void      processExternalEvent(ExternalEvent p_event);
public abstract void      finish();
```

FIG. 4.5 – Définition des fonctions du *wrapper* en Java. Cette interface est définie par six fonctions issues des simulateurs abstraits de DEVS. L'ensemble des états S n'apparaît pas ici ; il fait partie des attributs du modèle (au sens de la programmation objet).

Rappelons que la spécification DEVS est une structure composée par un vecteur d'états S, un ensemble de ports d'entrée X et de ports de sortie Y, de deux fonctions de transitions (l'une interne δ_{int} et l'autre externe δ_{ext}), d'une fonction de sortie $\lambda(S)$ et d'une fonction d'avancement du temps $ta(S)$. Nous retrouvons dans l'interface DEVS (le *wrapper*) les deux fonctions de transitions sous forme de fonctions de traitement des évènements externes (*processExternalEvent*) et de fin d'état (*processInternalEvent*). Ces deux fonctions respectent la spécification DEVS telle que : $\delta_{int} : S \rightarrow S$ et $\delta_{ext} : S \times X \rightarrow S$. Le paramètre de la fonction *processInternalEvent* ne fait pas partie de la spécification DEVS mais permet dans notre cas de transporter certaines informations telles que la date d'occurrence de l'évènement, par exemple.

Le traitement de l'évènement d'initialisation est représenté par une simple fonction (*init*). La fonction de sortie se traduit par une fonction admettant en paramètre la date courante afin d'estampiller les évènements qui seront générés et retourne une liste d'évènements. La notion d'évènement est mise en œuvre par la classe *Event*. Deux sous-classes sont disponibles *InternalEvent* et *ExternalEvent*. Seule la classe *ExternalEvent* est instantiable par le concepteur d'un *wrapper*. Les instances de la classe *InternalEvent* sont créées par les simulateurs et représentent l'évènement de fin d'état. L'opération de construction d'un évènement externe par la fonction de sortie est très simple : cette fonction connaît seulement le port de sortie sur lequel l'évènement doit être émis et la date d'occurence de cet évènement.

La dernière fonction à implémenter est la fonction d'avancement du temps. Elle est invoquée par le simulateur abstrait pour déterminer la date de fin de l'état courant. L'en-tête de la fonction fait donc apparaître tout logiquement une date (*Time*) en retour d'appel de cette fonction. Si ces six fonctions sont implémentées, alors nous disposons d'un *wrapper* DEVS. Celui-ci peut être intégré dans une simulation distribuée et le modèle qu'il met en œuvre peut être couplé de manière totalement transparente.

Comme le montre la figure 4.4, un simulateur s'intègre à la simulation globale en se connectant sur un bus logiciel défini par l'interface DEVS. L'architecture du bus repose sur la relation entre un coordinateur et des simulateurs. Telle que nous la définissons ici, la couche simulation est une couche logicielle qui a pour principal objectif d'exécuter les modèles. Elle doit donc intégrer les algorithmes nécessaires pour les bases de la simulation distribuée et le contrôle du cycle d'exécution de la simulation. Ces algorithmes ont été développés notamment par Kim [KK96]

qui introduit la notion de DEVS-bus. Il a poursuivi ses travaux par une intégration de DEVS-bus dans HLA [KK98]. La figure 4.6 schématise l'architecture que nous avons choisie pour notre couche simulation. Cette figure reprend celle présentée précédemment pour la couche simulation et celle présentée pour la couche opérationnelle. De plus, nous nous sommes fortement inspirés de l'architecture proposée par Kim et Zeigler [ZKP00].

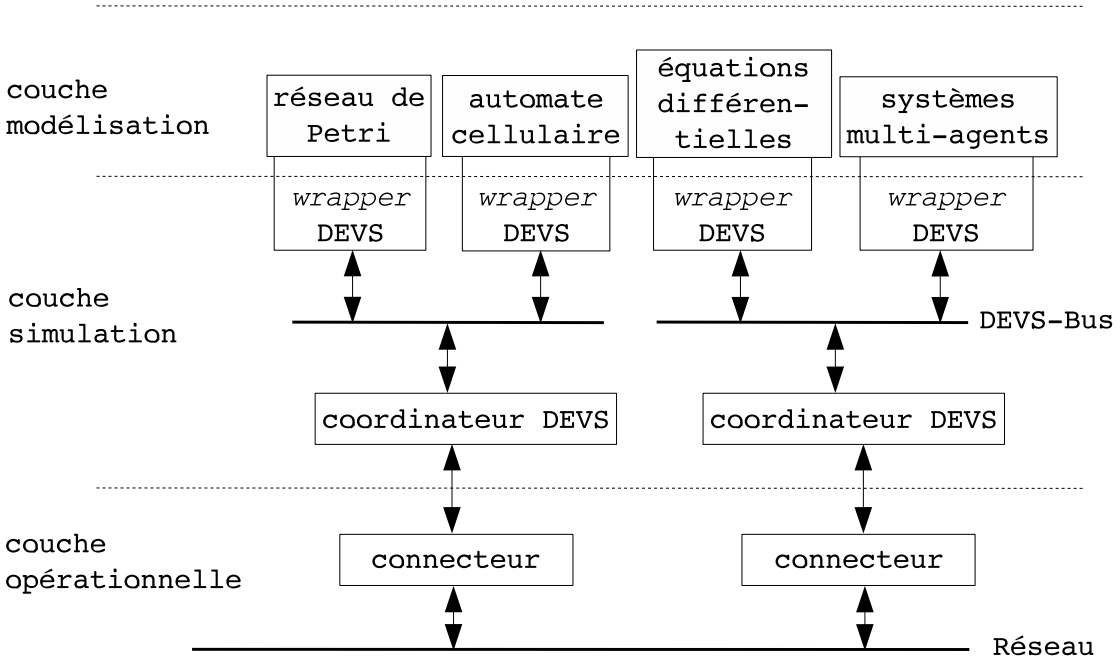


FIG. 4.6 – Intégration de DEVS-bus dans notre *framework*. Les coordinateurs DEVS peuvent être exécutés sur une même machine ou être distants. Dans ce cas, ils échangent les données *via* les connecteurs au réseau.

Les choix conceptuels qui ont été opérés pour la couche simulation permettent au concepteur de simulateurs de s'affranchir des mécanismes liés au couplage. En contrepartie, il doit développer une interface minimale de communication externe qui s'exprime en terme d'évènements. Si son modèle est implémenté en terme d'évènements discrets ou de temps discret, alors l'interfaçage ne pose pas de problème majeur. Par contre, si le temps n'est pas explicite dans son modèle, des choix doivent être faits en ce qui concerne le traitement des évènements en entrée ou en sortie qui eux, doivent être obligatoirement datés. Ce type de questions concerne plus particulièrement la couche modèle que nous allons maintenant décrire.

4.2.3 La couche modèle

Que signifie s'intéresser seulement au couplage au niveau modèle ? Comme nous l'avons dit en introduction de cette thèse, cette problématique est identifiée, notamment par Fishwick, sous le terme de modélisation multiple (*multi-modeling*) [Fis95]. La modélisation multiple considère qu'un modèle peut être composé de plusieurs autres modèles sous la forme d'un graphe où chaque nœud représente un modèle et les arcs les connexions entre ces modèles. Le couplage de modèles est alors vu comme la connexion de sorties à des entrées de modèles. La modélisation multiple

peut être vue comme l'agrégation de modèles. Fishwick parle également de raffinement lorsqu'un élément du comportement d'un modèle est exprimé plus précisément à l'aide d'un autre modèle. Cette opération définit *de facto* un nouveau niveau d'abstraction. La notion de raffinement intervient au niveau de la spécification de la dynamique. Nous pouvons par exemple définir un système d'équations différentielles pour expliquer les conditions de changements d'états d'un autre modèle (ce que nous avons fait dans le chapitre précédent avec le contrôle de l'état de satiété du copépode). Le système d'équations est alors un raffinement d'un état du modèle.
Cette approche du couplage de modèles est intéressante au niveau conceptuel mais ne répond pas à la question du couplage au niveau formel ou opérationnel. Le modélisateur est contraint de repenser ses simulateurs en fonction du multi-modèle construit.

Comme les autres couches, la couche modèle doit apporter un *medium* de communication entre les éléments de la même couche. Il est donc nécessaire de définir un bus modèle, où les formalismes et paradigmes utilisés peuvent être couplés. Il est possible d'adopter deux approches à ce niveau :

- un couplage à la fois formel et opérationel lorsque c'est possible,
- offrir un mode de représentation opérationnel des modèles pour permettre leur couplage.

Dans les deux cas, nous devons disposer d'un système d'écriture des modèles et de leur couplage.

Le premier cas correspond à la formalisation du couplage des modèles. Le concept de DEVS-bus est, là encore, utile. Il est possible de le considérer au niveau de la couche modèle. En effet, le choix de DEVS au niveau simulation est essentiellement lié au fait que DEVS est un formalisme indépendant des simulateurs qui offre également une sémantique opérationnelle. Ainsi, le lien entre la couche simulation et la couche modèle est établi de fait. Il faut pour cela formaliser le modèle en DEVS.

Comme nous l'avons dit dans le chapitre précédent, de nombreux auteurs proposent des spécifications DEVS de formalismes qui mènent parfois à des extensions de DEVS. Vangheluwe en fait état dans [Van00] et développe son approche $ATomM^3$ [LV02] de méta-modélisation et de couplage de différents formalismes[75]. Citons ici quelques exemples de spécifications DEVS de formalismes ou paradigmes connus (nous en avons déjà cité certains dans le chapitre précédent) :

- DEV&DESS [ZKP00] [BV02] pour une spécification des équations différentielles,
- Quantized Systems [Kof02], une méthode pour la résolution d'équations différentielles basée sur DEVS,
- G-DEVS pour une généralisation aux systèmes continus [GEG00],
- Cell-DEVS [WG01]) et DEVS et les automates cellulaires [VV00],
- DS-DEVS [Bar96],
- ... mais aussi des propositions concernant les state-charts [BV03], ...

Contrairement à la couche simulation, pour unifier conceptuellement les modèles, il est nécessaire de proposer les spécifications DEVS citées précédemment. Nous ne pouvons pas nous contenter d'une simple encapsulation du modèle par un *wrapper*. Le modélisateur doit faire l'effort de spécifier son modèle en DEVS soit en se référant à des travaux existants (si le formalisme a fait l'objet d'une spécification DEVS) soit en proposant sa propre spécification DEVS.

La deuxième approche, que nous pouvons adopter dans la couche modèle, est celle d'une

[75] http ://atom3.cs.mcgill.ca

description non formelle mais opérationnelle des modèles. En effet, certain modèles peuvent ne pas avoir d'équivalent en DEVS[76]. Si nous voulons communiquer ce type de modèles, nous devons mettre au point un vocabulaire commun pour leur description et une syntaxe pour leur représentation.

Ces réflexions, menées dans le cadre général du couplage de modèles hétérogènes, nous amènent à proposer le langage XML (eXtensible Markup Language) [HM01] pour la représentation des modèles de simulation des systèmes complexes. XML permet de définir une syntaxe de langage particulière à l'aide de balises du type de celles qui existent en HTML. La définition d'une telle syntaxe s'appelle une application XML. Dans ce qui suit, nous définissons une telle application appelée MLMC (Markup Language for Model Coupling). Un des grands avantages d'XML est sa flexibilité et son extensibilité, ce qui permet au format de fichier d'être étendu *ad infinitum*, tout en maintenant une compatibilité ascendante entre versions successives des formats de fichiers. De plus, un nombre croissant d'applications utilisent XML pour la sauvegarde des données, ce qui entraîne le développement rapide d'outils permettant le parcours des fichiers XML (on parle de *parser* XML). Ces *parsers* sont intégrés sous forme d'API (comme la « libxml » du langage C).
MLMC définit l'ensemble des modèles couplés et les connexions entre modèles. Il possède également des éléments de sémantique. Nous le décrivons donc après avoir présenté la dernière couche de notre *framework*.

4.2.4 La couche sémantique

Nous avons commencé à nous intéresser à la couche sémantique dans un but de couplage contrôlé des simulateurs. Plus précisément, nous voulons disposer d'une méthode et d'un outil qui rendent compte de la compatibilité des données échangées par les modèles au niveau du sens de ces données. Si un modèle calcule une vitesse, il est évident qu'il ne doit pas fournir ces données à un modèle qui a besoin d'une température en entrée. L'exemple paraît trivial mais il s'avère que l'automatisation d'un tel contrôle de cohérence au niveau sémantique est en réalité très complexe.
C'est dans le domaine des bases de données que l'on peut trouver les avancées les plus significatives en ce qui concerne l'intégration de données hétérogènes au niveau du sens [Jou01]. Cette discipline nous apprend que la fusion de deux ontologies (un vocabulaire et ses définitions) différentes est difficile. Néanmoins des outils existent, avec par exemple la mise au point de calcul de distance sémantique permettant de dire si deux termes sont proches ou différents [Jou01]. Il existe également des avancées dans le domaine de l'intégration d'applications hétérogènes qui peuvent servir de base à des outils d'intégration sémantique de modèles [XSDJ03].

Cette question nous a animés et a donné lieu à une publication qui propose XML pour la représentation de la sémantique des données pour le couplage de modèle [Dub02]. Néanmoins, nous pensons que cette question est trop vaste pour être traitée dans le cadre de cette thèse. Nous allons toutefois proposer dans l'application MLMC des éléments de sémantique et donc le support opérationnel pour relier données et sens des données.

De plus, la dernière couche de notre *framework* doit rendre compte de certains des choix du modélisateur pour la représentation des données. Dans l'état actuel de la réflexion, nous consi-

[76]Nous pensons ici à des SMAs très spécialisés ou même des plateformes de simulation existantes...

dérons quatre aspects de modélisation qui nous semblent les plus importants pour les systèmes dynamiques :

- la représentation du temps,
- la représentation de l'espace,
- la catégorisation des variables en terme d'unités,
- la classe d'appartenance des entités du système et le lien avec les variables du système.

Ce travail s'inscrit en partie dans un ensemble de réflexions menées au sein du groupe MIMOSA (Méthodes Informatiques pour la MOdélisation et la Simulation à base d'Agents[77]). Les membres du groupe MIMOSA réfléchissent sur les concepts manipulés en modélisation comme le temps, l'espace ou les entités d'un modèle. Ces réflexions permettent notamment d'adopter « un langage » commun pour la description des modèles. Ce « langage » peut être décrit par XML, ce qui rend opérationnel la comparaison, l'échange ou le couplage de modèles différents.

4.3 Description des modèles

Des travaux récents se sont orientés vers l'utilisation d'XML pour la représentation des modèles de simulation. Par exemple, F. Villa [Vil01] propose une syntaxe pour la description et l'échange des modèles. Seulement cette syntaxe s'applique principalement aux modèles formalisés avec des équations différentielles. P.A. Fishwick [Fis02] introduit MXL comme un langage basé sur XML pour la représentation des modèles. Néanmoins, ce langage reste pour l'instant limité et peu adapté à nos besoins de couplage. Les travaux les plus proches de nos préoccupations sont certainement ceux de l'équipe allemande du projet Man Model Measurement (M3) [Hoh02]. Ce projet a pour but de créer un monde virtuel dans lequel sont modélisées les interactions complexes entre l'homme et son environnement. M3 est un logiciel d'intégration ouvert dans le sens où n'importe quel modèle peut y être intégré du moment qu'il implémente les protocoles et interfaces définis pour les composants M3. Les concepteurs du projet ont décidé d'utiliser XML pour la représentation des données échangées par les différents modèles qui composent la plateforme. Néanmoins, les données et les entités réelles qu'elles représentent sont confondues dans la même application XML, ce qui oblige tous les modèles à « parler la même langue ». Nous avons préféré séparer les données des entités qu'elles représentent de manière à offrir plus de souplesse dans le couplage de modèles hétérogènes au niveau sémantique.
Pour la construction de la syntaxe XML, nous nous sommes encore une fois basés sur les travaux de Zeigler *et.al.* [ZKP00]. Nous reprenons ici les principales définitions apportées par Zeigler, comme les notions de ports d'entrées et ports de sorties auxquels sont attachées les données échangées entre modèles couplés. Nous introduisons d'autres types de ports, à savoir les ports d'états et d'initialisation. Les premiers permettent d'identifier les variables sur lesquelles le modèle permet d'effectuer des mesures. Les deuxièmes permettent d'initialiser certains paramètres comme les constantes du modèle. Zeigler définit également plusieurs niveaux hiérarchiques qui correspondent aux niveaux de connaissances que l'on possède d'un système. Dans notre approche, nous définissons un niveau hiérarchique minimal en dessous duquel nous n'avons pas connaissance du fonctionnement interne du modèle. À ce niveau, nous ne connaissons que la nature des données en entrée et en sortie et les dates associées (c'est le niveau 1 défini par Zeigler, voir figure 1.1 page 17). Le couplage des modèles décrits au niveau 1 s'effectue par la description des connexions entre les ports de sorties et les ports d'entrées des différents modèles. Ainsi, nous

[77] Site Web MIMOSA : `http://www-lil.univ-littoral.fr/Mimosa/`

décrivons un modèle couplé comme un « graphe » reliant des modèles décrits au niveau 1 ce qui engendre un modèle couplé au niveau 4.

Comme nous l'avons dit dans la section précédente, dans une perspective plus large que celle de la description du couplage qui est celle de la validation de ce couplage à différents niveaux (opérationnel et sémantique), nous avons besoin de connaître les choix de représentation du temps et de l'espace, les unités des données ou encore ce qu'elles représentent. Nous offrons donc une syntaxe de description des données ainsi qu'une description de l'espace et du temps à laquelle peuvent se référer les données.

4.3.1 Description de l'espace

L'espace peut être de trois types :
- un ensemble de lieux sans aucune relation de position des uns par rapport aux autres ;
- un espace topologique, où l'ensemble des lieux sont mis en relation par des liens de voisinage ;
- un espace métrique défini par un référentiel, une unité de mesure dans cet espace, sa caractéristique discrète ou continue, et dans le cas discret, le pas de discrétisation.

L'espace peut être décrit par un ensemble de places (*i.e.* de lieux), de relations de voisinage ou un référentiel. Il est décrit par la syntaxe XML ci-dessous (nous expliquons plus en détail cette syntaxe juste après l'avoir présentée) :

```
<SPACE name="space_name" type="topological | metric | set">

  <PLACES>

    <PLACE name="place_name1"/>

    <PLACE name="* | []" begin="n" end="m"/>

  </PLACES>

  <NEIGHBOURHOOD>

    <NEIGHBOUR link="place_name1,place_name2"></NEIGHBOUR>

    <NEIGHBOUR type="von_neumman | moore | none">
                  0 1 0 0 1 1 1 ...
    </NEIGHBOUR>

  </NEIGHBOURHOOD>

  <REFERENTIAL type="discrete | continuous" dimension="n">

    <AXIS id="x" min="val_min" max="val_max" step="val_step"/>

  </REFERENTIAL>

  <DISTANCE type="euclidian"/>

</SPACE>
```

Il est possible de définir l'ensemble des places (contenues dans la balise <PLACES>) en extension. Dans ce cas, il y aura autant de balises <PLACE> que de places dans le modèle. La première balise <PLACE> définit ce cas. Si les places sont définies en compréhension, alors il faut utiliser la syntaxe définie par la deuxième balise <PLACE> où l'attribut name="*" désigne un ensemble quelconque de places numérotées de 0 à n (n étant inconnu). Si name="[]" alors les attributs begin et end sont renseignés avec n et m désignant un ensemble de places numérotées de n à m, où n et m sont des entiers positifs, n < m et le nombre de places est égal à m-n.

L'attribut link de la balise <NEIGHBOUR> définit les relations de voisinage entre places. Si nous définissons ces relations en extension, nous utilisons la syntaxe donnée par la première balise <NEIGHBOUR>. Dans ce cas, il y aura autant de balises <RELATION> que de relations de voisinage. Si nous voulons définir ces relations en compréhension, l'ensemble des places doit impérativement avoir été défini en compréhension et nous devons utiliser la syntaxe de la deuxième balise <NEIGHBOUR>.
Le type de voisinage peut être connu ou inconnu. Dans ce dernier cas, la valeur de l'attribut type est égale à none. Dans ce cas, la suite de 0 et de 1 à l'intérieur de la balise <NEIGHBOUR> est le contenu de la matrice d'adjacence définie par l'ensemble des palces. La valeur 1 indique des voisins et 0 des non-voisins. La valeur de m-n est le nombre de lignes et de colonnes de cette matrice.

L'attribut type de la balise <REFERENTIAL> nous informe sur la nature discrète ou continue de l'espace. La valeur n de l'attribut dimension de cette balise est le nombre de dimensions de l'espace.

Un référentiel est défini à l'aide d'axes : c'est le rôle de la balise <AXIS>. Il y a autant de balises <AXIS> que de dimensions de l'espace. Si l'espace est discret, alors l'attribut val_step est renseigné. Sinon, nous définissons les bornes inférieures et supérieures val_min et val_max dans l'intervalle $]-\infty, +\infty[$. La balise <DISTANCE> a un rôle sémantique et nous renseigne sur la nature des distances mesurées (euclidiennes, d'ordre topologiques, etc.).

C'est le type de l'espace qui définit les balises contenues dans la balise <SPACE>. Si le type est « set », alors seule la balise <PLACES> et ses filles seront obligatoirement présentes. Si le type est « topological » alors les balises <PLACE>, <NEIGHBOURHOOD> et leurs filles seront obligatoirement présentes. Enfin, si le type est « metric », alors seule la balise <REFERENTIAL> et ses filles seront obligatoirement présentes. La balise <DISTANCE> est optionnelle. La balise <SPACE> peut apparaître plusieurs fois dans la définition d'un modèle, ce qui permet de représenter un modèle manipulant plusieurs types d'espace.

4.3.2 Description du temps

Le temps peut être de trois types :
- un ensemble de lieux temporels sans aucune relation d'ordre les uns par rapport aux autres ;
- un temps ordinal, où l'ensemble des instants sont mis en relation par une ou plusieurs relations d'ordre ;
- un temps cardinal défini par une base de temps, une unité de mesure dans ce temps, sa caractéristique discrète ou continue, et dans le cas discret, le pas de discrétisation.

Il y a une similitude assez évidente entre le temps et l'espace. En effet, pour les deux représentations il s'agit toujours de disposer d'un repère (ou référentiel). Le temps du modèle est

décrit par la syntaxe XML ci-dessous. Comme précédemment, nous expliquons plus en détails cette syntaxe juste après l'avoir présentée :

```
<TIME type="set | ordinal | cardinal">

    <TIME_SPANS>

       <TIME_SPAN name="time_spam_name"/>

       <TIME_SPAN name="* | []" begin="n" end="m"/>

    </TIME_SPANS>

    <ORDER>

       <RELATION sequence="name1,name2..."/>

       <RELATION sequence="Ti"/>

    </ORDER>

    <TIME_BASE type="discrete | continuous" unit="time_unit"
               begin="n" end="m" step="time_step"/>

</TIME>
```

La balise `<TIME_SPANS>` représente l'ensemble des lieux temporels utilisés dans le modèle (chez le voisin, le village, etc.). Ces lieux peuvent être définis en extension (première balise `<TIME_SPAN>`) ou en compréhension (deuxième balise `<TIME_SPAN>`), de la même façon que pour les lieux géographiques définis précédemment. La définition en compréhension implique une relation d'ordre de `n` à `m`.
Dans le cas d'une définition en extension, la balise `<ORDER>` contient les relations d'ordre (définies par la balise `<RELATION>`) entre les différents lieux temporels. Plusieurs syntaxes sont possibles pour la balise `<RELATION>`. Tout d'abord la définition en extension, où l'attribut séquence ordonne l'ensemble des lieux temporels, du plus récent au plus ancien (cas de la première balise `<RELATION>`). Il est également possible de définir les relations en compréhension (dernière balise `<RELATION>`).

La représentation du temps nécessite également de décrire le « type » de base de temps (discret ou continu). Dans le premier cas, l'attribut `step` de la balise `<TIME_BASE>` est renseigné. Les valeurs `n` et `m` sont des réels strictement positifs qui désignent la date de début et la date de fin du temps simulé par le modèle. Si la base de temps est continue (modèle à évènements discrets), alors nous ne connaissons pas à l'avance les dates d'occurrences des évènements ; nous ne spécifions donc pas de valeur pour l'attribut `step`.

Comme pour la définition de l'espace, le contenu de la balise `<TIME>` est conditionné par la valeur de son attribut `type`. Si le type de temps est `set`, alors seule la balise `<TIME_SPAN>` est présente, ainsi que ses filles. Si le type de temps est `ordinal`, alors les balises `<TIME_SPAN>` et `<ORDER>` sont présentes, ainsi que leurs filles. Si le type de temps est `cardinal`, alors seule la balise `<TIME_BASE>` est présente.
Une remarque importante s'impose : une donnée peut avoir une certaine représentation du temps

dans le modèle et être émise en sortie avec une autre représentation. Nous considérons que la représentation du temps attachée aux données correspond à leur représentation en sortie ou en entrée du modèle.

4.3.3 Description d'un modèle

La balise `<MODEL>` contient les deux balises `<TIME>` et `<SPACE>` plus d'autres balises qui spécifient les ports auxquels sont attachées les données. Chaque port est susceptible d'émettre ou de recevoir des évènements selon sa nature (ports d'entrée ou de sortie). À ces évènements sont attachés des variables structurées définies par la balise `<DATA>` (voir section 4.3.4). C'est la référence de la balise `<DATA>` au type de temps représenté par le modèle qui nous renseigne sur la fréquence ou les dates de sortie des évènements. Si le modèle ne manipule pas de temps (au sens courant), il émet une succession de résultats ordonnés. Cet ordre peut également être représenté dans la balise `<TIME>`.
Nous adoptons les notions de modularité et de hiérarchie des modèles définies dans [ZKP00]. Ainsi, un modèle peut être de type atomique (ou isolé) ou de type couplé. Ceci permet la construction d'un « graphe » de modèles, avec des modèles qui peuvent en contenir d'autres. Nous définissons d'abord la syntaxe pour les modèle atomiques, puis pour les modèles couplés.

4.3.3.1 Les modèles atomiques

La figure 4.7 montre les différents types de ports attachés à un modèle.

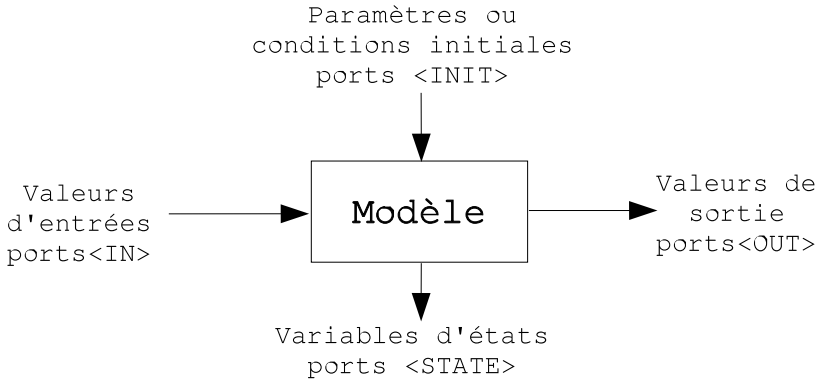


FIG. 4.7 – Représentation des différents types de ports attachés à un modèle. Par rapport à la définition des simulateurs abstraits DEVS (voir annexe D page 183), nous avons ajouté les ports d'états, qui permettent de définir les variables observables pour un modèle donné.

La syntaxe XML suivante définit la balise `<MODEL>` :

```
<MODEL name="model_name" type="atomic" autonomous="yes or no"
            xmlnx:xlink="http://www.w3.org/1999/xlink">

   <DESCRIPTION>

   </DESCRIPTION>
```

```
   <TIME>  </TIME>
   <SPACE> </SPACE>

   <INIT>

      <PORT name="port_name">  </PORT>

   </INIT>

   <IN>

      <PORT name="port_name">

   </IN>

   <OUT>

      <PORT name="port_name">  </PORT>

   </OUT>

   <STATE>

      <PORT name="port_name">  </PORT>

   </STATE>

</MODEL>
```

La balise `<MODEL>` contient les balises `<TIME>` et `<SPACE>` définies précédemment. De cette façon, nous unissons la description des modèles et leurs représentations du temps et de l'espace. L'attribut `autonomous` de la balise `<MODEL>` nous renseigne si le modèle peut être utilisé seul (*i.e.* sans être couplé à aucun autre). L'attribut `xmlns :xlink` définit un espace de nom. Ainsi tous les attributs définis peuvent prendre un attribut ayant comme préfixe `xlink` correspondant à la norme de définition des liens du *world wide web consortium* (w3c[78]).

La balise `<DESCRIPTION>` contient du texte libre. Cette balise permet de décrire le modèle par exemple.
Les balises `<INIT>`, `<IN>`, `<OUT>`, et `<STATE>` peuvent contenir plusieurs balises `<PORTS>`. Chaque port contient une ou plusieurs balises `<DATA>` de description des données qui lui sont attachées (nous décrivons la balise `<DATA>` un peu plus loin). Les ports d'initialisation ne peuvent pas être des ports d'entrée, et réciproquement.
La Balise `<IN>` permet de spécifier les données « réceptionnées » en entrée par le modèle. La balise `<OUT>` permet de spécifier les données « envoyées » par un modèle.

4.3.3.2 Les modèles couplés

Afin de spécifier le couplage de modèles, nous devons identifier les connexions externes (les ports d'entrée et de sortie du modèle couplé), ainsi que les différentes connexions internes qui

[78] URL du w3c : `http://www.w3c.org/`

relient les modèles composants. La figure 4.8 montre les connexions internes et externes sur un cas simple.

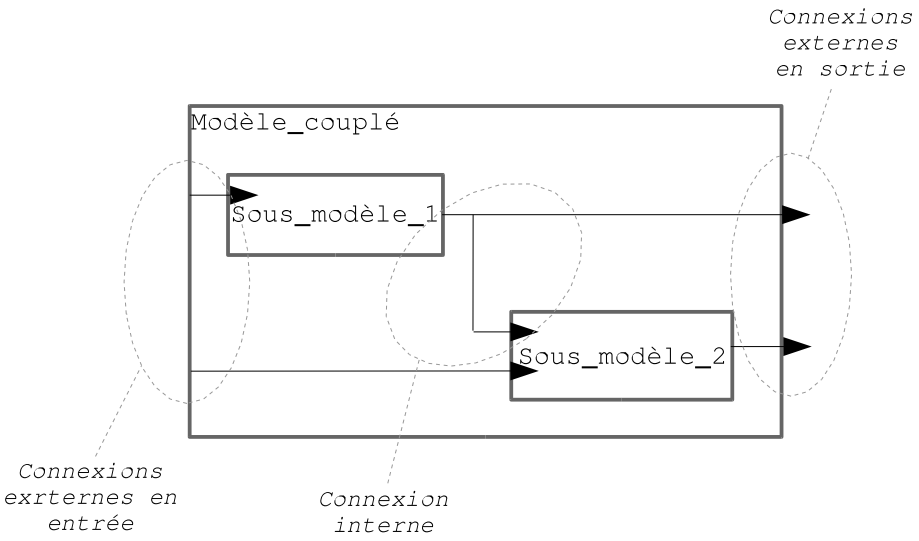


FIG. 4.8 – Exemple de modèle couplé composé de deux sous-modèles. Noter le fait qu'un port peut être connecté à plusieurs autres.

La syntaxe XML des modèles couplés diffère peu de celle des modèles atomiques. L'attribut `type` de la balise `<MODEL>` prend la valeur `coupled`, ce qui impose la définition des connexions internes et externes. Pour cela, nous introduisons de nouvelles balises définies dans la syntaxe XML ci-dessous.

```
<MODEL type="coupled" autonomous="yes | no"
       xmlns:xlink="http://www.w3.org/1999/xlink">

   <DESCRIPTION> </DESCRIPTION>

   <IN>          </IN>
   <OUT>         </OUT>
   <SPACE>       </SPACE>
   <TIME>        </TIME>

   <SUBMODELS>

      <LI xlink:type="simple"
          xlink:href="model_path#xpointer(XPath)"/>

   </SUBMODELS>

   <CONNECTIONS xlink:type="extended">

      <PORT_REF xlink:type="locator" xlink:label="port_label"
                xlink:href="model_path#xpointer(XPath)"/>
```

```
      <CONNECTION type="connection_type"
             xlink:type="arc"
             xlink:from="origin_port_label"
             xlink:to="destination_port_label"/>

   </CONNECTIONS>

</MODEL>
```

Nous voyons ici que la notion de décomposition hiérarchique est représentée par l'encapsulation des modèles dans la balise `<SUBMODELS>`. Les connexions externes du modèle couplé sont représentées par les mêmes balises que pour le modèle atomique.
Il y a autant de balise `<LI>` que de modèles composants. L'attribut `xlink :href` designe un lien qui pointe sur la racine d'un fichier de description d'un modèle. Ainsi, nous pouvons connaître la description des modèles composants.
La balise `<PORT_REF>` identifie un port qui entre en jeux dans la connexion des modèles. Le nombre de balises `<PORT_REF>` est égal à la somme des ports d'entrée et de sortie des modèles composants et du modèle composé.
La balise `<CONNECTION>` précise les connexions entre ports. Ainsi, le type des connexions peut être (voir figure 4.8) : `EIC` (connexion externes en entrée), `IOC` (connexion internes en sortie) ou `IC` (connexion interne). Chaque arc est orienté d'un port de sortie vers un port d'entrée.

Notons que dans cette syntaxe XML, les balises `<INIT>`, `<STATE>`, `<TIME>` et `<SPACE>` peuvent ne pas apparaître. En effet, il est possible de les déduire des modèles composants comme suit :

- l'ensemble des ports `<INIT>` correspond à l'union de tous les ports `<INIT>` des modèles composants,
- l'ensemble des ports `<STATE>` correspond à l'union de tous les ports `<STATE>` des modèles composants,
- les balises `<TIME>` et `<SPACE>` sont définies pour chaque modèle atomique.

Le modèle composé peut également spécifier ses propres ports différents des ports du modèle qui le compose. Les ports contenus dans les balises `<IN>` et `<OUT>` sont respectivement certains des ports `<IN>` et `<OUT>` des modèles composants. Il est nécessaire de les spécifier du fait qu'un même port peut être connecté à plusieurs autres.

4.3.4 Description des données

Nous décrivons ici l'ensemble des données en entrée ou en sortie des modèles. Trois niveaux de signification sont attachés aux données :

- le niveau informatique (entier, réel, booléen...),
- le niveau unitaire (l'unité de la donnée, une cardinalité ou sans unité...),
- le niveau sémantique (ce que représente la donnée : une distance, un nombre d'individus etc.).

La donnée représente plus que son unité et sa valeur aux yeux du modélisateur. C'est pourquoi nous ajoutons un niveau sémantique attaché à la donnée. Ce niveau est divisé en deux sous-niveaux :

- le sens propre de la donnée (une distance, une concentration, un objet...),
- sa représentation dans le temps et dans l'espace.

La représentation du temps ou de l'espace est très généralement indépendante de la donnée mais plutôt liée aux choix adoptés pour le modèle. La donnée fait référence à une certaine représentation. Par exemple, si la donnée est une matrice à deux dimensions représentant un espace topologique quadrillé, c'est la balise `<SPACE>` qui permettra de savoir que chaque élément de la matrice est situé sur une place particulière. Pour ce qui concerne le niveau du « sens », la donnée devra faire référence à une application XML particulière qui permet de représenter une « ontologie » du domaine scientifique dans lequel le modèle a été conçu. Une telle application reste à définir mais nous discuterons (brièvement) d'une première approche possible en ce qui concerne la modélisation en écologie.

La syntaxe XML pour la définition des données est la suivante :

```
<DATA name="data_name" xlink:type="extended">

        <TYPE class="data_type"/>

        <UNIT class="unit in the MKSA system"

        <TIME_REF xlink:type="locator" xlink:href="#xpointer(Xpath)"/>

        <SPACE_REF xlink:type="locator" xlink:href="#xpointer(XPath)"/>

        <METADATA xlink:type="locator"
                  xlink:href="metadata_path#xpointer(XPath)"/>

        <CONTENT dimension="matrix dimensions"
                 size="size of each dimension separated by ,">

        </CONTENT>

</DATA>
```

En ce qui concerne la balise `<UNIT>`, l'expression des unités se fait dans le système MKSA (Mètre, Kilogramme, Seconde, Ampère, etc.) pour les unités classiques. La donnée peut être également une simple cardinalité ou même sans unité.
La balise `<TIME_REF>` permet de faire référence à la représentation du temps définie précédemment et renseigne donc, par exemple, sur la fréquence de sortie ou d'entrée dans un modèle, d'une donnée particulière. La syntaxe de `path` est définie par le langage XPath[79]. Il en est de même pour la balise `<SPACE_REF>` qui concerne la représentation de l'espace.

Il est possible de décrire le contenu la donnée elle-même à l'intérieur de la balise `<CONTENT>`. Ce contenu sera toujours traité comme une chaîne de caractères. Des blancs séparent les contenus. Il est possible de traiter les matrices grâce aux attributs `dimension` et `size`.
Nous donnons également la possibilité de décrire des structures en incluant dans la balise `<DATA>` la balise `<AGREGATE>`. Cette inclusion récursive permet de décrire des données complexes comme l'ensemble des valeurs d'attributs d'un objet par exemple. Dans ce cas, l'argument `class` de le balise `<TYPE>` prend la valeur `agregate` et seules les balises `<UNIT>` et `<AGREGATE>` sont présentes comme filles directes de la balise `<DATA>`. Les balises `<UNIT>`, `<TIME_REF>` et `<SPACE_REF>` sont

[79] Voir par exemple [HM01] pour une présentation précise de ce langage.

toujours présentes au niveau des données agrégées, jamais au niveau de l'agrégat.

```
<DATA name="data_name" xlink:type="extended">

        <TYPE class="agregate"/>

        <AGREGATE>

           <DATA name="data_name" xlink:type="extended">
           </DATA>

        </AGREGATE>

</DATA>
```

La balise `<METADATA>` se réfère à une application XML qui reste à déterminer. Cette application doit permettre de dire à quelle entité s'applique la donnée et de quel point de vue. Néanmoins, une première piste existe. Elle consiste en la spécification, sous forme de diagramme de classes UML, des concepts de point de vue et d'entité dans un modèle, de représentation du temps ou de l'espace. Cette spécification est déjà bien avancée à l'intérieur du projet MIMOSA. À partir de cette dernière, il est possible de générer un fichier « .xmi » (XML meta-data interchange) avec des outils comme ARGOUML par exemple. Ce format XML permet de représenter les diagrammes UML. Ce fichier « .xmi » peut être transformé en un format XML propre à MIMOSA *via* une transformation XSLT[80]. Une fois que les concepts manipulés dans le domaine de la modélisation sont représentés dans une application XML, ils peuvent être mis en correspondance avec les entités de simulation manipulées dans des domaines particuliers. La figure 4.9, tirée d'un de nos articles [Dub02] et complétée, nous montre une association possible dans le domaine de la modélisation des écosystèmes.

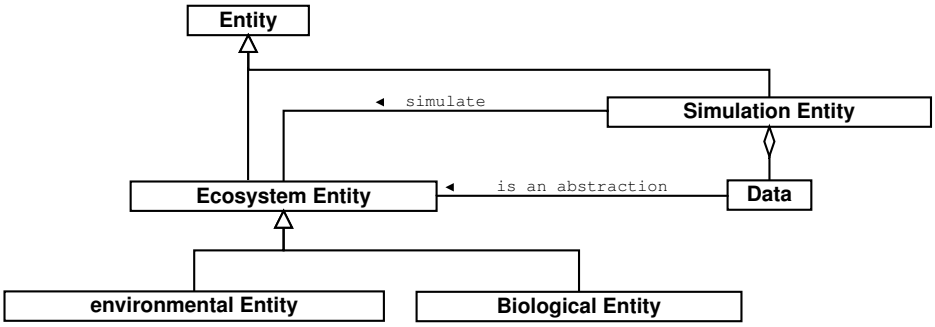


FIG. 4.9 – Diagramme de classes UML montrant les associations entre les entités de simulations et les entités simulées du point de vue des écosystèmes

Chaque donnée peut « pointer » sur une entité biologique ou environnementale, ce qui permet de la situer dans le contexte particulier du domaine de modélisation des écosystèmes. Dans la perspective de pouvoir valider, au moins partiellement, le couplage entre deux modèles au

[80]Se référer à l'ouvrage de référence sur XML [HM01] et le site de l'OMG sur le Model Driven Architecture (MDA) pour les définitions attachées à ces termes : `http://www.omg.org/mda/`

niveau sémantique, il apparaît nécessaire de définir une « ontologie » propre au domaine de modélisation. Ceci peut être fait en considérant le vocabulaire propre à une discipline. Il paraît difficile de construire un tel « dictionnaire » de manière exhaustive. Nous devrons donc mettre en place les outils qui permettront de la construire de façon empirique. Chaque modèle apportant des termes nouveaux construira l'arbre sémantique de sa discipline. Un effort particulier doit également être mené pour définir des termes communs entre disciplines pour permettre la validation du couplage de modèles issus de disciplines différentes. Ce travail reste à faire dans un cadre concerté.

Dans cette partie, nous avons défini une syntaxe XML particulière pour la définition des données, des modèles et des modèles couplés. Nous avons appelé cette application XML Meta Language for Model Coupling (MLMC). La DTD de ce langage est donnée en annexe A.1.

Jusqu'ici, nous avons élaboré un *framework* d'intégration de modèles hétérogènes en nous souciant des problèmes propres au couplage (opérationnel ou sémantique). Les modèles couplés que nous pouvons alors construire sont utilisés dans un cadre expérimental. Nous avons voulu définir une application XML qui permet la conservation, la modification et l'échange des expériences de simulation. Nous présentons cette application dans la section suivante.

4.4 Description des expériences

Comme nous l'avons dit précédemment, il y a un besoin important pour l'échange et la communication des modèles complexes. Nous sommes persuadés que ce besoin existe aussi pour les expériences. Traditionnellement, une expérience consiste à placer le système en cours d'étude dans des conditions expérimentales bien définies et à observer son évolution au cours du temps par l'intermédiaire de mesures. Deux approches peuvent être menées :

- on change les conditions expérimentales et on observe les variations de comportements du système ;
- on considère plusieurs systèmes placés dans les mêmes conditions expérimentales et on compare leur comportement.

La comparaison des comportements d'un ou plusieurs systèmes conduit à déployer de multiples stratégies statistiques. Ces stratégies correspondent aux études de sensibilité par exemple. L'avantage des expériences virtuelles (sur les modèles) par rapport aux expériences faites sur des systèmes réels est la maîtrise totale des conditions expérimentales et le fait que les mesures ne perturbent pas le système. Ainsi, nous pouvons construire des plans d'expériences rigoureux en vue de tests statistiques appropriés. Par analogie avec les expériences classiques menées en laboratoire ou sur le terrain, un plan d'expériences définit l'ensemble des conditions expérimentales, le nombre de réplicats (combien de fois on effectue la même simulation), éventuellement un blanc de simulation (une simulation de référence), une politique d'échantillonnage des résultats, etc. Pour ce qui concerne les expériences virtuelles, nous considérons que ce sont les valeurs des paramètres qui définissent les conditions expérimentales. Les paramètres sont les constantes du modèle ainsi que les valeurs d'initialisation des variables.

4.4.1 Laboratoire virtuel

Dans les sciences expérimentales classiques, il est nécessaire de décrire comment une expérience a été réalisée, le matériel utilisé et la démarche suivie. La description du protocole

expérimental prend une place importante dans les articles de journaux scientifiques sous le titre « *materials and methods* », le but évident de cette partie étant de permettre la reproductibilité de l'expérience. Dans cette partie, nous proposons une syntaxe XML (MLVE pour Markup Language for Virtual Experiments) pour la description des expériences virtuelles. Elle a pour but de décrire et stocker les plans d'expériences de simulations effectuées sur un modèle. L'adoption d'une syntaxe commune et portable permet l'échange et la validation des plans d'expériences sur plusieurs plateformes de modélisation ainsi que la reproductibilité des expériences. Indépendante de la plateforme d'exécution, cette syntaxe pourrait également permettre la mise au point d'une expérience standard sur un modèle standard pour la validation d'un outil de simulation par exemple. Dans la partie précédente, en définissant une syntaxe pour les modèles, nous avons introduit les termes de « port d'initialisation », « port d'entrée », « port de sortie » et « port d'état ». Ces définitions restent valables ici. Nous y ajoutons les notions de conditions expérimentales et de mesures.

Les conditions expérimentales fixent les valeurs à appliquer aux ports d'initialisation. Seulement une partie des ports peut être renseignée. L'autre partie sera renseignée par la valeur par défaut définie par le modèle. Si une valeur par défaut n'est pas définie, alors les conditions expérimentales doivent impérativement spécifier une valeur. Selon la politique d'exploration du modèle, les conditions expérimentales doivent être définies comme des ensembles de valeurs à parcourir pour toute ou partie des ports d'initialisation. La description des ensembles de valeurs peut être :

- simple : le paramètre prendra successivement les valeurs d'un ensemble de valeurs ou les valeurs comprises dans un intervalle selon un certain pas ;
- complexe : le paramètre prendra des valeurs qui sont fonction de la valeur prise par d'autres paramètres (on parlera de paramètres contraints).

Les modèles offrent la lecture des variables d'états par les ports d'états. La définition d'une expérience inclut la liste des points de mesures effectuées sur les variables d'états. La syntaxe indique tout simplement quelles sont les variables d'états que l'on désire observer. La lecture des mesures fait l'objet d'une politique d'échantillonnage. Cette politique définit le type d'acquisition des données. L'échantillonnage peut être de deux types :

- selon une certaine fréquence définie par le plan d'expériences,
- selon l'ordre d'arrivée des mesures défini par le modèle (évènements de sortie du modèle).

Nous parlons dans le premier cas de mesure active : le modèle est capable de générer les sorties en accord avec le plan d'expériences. Dans le deuxième cas, on parlera de mesure passive : chaque mesure est datée par la date d'occurrence de l'évènement de sortie de sa valeur. Pour ce qui concerne les mesures actives, on peut imaginer vouloir ne conserver qu'une valeur synthétisant un ensemble de valeurs en sortie d'un modèle. Jusqu'à maintenant, nous avons toujours supposé que le modèle n'effectuait pas lui-même de traitement sur les données. Cela implique de pouvoir appliquer des opérateurs statistiques (moyenne, par exemple) afin de synthétiser des indicateurs de comportement du modèle. Nous offrons la possibilité d'exprimer un tel type de mesure active, étant entendu que le traitement à la volée des mesures en sortie du modèle se fait par une application particulière.

Si le modèle ou le système intègre une partie stochastique (autrement dit, si deux simulations dans les mêmes conditions expérimentales conduisent à des réponses différentes), il est nécessaire d'effectuer des réplicats. Un ensemble de réplicats est un ensemble de simulations dans les même conditions expérimentales. Voyons maintenant l'utilisation d'XML pour la spécification des

expériences.

4.4.2 Spécification des expériences

Nous posons :
- qu'un modèle possède son simulateur ;
- qu'une simulation correspond à une exécution du simulateur ;
- que les mesures sont attachées à des ports d'états du modèle ;
- que l'initialisation se fait *via* les ports d'initialisation ;
- que les sorties d'un modèle se font dans un ou plusieurs fichiers.

Nous donnons dans ce qui suit l'ensemble des éléments de notre application XML. Ce document est bien formé (au sens d'XML), ce qui donne une idée de l'arborescence décrite par notre application. Nous décrirons ensuite de façon plus précise les balises qui définissent application.

```
<EXPERIMENT name="experience_plan_name" date="creation_date">

   <NOTES>

   </NOTES>

   <MODEL xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type="extended">

      <DESCRITPION xlink:type="locator"
                   xlink:href="model_description_path#xpointer(/)"/>


      <EXECUTABLE  xlink:type="locator"
                   xlink:href="model_executable_path"/>


      <OUTPUT_STREAM xlink:type="locator"
                   xlink:href="output_model_path"/>


      <EXECUTION type="mono | distributed | parallel">
         <EXECUTION_NODE xlink:type="locator"
                         xlink:href ="node_adress"/>

      </EXECUTION>


      <EXPERIMENTAL_CONDITIONS replicat="number_of_replicat"
                               apply="experimental policy on parameter">



         <CONDITION  xmlns:pathtoport="http://www.w3.org/1999/xlink"
                     pathtoport:type="simple"

            <SET>
               <ITEM value="a number"/>
```

```
            </SET>
         </CONDITION>


         <CONDITION xmlns:pathtoport="http://www.w3.org/1999/xlink"
                    pathtoport:type="simple"
              pathtoport:href="model_description_path#xpointer(XPath)">

            <INTERVAL begin="begin_value"
                      end="end_value"
                      step="step_value"/>

            <RANDOM function="random_function" average="value"
                    standard_deviation="value" nget="value"/>


         </CONDITION>


         <CONDITION xmlns:pathtoport="http://www.w3.org/1999/xlink"
                    pathtoport:type="simple"
              pathtoport:href="model_description_path#xpointer(XPath)">
            <CONSTRAINST>
               <EQUAL_TO>
                  <MULT>
                     <CONST value="a numbre"/>
                     <VAR pathtoport:type="simple"
               pathtoport:href="model_description_path#xpointer(XPath)"/>
                  </MULT>
               </EQUAL_TO>
            </CONSTRAINST>
         </CONDITION>

      </EXPERIMENTAL_CONDITIONS>

      <MEASURES xmlns:pathtoport="http://www.w3.org/1999/xlink"
                pathtoport:type="extended">

         <MEASURE pathtoport:type="locator"
               pathtoport:href="model_description_path#xpointer(XPath)"
                  type="active | passive"
                  frequence="x" apply="average"/>

      </MEASURES>

   </MODEL>

</EXPERIMENT>
```

La balise <NOTES> permet d'inscrire du texte libre (des remarques générale sur l'expérience). <DESCRIPTION> est un pointeur vers un fichier de description d'un modèle, ce qui permet de naviguer vers les modèles impliqués dans le plan d'expériences. La balise <EXECUTABLE> pointe vers le fichier exécutable du modèle. <OUTPUT_STREAM> désigne le répertoire de sortie des don-

nées.
La balise `<EXECUTION_NODE>` permet spécifier la distribution les simulations sur un ensemble de nœuds d'exécution (autant de balises `<EXECUTION_NODE>` que de nœuds). Si le modèle est distribué (parallèle), alors nous spécifions l'adresse de base de la machine parallèle. Si le modèle s'exécute sur la machine où est traité le fichier XML, alors l'attribut `xlink :href` prend la valeur `this`.

La balise `<EXPERIMENTAL_CONDITIONS>` contient l'ensemble des valeurs possibles prises par les ports d'initialisation. Ainsi, l'ensemble des conditions s'appliquent sur les ports d'initialisation du modèle. L'attribut `replicat` nous indique le nombre de simulation que nous voulons faire avec le même jeu de paramètre en entrée (cas d'une simulation stochastique par exemple). L'attribut `apply` définit la politique appliquée pour définir le nombre de simulations faites avec des jeux de paramètres différents. Cet attribut peut avoir deux valeurs : `exhaustive` ou `sequential`. Dans le premier cas, le nombre de simulations sera égal au produit cartésien des ensembles définis sur chaque port d'initialisation. Dans le deuxième cas, on prend les valeurs de ces ensembles dans l'ordre. Tous les ensembles doivent donc avoir le même nombre de valeurs et il y aura autant de simulations que de valeurs dans ces ensembles.

L'ensemble des valeurs successives que prend un port est décrit par la balise `<CONDITION>`. Trois cas sont possibles :

- dans le premier cas, les valeurs sont prises dans un ensemble décrit par l'ensemble des balises `ITEM`,
- dans le deuxième cas, les valeurs sont prises dans un intervalle avec un certain pas (défini dans la balise `<INTERVAL>`). Il est possible de définir un intervalle à l'intérieur duquel nous tirons un nombre aléatoire. Pour cela nous utilisons la balise `<RANDOM>`. Les types possibles de fonctions utilisées sont donnés avec la définition du type de document XML en annexe A.2. Il est bien entendu possible d'étendre leur nombre. La balise `<RANDOM>` peut également aparaître dans une condition avec la balise `<SET>`,
- dans le troisième cas, les valeurs possibles sont contraintes par rapport à d'autres. Par exemple, dans la syntaxe présentée précédemment, les valeurs du port d'initialisation doivent toujours être deux fois supérieures à celles prises par un port de référence (troisième balise `<CONDITION>`). L'application qui traite le fichier d'expérience peut avoir deux politiques : soit elle génère elle-même les paramètres contraints, soit elle vérifie la validité de la contrainte.

La balise `<MEASURE>` contient des références sur l'ensemble des ports d'états du modèle affectés par une mesure. L'attribut `fréquence` est défini si l'attribut `type` a la valeur `active`. C'est un entier positif qui définit le nombre de sortie que l'on désire prendre en compte dans le calcul d'une valeur de synthèse de type moyenne par exemple, en affectant la valeur `average` à l'attribut `apply`.

L'application qui lit un tel fichier doit être capable de déterminer le nombre de simulations qui devront être réalisées en analysant l'ensemble des conditions attachées aux ports d'initialisation. Cette application est libre de mener la politique de son choix lorsqu'elle trouve par exemple dix valeurs possibles en entrée pour le port A et cinq pour le B. Faut-il faire dix simulations ou seulement cinq ? Nous offrons seulement la possibilité d'exprimer cette politique dans les cas simples décrit dans la syntaxe.
De manière générale, l'utilisation des contraintes conduit à la résolution de systèmes d'inéqua-

tions. Cette résolution peut s'avérer difficile. La partie expression d'une contrainte peut faire intervenir :

- tout opérateur arithmétique (+,−,×,/,%),
- toute fonction exponentielle, logarithmique et trigonométrique,
- toute fonction aléatoire (tirage aléatoire dans un ensemble discret ou continu de valeurs selon une certaine loi).

Pour ce qui concerne le nommage de fichiers de sortie, le choix est laissé à l'application qui lit le fichier de description des expériences. Ce fichier contient l'ensemble des simulations à effectuer. Pour savoir quel paramétrage de simulation particulière correspond à quels résultats particuliers, il faut établir une correspondance entre les noms des fichiers, l'ordre des simulation et les nœuds d'exécution par exemple. À partir du fichier d'expériences, on peut générer des fichiers de paramétrage correspondant à une simulation particulière, dans un format que le modèle sait lire (XML ou non).
Dans cette partie, nous avons défini une syntaxe XML particulière pour la définition des expériences. Nous avons appelé cette application XML Meta Language for Virtual Experiment (MLVE). La DTD de ce langage est donnée en annexe A.2.

4.5 Discussion et conclusion

Dans cette partie, nous avons proposé un *framework* pour l'intégration de modèles hétérogènes. Ce *framework* se compose de quatre couches (ou niveaux) : la couche opérationnelle, la couche simulation, la couche modèle et la couche sémantique. Ce cadre conceptuel nous a permis d'intégrer différentes propositions existantes dans notre propre architecture.

Pour la couche opérationnelle, nous proposons des connecteurs réseaux pour différents langages de programmation qui reposent sur des technologies standards allant des *sockets* jusqu'à SOAP. Ces connecteurs permettent à des coordinateurs DEVS de communiquer *via* le réseau (voir figure 4.6 page 103).

Pour la couche simulation, nous proposons l'intégration des simulateurs abstraits et du principe de DEVS-BUS pour disposer d'une base solide en matière de simulation distribuée. Ces algorithmes sont basés sur les notions d'évènements discrets et de fonctions de transition « classiques » de DEVS. DEVS nous permet de bénéficier des notions de couplage et de décomposition hiérarchique inhérentes à ce formalisme. Ainsi, nous pouvons composer des modèles par couplage au sens de DEVS. L'adoption de DEVS au niveau simulation nous oblige à entrer dans une certaine logique. Ainsi, nous considérons deux types de simulateurs :

- soit des simulateurs DEVS, auquel cas aucun problème de couplage n'est posé,
- soit des interfaces fonctionnelles, que nous avons appelées *wrappers*, qui permettent de rendre les simulateurs compatibles avec DEVS.

La deuxième proposition permet d'intégrer des simulateurs préexistants sans forcément se ramener dans une logique DEVS. Une difficulté apparaît avec le *wrapping* lorsque le temps n'est pas explicite dans les modèles que l'on veut coupler.
En effet, tous les formalismes n'intégrent pas le temps (les automates à états ou les réseaux de Petri, par exemple). Or, le temps et la base de temps utilisés sont fondamentaux pour le couplage de modèles dans notre *framework*. Dans le cas des réseaux de Petri, le comportement du réseau fait évoluer le marquage uniquement en fonction des transitions franchissables. Si une transition est franchissable, alors les jetons des places amonts sont retirés et des jetons sont injectés dans les

places avales. À aucun moment, la notion de temps n'a été évoquée. Pour résoudre le problème, nous pouvons considérer que le temps est discret et prend ses valeurs parmi les entiers positifs. À chaque transition ou lorsque l'on a épuisé l'ensemble des transitions franchissables à l'instant t, le temps avance. Cette approche n'a aucune conséquence sur le modèle lui-même, mais quelles sont les conséquences pour les modèles couplés ? Il faut en effet que la base de temps soit compatible. La réponse à cette question se situe aussi bien dans le modèle à base de réseaux de Petri que dans les modèles couplés. Dans certains cas, le réseau de Petri représente un processus non temporel ce qui implique que le choix d'une base de temps dépendra des modèles auxquels il est couplé. Si le réseau de Petri représente un processus temporel, il faut alors adopter une base de temps conforme au processus modélisé.
Une autre difficulté des *wrappers* concerne la définition des fonctions de transitions externes et de sorties pour des modèles pré-existants ? Là encore, la réponse est directement liée au modèle que l'on désire coupler. Le modélisateur doit fournir un effort minimal de « traduction » de la réception ou de l'envoi d'évènements dans son formalisme ou son paradigme.

En ce qui concerne la couche modèle, nous avons proposé une application XML (MLMC) qui permet de décrire le couplage de simulateurs. En effet, cette syntaxe décrit les interfaces entre modèles en terme de ports d'entrée-sortie interconnectés. De plus, cette syntaxe intègre des notions de sémantique attachée aux données qui transitent par ces ports. Par définition, XML permet d'encapsuler les données dans une hiérarchie qui peut représenter les différents niveaux sémantiques d'une donnée (de l'unité utilisée jusqu'à sa représentation pour le modélisateur). Une application qui permet de valider le couplage de modèle au niveau sémantique reste à définir mais nous pensons avoir posé des bases intéressantes dans ce sens. Dans un contexte scientifique où les ontologies sont assez précises, nous pensons que ce type d'applications pourrait être développé sans problèmes majeurs.
MLMC travaille au niveau 1 et 4 des niveaux de spécification donnés par Zeigler (voir tableau 1.1 page 17). Ceci implique que nous n'avons pas de représentation de la dynamique des modèles dans notre syntaxe. Là encore, nous pensons qu'il est possible d'arriver à une application XML augmentée de cette description. Comme nous l'avons dit, XML a une compatibilité ascendante ce qui nous assure de pouvoir étendre notre syntaxe. Nous avons commencé des travaux sur la définition XML des équations différentielles et des réseaux de Petri. Il existe également des spécifications XML en cours d'élaboration pour les modèles DEVS[81].

Devant la complexité des modèles qu'il est envisageable de construire avec une telle approche, nous pensons, en accord avec V. Grimm [Gri02], qu'il peut être utile d'adopter l'attitude du naturaliste devant son sujet. Ici, la notion d'expériences virtuelles prend tout son sens. Dans bien des cas, le modélisateur doit faire des plans d'expériences pour valider une hypothèse ou simplement faire une calibration. Notre proposition de spécification d'expériences virtuelles avec XML peut permettre de mieux appréhender la problématique des plans d'expériences bien connue des naturalistes. De plus, cette spécification peut aider à communiquer nos expériences. Il est évident que nous aurions pu établir nous-mêmes un format de fichier, mais ce choix ne permet pas de s'ouvrir aux outils qui se développent actuellement pour XML, et notamment ceux orientés vers internet. Dans ce cadre, nous avons pour objectif à court terme de mettre en place un laboratoire virtuel.
Par analogie avec un laboratoire réel, un laboratoire virtuel doit fournir l'ensemble des outils nécessaires pour la préparation, la réalisation et l'analyse des résultats d'expériences virtuelles.

[81] M.K. Traoré au LIMOS à Clermont-Ferrrand.

Une expérience virtuelle est la simulation d'un modèle informatique et l'observation de son comportement *via* les résultats (ou traces) de simulations. Ces traces peuvent être considérées comme un résultat émergent [SPTD98]. Nous pouvons fixer trois étapes dans la réalisation d'une expérience virtuelle :

- la préparation du système correspond à la définition de l'ensemble des valeurs de paramètres et conditions initiales affectées au modèle (ce sont les conditions expérimentales),
- la réalisation se fait *via* une ou plusieurs simulations sur un ou plusieurs nœuds d'exécution (simulation distribuée),
- l'analyse des résultats se fait *via* des outils de visualisation ou de traitement des données.

La logique et la problématique liées à l'expérimentation restent les même que pour une expérience classique : étudier le comportement d'un système sous certaines conditions et étudier la variabilité de ce comportement. L'étude de la validité du modèle par rapport au système réel fait partie de l'« analyse des résultats ». À ce niveau, il existe beaucoup de techniques statistiques (tests d'ajustements, analyse de variance, etc.) qui peuvent servir d'outils. Nous voulons proposer un service *web* qui assure non seulement la fonction de laboratoire virtuel, mais également de plateforme de couplage dynamique de modèles hétérogènes. Ce travail a commencé cette année dans le cadre de la thèse de G. Quesnel au Laboratoire d'Informatique du Littoral à Calais.

Notre démarche est assez proche de celle adoptée par l'OMG et connue sous le nom de MDA (pour *Model Driven Architecture*). En effet, nous voulons intégrer des logiciels (ici des modèles) pré-existants. C'est le problème bien connu en génie logiciel qui concerne les architectures légataires (*legacy architectures*), c'est-à-dire la maintenance, l'évolution, la réutilisation ou l'adaptation de l'existant. De plus, nous utilisons les mêmes technologies, comme UML et XML, pour les ontologies en cours de définition dans le groupe de travail MIMOSA du GdR I3. Ces technologies nous garantissent de bénéficier des dernières avancées en terme d'évolution du logiciel. Seulement, l'approche MDA, bien que basée sur la notion de modèle, s'adresse essentiellement au logiciel en tant qu'outil effectuant une tâche pour une entreprise ou une organisation, pas en tant que modèle de systèmes dynamiques avec lequel nous nous interrogons et nous expérimentons sur la réalité. Nous avons donc choisi une approche intégrative qui est relativement ouverte et centrée sur les modèles comme outils de connaissances et d'expériementations.

Maintenant que nous avons présenté une intégration formelle (chapitre 3) permettant de spécifier un modèle d'agents réactifs situés comme un modèle DEVS et un *framework* pour l'intégration de modèles hétérogènes, nous allons présenter l'application d'une intégration en écologie marine. Cette application illustre non seulement notre méthode de transfert d'échelles définie au chapitre 2.2, mais aussi le concept d'expériences virtuelles introduit dans ce même chapitre et complété ici d'une spécification opérationnelle.

5

Intégration pour la simulation du transfert d'échelles : application à un modèle proie-prédateur en écologie marine

Résumé

Dans ce chapitre nous illustrons la méthode présentée au chapitre 2, section 2.2. Le but est de faire cohabiter des modèles de différentes natures au sein d'une même simulation. Nous avons d'un côté un modèle d'agents réactifs qui simule le comportement alimentaire de copépodes se nourrissant de phytoplancton et, de l'autre, deux équations différentielles qui simulent la dynamique d'un système proies-prédateurs. Le premier modèle est caractéristique de l'échelle des individus et le deuxième de l'échelle des populations.
En adoptant une attitude d'expérimentateur, nous sommes capables de construire et de paramétrer une fonction mathématique qui caractérise l'activité alimentaire du copépode (fonction trophique). Cette fonction intervient dans la définition d'un système d'équations différentielles modélisant l'interaction proies-prédateurs. En couplant les deux modèles à l'aide de cette fonction, nous sommes capable de faire cohabiter deux échelles différentes dans une même simulation. Il apparaît que ce couplage apporte une réelle plus-value par rapport au simple paramétrage de la fonction trophique. En effet, les dynamiques du système couplé et du système paramétré sont différentes. Dans le système couplé, nous prenons en compte directement l'hétérogénéité spatiale des distributions de proies. Cette hétérogénéité est caractéristique de l'échelle des individus. Le couplage permet de prendre en compte cette caractéristique individuelle à l'échelle de la population. Ainsi, nous simulons un transfert d'échelle.
Nous pensons que la méthode de couplage que nous utilisons ici peut-être appliquée dans d'autres contextes de modélisation.

Sommaire

5.1 Sensibilité du modèle

Dans cette partie, nous étudions le système présenté au paragraphe 2.3 de la page 36. Dans un premier temps, nous sommes intéressés par le modèle d'agents réactifs en discutant des choix de représentations dans le modèle. Nous présentons ensuite l'utilisation du modèle comme un laboratoire virtuel pour la détermination d'une fonction émergente. Puis, nous couplons le modèle d'agents réactifs avec un modèle d'équations différentielles modélisant le même système.

Au départ, nous avions conçu une version du modèle en temps discret. Nous avons d'abord effectué des tests de sensibilité sur cette version. Nous avons ensuite vérifié que pour ces tests, les résultats entre la version temps discret et évènements discrets (cf. formalisation du modèle au paragraphe 3.5.2 page 71) étaient les mêmes. Notre travail de comparaison des deux modèles est resté informel, il mériterait d'être approfondi[82]. Dans ce qui suit, seule la partie « sensibilité au choix de représentation » concerne les résultats obtenus pour la version temps discret du modèle tels qu'ils ont été publiés en 2001 [DRP01]. Nous nous sommes intéressés aux problèmes liés à notre représentation discrète du copépode. Le modèle mathématique des processus physiologiques (voir annexe B) a été étudié de façon précise par P. Caparroy. Les résultats sont disponibles dans sa thèse [Cap96]. La suite du chapitre est écrite à partir des résultats obtenus avec la version à évènements discrets du modèle.

Dans ce qui suit, nous allons nous intéresser principalement à l'efficacité des copépodes en tant que prédateurs. Cette efficacité peut se mesurer à l'aide d'un taux, appelé taux d'ingestion, qui mesure la quantité de proies capturées par le copépode par unité de temps. Ce taux d'ingestion est le reflet de l'activité alimentaire du copépode. Il est prépondérant dans les modèles mathématiques en dynamique de populations (nous y reviendrons au paragraphe 5.2). Dans notre modèle, nous devons faire des choix particuliers de représentation. Ces choix concernent :

1. la dimension de l'espace continu dans lequel évolue le copépode,
2. la politique adoptée par le copépode quand il arrive aux limites de cet espace,
3. la nature de la distribution des cellules de phytoplancton dans le milieu.

Nous devons connaître l'influence de ces choix sur la mesure du taux d'ingestion avant d'utiliser le modèle dans un cadre expérimental. C'est ce que nous allons faire juste après une courte discussion sur la génération de nombres pseudo-aléatoires.

5.1.1 À propos des générateurs de nombres pseudo-aléatoires

De toute évidence, un ordinateur est une machine totalement déterministe. Pour simuler l'aléatoire, de nombreux travaux proposent des algorithmes de génération de nombres dit « pseudo-aléatoires » [L'E98]. Le préfixe « pseudo » est là pour nous rappeler qu'ici l'aléatoire n'est qu'une illusion, illusion acceptable dans certaines limites. C'est de ces limites qu'il faut être conscient.
Dans toute modélisation stochastique, *a fortiori* dans notre modèle, les résultats de simulations sont étroitement liés à la qualité du générateur de nombre pseudo-aléatoires. Dans une simulation multi-agents par exemple, il est une des bases des comportements des agents et donc des propriétés émergentes de ces systèmes. Des questions comme la part de responsabilité du générateur aléatoire dans les résultats sont difficiles et très rarement abordées. Sans vouloir entrer dans

[82] Par exemple en montrant clairement le gain en temps de calcul de la version évènement discret.

une étude statistique complexe d'un modèle, il s'avère tout de même essentiel de connaître les points importants liés à la génération de nombres pseudo-aléatoires pour connaître le domaine de validité des résultats d'une simulation. Dans ses travaux, P. Lecuyer s'est beaucoup intéressé à ces problèmes ; ainsi il a pu mettre en évidence les défauts majeurs des générateurs standards utilisés par Java, Excel ou encore Visual Basic [L'E01]. Il est nécessaire de choisir un générateur qui satisfasse le compromis entre temps de calcul, durée de simulation et qualité des distributions générées. Dans un contexte distribué, il serait également nécessaire de choisir une implémentation adaptée pour éviter les corrélations malheureuses entre les séries de nombres générés sur chaque nœud de simulation [TH01]. Dans le cas de simulation mono-processeur, comme ici, nous nous focalisons sur deux points :

1. les problèmes d'auto-corrélations internes du générateur,
2. la longueur de la suite de nombres générés sans répétition (la période).

Il est possible d'estimer un nombre maximum de tirages au cours d'une simulation. Nous considérons une simulation avec un nombre de changements de direction suivant une fréquence maximum de $20s^{-1}$(c'est-à-dire toutes les $0,05s$, ce qui correspond au temps de manipulation d'une cellule par le copépode). Chaque changement implique de tirer trois nombres correspondants aux trois nouvelles directions d'un copépode sur les trois axes. Au début de la simulation, nous positionnons les cellules de phytoplancton, ce qui nécessite là aussi trois tirages par cellule. Pour un copépode, nous avons la formule suivante :

$$N \approx T \times 20 \times 3 + (n \times 3) \tag{5.1}$$

avec : N le nombre de tirages, T le temps simulé en seconde, n le nombre de cellules de phytoplancton.

L'équation 5.1 ne prend pas en compte le nombre de rebonds (dépendant de la trajectoire du copépode) et les changements de directions aléatoires intervenant entre deux captures de phytoplancton. Ici, nous considérons un copépode qui mange en continu et change de direction entre deux captures sans se déplacer. Ainsi, nous maximisons le nombre de tirages calculés. Il reste néanmoins que cette équation est approximative. Nous allons donc considérer un temps simulé très grand (relativement aux expériences que nous allons faire par la suite) et un nombre maximum de cellules et de copépodes dans le milieu pour minimiser au maximum les erreurs d'approximation, soit :

avec $T = 86400$ (un jour),

$n = 10^6$ (nombre maximum de cellules dans le milieu),

et $nc = 100$ (le nombre maximum de copépodes dans le milieu),

on a $N \approx 521,4.10^6$ tirages.

Dans ce qui suit, nous allons être amenés à effectuer un grand nombre d'expériences successives sur ce modèle (surtout dans la partie couplage). Nous majorons ce nombre en le fixant à 10^5. En le multipliant par N, nous obtenons une estimation du nombre maximum de tirages effectués lors de nos simulations, $N_{max} \approx 5,214.10^{13}$ tirages.
Pour s'assurer de la bonne uniformité des distributions, P. Lecuyer conseille de prendre un générateur dont la période p est deux fois supérieure au nombre maximum de tirages estimé [L'E98]. Ce qui nous amène à $p \approx 10^{14}$. Nous nous sommes donc orientés vers le générateur appelé *Mersenne Twister* développé par M. Matsumoto et T. Nishimura [MN98]. Il a deux avantages majeurs, il demande peu en temps de calcul et possède une période très longue, égale à $2^{19937}-1$.

P. Lecuyer le qualifie de « bon générateur aléatoire » [L'E01] et garantit qu'il ne possède pas d'auto-corrélations internes. Son utilisation nous assure que les résultats de nos simulations ne seront pas biaisés par des artefacts liés au nombre pseudo-aléatoires.

5.1.2 Sensibilité aux choix de représentation

Pour étudier l'impact des trois choix de représentation cités plus hauts, nous nous intéressons à l'évolution du nombre de cellules de phytoplancton (que nous appelons aussi particules) consommées par un copépode au cours du temps. En comparant ces évolutions, nous voulons choisir les meilleures représentations pour notre modélisation. Pour cela, nous éliminons certaines caractéristiques aléatoires du copépode dans notre modèle.
La source principale de stochasticité du copépode est son déplacement aléatoire. Nous supprimons cet aspect et le remplaçons par un déplacement rectiligne. Le changement de direction du copépode est alors uniquement lié à la perception d'une proie ce qui nous permet d'étudier l'effet de la distribution des proies sur l'ingestion du copépode. Nous désactivons également le modèle de la gestion de l'énergie, ainsi tous les copépodes ont faim en continu. Le déplacement rectiligne du copépode permet de faire ressortir l'effet du choix du type de rebonds aux limites de l'espace. Pour toutes les simulations, nous considérons les valeurs de paramètres données par le tableau 5.2. Nous simulons l'activité d'un seul individu.

Tab. 5.1 – Tableau des valeurs des principaux paramètres utilisés dans le modèle.

Paramètres	Valeurs
Angle de perception (rad)	π
Distance de perception	$2mm$
Distance de capture	$1mm$
Vitesse de nage	$0,5mm.s^{-1}$
Position initiale	centre de l'espace
Angle initial du vecteur directeur (rad)	$\widehat{xOy} = 1\ \widehat{xOz} = 1\ \widehat{yOz} = 1$
Pas de temps	$0,05s$
Concentration initiale en cellules de phytoplancton	$0,5cell.mm^3$

Nous choisissons trois volumes pour l'espace modélisé afin de montrer l'impact des dimensions de l'espace sur l'efficacité de capture du copépode :

1. 10^3mm^3,
2. 8.10^3mm^3,
3. 64.10^3mm^3.

Pour toutes les expériences, la concentration en phytoplancton (ici, nous emploierons parfois le terme de particules) est de $0,5$ particules.mm^{-3}, soit 32000 particules pour un espace de 64.10^3mm^3, ce qui correspond à une forte concentration [CC96].
Nous choisissons de tester trois types de conditions aux limites de l'espace :

1. la première consiste en des rebonds de type réflexion (respect de la loi de Descartes),
2. la deuxième supprime le problème en considérant un espace torique,

3. la troisième consiste en des rebonds aléatoires (*i.e.* angles de rebonds aléatoires).

Dans tous les cas, la distance totale de déplacement du copépode pendant une itération reste constante. Pour tester l'impact de la nature de la distribution des cellules de phytoplancton sur l'ingestion du copépode, nous utilisons trois méthodes de génération d'une distribution :

1. distribution régulière : la distribution des cellules de phytoplancton se fait sur une grille discrétisant l'espace cubique. La taille du côté Δx d'une case correspond à $\Delta X / \sqrt[3]{n}$ avec ΔX la taille d'une arête du cube et n le nombre de particules,
2. distribution homogène ou uniforme : les coordonnées des cellules sont générées selon une loi uniforme,
3. distribution hétérogène ou en paquet : nous tirons un nombre de points virtuels à l'intérieur du cube, correspondant à 10% du nombre total de particules à l'aide d'une loi de distribution uniforme. Ensuite, nous distribuons les particules de manière aléatoire autour de ces points virtuels. Pour chaque particule, nous tirons ses coordonnées suivant une loi normale, centrées sur un point virtuel tiré au hasard, suivant une loi uniforme.

Le tableau 5.2 présente le plan d'expériences adopté pour l'ensemble des tests.

TAB. 5.2 – Plan d'expériences pour chaque volume spatial considéré

Type de distribution du phytoplancton	Rebonds déterministes		Rebonds aléatoires
	réflexion	espace torique	
Homogène ou uniforme	30 simulations (distributions différentes)	30 simulations (distributions différentes)	30 simulations (une distribution) et 30 simulations (distributions différentes)
Hétérogène ou en paquet	30 simulations (distributions différentes)	30 simulations (distributions différentes)	30 simulations (une distribution) et 30 simulations (distributions différentes)
Régulière	une simulation	une simulation	30 simulations

Pour ce qui concerne les simulations avec des distributions hétérogènes et des rebonds aléatoires, le fait de faire des simulations avec une même distribution et des simulations avec des distributions différentes permet de séparer les effets des deux choix de représentations.
Dans tous les cas, la simulation se termine lorsque le copépode a consommé toutes les particules présentes dans le milieu ou lorsqu'il n'est plus capable d'en consommer. Dans ce qui suit, nous exposons les résultats principaux qui nous ont permis de choisir une représentation optimale pour utiliser le modèle.

Influence du type de rebonds

La figure 5.1 présente une comparaison entre le nombre cumulé de rebonds et le nombre cumulé de particules ingérées par le copépode au cours du temps dans un volume de $64.10^3 mm^3$

pour une distribution uniforme des particules, dans le cas de rebonds déterministes. Nous observons qu'à partir de 2000 secondes simulées, l'augmentation du nombre de rebonds devient linéaire et que le copépode ne consomme plus de cellules.

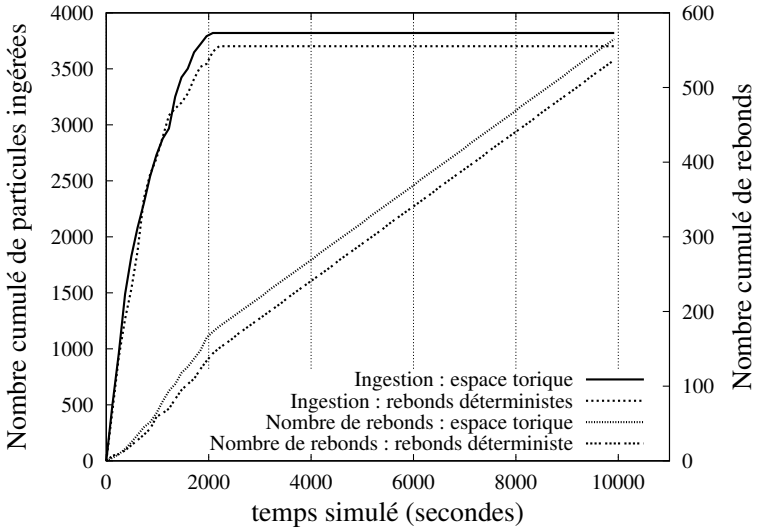


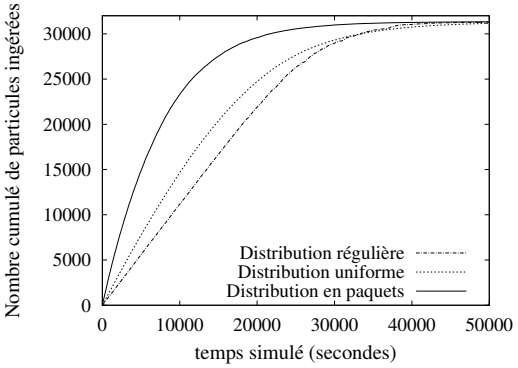
FIG. 5.1 – Comparaison entre le nombre cumulé de rebonds et le nombre cumulé de cellules de phytoplancton ingérées par le copépode pour une simulation particulière dans le cas de rebonds déterministes. On note que lorsque le premier devient linéaire, le second n'augmente plus.

Les rebonds déterministes semblent induire un artefact de simulation important. Il s'avère que l'agent copépode n'est pas capable d'explorer tout l'espace dans ce cas précis. En fait, la trajectoire devient totalement déterministe au bout d'un certain temps, il ne perçoit plus de proie. Cet artefact serait certainement diminué en considérant une trajectoire aléatoire en l'absence de perception de nourriture. Il induirait néanmoins des choix de trajectoire, et donc une vitesse de consommation, partiellement liée à l'algorithme de gestion des rebonds dans ce cas[83]. Nous avons donc décidé d'adopter un comportement de rebonds aléatoires.
La figure 5.2 présente les courbes moyennes de 30 simulations effectuées en considérant un espace de $64.10^3 mm^3$ et des rebonds aléatoires pour les trois types de distribution. Dans tous les cas, le copépode consomme la totalité des particules présentes. Il est donc capable d'explorer tout l'espace.

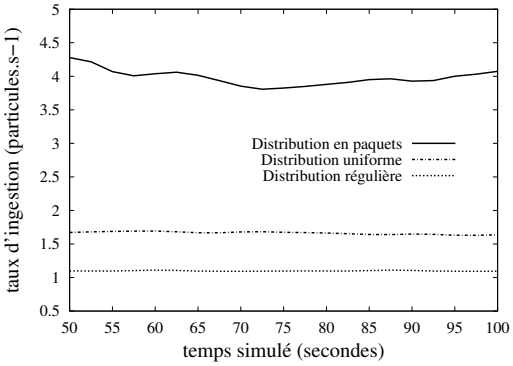
Influence du type de distribution

La figure 5.2(a) montre trois courbes (une courbe par type de distribution de particules) représentant le nombre cumulé de particules consommées par le copépode en fonction du temps. Les trois pentes à l'origine dépendent du type de distribution. Nous observons ici une augmentation de ces pentes avec l'augmentation de l'hétérogénéité de la distribution. Le même résultat qualitatif est obtenu avec les autres volumes d'espace considérés. Cette observation est importante car elle reflète la validité d'une modélisation agent. Ici les individus sont discrets et les interactions purement « mécaniques » fournissent des résultats en accord avec ceux obtenus avec

[83]Il serait difficile de savoir dans quelle mesure.



(a) Courbes d'ingestion moyenne simulées pour un espace de $64.10^{-3}mm^3$ (30 simulations)



(b) Évolution moyenne du taux d'ingestion en début de simulation pour un espace de $64.10^{-3}mm^3$ (30 simulations)

FIG. 5.2 – Nous observons qu'une augmentation de l'hétérogénéité de la distribution accélère la consommation de cellules de phytoplancton, ce qui se traduit par une augmentation de la pente des courbes figure (a). La valeur de cette pente correspond au taux d'ingestion donné figure (b). Les même résultats qualitatifs sont obtenus pour d'autres dimensions de l'espace.

des modèles continus [CC96] qui simulent l'effet de la turbulence sur le taux d'ingestion du copépode. Dans ces modèles, la turbulence est censée augmenter le taux de rencontre entre les proies et les prédateurs en induisant une hétérogénéité de la distribution des cellules. Nous sommes capables ici de reproduire cet effet avec une autre approche. Nous revenons sur ce point dans la discussion de ce chapitre. La pente à l'origine des courbes de la figure 5.2(a) correspond au taux d'ingestion du copépode. La figure 5.2(b) nous montre l'évolution de ce taux au début de la simulation.

La figure 5.2(b) donne une première vision quantitative de l'effet du type de distribution sur le taux d'ingestion du modèle. En adoptant une position d'expérimentateur, nous sommes capables de relier distribution et taux d'ingestion. Ceci nous amène à étudier plus précisément cette relation et à préciser le calcul du taux d'ingestion au paragraphe 5.2.2. Le fait que ce taux soit constant au début de la simulation nous intéresse particulièrement pour utiliser le modèle du copépode. Nous revenons plus tard sur ce point. Pour ce qui nous concerne ici, nous pouvons dire que le type de distribution influence fortement les résultats de simulations.
Cet état de fait est connu en écologie [FPV95] et constitue une des raisons de l'utilisation des simulations centrées agents [CL96] dans cette discipline. En effet, l'hétérogénéité spatiale est facilement représentable par un environnement discret dans un SMA. Seulement, à notre connaissance, il n'a jamais été établi de lien quantitatif entre hétérogénéité de la source de nourriture dans un espace continu et dynamique de population. Un peu plus loin dans ce chapitre, nous proposons d'établir ce lien.

Influence de la dimension de l'espace

Pour finir, nous voulons connaître l'influence de la dimension de l'espace sur l'efficacité de prédation du copépode. Pour cela, nous considérons 30 simulations pour les trois dimensions citées plus haut. Le copépode adopte des rebonds aléatoires. Nous dessinons l'évolution de l'écart-

type du nombre de particules ingérées comme un pourcentage de particules restantes dans le milieu (figure 5.3). Ceci nous permet de comparer les résultats obtenus avec un nombre différent de particules au départ (fonction de la taille de l'espace).

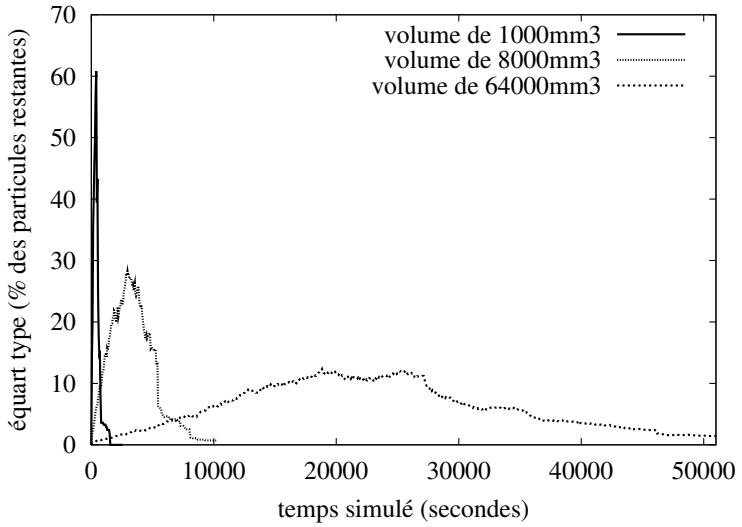


FIG. 5.3 – Écart-type du nombre de cellules de phytoplancton ingérées (en pourcentage du nombre de particules restantes) en fonction du temps. Chaque courbe représente la variabilité des résultats de simulations effectuées pour trente distributions hétérogènes différentes (avec rebonds aléatoires du copépode). Nous observons une décroissance des maxima avec une augmentation de la taille de l'espace, ainsi qu'un décalage dans le temps des maxima de variabilité.

La figure 5.3 met en évidence l'influence du choix des dimensions de l'espace sur la variabilité des résultats de simulation. Dans le cas du plus petit espace, la variabilité est très importante (l'espace influe beaucoup sur les résultats). Cette variabilité diminue avec l'augmentation des dimensions de l'espace. Nous observons également un décalage dans le temps du maximum de variabilité avec l'augmentation de la taille de l'espace, puis une décroissance jusqu'à 0 due au fait que toutes les cellules sont consommées.
En choisissant un espace de $64.10^3 mm^3$, nous garantissons que la variabilité de la moyenne du nombre de cellules ingérées pour 30 simulations ne dépasse pas 20% du nombre de cellules restantes dans le milieu. Ce qui nous intéresse particulièrement pour la suite est que la variabilité est presque nulle pour des durées de simulations très courtes.

En résumé, en simulant des rebonds aléatoires et un espace cubique de $64.10^3 mm^3$, nous avons deux choix de représentation qui influencent très peu le taux d'ingestion mesuré. Nous considérons alors la moyenne du taux d'ingestion de 30 simulations comme un bon estimateur. Nous avons observé une grande influence de la nature de la distribution sur le taux d'ingestion.

5.1.3 Observation d'une première fonction émergente

Ce travail sur l'effet des choix de représentation sur les résultats de simulations nous a conduit à identifier une première fonction émergente. Cette observation est faite en considérant notre modèle comme une laboratoire virtuel pour le paramétrage d'un modèle agrégé, de la même façon qu'au paragraphe 2.2.2 pour le modèle de diffusion.

En faisant un effort d'abstraction, nous pouvons considérer les résultats illustrés par la figure 5.2(a) comme étant l'apparition d'un composé dans une réaction chimique mettant en jeu deux réactifs (représentés par le copépode et le phytoplancton). Dans ce contexte, le nombre cumulé de cellules ingérées correspond au nombre de molécules produites par la réaction (la rencontre des deux réactifs). Ce type de réaction est formalisée par une équation de type monomoléculaire ou fonction de Mitscherlisch [Pav94] comme suit :

$$\frac{dX}{dt} = a(1 - X/k) \tag{5.2}$$

avec X la concentration en réactant, k le maximum de produit formé et a la pente à l'origine.

Dans notre modèle, les variables prennent un autre sens avec X le nombre cumulé de cellules ingérées, k le nombre total de cellules dans le milieu et a le taux d'ingestion du copépode. Nous pouvons donc identifier tous les paramètres de cette équation comme des caractéristiques statiques (k) ou dynamiques (X et a) de notre modèle. Nous sommes donc en présence d'un calcul émergent au sens de S. Forrest [For90]. Comme nous l'avons dit plus haut, le taux d'ingestion est une caractéristique importante des modèles de simulations de la dynamique de population des copépodes. En mettant en évidence l'équivalence en terme de trace de simulation entre notre modèle agent et un modèle mathématique classique, nous commençons à voir comment nous allons coupler ces deux approches. Comme si nous faisions un paramétrage classique à partir d'expériences réelles, nous allons construire une fonction basée sur les traces de simulations du modèle agent.
D'un côté, nous avons un modèle discret stochastique (le SMA) et de l'autre un modèle continu déterministe (équation 5.2). Le passage de l'un à l'autre se fait par ajustement d'une fonction à un ensemble de valeurs moyennes. Cette « moyennisation » du comportement du modèle agent est indispensable pour trouver une équivalence entre les traces de simulations des deux modèles. Comme le montre la figure 5.4, lorsqu'on fait un zoom sur une courbe particulière ayant servi à dessiner les courbes de la figure 5.2(a), nous observons une croissance en escalier du nombre cumulé de cellules consommées. Ceci est le reflet du comportement individuel du copépode, alternant entre capture et recherche de proies.

Dans ce paragraphe, nous avons mis en évidence que le type de distribution a une influence sur le taux d'ingestion du copépode. Nous avons également montré qu'il était possible de représenter le comportement global du système à l'aide d'une équation, à l'instar des expérimentateurs qui construisent un modèle à partir d'expériences réelles. Dans ce qui suit, nous voulons aller un peu plus loin en montrant qu'une approche agent permet de retrouver des fonctions classiques issues de l'écologie théorique et apporte un nouveau moyen d'expression pour les modélisateurs de cette discipline. De plus, le couplage entre un modèle mathématique et le modèle agent que nous allons mettre au point augmente encore ces possibilités.

5.2 Construction et paramétrage d'une réponse fonctionnelle à l'aide du modèle agent

Dans un article récent, nous avons montré qu'il est possible de paramétrer une équation différentielle ordinaire à l'aide d'un modèle d'agents réactifs [DRP03]. Dans ce même travail, nous avons proposé une méthode opérationnelle pour réaliser un tel couplage de modèles. Nous avons également utilisé cette méthode pour le paramétrage de la réponse fonctionnelle d'un modèle mathématique proies-prédateurs. Dans ce qui suit, nous présentons et approfondissons ce

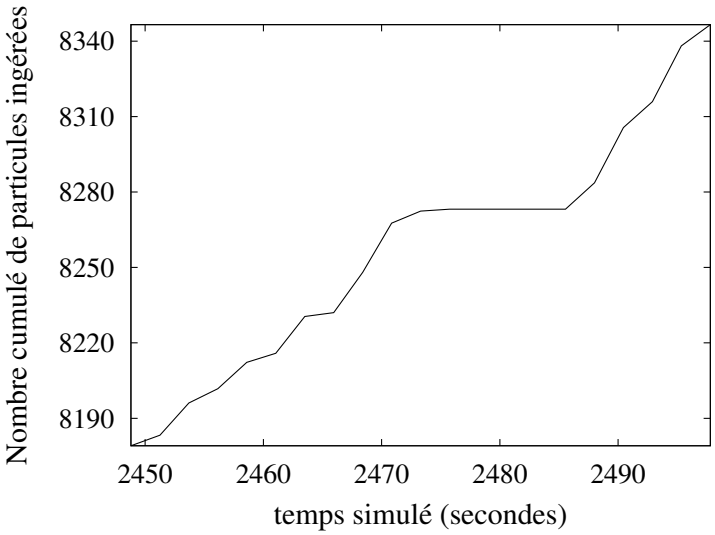


FIG. 5.4 – Évolution du nombre de cellules ingérées par le copépode au cours du temps pour une distribution en paquets (ou hétérogène) des cellules. Cette courbe est un agrandissement d'une portion de courbe particulière ayant servi au calcul des courbes moyennes de la figure 5.2(a).

travail afin d'étudier la forme de cette réponse et de choisir la formulation la mieux adaptée pour décrire les résultats de simulations effectuées à l'aide du modèle d'agents réactifs de copépodes. Nous voulons montrer que l'utilisation conjointe des deux types de modèles peut être un outil efficace d'aide à l'étude théorique de la dynamique des systèmes au travers d'un exemple en écologie marine.

5.2.1 Contexte théorique

Dans un système proie-prédateur, la réponse fonctionnelle correspond à la forme de la fonction trophique des prédateurs *i.e.* l'évolution de la quantité de proies mangées par prédateur et par unité de temps (ou taux d'ingestion) en fonction de paramètres du milieu. Traditionnellement, cette fonction $G(.)$ est considérée comme dépendante uniquement de la quantité de proies. Dans la version originale du modèle de lotka-volterra de 1925 [Lot25][Vol26], elle n'est rien d'autre que la loi d'action de masse en chimie et s'écrit $G(N) = aN$ (où N est la quantité de proies et a un coefficient de proportionnalité). En 1959, C.S. Holling postule que cette réponse fonctionnelle dépend de la concentration de la ressource [Hol59]. Dans ce contexte, la fonction trophique est dite « densité-dépendante » et s'écrit (équation 5.3) :

$$G(N) = \frac{\alpha N}{\beta + N} \tag{5.3}$$

où : N est la concentration en proies, α le taux d'ingestion maximum et β la constante de demi-saturation, c'est-à-dire la quantité de proies pour laquelle le taux d'ingestion est égal à $\frac{1}{2}$ de α.

L'équation 5.3 est également appelée « équation du disque de Holling » ou « équation de type II ». Holling a également suggéré que la forme de cette fonction dépend de la nature même des prédateurs et des proies. Ainsi, en élevant les termes β et N au carré, Real [Rea77] propose une

variante de l'équation de Holling (même si Holling l'avait suggérée auparavant) rendant mieux compte de la possibilité pour les prédateurs d'augmenter rapidement leur efficacité de prédation (*i.e.* leur taux d'ingestion) en fonction de la concentration en proie. La valeur du terme puissance est alors une caractéristique de l'espèce. L'équation de Real (de type III) est construite sur des bases théoriques et nous intéresse un peu plus loin.

En 1989, P. Arditi propose une équation qui prend en compte le nombre de prédateurs. Dans l'un de ses articles [AG89], il montre qu'une telle réponse ajuste mieux les données de nombreuses campagnes de mesures *in situ* et expériences de laboratoires. La forme de cette réponse est donnée par l'équation 5.4 :

$$G(N,P) = \frac{\alpha N}{\beta N + \gamma P} \tag{5.4}$$

avec N la quantité de proies, P la quantité de prédateurs α le taux d'ingestion maximum, β le coefficient de demi-saturation et γ un coefficient de dépendance au nombre de prédateurs dans le milieu.

Cette réponse fonctionnelle est dite ratio-dépendante. Arditi suggère par la suite qu'elle est liée au niveau d'hétérogénéité de la distribution des proies et de prédateurs dans le milieu. En 1992, il réalise une expérience et vérifie son hypothèse [AS92].

D'autres formes ont été proposées pour cette équation, notamment avec les équations de type III introduisant une valeur seuil de la concentration des proies en-dessous de laquelle le taux d'ingestion est nul [CGW00]. Néanmoins, quelle que soit la loi considérée, deux hypothèses fortes sont émises :

- les proies et les prédateurs sont supposés se rencontrer de façon aléatoire,
- les distributions spatiales des proies et des prédateurs sont homogènes. Par la suite, nous qualifierons cette dernière hypothèse « d'hypothèse de Holling », même s'il s'agit de l'hypothèse ergodique classique de la physique.

La réponse fonctionnelle reflète un ensemble de processus qui existent à l'échelle individuelle. Il s'agit de caractéristiques telles que le temps de manipulation des proies par les prédateurs, la recherche des proies ou encore le mode de perception. Ces caractéristiques sont dépendantes de la stratégie de déplacement des prédateurs par rapport aux proies, de leur vitesse relative, de la nature de la distribution des proies, etc. La réponse fonctionnelle intègre tous ces processus à l'échelle de la population, ce qui implique qu'elle représente un individu moyen. Le modèle d'agents réactifs individualise les entités du système ; ainsi il est capable de représenter la variabilité individuelle. Il sera donc nécessaire de moyenner les résultats de simulations sur la population d'agents pour pouvoir ajuster une réponse fonctionnelle représentant la population. Nous avons identifié deux classes de réponses fonctionnelles. Une première classe dite densité dépendante et une deuxième dite ratio-dépendante. Dans ce qui suit, nous proposons de réaliser des expériences virtuelles à l'aide de notre modèle d'agents réactifs situés de copépode. Ces expériences nous permettrons de « valider qualitativement notre modèle » en vérifiant qu'il reproduit bien une réponse fonctionnelle densité dépendante en milieu homogène et ratio-dépendante en milieu hétérogène. Comme nous maîtrisons tous les paramètre de l'expérience, et notamment le niveau d'hétérogénéité des distributions, nous pouvons étudier par la suite l'impact de cette hétérogénéité sur la dynamique d'un système proies-prédateurs classique.

5.2.2 Méthodes

Nous nous intéressons ici au paramétrage de la réponse fonctionnelle à l'aide des résultats de simulations du modèle d'agents réactifs. Pour cela, nous plaçons notre modèle dans des conditions similaires à celles déterminées par la fonction de Holling (*i.e.* la distribution des proies et des prédateurs est homogène) puis dans les conditions énoncées par Arditi (*i.e.* la distribution des proies et des prédateurs est hétérogène). Une des hypothèses fortes est que la rencontre entre les prédateurs et les proies est aléatoire. Cette hypothèse n'est pas vérifiée dans notre modèle où les copépodes adoptent un comportement de chasse. C'est en fait ce qui fait l'intérêt de notre approche. En effet dans la nature, la rencontre entre les proies et les prédateurs n'est pas totalement aléatoire et pourtant cette hypothèse est très souvent faite dans les approches mathématiques, même individus-centrées, de par la nature même des équations.

À partir des deux classes de réponses fonctionnelles identifiées (densité-dépendante et ratio dépendante), nous fixons deux classes d'expériences basées sur le niveau d'hétérogénéité de la distribution des proies :

- expériences en milieu homogène,
- expériences en milieu hétérogène.

Dans les deux classes d'expériences, nous faisons varier le nombre de prédateurs afin de vérifier les hypothèses sous-jacentes aux formulations de Holling et d'Arditi. Étant donnée la nature stochastique du modèle agent, nous effectuons 30 simulations indépendantes pour chaque expérience et considérons la valeur moyenne du taux d'ingestion. Les valeurs des différents paramètres du modèle agents sont identiques à celles utilisées pour l'étude de sensibilité au choix de représentation. Nous considérons donc un espace cubique de $64.10^3 mm^3$ et des rebonds aléatoires. De plus, nous considérons le modèle physiologique décrit en annexe B. Ce modèle contrôle l'appétit du copépode. Chaque individu est initialisé avec une position aléatoire est une quantité fixe de cellules de phytoplancton dans l'estomac (130 cellules). Cette quantité est inférieure au seuil de satiété (165 cellules), le copépode a donc faim en début de simulation.

Calcul du taux d'ingestion par simulation

Pour calculer le taux d'ingestion, nous fixons d'abord le nombre de proies et de prédateurs puis nous mesurons le nombre de proies mangées par prédateur pendant un temps t. Malgré l'apparente simplicité d'une telle méthode, une difficulté demeure. La quantité $G(N)$ mesure un taux d'ingestion instantané pour une valeur fixe de N, alors que dans une expérience d'une certaine durée t, le nombre de proies diminue au fur et à mesure qu'elles sont consommées par les prédateurs. Pour répondre à cette difficulté, Arditi [AS92] propose une équation pour déterminer le taux d'ingestion instantané à partir d'expériences directes. Il pose :

$$g(N, P, T) = \frac{\Delta N}{PT} \tag{5.5}$$

avec ΔN la variation de la quantité de proies décrit par l'équation implicite suivante :

$$\Delta N = N[1 - e^{-\Omega P^{1-m} T + \Omega P^{-m} t_h \Delta N}] \tag{5.6}$$

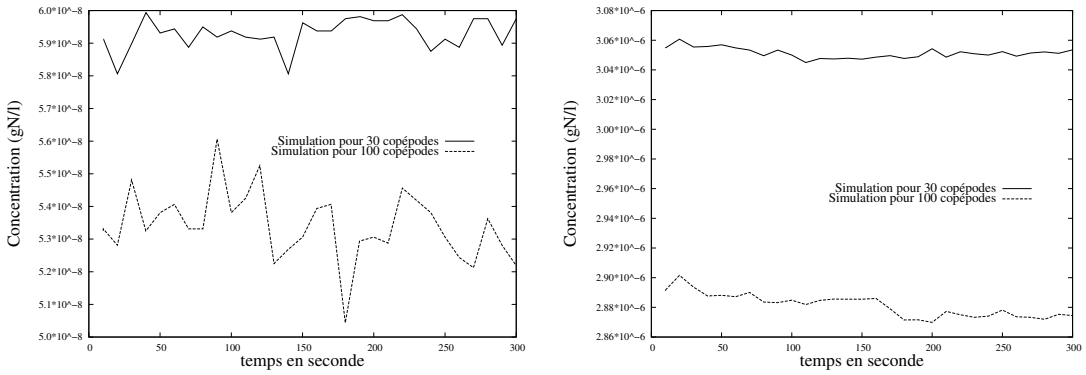
où : N est la quantité de proies en début d'expérience,
P est la quantité de prédateurs (constante),
T est la durée de l'expérience,

t_h est le temps de manipulation d'une proie par un prédateur,
Ω est un paramètre d'efficacité de recherche,
m quantifie la dépendance de la réponse fonctionnelle par rapport au nombre de prédateurs.

Une difficulté demeure avec l'équation 5.5. En effet, les paramètres Ω et m doivent être connus pour pouvoir évaluer $g(N, P, T)$. Ne connaissant pas ces paramètres, il faudrait résoudre cette équation implicite pour les évaluer, ce qui est impossible. Nous nous sommes donc dirigés vers une méthode informatique. L'avantage par rapport aux modèles mathématiques est la possibilité d'introduire des artefacts dans une simulation, c'est-à-dire des effets non désirés[84]. Le terme rend bien compte d'une condition « artificielle » qui produit un effet de bord. Nous conservons donc ce terme. L'artefact de simulation introduit assure une concentration constante de proies, comme suit :

– chaque cellule de phytoplancton consommée disparaît pendant un temps δt, puis réapparaît au même endroit. δt doit être assez grand pour que le copépode s'éloigne suffisamment de la cellule afin de ne pas la percevoir aussitôt après l'avoir consommée. Une valeur de 10 secondes pour δt s'est avérée être suffisante.

L'artefact de simulation proposé ne perturbe pas le comportement du copépode, celui-ci n'ayant pas de mémoire des évènements. Ainsi, nous mesurons le taux d'ingestion du copépode en utilisant la formule simple (équation 5.5) où ΔN correspond dans notre simulation au nombre de cellules consommées par l'ensemble des individus. En utilisant cet artefact, la concentration va osciller autour d'une valeur moyenne légèrement inférieure à la concentration initiale en proies du milieu. La figure 5.5 représente graphiquement des résultats d'expériences montrant l'évolution de la concentration dans le milieu au cours du temps en début de simulation.



(a) concentration initiale = $6,25.10^{-8}$ gN/l (*i.e.* gramme d'azote par litre) Hétérogénéité=$2,8$

(b) concentration initiale = $3,12.10^{-6}$ gN/l (*i.e.* gramme d'azote par litre) Hétérogénéité=2

FIG. 5.5 – Évolution de la concentration en proies dans le milieu au cours du temps au début de la simulation. En (a) les oscillations sont importantes. En (b) les oscillations sont amorties. Voir le texte pour les commentaires et le paragraphe 5.2.2 pour le sens des mesures d'hétérogénéités.

Nous avons délibérément choisi des concentrations faibles de proies pour réaliser ces expériences. En effet, c'est aux faibles concentrations que l'impact de l'artefact de simulation est le

[84]Ces effets maintenant volontaires ne sont plus des artefacts *stricto sensu*.

plus grand. La figure 5.5(a) montre des oscillations de la concentration en proies durant les trois cents premières secondes de la simulation. Ces oscillations reflètent les effets conjugués de la consommation des proies et de leur réapparition dix secondes plus tard. L'amplitude maximale de ces variations correspond à $9,5\%$ de la concentration initiale. La figure 5.5(b) montre des oscillations très amorties avec une amplitude maximale qui correspond à $0,1\%$ de la concentration initiale. La comparaison de ces deux figures nous montre que l'effet de l'artefact de simulation diminue avec l'augmentation de la concentration en proies. Au regard de ces figures, deux autres conclusions sont importantes pour la suite :

1. nous constatons que l'augmentation du nombre de copépodes fait diminuer la concentration moyenne en proies dans le milieu pour une même concentration initiale de proies. Ceci est dû à l'augmentation de l'intensité de prédation avec l'augmentation du nombre de copépodes,
2. nous constatons également que le système s'équilibre autour d'une concentration moyenne très rapidement (une centaine de secondes). Cet équilibre dépend du temps de réapparition des proies.

Cette deuxième remarque implique que le temps t nécessaire à simuler pour calculer le taux d'ingestion moyen par copépode doit être supérieur à 100 secondes. Dans la pratique, nous avons choisi $t = 150s$ simulées. Ainsi le temps T pour le calcul du taux d'ingestion est égal à : $T = t - 100$. La première remarque implique que la concentration réelle en proies dépend du nombre de prédateurs, ce dont il faudra tenir compte pour les simulations de la réponse fonctionnelle (*cf.* paragraphe 5.2.3).
Nous avons discriminé deux classes de réponses fonctionnelles sur la base d'un milieu homogène ou hétérogène. Nous allons donc réaliser des expériences où le degré d'hétérogénéité varie. Une des difficultés est de maintenir constante la mesure de l'hétérogénéité de la distribution des proies quelle que soit leur concentration. Dans ce qui suit, nous présentons nos algorithmes de distribution des proies dans l'espace ainsi que la méthode de calcul de l'hétérogénéité.

Techniques de distribution des cellules de phytoplancton et mesure de l'hétérogénéité

Le problème ici est la génération de la distribution des cellules de phytoplancton (ou particules dans ce qui suit) dans un espace en trois dimensions et la mesure de l'hétérogénéité de cette distribution. Pour ce qui est de la distribution, le problème est de tirer les trois coordonnées (x, y, z) des particules représentées par un point. Il existe de nombreuses façons de faire suivant la loi de distribution utilisée. Une des contraintes est que la concentration des particules doit varier. Nous pourrions penser que pour obtenir une distribution uniforme (ou homogène, *i.e.* les cellules occupent tout l'espace et les distances inter-cellules tendent vers la même valeur), une loi de distribution aléatoire uniforme pourrait convenir. Ce n'est pas le cas. En effet, l'uniformité de cette loi est valide (du point de vue spatial) pour un très grand nombre de tirages. Ainsi, aux faibles concentrations, la loi uniforme fait apparaître « des trous » dans la distribution, augmentant ainsi l'hétérogénéité du milieu. De plus, elle n'offre aucun degré de liberté, ce qui nous empêche de jouer sur la distribution.
Pour générer les distributions de cellules, nous avons choisi deux des trois techniques décrites au paragraphe 5.1.2, à savoir :

1. distribution régulière,
2. distribution hétérogène (ou en paquets). En jouant sur l'écart-type de la loi normale de

distribution des cellules autour des paquets, nous jouons sur « l'étalement » de ceux-ci et donc sur l'hétérogénéité de la distribution.

Nous voulons maintenant définir une mesure du degré d'hétérogénéité d'une distribution. Cela revient à définir une grandeur allant d'une distribution régulière à une distribution très hétérogène, où les cellules sont réparties dans des paquets très éloignés les uns des autres.
Il y a une façon simple de mesurer l'hétérogénéité d'une distribution qui est souvent utilisée en écologie [FPV95]. Adaptée à notre cas d'étude, cette méthode doit :

- diviser l'espace cubique en petits cubes de $1mm$ de côté correspondant à la distance de capture du copépode. En-dessous de cette distance le copépode ne perçoit pas l'hétérogénéité puisqu'il n'effectue pas de déplacement pour capturer ces proies,
- compter le nombre de cellules par petit cube,
- calculer l'écart-type σ et la moyenne m du nombre de cellules par petit cube.

Plus l'écart-type σ est grand par rapport à la moyenne m, plus la valeur de l'hétérogénéité augmente (les paquets sont de plus en plus denses). Un écart-type faible devant la moyenne correspond à une distribution homogène ou uniforme. Où est la limite entre hétérogénéité et homogénéité ? Considérant notre définition, et en toute rigueur, seul un écart-type nul ($\sigma = 0$) nous assure d'une parfaite homogénéité. Dans ce qui suit, nous appelons degré d'hétérogénéité la valeur ξ telle que $\xi = \frac{\sigma}{m}$. Nous posons qu'avec $\xi < 1$ (*i.e.* $\sigma < m$) la distribution est homogène.

La technique de distribution régulière assure une valeur de ξ constante et $\xi << 1$ à partir d'une certaine concentration. En effet, pour les faibles concentrations, la distance inter-particules est importante et induit une hétérogénéité de la distribution. Dans les faits, le degré d'hétérogénéité ξ du milieu est proche de 0 à partir d'une concentration de $2.10^{-6} gN.l^{-1}$. Cette concentration correspond à une cellule de phytoplancton par cube de $1mm$ de côté (correspondant à la discrétisation de l'espace pour la mesure de ξ). En-dessous de cette concentration, ξ augmente. Ceci est dû aux « vides » entre les particules. Nous verrons par la suite les conséquences d'une telle distribution sur le taux d'ingestion (figure 5.7).

Dans le cas de la technique de distribution hétérogène, nous avons le moyen de faire varier ξ. Nous effectuons alors une série d'expériences de mesure du taux d'ingestion à concentration constante en faisant varier ξ et le nombre de copépodes. La figure 5.6 nous montre deux exemples de courbes obtenues par simulation.

Nous pouvons découper les courbes de la figure 5.6 en trois parties :

1. $0,5 < \xi < 1$:
   les valeurs de ξ considérées ici correspondent à une distribution homogène. Le taux d'ingestion simulé oscille avec une tendance à croître,
2. $1 \leq \xi \leq \{2; 2,5\}$:
   la distribution devient de plus en plus hétérogène et le taux d'ingestion croît jusqu'à un maximum dépendant du nombre de prédateurs,
3. $\xi > \{2; 2,5\}$:
   l'augmentation de ξ a ici l'effet opposé. Le taux d'ingestion est décroissant avec l'augmentation de l'hétérogénéité.

La forme générale en cloche des courbes de la figure 5.6 nous montre que l'hétérogénéité n'a pas le même effet sur le taux d'ingestion selon son importance. Lorsque ξ augmente jusqu'à $2,5$, l'hétérogénéité est bénéfique aux prédateurs, le taux d'ingestion augmente. À partir de $\xi = 2,5$, l'augmentation de l'hétérogénéité devient défavorable aux prédateurs. Nous obtenons la même

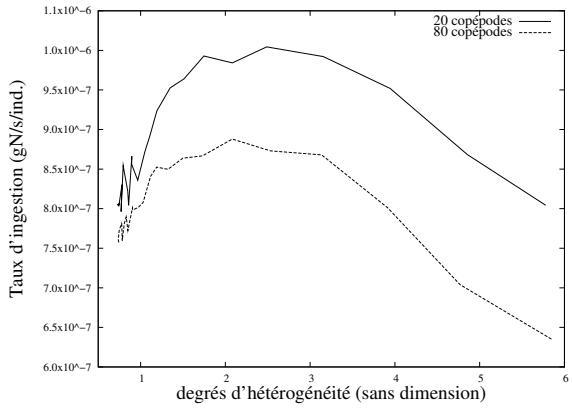


FIG. 5.6 – Variation du taux d'ingestion moyen par individu en fonction du degré d'hétérogénéité de la distribution des proies (ξ). Chaque courbe correspond à la moyenne de 30 simulations effectuées pour 20 copépodes (trait plein) et 80 copépodes (pointillés) dans le milieu, pour une concentration constante de proies de $6,25.10^{-6} gN/l$.

tendance générale quelle que soit la concentration en proies ou prédateurs. En augmentant le nombre de prédateurs, le taux d'ingestion moyen par prédateur diminue pour un même degré d'hétérogénéité. Il y a donc compétition intra-prédateurs pour la ressource. Nous aurions pu penser que l'artefact de simulation introduit au paragraphe 5.2.2 aurait atténué, voire éliminé cette compétition, mais ce n'est pas le cas.
La première partie de la courbe ($\xi < 1$) montre qu'en milieu homogène, la variabilité du taux d'ingestion augmente, mais les deux courbes sont ici plus proches l'une de l'autre que pour des valeurs de ξ élevées, où l'écart important entre les courbes ne semble pas se réduire. Cette tendance au rapprochement nous fait penser qu'en milieu homogène, le taux d'ingestion n'est pas dépendant du nombre de prédateurs (hypothèse de Holling), hypothèse que nous vérifions un peu plus bas. À ce niveau, nous pouvons confirmer l'hypothèse d'Arditi d'une relation entre l'hétérogénéité de la distribution des proies et la forme de la réponse fonctionnelle des prédateurs. Dans ce qui suit, nous allons de nouveau confirmer l'hypothèse d'Arditi en simulant la fonction trophique des prédateurs. Nous verrons également comment paramétrer cette fonction à partir des résultats de simulations.

5.2.3 Simulations et construction de la réponse fonctionnelle

Simuler la réponse fonctionnelle revient à reproduire *in silico* les expériences classiques menées d'habitude en laboratoire. Ces expériences consistent à mesurer le taux d'ingestion moyen instantané des prédateurs pour différentes concentrations de proies et de prédateurs. La forme de la réponse fonctionnelle est donnée par la représentation graphique de la valeur du taux d'ingestion instantané calculée (en utilisant la méthode donnée au paragraphe 5.2.2) en fonction de la concentration en proies dans le milieu.

Analyse qualitative de la réponse

Au regard des deux grandes catégories de réponses fonctionnelles identifiées précédemment, nous proposons deux séries d'expériences. Une série avec une distribution homogène des proies et une série avec une distribution hétérogène. Notre modèle étant stochastique, nous répétons

chaque simulation trente fois, ce qui, au regard du théorème central limite et sous l'hypothèse d'une distribution normale des résultats, garantit que la moyenne est un bon estimateur. La figure 5.7 nous montre la réponse fonctionnelle obtenue avec une distribution homogène des proies.

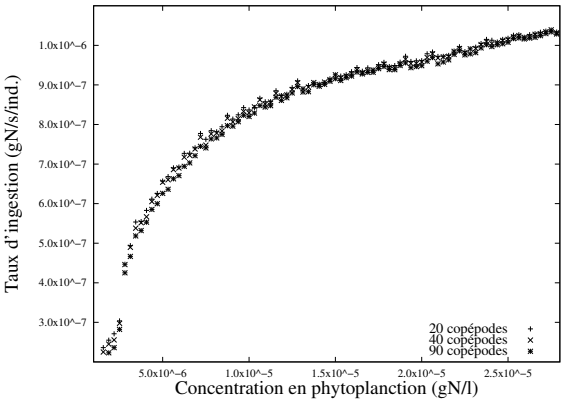


FIG. 5.7 – Réponses fonctionnelles simulées par le modèle agents pour différentes quantités de prédateurs. La distribution des cellules est régulière. Chaque point correspond à la moyenne de 30 simulations. Le taux d'ingestion reste identique avec l'augmentation du nombre de prédateurs. Cette expérience confirme l'hypothèse d'une réponse fonctionnelle non ratio-dépendante en milieu homogène.

Nous constatons que les courbes se superposent quel que soit le nombre de prédateurs. Ceci confirme encore une fois l'hypothèse de Holling. Au regard du côté artificiel de la technique de distribution utilisée ici, nous pouvons nous demander si ces conditions existent dans la nature. La réponse est évidemment non. La distribution des proies n'est jamais uniforme et, comme nous le voyons avec la figure 5.8, ceci a des conséquences sur la forme de la réponse fonctionnelle [85].

La figure 5.8 nous montre la réponse fonctionnelle obtenue avec une distribution hétérogène des proies. Nous constatons la diminution de la vitesse d'accroissement du taux d'ingestion avec l'augmentation du nombre de prédateurs. Ceci vérifie l'hypothèse d'Arditi d'une réponse fonctionnelle ratio-dépendante en milieu hétérogène.

Détermination et paramétrage de la réponse fonctionnelle

Comment ajuster une réponse fonctionnelle à nos résultats de simulations ? Comment choisir une réponse fonctionnelle représentative des hypothèses posées et des caractéristiques de notre modèle ? Effectivement, il est toujours possible, dans un cas comme le nôtre, d'ajuster un modèle « au mieux », en utilisant le critère des moindres carrés par exemple. Seulement, la réponse choisie doit avoir « du sens » dans un contexte particulier, de conditions expérimentales et dans notre cas, de la nature du modèle produisant cette réponse.

Notre modélisation mécanique de la relation proie-prédateur implique « un seuil d'ingestion limite » rapidement atteint par le modèle. En effet, le temps de manipulation de $5.10^{-2}s$ implique

[85] Comme nous l'avons expliqué au paragraphe 5.2.2, nous ne prenons en compte que la partie de la courbe où la concentration en proies est supérieure à $2.10^{-6}gN.l^{-1}$. En-dessous de cette valeur, la distribution n'est pas homogène.

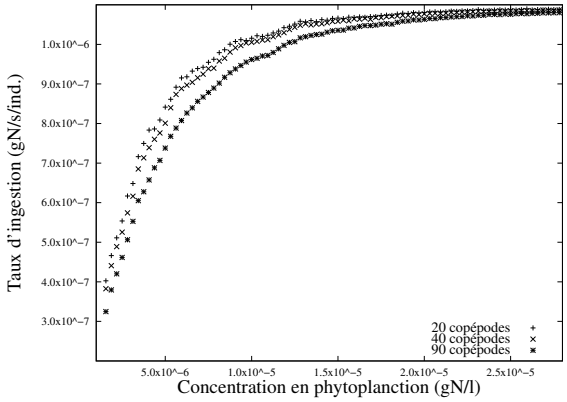


FIG. 5.8 – Réponses fonctionnelles simulées par le modèle d'agents réactifs pour différentes quantités de prédateurs en milieu hétérogène ($\xi = 1,1$). Chaque point correspond à la moyenne de 30 simulations. La diminution de la vitesse d'accroissement du taux d'ingestion avec l'augmentation du nombre de prédateurs dans le milieu vérifie l'hypothèse d'Arditi d'une réponse fonctionnelle ratio-dépendante en milieu hétérogène.

qu'au maximum, le taux d'ingestion est de 20 $cell.s^{-1}$. Dans un milieu très concentré, dans lequel le copépode effectue très peu de déplacements pour rechercher sa nourriture et sans restriction liée à son appétit, il atteindra rapidement cette limite. De plus, au cours de l'analyse des résultats de simulations, nous nous sommes rendus compte que lorsque le plateau (le taux maximum d'ingestion) était atteint, il n'y avait pas de croissance infinie, même très faible, du taux d'ingestion. C'est pourquoi nous avons choisi la réponse fonctionnelle énoncée par Real [Rea77] qui rend mieux compte de « l'effet plateau ». Cette réponse apparaît comme un cas particulier d'une forme plus générale de l'équation de Holling, elle s'écrit :

$$g(N) = \frac{\alpha N^2}{\beta^2 + N^2} \tag{5.7}$$

où : N est la concentration en proies, α le taux d'ingestion maximum et β la constante de demi-saturation.

L'équation 5.7 suppose les mêmes hypothèses d'homogénéité des distributions que l'équation de Holling. Nous avons vu que le modèle agent reproduisait qualitativement une réponse fonctionnelle de type ratio-dépendant en milieu hétérogène. Nous allons donc commencer par ajuster la réponse fonctionnelle de Real (équation 5.7) aux valeurs simulées, puis nous étudierons l'évolution du paramètre β pour construire une réponse ratio-dépendante à partir de cette équation.

Pour ajuster la réponse fonctionnelle aux données simulées, nous commençons par étudier statistiquement les données. Nous considérons chaque ensemble de simulations du taux d'ingestion pour une concentration de proies donnée comme un échantillon et, à l'aide d'une analyse de variance, nous étudions l'effet de la concentration sur les résultats (tableau 5.3).

À partir des résultats du tableau 5.3, nous pouvons conclure que la concentration en proies a un effet significatif sur le taux d'ingestion. Ces résultats nous permettent, sous l'hypothèse d'une distribution normale des résultats par échantillon, de considérer la moyenne du taux d'ingestion comme représentatif de celui de la population de prédateurs.
L'ajustement de la réponse fonctionnelle se fait donc sur les moyennes des trente simulations

TAB. 5.3 – Analyse de variance des résultats de simulations pour trois niveaux d'hétérogénéité. Ces résultats nous amènent à rejeter l'hypothèse d'une égalité des moyennes avec un niveau de confiance de 99%

hétérogénéité ξ	1	1,5	2
nombre total de mesures	570	600	600
nombre d'échantillons	19	20	20
moyenne totale	1.10^{-06}	1.10^{-06}	1.10^{-06}
variance due à la concentration des proies	$1,5.10^{-14}$	$2,5.10^{-14}$	$1,5.10^{-14}$
variance résiduelle	$1,4.10^{-16}$	$1,8.10^{-16}$	$1,6.10^{-16}$
Valeur de la variable de Fisher-Snedecor	3247	4254	2884

pour chaque concentration en proies donnée. Pour cela, nous utilisons l'algorithme de Marquardt-Levenberg [Mar63]. Cet algorithme consiste en une régression non linéaire basée sur le critère des moindres carrés. La figure 5.9 présente les résultats d'un tel ajustement pour les trois valeurs d'hétérogénéité considérées dans le tableau 5.3

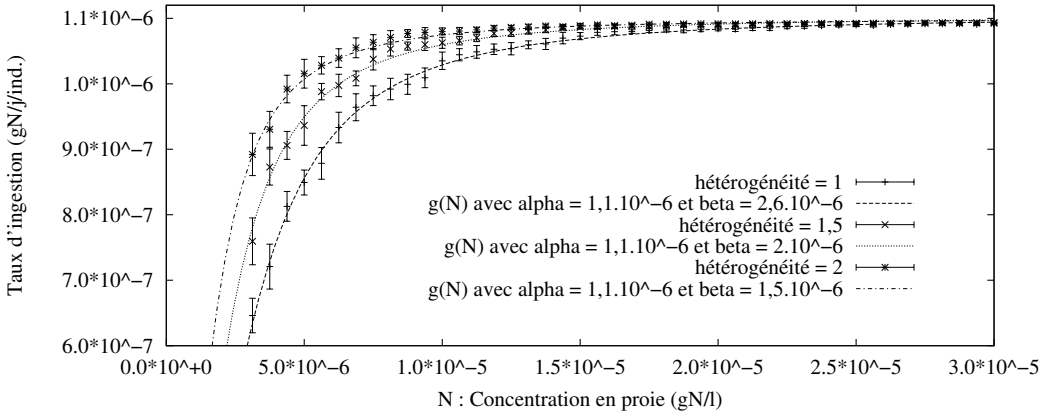


FIG. 5.9 – Ajustement de la réponse fonctionnelle $g(N)$ de Real (équation 5.7) aux données simulées pour trois valeurs d'hétérogénéité et un nombre constant de 30 prédateurs. Les segments verticaux représentent l'écart-type et les points la moyenne de 30 simulations. Voir le texte pour les commentaires.

Nous avons effectué des tests statistiques d'ajustements classiques (test de Kolmogorov-Smirnov) adaptés à la comparaison de distributions théoriques et empiriques de variables continues. Dans tous les cas, nous concluons à un très bon accord entre la loi de Real et les données simulées. La figure 5.9 montre que le coefficient β de l'équation de Real n'est pas constant, mais varie selon le degré d'hétérogénéité de la distribution des proies. Nous avons vu au paragraphe précédent que nos simulations reproduisaient également la propriété de densité dépendance énoncée par Arditi. Le modèle de Real utilisé pour ajuster la réponse fonctionnelle aux données si-

mulées ne fait pas intervenir cette propriété. Aussi, la variation du coefficient β est due à l'effet de l'hétérogénéité des distributions des proies et des prédateurs sur la réponse fonctionnelle. Nous proposons donc de représenter la variation de β en fonction du nombre de prédateurs et du niveau d'hétérogénéité. Ceci doit nous permettre de mieux comprendre la relation complexe existant entre ces deux variables dans l'ajustement de la réponse fonctionnelle. Pour cela, nous effectuons un ensemble de simulations pour différentes concentrations de prédateurs et différents niveaux d'hétérogénéité. La figure 5.10 représente les isolignes (lignes de même valeur) des valeurs de β dans le plan des prédateurs et de l'hétérogénéité. Ainsi, nous avons un aperçu de la variation de β en fonction de ces deux variables.

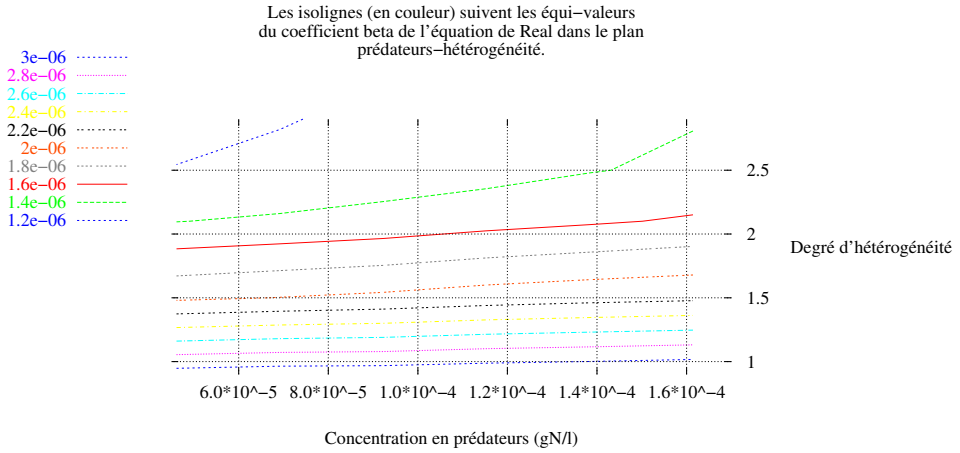
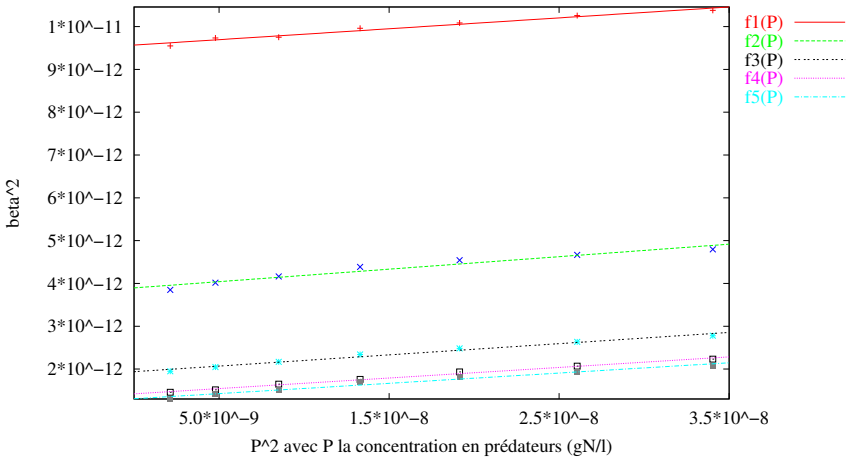


FIG. 5.10 – Isolignes des valeurs du paramètre β de l'équation 5.7. Les isolignes montrent l'évolution de la topographie attachée à β. Voir le texte pour les commentaires.

La figure 5.10 nous montre l'accroissement des valeurs de β avec la diminution de l'hétérogénéité. Cet accroissement est d'autant plus rapide que l'hétérogénéité est faible, ce qui est en accord avec la figure 5.6 de la page 141 où le taux d'ingestion augmente avec le degré d'hétérogénéité jusqu'à un plateau. Si nous observons cette figure comme des courbes de niveaux, nous pouvons dire que la pente est principalement dans le sens d'une hétérogénéité croissante. Néanmoins nous observons une légère pente dans le sens d'une concentration en prédateurs décroissante. Nous remarquons également que plus l'hétérogénéité diminue, plus les isolignes tendent à être parallèles à l'axe des abscisses. Ceci montre que le nombre de prédateurs est bien une variable qui doit être utilisée pour relier le niveau d'hétérogénéité des proies et la réponse fonctionnelle. En effet, si le nombre de prédateurs n'avait aucune influence sur le taux d'ingestion, alors les isolignes seraient parallèles à l'axe des abscisses.
La figure 5.10 fait néanmoins ressortir que la principale cause de variation de β est le degré d'hétérogénéité. À notre connaissance, les formes de réponse fonctionnelle ne font jamais intervenir ce paramètre lorsqu'elles sont utilisées dans des modèles de dynamique de populations. Il y a une raison pragmatique à cet état de fait. Il est en effet extrêmement difficile de mesurer l'hétérogénéité d'une distribution *in vivo*. De plus, les modèles de dynamique de population ne prennent pas directement en compte les processus à l'échelle individuelle. Nous devons donc trouver un moyen simple d'exprimer la relation complexe qui existe entre la densité de prédateurs, le degré

d'hétérogénéité et la variation de β. Nous pourrions ajuster un modèle sur la surface décrite par β (figure 5.10) mais, encore une fois, le degré d'hétérogénéité n'est pas une variable des modèles de dynamique de populations et nous ne pourrions pas travailler dans un contexte classique. De plus, notre mesure de l'hétérogénéité est très relative à notre modélisation par agents (la discrétisation de l'espace pour son calcul est fonction de la distance de capture du copépode), il serait nécessaire d'avoir une mesure plus générale. Nous reviendrons sur ce point dans la discussion.

Pour trouver une relation simple entre le degré d'hétérogénéité de la distribution des proies, le nombre de prédateurs et la réponse fonctionnelle, nous nous intéressons à la relation entre β (équation 5.7 page 143) et la densité de prédateurs P à un niveau d'hétérogénéité donné. Pour cela, nous ajustons l'équation 5.7 à un ensemble de simulations effectuées avec des concentrations différentes en prédateurs à cinq niveaux d'hétérogénéité différents. Ainsi, nous pouvons établir une relation de la forme $\beta = f(P)$ pour chaque niveau d'hétérogénéité. La figure 5.11 nous montre cette relation.



$$
\begin{aligned}
f1(P^2): &\quad \gamma^2 = 5.10^{-3} \quad \mu^2 = 9.10^{-12} \quad pour\ \xi = 0,9\\
f2(P^2): &\quad \gamma^2 = 5,3.10^{-3} \quad \mu^2 = 3.61.10^{-12}\ pour\ \xi = 1,5\\
f3(P^2): &\quad \gamma^2 = 5.10^{-3} \quad \mu^2 = 1,69.10^{-12}\ pour\ \xi = 1,9\\
f4(P^2): &\quad \gamma^2 = 4,8.10^{-3} \quad \mu^2 = 1,44.10^{-12}\ pour\ \xi = 2,5\\
f5(P^2): &\quad \gamma^2 = 4,8.10^{-3} \quad \mu^2 = 1,21.10^{-12}\ pour\ \xi = 2,9
\end{aligned}
$$

FIG. 5.11 – Relation entre la concentration en prédateurs P^2 et β^2. Les différents types de points indiquent les valeurs simulées. Les droites de régression linéaire sont de la forme $f(P^2) = \beta^2 = \gamma^2 P^2 + \mu^2$ avec un coefficient de corrélation toujours supérieur à $0,9$

La figure 5.11 montre qu'il existe une relation linéaire entre β^2 et P^2. Nous notons cette relation comme suit :

$$\beta^2 = \gamma^2 P^2 + \mu^2$$

Les coefficients γ et μ sont caractéristiques d'un niveau d'hétérogénéité donné. Nous notons que γ varie peu alors que μ varie du simple au triple. Comme nous l'avons vu avec la figure 5.10,

il existe une relation complexe entre la densité de prédateurs et le niveau d'hétérogénéité. Néanmoins, nous pouvons approximer cette relation par une équation de droite, ce qui permet de donner facilement du sens aux paramètres.
Ainsi, la formulation de Real (équation 5.7), qui ne rend pas compte de l'effet ratio-dépendant identifié plus haut peut être réécrite pour rendre compte de cet effet. Pour cela, nous introduisons P (la densité de prédateurs) dans l'équation de Real en remplaçant le terme β^2 par l'équation de droite que nous venons de déterminer. Ainsi, nous obtenons une équation ratio-dépendante similaire à celle énoncée par Arditi (équation 5.4). Nous généralisons donc l'équation 5.7 comme suit :

$$G(N,P) = \frac{\alpha N^2}{\mu^2 + N^2 + \gamma^2 P^2} \tag{5.8}$$

Considérant l'équation 5.8, nous pouvons noter des valeurs particulières de μ et γ. Si le milieu est homogène, alors le nombre de prédateurs n'a pas d'influence sur la réponse fonctionnelle, d'où $\gamma = 0$. Au contraire, si le milieu est hétérogène, le nombre de prédateurs a une influence sur la réponse fonctionnelle et $\gamma > 0$. De plus, γ peut être considéré comme une constante. Nous pouvons donc dire que ce paramètre est caractéristique des prédateurs. Il permet de rendre compte des effets de l'hétérogénéité pour un type de prédateur donné, sur la réponse fonctionnelle.
En ce qui concerne le paramètre μ, nous avons vu que même avec une densité constante de prédateurs, ce paramètre varie significativement (figures 5.9 et 5.10) en fonction de l'hétérogénéité. Ainsi, ce paramètre caractérise la distribution des proies elles-mêmes. Dans la pratique, des valeurs nulles de ces paramètres ne sont jamais observées. En effet, Arditi [AS92] constate que pour sa réponse ratio-dépendante (équation 5.4 page 136), β n'est jamais nul. Ceci revient à dire que le milieu n'est jamais homogène en pratique.

Dans ce qui précède, nous avons construit une réponse fonctionnelle à partir des résultats de simulations d'un modèle agents. Cette réponse ne se réduit pas à un ajustement polynomial mais résulte de la construction d'une fonction mathématique qui a du sens. Nous sommes donc bien dans le cas d'un calcul émergent au sens de Forrest [For90]. Il est toujours possible d'ajuster un modèle à un ensemble de points. Seulement, nous venons de montrer que, dans notre cas, l'ajustement est très bon, ce qui nous permet de valider qualitativement notre modèle comme étant une version *bottom up* d'un modèle *top down*. Nous sommes en présence de deux modèles du même système, définis à deux niveaux d'abstraction différents, individus et population.

5.3 Couplage de modèles et transfert d'échelle

Nous proposons ici d'illustrer la méthode proposée au paragraphe 2.2 pour simuler un transfert d'échelles. En effectuant différentes expériences sur le modèle d'agents réactifs, nous avons construit une fonction mathématique (la réponse fonctionnelle). Cette fonction peut entrer dans la définition d'un système d'équations différentielles décrivant la dynamique de populations de proies et de prédateurs. Nous pouvons alors choisir de paramétrer la réponse fonctionnelle à l'aide de notre modèle agents ou de remplacer cette fonction par notre modèle agent. Dans le deuxième cas, il s'agit de coupler le modèle d'agents réactifs avec le système d'équations différentielles. Un tel couplage permet la coexistence de deux niveaux d'organisations dans la même simulation et ainsi d'étudier les conséquences des propriétés modélisées à l'échelle de l'individu sur une dynamique globale de population. Cette question est fondamentale en écologie. Nous

ne prétendons pas y répondre ici mais voulons illustrer une nouvelle approche de modélisation permettant de l'appréhender. Nous considérons le degré d'hétérogénéité de la distribution des cellules de phytoplancton comme une caractéristique de l'échelle des prédateurs : nous voulons étudier son impact sur la dynamique globale du système proies-prédateurs.

5.3.1 Paramétrage du modèle proies-prédateurs

Dans un premier temps, nous considérons le paramétrage de la réponse fonctionnelle par notre modèle d'agents. Ce paramétrage nous permet de relier le degré d'hétérogénéité de la distribution des proies sur la dynamique globale. Pour cela, nous considérons le modèle classique de l'interaction proie-prédateur [Pav94] [Lot25] [Vol26] [BR93] décrit au paragraphe 2.3.3 page 44. Nous en rappelons ici la formulation avec les deux équations qui décrivent respectivement la dynamique des proies et de prédateurs :

$$\frac{dN}{dt} = rN(1-\frac{N}{K}) - G(N,P)P \qquad \text{avec } G(N,P) \text{ la réponse fonctionnelle} \tag{5.9}$$

$$\frac{dP}{dt} = eG(N,P) - mP \tag{5.10}$$

Les dynamiques de tels systèmes peuvent être étudiées analytiquement. Par exemple, si la réponse fonctionnelle est l'équation de Holling (équation 5.3 page 135) alors il apparaît des équilibres dans l'évolution temporelle des variables N et P sous forme de cycles limites stables pour certaines valeurs de paramètres [BR93]. Il en est de même pour les réponses fonctionnelles ratio-dépendantes [Jos98]. Ces études théoriques permettent notamment de déterminer les valeurs des paramètres des équations du système pour lesquelles nous observons des dynamiques particulières. Ici, nous allons simplement observer la dynamique du système proies-prédateurs pour différentes valeurs de paramètres en fonction du degré d'hétérogénéité de la distribution des proies.
Dans le paragraphe précédent, nous avons construit une réponse fonctionnelle ratio-dépendante et nous avons vu que la valeur du paramètre μ de l'équation 5.8 est liée au niveau d'hétérogénéité. Au travers de cette paramétrisation, nous pouvons relier des propriétés caractéristiques des échelles individuelles d'espace de quelques millimètres (distribution des proies par rapport au prédateur) et de temps de l'ordre de quelques secondes (chasse, capture, ingestion) à une dynamique globale d'échelle temporelle journalière.
Considérons les valeurs suivantes de paramètres des équations 5.9 et 5.10 :

- $r = 1j^{-1}$ correspondant à un renouvellement journalier des cellules,
- $K = 3,4.10^{-5}\ gN/l$ $(1,56\ cellules.mm^{-3})$ la concentration maximale en phytoplancton dans le milieu,
- $e = 0,02$ coefficient de proportionnalité sans dimension,
- $m = 0,02\ j^{-1}$ correspondant à une durée de vie de cent jours des copépodes.

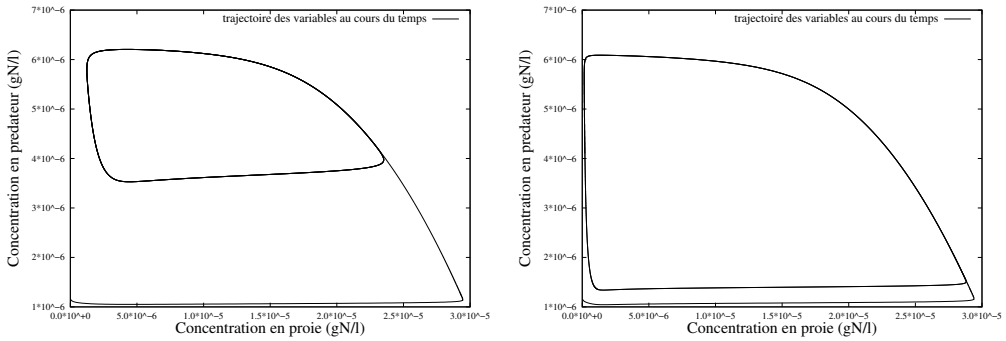
Nous choisissons la réponse fonctionnelle $G(N,P)$ définie précédemment (équation 5.8). Les valeurs des paramètres de cette équation sont pris pour deux niveaux d'hétérogénéité (correspondant aux résultats de simulations, figure 5.11). Nous définissons deux expériences :

	paramètre	expérience 1	expérience 2
hétérogénéité	ξ	$0,9$	$2,9$
taux d'ingestion maximum	α	$1,1.10^{-6}/738.10^{-9}$	$1,1.10^{-6}/738.10^{-9}$
coefficient de ratio-dépendance	μ	$3,1$	$1,1$
coefficient de densité-dépendance	γ	$5,1$	$4,8$

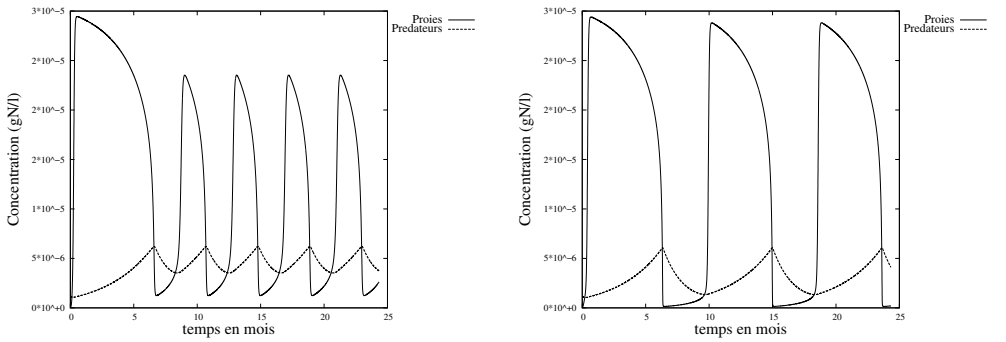
La valeur de α est divisée par la masse d'un copépode *Acartia tonsa* adulte [KFM85], ceci pour conserver la cohérence des unités du système d'équations. En effet, dans un tel système, les proies et les prédateurs sont modélisés par des concentrations alors que nous avons fait nos expériences avec un nombre fini de copépodes. Les valeurs de μ et γ sont ainsi définies respectivement pour une distribution plutôt homogène ($\xi = 0,9$) et hétérogène ($\xi = 2,9$) des proies. La figure 5.12 nous montre les portraits de phase (évolution de P en fonction de N) du système formé par les équations 5.9 et 5.10 pour les deux expériences considérées. Pour la simulation d'un tel système, nous avons utilisé le schéma d'intégration numérique de Runge-Kutta d'ordre 4.

La figure 5.12 montre l'apparition d'un cycle limite dans les deux expériences. Les concentrations représentant les populations oscillent autour d'une valeur moyenne au cours du temps. C'est un équilibre stable. Néanmoins, les deux dynamiques sont différentes. En milieu homogène, on observe une fréquence d'oscillation plus rapide qu'en milieu hétérogène. L'intensité de prédation est plus forte en milieu hétérogène, ce qui ralentit la croissance du phytoplancton. Cette croissance plus faible influence à son tour la croissance des copépodes. Comme nous pouvons le voir sur les figures 5.12(a) et (b), le maximum de copépodes est identique en milieu homogène et hétérogène, mais le minimum est inférieur en milieu hétérogène, ce qui a pour conséquence un maximum plus important pour le phytoplancton. Cette observation du comportement des équations différentielles pour deux niveaux d'hétérogénéité différents montre l'importance cruciale de la prise en compte des phénomènes de petites échelles sur les dynamiques globales. Ceci n'est pas nouveau en écologie. Néanmoins peu d'outils opérationnels permettent de tester des hypothèses à ce sujet ; les expériences virtuelles sur les modèles agents semblent être un outil prometteur. Nous ne faisons pas ici d'analyses écologiques, notre modèle de population se limite à un interaction proies-prédateurs alors que la dynamique de populations des copépodes est très complexe. Il s'agit simplement d'illustrer la potentialité de l'approche dans le cas où il est très difficile d'exercer un contrôle systématique sur certains paramètres. C'est le cas ici du degré d'hétérogénéité.

Comme nous l'avons vu avec la diffusion, il est possible d'aller plus loin dans l'exploitation de l'approche avec un couplage des deux modèles. Ce couplage se justifie ici du fait que l'hétérogénéité n'est pas constante, mais est fonction de la concentration. En effet, les cellules sont très dispersées aux faibles concentrations et au contraire très regroupées aux fortes concentrations, ce qui implique une hétérogénéité plus grande dans le premier cas. Nous ne disposons pas de modèle mathématique qui nous permettrait de formaliser ce phénomène. En couplant les deux modèles, nous pouvons pallier ce manque.



(a) distribution homogène ($\xi = 0, 9$) (b) distribution hétérogène ($\xi = 2, 9$)



(c) distribution homogène ($\xi = 0, 9$) (d) distribution hétérogène ($\xi = 2, 9$)

FIG. 5.12 – Résultats du paramétrage du modèle proies-prédateurs à l'aide du modèle agents du copépode. (a) et (b) : portrait de phase des expériences en milieux homogène et hétérogène. (c) et (d) : évolution au cours du temps des concentrations en proies et en prédateurs.

Nous nous apprêtons à coupler deux modèles de nature très différente. De ce fait, nous ne pouvons pas faire d'étude théorique de la dynamique du système. La simulation reste le seul moyen d'investigation, même s'il est possible d'écrire ce modèle couplé dans un formalisme opérationnel unifié (cf. chapitre 3). Nous reviendrons dans la discussion sur les possibilités d'études théoriques liées à notre approche.

5.3.2 Couplage des modèles

Pour le couplage des deux modèles, nous suivons exactement la même démarche que pour le modèle de diffusion exposé au paragraphe 2.2.3. La figure 5.13 présente de façon informelle le système d'équations couplé avec le modèle agents du copépode. Elle est à mettre en correspondance avec la partie droite de la figure 2.5 de la page 35 pour l'aspect opérationnel de la méthode.

À chaque pas de temps Δt de résolution du schéma numérique, le modèle agrégé fournit le nombre de copépodes et le nombre de cellules de phytoplancton au modèle agents. Celui-ci simule le taux d'ingestion et donne en retour la valeur de $G(x)$ au modèle agrégé. Conformément à ce qui a été dit précédemment (paragraphe 5.2.2), le modèle agent simule une durée de 150s

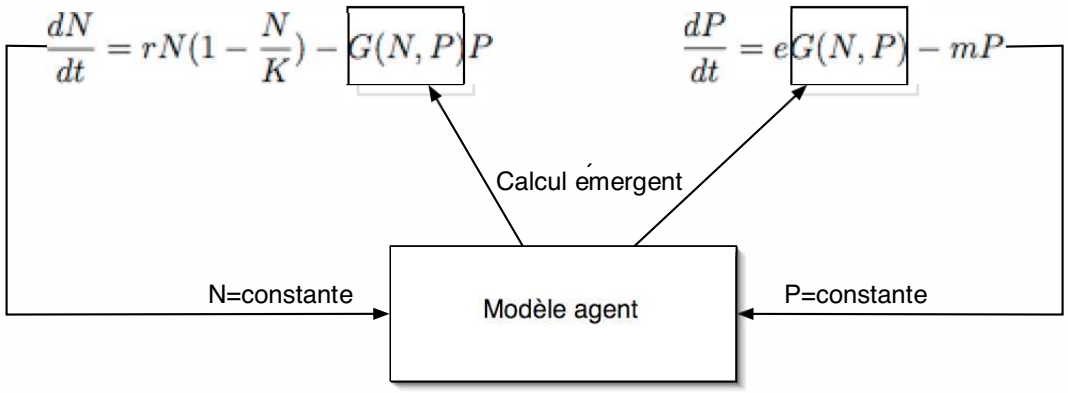


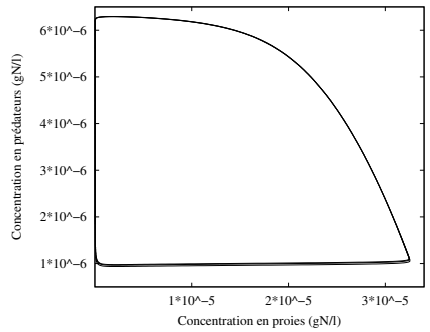
FIG. 5.13 – Schéma du couplage entre le modèle proies-prédateurs et le modèle agents du copépode. Le modèle agent « simule » $G(x)$ et le modèle proie-prédateur détermine le nombre de copépodes et de cellules de phytoplancton à chaque itération du schéma numérique.

pour effectuer le calcul. Nous considérons un pas de temps $\Delta t = 0,1$j pour le schéma numérique et une durée totale simulée de 720 jours, soit deux ans. Pour chaque Δt, la valeur de $G(x)$ est constante.
Comme nous l'avons dit dans le paragraphe précédent, nous ne pouvons pas garantir le même niveau d'hétérogénéité pour toutes les distributions simulées. Aux faibles concentrations, l'hétérogénéité devient très importante. Dans le cas du paramétrage du système d'équations différentielles, les paramètres de la réponse fonctionnelle étaient évalués à l'avance. Ceci implique que le niveau d'hétérogénéité soit constant. Cette dernière hypothèse est une hypothèse forte. En effet, à l'échelle du copépode, elle n'est jamais vérifiée puisque dépendante de la concentration autant que du type de distribution. En couplant les deux modèles, nous garantissons que la réponse fonctionnelle $G(N,P)$ reflète systématiquement les conditions d'hétérogénéité expérimentées par le copépode à un instant donné.
Pour connaître l'influence de la prise en compte dynamique du niveau d'hétérogénéité, nous avons donc décidé de faire deux expériences de couplage avec les deux types de distributions suivantes (décrites au paragraphe 5.1.2 page 5.1.2) :

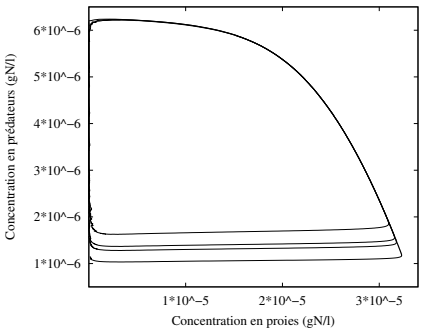
1. distribution hétérogènes,
2. distribution homogène.

La figure 5.14 présente les résultats de simulations du couplage en utilisant les mêmes paramètres pour le système d'équations différentielles que pour le paramétrage précédent (paragraphe 5.2.3). La différence est que la fonction $G(N,P)$ est « remplacée » par le modèle agents. À présent, nous allons brièvement décrire ces résultats simplement pour montrer que le couplage induit une dynamique nouvelle du système.

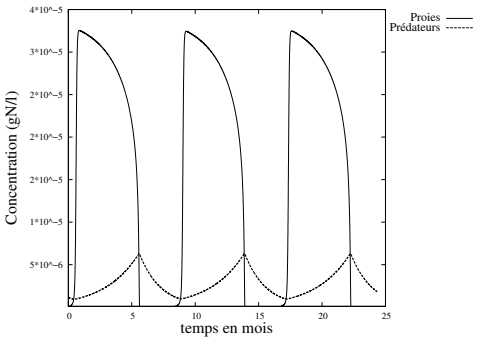
La figure 5.14 nous montre des résultats qui sont opposés à ceux obtenus par paramétrage (figure 5.12) et ce pour les deux types de distribution. En effet, la simulation avec couplage en milieu homogène présente une dynamique analogue à la simulation avec paramétrage en milieu hétérogène. Dans les deux cas, pour 25 mois simulés, il apparaît trois périodes dans l'évolution de la concentration en phytoplancton et en copépodes. Les simulations sont également proches quantitativement. Une différence notable est que dans le cas du couplage, la concentration en phytoplancton reste pratiquement nulle sur une durée plus longue (environ 3 mois) que dans la cas du paramétrage. La dynamique du système à faible concentration semble jouer un rôle



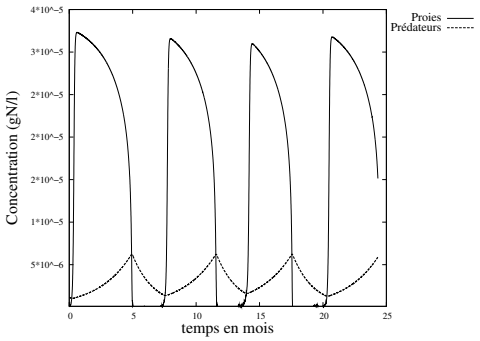
(a) homogénéité de la distribution des proies.



(b) forte hétérogénéité de la distribution des proies.



(c) homogénéité de la distribution des proies.



(d) forte hétérogénéité de la distribution des proies.

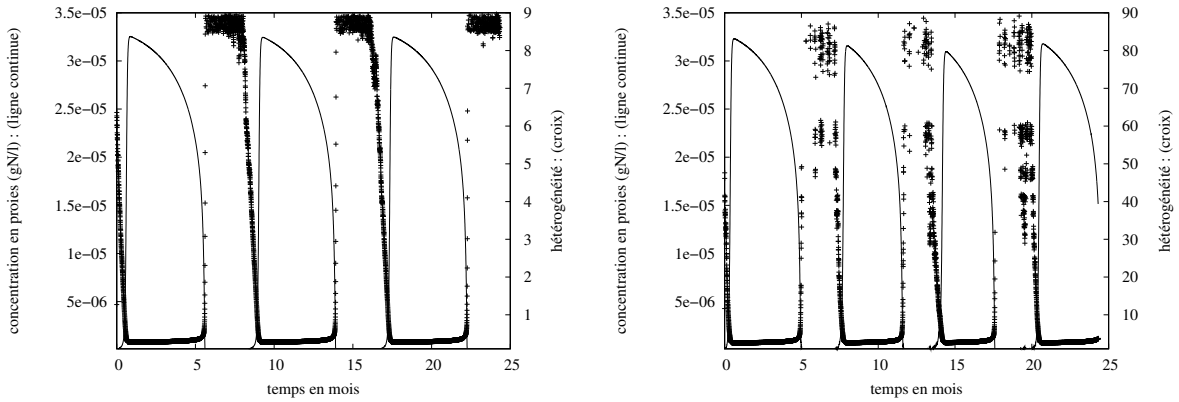
FIG. 5.14 – Résultats du couplage du modèle agents avec le système d'équations différentielles. En haut figurent les portraits de phase, en bas l'évolution des concentrations en fonction du temps. (a) et (c) : la distribution des proies suit une loi aléatoire uniforme (distribution homogène). Voir le texte pour les commentaires. (b) et (d) : la distribution des proies suit une loi normale d'écart-type 0,1 centrée sur des points tirés selon une loi uniforme (distribution hétérogène ou en paquets).

important dans la dynamique globale.

La simulation avec couplage en milieu hétérogène présente une dynamique assez similaire à la simulation avec paramétrage en milieu homogène. Les deux simulations sont quantitativement assez différentes. Dans le cas du couplage, l'amplitude des cycles de variation de la concentration en phytoplancton n'est pas stationnaire. De plus, comme dans le cas d'une distribution homogène, la concentration en phytoplancton reste pratiquement nulle pendant une assez longue période (2 mois environ), ce qui n'est pas le cas de la simulation par paramétrage.

Aussi bien d'un point de vue qualitatif que quantitatif, le couplage des deux modèles implique une dynamique nouvelle du système avec notamment la disparition presque totale du phytoplancton pendant une durée mesurable. Nous pensons que cette nouvelle dynamique est liée à la prise en compte de la variation du degré d'hétérogénéité de la distribution des proies par le modèle couplé. Comme nous l'avons dit plus haut, aux faibles concentrations, et quel que

soit le type de distribution, l'hétérogénéité est importante. Elle diminue avec l'augmentation de la concentration. Nous représentons l'évolution de la valeur ξ au regard de la concentration en phytoplancton sur la figure 5.15 pour les deux simulations avec couplage considérées précédemment.



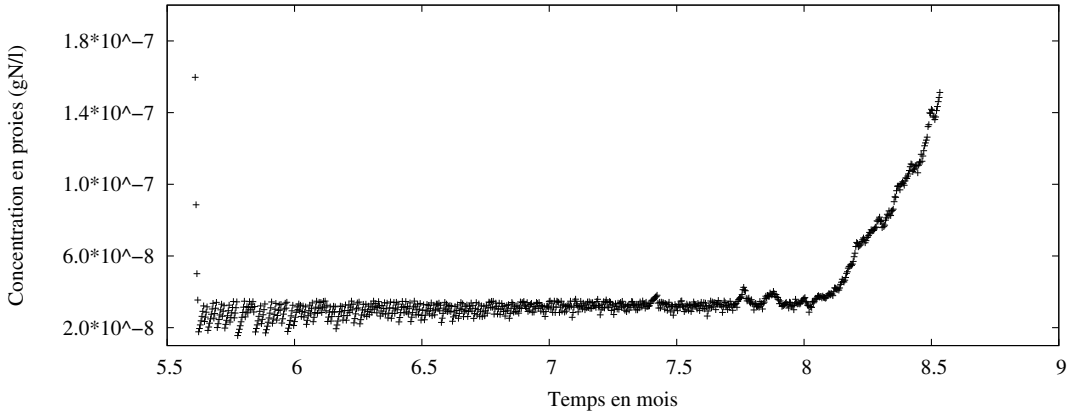
(a) Distribution homogène des proies

(b) Distribution hétérogène des proies

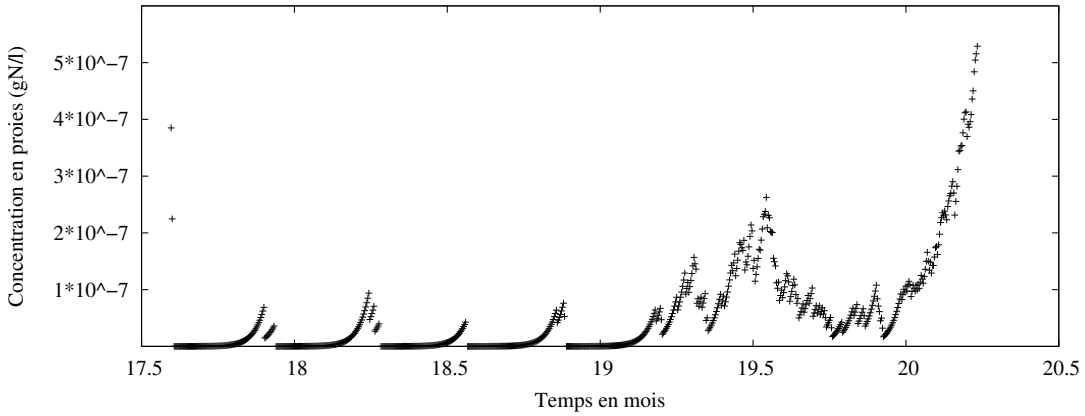
FIG. 5.15 – Évolution de l'hétérogénéité (valeur de ξ sur l'axe des ordonnées à droite) au regard de l'évolution de la concentration en proies (axe gauche des ordonnées). (a) : la distribution des proies suit une loi aléatoire uniforme. (b) : la distribution des proies suit une loi normale d'écart type 0,1 centrée sur des points tirés selon une loi uniforme (distribution en paquets). Dans les deux cas, les copépodes expérimentent principalement deux degrés d'hétérogénéité. Voir le texte pour les commentaires.

La figure 5.15 confirme que l'hétérogénéité de la distribution des proies n'est pas constante au cours du temps. Il apparaît deux « plateaux d'hétérogénéité » en phase avec l'évolution de la concentration en phytoplancton. Pour les deux types de distribution, l'hétérogénéité est maximale pour une très faible concentration en phytoplancton. Les valeurs des plateaux maximum sont intéressantes : en milieu homogène $\xi = 9$ et en milieu hétérogène $\xi = 90$. Nous avons mesuré l'impact de ξ sur le taux d'ingestion (figure 5.6 page 5.6). Il apparaît qu'à partir de $\xi > 4$, l'hétérogénéité a un effet négatif. Dans les deux cas, l'ingestion du copépode est donc extrêmement limitée lorsque la concentration en phytoplancton est faible. Cette relation connue est exprimée par des fonctions continues dans les modèles mathématiques classiques. Ici, nous apportons une information supplémentaire : pour un type de distribution donné, le copépode évolue principalement dans deux types de distributions caractérisées par deux degrés d'hétérogénéité. De plus, si la distribution est très hétérogène, le copépode est particulièrement peu efficace aux faibles concentrations de phytoplancton. Ceci se traduit par une pression de prédation moindre et donc par une croissance plus rapide du phytoplancton. Dans notre modèle, lorsque la concentration en phytoplancton est très faible, le modèle agents du copépode peut retourner une valeur nulle pour $G(x)$ (le copépode n'a pas pu s'alimenter). Dans ce cas, seule la mortalité des copépodes intervient dans leur dynamique, favorisant encore la croissance du phytoplancton par une diminution accrue de la quantité de prédateurs.
La dynamique de notre système pour les faibles concentrations de phytoplancton doit montrer des particularités qui sont peut-être à l'origine des différences de résultats entre paramétrage du modèle et couplage. La figure 5.16 présente une partie des courbes d'évolution de la concentra-

tion en phytoplancton des figures 5.15(a) et 5.15(b). Les « agrandissements » sont effectués sur des périodes de faible concentration en proies.



(a) Distribution homogène des proies



(b) Distribution hétérogène des proies

FIG. 5.16 – Évolution de la concentration en proies en fonction du temps. Les deux courbes (a) et (b) sont des agrandissements des courbes en traits pleins des figures 5.15(a) et 5.15(b) respectivement. Les agrandissement sont effectués sur des périodes de très faible concentration en proies. (a) : la concentration oscille de façon amortie autour d'un minimum puis augmente rapidement. (b) : la concentration augmente par à-coups successifs pour finir par croître rapidement. Voir le texte pour les commentaires.

La figure 5.16 nous montre deux dynamiques bien différentes. Pour une distribution homogène, la pression de prédation est faible mais relativement constante. Pour une distribution hétérogène des proies, la concentration augmente de façon discontinue avec une succession de ruptures où la concentration redevient proche de 0. Cette évolution reflète la grande variabilité de la pression de prédation exercée. Dans ce second cas, la distribution en paquets du phytoplancton implique que les copépodes sont parfois incapables de s'alimenter. Le phytoplancton peut alors croître rapidement jusqu'à ce que les copépodes puissent à nouveaux exercer une prédation. Ensuite, le phytoplancton se remet à décroître. Ce scénario se reproduit plusieurs fois,

jusqu'à ce que la population de copépodes soit suffisamment peu importante pour permettre une croissance importante du phytoplancton.
Nous observons que la durée des phases de faibles concentrations sont similaires dans les deux expériences. Ceci nous indique que les deux dynamiques particulières présentées ont un effet similaire sur le comportement global du système. Ce sont les fortes concentrations qui impliquent une dynamique particulière. À ces concentrations, le degré d'hétérogénéité influence la prédation. Néanmoins, la dynamique à l'échelle mensuelle en milieu hétérogène paraît particulièrement intéressante. Elle reflète le caractère ponctuel et intense de la prédation sur des petites populations dispersées. On peut penser en effet que lorsque les prédateurs trouvent des paquets de proies, il les éliminent toutes. Ces évènements ponctuels sont difficiles à modéliser dans une dynamique globale à grande échelle ; nous offrons ici une technique possible d'investigation.
Dans la discussion suivante, nous abordons notamment les utilisations possibles du modèle couplé dans un contexte purement écologique puis nous généraliserons aux système complexes.

5.4 Discussion

Dans l'étude du modèle couplé, notre analyse est restée principalement qualitative. Il s'agit ici de montrer le potentiel de notre méthode de couplage. Nous nous positionnons en tant qu'expérimentateurs d'un modèle, considéré comme un laboratoire virtuel. Dans cette perspective, les simulations ont pour premier objectif de faire progresser la connaissance du fonctionnement du modèle avant toute progression de la connaissance du phénomène représenté. Comme nous l'avons dit dans un article [DADR03] :

> « Une telle démarche peut probablement paraître choquante, au moins pour deux raisons. Tout d'abord, ne faisons-nous pas des modèles pour comprendre des phénomènes que nous ne comprenons pas ? Alors quel intérêt d'y substituer des modèles que nous ne comprenons pas non plus (puisqu'il faut les étudier pour les comprendre) ? Mais plus profondément, un modèle est totalement spécifié par son concepteur, donc les moindres mécanismes en sont connus. Comment se peut-il qu'il faille encore l'étudier pour le comprendre ? »

Ici, notre connaissance intime des mécanismes du modèle couplé ne nous permet pas de comprendre son comportement global. Cette caractéristique, souvent associée à une certaine conception de la complexité, permet d'envisager pleinement notre modèle comme objet d'expérimentations, au même titre qu'un phénomène naturel. Dans notre cas, nous ne connaissons pas l'influence de l'échelle individuelle sur le comportement de la population, car ces expériences sont très difficiles à mettre en œuvre sur le système réel. Ainsi, l'expérimentation virtuelle sur notre modèle couplé peut offrir un cadre pour formuler des hypothèses sur le système réel. Nous allons commencer par discuter de cet aspect.

5.4.1 À propos du copépode

Nous avons commencé par travailler sur la détermination d'une réponse fonctionnelle. Cette réponse dépend des caractéristiques intrinsèques de notre modèle. L'approche agent nous permet d'exprimer simplement des paramètres comme la vitesse de déplacement, la géométrie de perception ou la stratégie de chasse (ici la nage libre) au niveau du copépode. Elle permet également l'étude de l'influence des caractéristiques environnementales sur la dynamique du système. Ce grand pouvoir d'expressivité a une conséquence néfaste évidente : l'analyse de sensibilité du

modèle peut s'avérer très longue et extrêmement complexe à mettre en œuvre. C'est la raison pour laquelle cette étape est très largement négligée en modélisation orientée agents. Dans notre cas, les paramètres du modèle ont été empruntés à la littérature. Nous avons donc considéré ces paramètres comme « valides ». Nous nous sommes focalisés sur les conséquences de nos choix propres de modélisation. Nous n'avons pas appelé cette étape « analyse de sensibilité » justement parce que l'étude menée n'est pas exhaustive. Elle a néanmoins permis de choisir une bonne représentation de l'espace et du comportement de l'agent aux frontières pour notre travail sur le taux d'ingestion.

Nous nous sommes focalisés sur l'influence de la nature discrète de la distribution des proies sur le taux d'ingestion du copépode. Cette question est difficilement abordable expérimentalement, c'est pourquoi ce sont des facteurs comme la concentration en proies ou la température (facteurs contrôlables) qui sont traditionnellement étudiés [PBLM95]. De par la nature spatialement discrète de notre approche, nous pouvons aborder cette question avec une mise en relation d'une mesure du degré d'hétérogénéité des distributions et du taux d'ingestion, c'est-à-dire l'efficacité du prédateur. Cette mesure est dépendante de la distance de capture du copépode et ne s'applique donc pas à l'ensemble des copépodes, ni même à une espèce dans son ensemble puisque cette distance varie en fonction de l'âge des individus. Elle est donc caractéristique d'*A. Tonsa* adulte dans notre exemple. Cette hétérogénéité relative à *A. Tonsa* fait apparaître que l'efficacité du copépode est optimale dans une certaine fourchette d'hétérogénéité (voir figure 5.6 page 141). Nous pouvons associer cet intervalle à la définition d'un « habitat favorable » en terme de distribution des proies, c'est-à-dire des conditions environnementales optimales pour la vie des copépodes. Cet intervalle est naturellement lié au paramétrage du modèle de l'individu, donc à un type d'individus particulier.
Nous sommes donc capables de mettre en relation le degré d'hétérogénéité et la prédation d'un copépode particulier. La biologie du copépode est assez complexe. L'individu évolue en passant d'un stade de développement à un autre par des mues successives. Il existe 11 stades chez les copépodes. En dehors du stade d'œuf, les différents stades d'évolution (du *nauplii* jusqu'à l'adulte) sont caractérisés par des vitesses de déplacement et des distances de perception particulières [Tr03]. Ces deux paramètres influencent le taux d'ingestion. Le travail de mise en relation du niveau d'hétérogénéité avec le taux d'ingestion effectué ici peut être fait pour les différents stades du copépode. Ainsi, il serait possible de définir des niveaux d'hétérogénéités optimaux pour chaque stade de développement, définissant des habitats favorables.

Nous sommes donc capables de proposer une méthode pour répondre à la première partie de la question posée en introduction page 38[86]. Pour répondre à la deuxième partie de la question, nous avons montré que la notion d'expériences virtuelles sur notre modèle agents du copépode pouvait mettre en relation ce modèle avec un modèle mathématique, appelé « réponse fonctionnelle » des prédateurs. La construction de ce modèle nous a permis de faire une validation qualitative de notre modèle agents. En effet, nous sommes capables de reproduire les modèles densité dépendants et ratio dépendants décrits par la littérature [CGW00][AS92] et classiquement utilisés dans les modèles de dynamique de population. Là aussi, notre étude reste théorique et étroitement liée au paramétrage de notre modèle agents. Néanmoins, nous mettons en évidence la possibilité de paramétrer ces réponses fonctionnelles à l'aide d'expériences virtuelles menées comme des expériences classiques. Pouvons nous identifier des réponses fonctionnelles

[86] Comment l'hétérogénéité de la distribution des proies influence-t-elle la prédation des copépodes et a-t-elle une influence sur la dynamique de population ?

différentes ? Nous pensons que oui, en jouant sur le paramétrage du modèle agents ou en introduisant d'autres facteurs de variabilité individuelle. Nous avons donc là un moyen de relier une dynamique individuelle à un comportement global de façon opérationnelle.
C'est à partir de cette mise en correspondance que nous avons proposé le paramétrage de la réponse fonctionnelle ou son remplacement total dans un modèle proies-prédateurs classique pour simuler un transfert d'échelle entre le niveau individuel et le niveau de la population. Ainsi nous avons apporté une méthode pour répondre à la deuxième partie de la question posée en introduction.

Cette question est directement liée à celle de la turbulence [SSL+99][SSL+96]. *A priori*, il n'y a pas d'outil qui permette de relier directement la turbulence au niveau d'hétérogénéité des proies[87]. Si cet outil est développé, il sera relativement simple de l'intégrer dans notre modèle pour la génération des distributions de particules et nous pourrons ainsi proposer une méthode de modélisation pour l'étude de l'influence de la turbulence sur la dynamique de populations.

D'un point de vue plus théorique, le fait de pouvoir paramétrer une réponse fonctionnelle particulière en fonction du degré d'hétérogénéité permet de mettre en relation ce dernier avec la dynamique d'un système d'équations différentielles. Il est possible, dans certains cas, de calculer les points d'équilibres, le type de cycles limites et d'oscillations d'un système d'équations différentielles. Cette étude permet de savoir pour quelles valeurs de paramètres le modèle présente un équilibre ou voit son évolution mener à l'extinction d'une espèce, etc. Les modèles ratio-dépendants, tel que le nôtre, présentent un fort bassin d'attraction en $N = 0$ et $P = 0$ mais peuvent également montrer des cycles limites tels que nous les observons [JAA99]. Ces études nécessitent le tracé de diagrammes de bifurcations[88]. À partir des paramètres simulés par notre modèle agents, ces diagrammes nous permettront d'étudier l'influence théorique de l'hétérogénéité sur la dynamique globale d'un système proies-prédateurs. Cette étude a commencé en partenariat avec J.C. Poggiale, mathématicien au centre d'océanologie de Marseille.

Le système couplé que nous avons développé permet d'élargir le spectre des échelles de temps observables dans une simulation. Dans notre exemple, nous sommes capables d'observer des phénomènes allant de la seconde jusqu'à l'année en considérant le modèle agrégé, en passant par des échelles intermédiaires comme celle du mois (voir figures 5.16(a) et (b) page 154). Nous pensons que cette possibilité est une réelle plus-value pour la compréhension de la dynamique des systèmes en général.

5.4.2 Sur l'apport à la modélisation des systèmes complexes

La modélisation des systèmes complexes passe par un aller-retour permanent entre l'expérience sur le système réel et la modélisation. Dans ce contexte, l'activité de modélisation fait partie du cycle empirique de développement des connaissances. La figure 5.17 (rectangle du haut) présente ce cycle et le lien avec l'activité de modélisation. V. Grimm attribue aux modèles mathématiques une qualité généralisable aux autres types de modèles [Gri94]. Cette qualité est le questionnement permanent du modélisateur sur son modèle qui permet la formulation de nouvelles hypothèses sur le système modélisé.

[87] C'est l'une des questions abordées lors de la conférence internationale ASLO Ocean Research à Honolulu en Février 2004.

[88] Variation des valeurs de paramètres dans l'espace des paramètres.

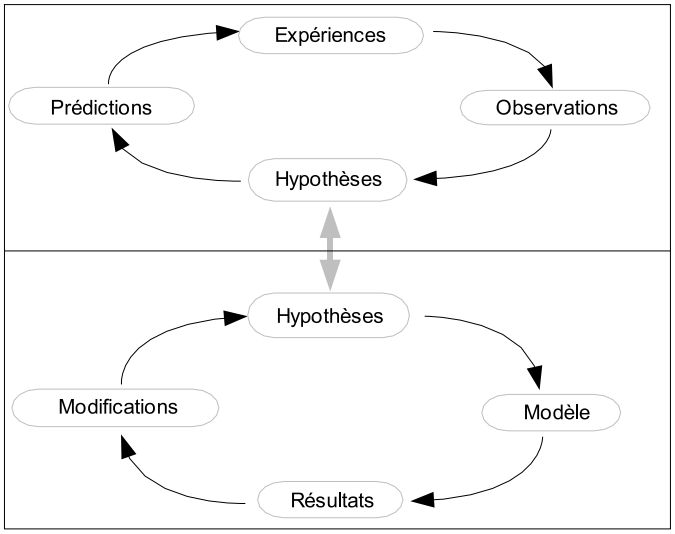


FIG. 5.17 – Schéma sur l'apport de la modélisation dans l'enrichissement des connaissances dans les sciences naturelles (modifié d'après [Gri94]). Dans le rectangle du haut figure le diagramme du cycle de l'enrichissement des connaissances. Dans le rectangle du bas figure le cycle de la création d'un modèle. C'est la formulation de nouvelles hypothèses qui fait le lien entre les deux cycles.

En écologie, cette démarche concernait principalement les modèles mathématiques. Il est évident que les autres types de modèles participent également à cette démarche, en apportant de nouveaux moyens d'expressions. C'est typiquement le cas de notre modèle. Jusque-là, les IBMs concernant les copépodes étaient principalement des modèles basés sur des équations différentielles. Nous avons proposé une modélisation mécaniste des individus qui permet de prendre en compte des comportements plus complexes que ceux exprimables par des moyens plus « traditionnels ». Dans le contexte décrit par la figure 5.17, les SMAs prennent une place de plus en plus importante en posant de nouvelles questions aux expérimentateurs, notamment sur les mécanismes à la base des comportements. Dans le cas des copépodes, les expériences nécessaires pour identifier de tels mécanismes sont fastidieuses et nécessitent souvent un appareillage sophistiqué et coûteux. On voit ici l'intérêt des expériences virtuelles, basées sur des modèles décrivant des processus fins, pour l'aide à la mise en œuvre d'expériences sur le système réel. L'avantage évident des modèles est de pouvoir explorer un très grand nombre de scénarios, ce qui permet d'élaborer un ensemble d'hypothèses *a priori* sur le fonctionnement d'un système et donc de construire des protocoles expérimentaux adaptés pour vérifier ces hypothèses. Jusqu'à maintenant, dans la majorité des cas, le modélisateur construit son modèle indépendamment de l'expérimentateur et compte sur les données de celui-ci pour construire ou vérifier ses hypothèses.

Depuis les méthodes de Monte-Carlo, il est connu que la simulation ayant recours aux nombres pseudo-aléatoires peut par exemple servir au calcul d'intégrales présentant des limites d'intégration qui ne sont pas aisées à exprimer analytiquement [PTVF94]. La simulation vient ici « au secours » des mathématiques, lorsque celles-ci atteignent leurs limites calculatoires opérationnelles. De même, les méthodes d'intégration numérique, bien qu'issues d'axiomes et théorèmes démontrés formellement, sont nées du besoin de résoudre numériquement, c'est-à-dire en les simulant, des systèmes d'équations différentielles le plus souvent insolubles analytiquement.

Notre approche se situe dans la même lignée, en ajoutant l'idée d'expérimentation virtuelle sur un modèle centré-individus.
Nous pensons que cette approche permet, dans certains cas, d'utiliser au mieux les avantages des deux types de modélisation, et d'obtenir une réelle plus-value par rapport à l'utilisation d'un seul des modèles. En effet, le modèle agrégé permet d'obtenir une représentation compacte et facilement utilisable de l'ensemble de la dynamique du système. Le modèle individus-centré permet d'assouplir l'utilisation du modèle agrégé en identifiant localement ses paramètres.
On peut par ailleurs se poser la question de l'efficacité du couplage. Il est clair que dans certains cas, il peut être plus avantageux d'établir *a priori* un protocole de tests dans le laboratoire virtuel et d'en tirer l'ensemble des valeurs intéressantes pour le modèle agrégé. Dans notre exemple basé sur la diffusion de particules (paragraphe 2.2 page 27), si l'ensemble des valeurs de v_{max} atteignables sont connues *a priori*, il suffit alors d'établir des expérimentations pour différentes valeurs de v_{max} dans cet intervalle, afin d'obtenir par interpolation la fonction qui donne la valeur du coefficient de diffusion D en fonction de v_{max}. Cette fonction est ensuite directement utilisée dans le modèle agrégé. Il s'agit là d'une méthode généralisable à de nombreux autres modèles, la difficulté étant de connaître à l'avance le domaine de variation des variables du modèle agrégé.

Nous avons considéré un modèle mathématique dont l'évolution est uniquement temporelle. Il est maintenant admis que la prise en compte de la structure spatiale des écosystèmes est fondamentale dans la compréhension de leur dynamique [FPV95]. Peut-on alors généraliser notre méthode pour prendre en compte l'espace au niveau du modèle agrégé ?
Introduire l'espace, c'est introduire une discrétisation supplémentaire du modèle mathématique. Ainsi, il est possible de considérer un IBM local à chaque maille de discrétisation de l'espace et, sous la même hypothèse de constance des variables à chaque pas de temps avec celle de la constance des variables dans l'espace sous mailles, il semble possible d'utiliser notre méthode. Nous parlons ici d'IBM indépendamment des formalismes utilisés. La figure 5.18 illustre un couplage avec prise en compte de l'espace au niveau du modèle agrégé.

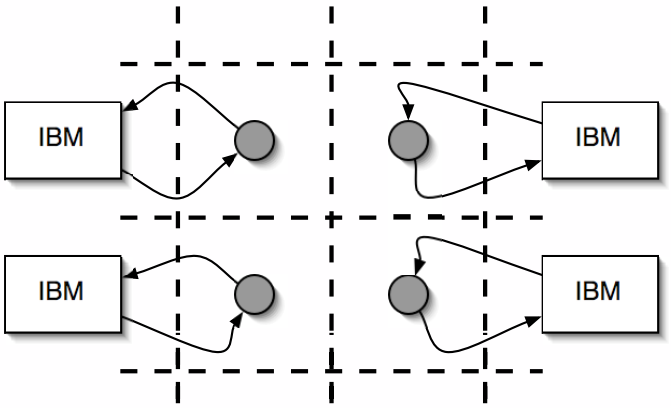


FIG. 5.18 – Schéma du couplage entre un modèle spatial implémenté à l'aide d'un schéma numérique et des IBMs. Pour chaque pas d'espace (cercles), et à chaque pas de temps, un couplage bidirectionnel est réalisé. Le modèle agrégé fixe l'environnement local de l'IBM, celui-ci venant paramétrer le premier.

Notre méthode s'appuie sur l'identification de propriétés stables du modèle agents dans certaines conditions. Dans notre exemple, il s'agit du taux d'ingestion pour une certaine concen-

tration en phytoplancton et en copépodes. Ces propriétés stables « quantifient » le comportement individuel, c'est-à-dire en donne une mesure. Nous pouvons donc coupler le modèle des individus avec un modèle « quantitatif », manipulant des scalaires. D. Servat [Ser00] propose une approche « qualitative » pour définir des structures stables pendant un certain temps en réifiant (ou réunitarisant) des agents en un groupe partageant les mêmes caractéristiques. En conservant notre approche, il apparaît que ce type de modèle devrait être couplé avec un autre modèle agent, capable de manipuler ces objets réifiés. Dans ce cas, c'est le modèle de bas niveau qui contrôle la durée d'existence de ces structures émergentes, alors que dans notre cas, c'est le pas de temps du modèle de haut niveau qui cadence les changements d'environnements du modèle de bas niveau. Pour l'approche de D. Servat, le concept d'évènements peut alors être utile pour modéliser l'apparition de ces structures au niveau supérieur.
Cet exemple permet d'illustrer le fait que les méthodes de séparation des échelles de temps et de variation de la constante présentées au paragraphe 2.2.4 page 33, enrichies de la notion d'expérience virtuelle, sont applicables dans d'autres contextes de modélisation. Cette méthode est un moyen de simuler le transfert d'échelles, c'est-à-dire la prise en compte des propriétés d'un modèle micro dans un modèle macro.

Nous pouvons assimiler notre modèle à un modèle particulaire, c'est-à-dire un modèle de déplacements lagrangiens. Ce type de modèles a l'inconvénient majeur d'être limité quant au nombre de particules qu'il est possible de simuler, surtout lorsqu'elles ont un comportement complexe, nécessitant encore plus de temps de calcul[89]. Dans ce contexte, peut-on voir notre méthode comme une optimisation pour les modèles particulaires, et plus particulièrement pour les SMAs ? Le terme « optimisation » est un peu usurpé ici. En effet il n'est pas à comprendre comme « la mise au point d'un algorithme de résolution plus rapide », mais comme « la possibilité pratique d'augmenter les dimensions du système modélisé ». En identifiant des paramètres, des variables ou des fonctions constants pendant un certain temps de simulation d'un SMA et en utilisant un couplage tel que nous l'avons présenté, il est possible de manipuler à la fois des particules et des scalaires représentant totalement ou partiellement le même système. Ainsi, nous pouvons simuler un nombre d'agents très grand et des durées très longues par rapport à celles atteignables en considérant seulement le modèle particulaire. Par exemple, la figure 5.19 nous donne l'évolution du temps de simulation en fonction du temps simulé pour notre modèle d'agents dans certaine conditions.

L'évolution du temps de simulation est linéaire. À l'aide du coefficient de la droite présentée figure 5.19 d'une valeur de $27,07 s.j^{-1}$, nous pouvons inférer le temps de simulation nécessaire pour simuler 720 jours (durée simulée dans les expériences de couplage). Une telle simulation prendrait environ 5 jours et demi alors qu'en utilisant le couplage, la durée de simulation tombe à 7h 42mn 12s sur la même machine ! [90]. Notre technique de couplage peut être une réponse

[89] Cette limitation est directement liée à la puissance de traitement des ordinateurs et à leur capacité mémoire. Ces deux aspects sont en perpétuelle évolution et nous pourrions penser que dans le futur les modèles particulaires n'auront plus de limites liées au matériel. Ceci est partiellement vrai. En effet, à un niveau de détail donné, les modèles particulaires ne sont pas prohibés. Toutefois, il est possible d'augmenter le niveau de détail d'un modèle presque à l'infini.

[90] Une remarque s'impose ici. Si nous avions un modèle agents de la dynamique de population du copépode, il devrait prendre en compte la croissance (11 stades chez les copépodes), la mortalité naturelle, la reproduction, la ponte etc. Il serait également nécessaire d'introduire une dynamique au phytoplancton. Il serait donc beaucoup plus complexe et certainement beaucoup plus long à simuler. En considérant un modèle complet de dynamique de population du copépode sous forme d'équations différentielles, nettement plus compliqué lui aussi mais manipulant des scalaires, nous pouvons raisonnablement penser que l'écart

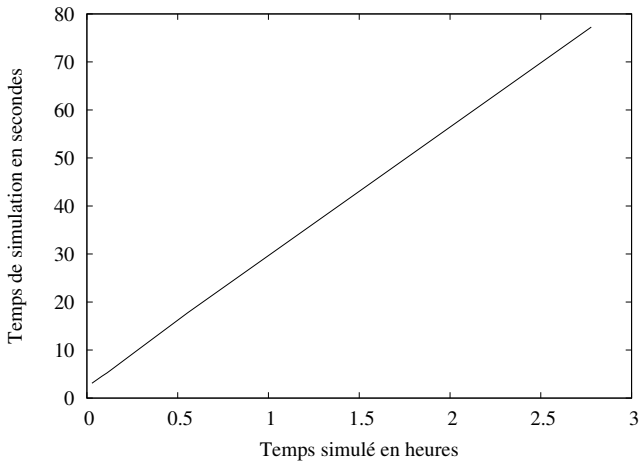


FIG. 5.19 – Évolution du temps de simulation en fonction du temps simulé pour le modèle d'agents. Le résultat donné ici concerne des simulations avec 10 copépodes, des distributions uniformes et un renouvellement des cellules de phytoplancton. Les simulations ont été effectuées sur un PC muni d'un processeur Atlon 1GHz et de 128M de RAM. L'évolution est linéaire (pente de la droite = $27,07 s.j^{-1}$). La droite ne passe pas par l'origine, ce temps minimal est celui nécessaire pour analyser les fichiers XML de description du plan d'expérience.

aux problèmes de « montée en charge » des SMAs lorsqu'ils se complexifient ou que le besoin de simuler beaucoup plus d'agents se fait sentir.
La difficulté majeure peut être de trouver le modèle mathématique agrégé équivalent au SMA en terme de comportement global. Néanmoins, comme nous l'avons montré dans notre exemple, il est possible de construire ce modèle mathématique à partir des traces de simulation du SMA. Dans ce contexte, notre approche présente une grande similarité avec la méta-modélisation[91]. Des travaux comme ceux de A. Aussem et D. Hill [AH99] proposent d'utiliser des modèles de type réseaux de neurones pour représenter le comportement d'un modèle discret complexe et beaucoup plus coûteux à simuler. Cette approche de méta-modélisation offre la possibilité d'effectuer plus rapidement les simulations mais est restreinte à « l'apprentissage » du méta-modèle, donc au domaine de validité défini par le modèle de base. Dans notre approche, la simulation n'est pas bornée sur un domaine fixé *a priori*. De plus, nous conservons le degré de détail du modèle discret, c'est-à-dire son expressivité.

5.5 Conclusion et perspectives

Au paragraphe 2.2 page 27, nous avons présenté une méthode de couplage entre un modèle particulaire et un modèle mathématique, basée sur un exemple simple de diffusion de particules. Cette méthode est principalement basée sur les notions de calcul émergent et d'expériences virtuelles. Dans ce chapitre, nous avons montré que cette méthode était applicable dans le contexte de l'écologie théorique où elle offre de nouvelles perspectives d'étude et de modélisation. Cet exemple d'applications nous a également permis d'illustrer la richesse entraînée par l'adoption d'un comportement d'expérimentateur sur les modèles en mettant en évidence de façon méca-

en terme de temps de calcul serait encore plus important.
[91]La construction de modèle à partir de modèle.

niste une fonction de l'écologie théorique, permettant une validation qualitative de notre modèle. Une des originalités de notre approche par rapport aux modèles mathématiques classiques est que nous pouvons étudier l'influence de la variabilité du comportement individuel sur la dynamique globale. Cette question est actuellement centrale en écologie si l'on en croit les nouvelles orientations d'une ART du PNEC dans laquelle nous nous sommes inscrits.

La modélisation en écologie est historiquement très liée aux modèles purement mathématiques. Un des défis lancés aux informaticiens qui étudient les SMAs dans le contexte de l'écologie est de communiquer les apports techniques et les nombreux apports théoriques possibles auprès des écologues[92]. De façon générale, nous pensons que la notion d'expériences virtuelles permet un rapprochement enrichissant des informaticiens et des modélisateurs de disciplines particulières. Les premiers offrent des outils théoriques et opérationnels nouveaux pour les seconds, ceux-ci posant de nouvelles questions sur la représentation rationnelle des systèmes nécessitant la mise au point de nouvelles techniques par les premiers.
Dans notre cas, nous avons apporté le paradigme agent pour simuler les caractéristiques individuelles des copépodes. Mais c'est la problématique du transfert d'échelles en écologie qui nous a guidé dans la mise au point d'une technique de couplage formelle et opérationnelle pour l'étude de ce phénomène.

Une des caractéristiques les plus intéressantes des SMAs pour un écologue est la possibilité de regarder le système à plusieurs niveaux d'abstraction. Nous nous sommes donc posé la question de savoir si une modélisation agent (au niveau individus) pouvait représenter dans sa totalité la dynamique des modèles classiques de population. Cette expérience a déjà été faite en ce qui concerne les équations de Lotka-Volterra par Cazoulat [CV94]. À partir des hypothèses simples du modèle mathématique, il est possible de construire un modèle agent dont le comportement émergent fera apparaître une dynamique de populations stable. Comme nous l'avons dit plus haut, si nous voulons considérer le cycle de vie complet des copépodes pour une telle expérience, nous serons très vite limités par la puissance de traitement de nos ordinateurs de bureau. Nous avons donc commencé une étude de la distribution de modèles particulaires de diffusion à évènements discrets sur un cluster de PC [QDNR03] afin de mener à bien cette expérience.

Un autre point intéressant les écologues est l'apport de la modélisation à la compréhension, ou à la mesure de phénomènes pratiquement inatteignables par l'expérimentation *in vivo*. Typiquement, le taux de mortalité est un paramètre essentiel dans les modèles de dynamique de population [CGW00] et pourtant il est très peu étudié. Cette lacune ne reflète pas un manque d'intérêt pour ce processus mais plutôt la difficulté de le mettre en évidence de manière expérimentale. En effet, la mort d'un organisme peut avoir différentes causes, les principales étant le vieillissement, la prédation, la maladie et le jeûne. Dans le cas particulier du copépode, il faut ajouter l'échec du passage d'un stade à l'autre lors des mues successives qui joue un rôle important dans la dynamique de populations [SS01][CS89]. Le problème est de pouvoir isoler ces différentes causes de mortalité pour être capable de quantifier leurs effets respectifs sur la mortalité totale. Il est presque impossible de le faire de façon expérimentale sauf pour la mortalité par prédation qui masque, par son intensité et sa ponctualité, les autres causes de mortalité.
L'utilisation de la modélisation apparaît comme un moyen efficace de discrimination des différentes causes de mortalité. Dans notre modèle, la mort du copépode ne peut intervenir que parce qu'il ne se nourrit pas suffisamment ; c'est la mortalité due au jeûne. Ainsi, nous sommes

[92]Nous avons en tête les nombreuses questions et les discussions qui ont suivis notre présentation aux Journées de la Société Française de Biologie Théorique (SFBT) à Saint-Flour, en Juin 2002.

capables de faire évoluer le copépode dans un milieu riche en nourriture puis de faire disparaître celle-ci pour mesurer le temps de survie d'un individu et ainsi de calculer un taux de mortalité dû au jeûne. Ce travail a commencé dans le cadre d'une collaboration avec F. Carlotti de la station marine d'Arcachon.

Enfin, nous pouvons élargir le cadre d'application de notre technique de couplage. Celle-ci peut s'appliquer à des domaines proches comme l'écologie terrestre, par exemple pour la simulation des migrations d'animaux. Cette migration peut être modélisée à l'échelle régionale à l'aide d'équations différentielles. Sachant que le comportement individuel et les réactions à l'environnement influencent fortement la dynamique de l'ensemble du troupeau [DH01], les individus sont modélisés par un SMA. Le couplage des deux modèles permet de faire intervenir les caractéristiques des individus au niveau global. Dans des domaines plus éloignés comme l'écoulement des fluides, notre approche pourrait avoir un intérêt certain. Le modèle agent de « boules d'eau » proposé par D. Servat [Ser00], pourrait faire l'objet d'un couplage avec différents modèles agrégés, utilement inspiré par notre approche. On peut supposer que ce soit également le cas de nombreuses autres applications dans des domaines très variés comme la modélisation de systèmes socio-économiques par exemple.

6

Conclusion générale

Résumé

Nous donnons ici une vision synoptique de nos travaux en insistant sur les apports de cette thèse. Nous pouvons les résumer ici :

- rapprochement du paradigme d'agents et de la formalisation des systèmes dynamiques,
- formalisation DEVS du couplage d'un SMA réactif situé avec un système d'équations différentielles,
- élaboration d'un *framework* opérationnel pour l'intégration de modèles hétérogènes avec définition d'applications XML pour la description et l'échange des modèles,
- mise au point d'une méthode pour simuler le transfert d'échelles entre différents niveaux d'organisation,
- application de cette méthode en écologie marine pour montrer les relations entre la nutrition individuelle et la dynamique globale d'une population de copépode.

Des perspectives intéressantes se sont dégagées de ce travail. Nous avons commencé l'exploration de certaines d'entre elles.

Sommaire

6.1 Synoptique de nos travaux

La problématique de fond de cette thèse est de savoir comment intégrer, c'est-à-dire coupler, des modèles hétérogènes. Le but d'une telle intégration est la modélisation et la simulation des systèmes dynamiques complexes. En accord avec Fishwick [Fis95] ou encore H. Vangheluwe [VLM02], nous pensons que la multi-modélisation apporte une réelle plus-value en terme de description de systèmes. Cette description « augmentée » peut permettre en retour de mieux comprendre certains aspects des systèmes dynamiques modélisés. Pour traiter de cette question, nous avons choisi trois thèmes :

1. l'intégration formelle,
2. l'intégration opérationnelle,
3. l'utilisation d'un modèle couplé en écologie marine.

Dans le premier thème, nous traitons de la multi-modélisation sur le plan formel. Des travaux existent dans ce domaine où le formalisme DEVS joue un rôle central comme cadre d'intégration formelle. Nous avons traité cette question en considérant un modèle d'agents réactifs couplé avec un système d'équations différentielles. Pour cela, nous avons dû spécifier notre SMA et le simulateur du système d'équations différentielles totalement en DEVS. Ce formalisme intègre les notions de couplage et de décomposition hiérarchique. Nous disposons alors d'un modèle couplé formalisé à différents niveaux d'abstraction.

Dans le deuxième thème, nous nous sommes intéressés à la définition d'un *framework* pour l'intégration de modèles hétérogènes. Pour définir ce *framework*, nous avons intégré certains des concepts utilisés dans HLA ou encore dans DEVS-Bus. Nous reprenons notamment la logique des simulateurs abstraits. Nous avons défini quatre niveaux d'abstractions différents pour notre *framework* (opérationnel, simulation, modèle et sémantique). Des solutions sont proposées pour chacun d'eux.

Dans le troisième thème, nous couplons un modèle d'agents réactifs simulant le comportement alimentaire d'un copépode se nourrissant de phytoplancton avec un modèle classique d'équations différentielles simulant une interaction proies-prédateurs. Ce travail illustre tout l'intérêt des réflexions menées dans les autres thèmes. Nous montrons qu'il est possible de considérer dans un même modèle et dans une même simulation le comportement d'individus, les interactions discrètes entre proies et prédateurs et la dynamique globale des populations.

6.2 Apports de cette thèse

Pour mettre en évidence les apports de cette recherche, nous reprenons ici les trois thèmes mentionnés plus haut.

À l'intérieur du premier thème, nous avons rapproché le paradigme des SMAs de la théorie de la modélisation et de la simulation telle qu'elle est définie par B.P. Zeigler *et. al.* [ZKP00]. Cette théorie propose le formalisme DEVS comme cadre intégrateur pour la spécification de systèmes. Nous avons donc proposé une analogie des SMAs réactifs situés vers DEVS. Il existait déjà des formalisations DEVS de SMAs mais, à notre connaissance, il n'y a pas de travaux qui proposent clairement de définir une telle analogie. Dans ce travail sur l'intégration formelle, nous proposons également la simulation à évènements discrets comme une solution pour coupler deux systèmes

à des échelles de temps très différentes. Le concept d'évènements discrets permet en effet d'effectuer des « sauts irréguliers dans le temps » qui autorisent la cohabitation de phénomènes lents et rapides relativement les uns aux autres.

L'originalité du *framework* que nous avons proposé dans le deuxième thème se situe essentiellement au niveau modèle. Nous développons la notion de *wrapping* à l'aide d'une interface fonctionnelle, ce qui permet potentiellement de coupler un simulateur issu d'un paradigme quelconque, avec d'autres simulateurs issus d'autres paradigmes. Cette interface fonctionnelle permet la communication des modèles avec un coordinateur DEVS qui assure notamment la cohérence globale de la simulation au niveau du temps simulé.
Nous avons défini l'application MLMC basée sur XML qui offre une syntaxe opérationnelle pour la description des modèles et de leur couplage. Cette application permet d'encapsuler des éléments de sémantique au niveau du couplage des modèles.
Nous avons également abordé le concept d'expériences virtuelles en considérant les simulateurs comme des systèmes réels. L'utilisation classique de ce concept est de vouloir comprendre le système modélisé. Dans ce contexte, nous avons élaboré une syntaxe XML (MLVE) qui permet de représenter et d'échanger des plans d'expériences.

Ce concept nous a également permis d'élaborer une méthode pour simuler le transfert d'échelles entre différents niveaux d'organisations. Nous pensons qu'elle peut être appliquée dans de nombreux domaines. Nous avons illustré cette idée avec le troisième thème.
En identifiant un modèle fonctionnel connu de l'ingestion de phytoplancton par un copépode à partir de simulations d'un modèle d'agents réactifs, nous apportons un argument supplémentaire à l'utilisation des SMAs pour la simulation centrée individus en écologie. Les SMAs offrent une modélisation discrète des états, des entités et surtout des interactions, ce qui est impossible dans un modèle totalement continu. Or, nous avons vu que la nature discrète des interactions au niveau individuel a des conséquences importantes sur la dynamique globale du système.

6.3 Perspectives

Nous avons défini le cadre de réalisation d'une plateforme d'intégration de modèles hétérogènes. Nous voulons produire une version finalisée de cet outil à court terme. Ce travail a commencé l'année dernière dans le cadre de la thèse de G. Quesnel, au LIL. Aux briques de base existantes comme les connecteurs ou les coordinateurs DEVS, se sont ajoutés les *wrappers* pour les réseaux de Petri et les équations différentielles ordinaires. Nous pouvons d'ores et déjà coupler ces deux formalismes à l'intérieur de notre *framework*.

Dans le cadre de la simulation informatique des systèmes complexes, nous pensons qu'il est nécessaire de développer la notion de plans d'expériences pour une meilleure confrontation des résultats. Nous avons commencé à travailler sur ces questions en collaboration avec le Cemagref de Clermont-Ferrand.

Le travail sur les aspects sémantiques du couplage nous semble un point fondamental dans la perspective d'un couplage automatique (ou semi-automatique) des modèles de simulation. Nous devons développer des collaborations avec des chercheurs dans le domaine de l'intégration de bases de données hétérogènes. De telles recherches sont menées au LIL, et nous pouvons espérer des avancées dans cette direction.

Notre travail sur l'intégration formelle nous a amené à des réflexions sur les SMAs et notamment sur l'utilisation de DS-DEVS pour leur formalisation. Nous aimerions poursuivre dans cette voie en considérant les différents types d'environnement des agents. Dans ce cadre, le couplage des SMAs avec des équations différentielles spatialisées nous semble un travail intéressant. Nous avons déjà des pistes avec l'utilisation de méthodes de résolution de systèmes d'équations différentielles basées sur DEVS, comme QSS par exemple[93].

Notre application nous a amenés à réfléchir sur les notions de hiérarchie, de transfert d'échelles et de construction automatique de modèles. Nous avons commencé une réflexion sur ce sujet et nous voulons la poursuivre avec les membres de l'équipe MESC en travaillant sur la génération automatique de modèles par programmation génétique : un des objectifs du projet est la simplification de modèles résultant du couplage de modèles plus élémentaires, afin de permettre leur simulation avec un coût machine accessible.

Concernant les SMAs, qui intègrent un grand nombre de techniques et de formalismes, une question se pose alors. Pouvons-nous les considérer comme des multi-modèles ? Quoi qu'il en soit, les SMAs comme les multi-modèles sont bien placés pour la modélisation et la simulation des systèmes complexes. Nous pourions même aller un peu plus loin et considérer la modélisation en informatique comme l'instance d'une véritable science de la complexité.

[93] À l'heure actuelle, un étudiant de DEA travaille sur ce sujet

A

Document Type Definition

A.1 DTD pour MLMC (Meta Language for Model Coupling)

```
<!DOCTYPE MODEL [

<!ELEMENT MODEL (DESCRIPTION, TIME?, SPACE?, INIT?, IN?, OUT?, STATE?,
                        (SUBMODELS+, CONNECTIONS+)?)>

<!ELEMENT DESCRIPTION (#PCDATA)>

<!ELEMENT TIME (TIME_SPANS?, ORDER*, TIME_BASE*)>

<!ELEMENT TIME_SPANS (TIME_SPAN+)>

<!ELEMENT TIME_SPAN EMPTY>

<!ELEMENT ORDER (RELATION+)>

<!ELEMENT RELATION EMPTY>

<!ELEMENT TIME_BASE EMPTY>

<!ELEMENT SPACE (PLACES?, NEIGHBOURHOOD*, REFERENTIAL*, DISTANCE*)>

<!ELEMENT PLACES (PLACE+)>

<!ELEMENT PLACE EMPTY>

<!ELEMENT NEIGHBOURHOOD (NEIGHBOUR)>

<!ELEMENT NEIGHBOUR (#PCDATA)>

<!ELEMENT REFERENTIAL (AXIS+)>

<!ELEMENT AXIS EMPTY>

<!ELEMENT DISTANCE EMPTY>
```

```
35
    <!ELEMENT INIT (PORT+)>

    <!ELEMENT STATE (PORT+)>

40  <!ELEMENT IN (PORT+)>

    <!ELEMENT OUT (PORT+)>

    <!ELEMENT PORT (DATA+)>
45
    <!ELEMENT DATA (TYPE?, UNIT?, TIME_REF?, SPACE_REF?, METADATA?,
                                                  CONTENT?, AGREGATE?)>
    <!ELEMENT UNIT EMPTY>

50  <!ELEMENT TIME_REF EMPTY>

    <!ELEMENT SPACE_REF EMPTY>

    <!ELEMENT METADATA EMPTY>
55
    <!ELEMENT TYPE EMPTY>

    <!ELEMENT CONTENT (#PCDATA)>

60  <!ELEMENT AGREGATE (DATA+)>

    <!ELEMENT SUBMODELS (LI+)>

    <!ELEMENT LI EMPTY>
65
    <!ELEMENT CONNECTIONS (PORT_REF+, CONNECTION+)>

    <!ELEMENT PORT_REF EMPTY>

70  <!ELEMENT CONNECTION EMPTY>

    <!ATTLIST MODEL         name CDATA #REQUIRED
                            type (atomic | coupled) #REQUIRED
                            autonomous (yes | no) #REQUIRED
75                          xmlns:xlink CDATA
                                  #FIXED 'http://www.w3.org/1999/xlink'>

    <!ATTLIST TIME          type (set | ordinal | cardinal) #REQUIRED>

80  <!ATTLIST TIME_SPAN     name  CDATA #REQUIRED
                            begin CDATA #IMPLIED
                            end   CDATA #IMPLIED>

    <!ATTLIST RELATION      sequence CDATA #REQUIRED>
85
    <!ATTLIST TIME_BASE     type (discrete | continuous) #REQUIRED
                            unit  CDATA #REQUIRED
                            begin CDATA #REQUIRED
```

```
                          end   CDATA #REQUIRED
                          step  CDATA #IMPLIED>

    <!ATTLIST SPACE       name CDATA #REQUIRED
                          type (topological | metric | set) #REQUIRED>

    <!ATTLIST PLACE       name  CDATA #REQUIRED
                          begin CDATA #IMPLIED
                          end   CDATA #IMPLIED>

    <!ATTLIST NEIGHBOUR   link CDATA #IMPLIED
                          type (von_neumman | moore | none) #IMPLIED>

    <!ATTLIST REFERENTIAL type (discrete | continuous) #REQUIRED
                          dimension CDATA #REQUIRED>

    <!ATTLIST AXIS        id   CDATA #REQUIRED
                          min  CDATA #REQUIRED
                          max  CDATA #REQUIRED
                          step CDATA #IMPLIED>

    <!ATTLIST DISTANCE    type CDATA #REQUIRED>


    <!ATTLIST DATA        name CDATA #REQUIRED
                          xlink:type (extended) #FIXED 'extended'>

    <!ATTLIST TYPE        class CDATA #REQUIRED>

    <!ATTLIST UNIT        class CDATA #REQUIRED
                          power CDATA #REQUIRED>

    <!ATTLIST TIME_REF    xlink:type (locator) #FIXED 'locator'
                          xlink:href CDATA #REQUIRED>

    <!ATTLIST SPACE_REF   xlink:type (locator) #FIXED 'locator'
                          xlink:href CDATA #REQUIRED>

    <!ATTLIST METADATA    xlink:type (locator) #FIXED 'locator'
                          xlink:href CDATA #REQUIRED>

    <!ATTLIST CONTENT     dimension CDATA #REQUIRED
                          size CDATA #REQUIRED>

    <!ATTLIST PORT        name CDATA #REQUIRED>

    <!ATTLIST LI          xlink:type (simple) #FIXED 'simple'
                          xlink:href CDATA #REQUIRED>

    <!ATTLIST PORT_REF    xlink:type (locator) #FIXED 'locator'
                          xlink:label CDATA #REQUIRED
                          xlink:href CDATA #REQUIRED>

    <!ATTLIST CONNECTIONS xlink:type (extended) #FIXED 'extended'>
```

```
    <!ATTLIST CONNECTION    type (EIC | IOC | IC) #REQUIRED
145                         xlink:type (arc) #FIXED 'arc'
                            xlink:from CDATA #REQUIRED
                            xlink:to CDATA #REQUIRED>      ]>
```

A.2 DTD pour MLVE (Meta Language for Virtual Experiments)

```
 1  <!DOCTYPE EXPERIMENT [

    <!ELEMENT EXPERIMENT (NOTES, MODEL+)>

 5  <!ELEMENT NOTES (#PCDATA)>

    <!ELEMENT MODEL (DESCRIPTION, EXECUTABLE, OUTPUT_STREAM, EXECUTION,
                     EXPERIMENTAL_CONDITIONS, MEASURES)>

10  <!ELEMENT DESCRIPTION EMPTY>

    <!ELEMENT EXECUTABLE EMPTY>

    <!ELEMENT OUTPUT_STREAM EMPTY>
15
    <!ELEMENT EXECUTION (EXECUTION_NODE+)>

    <!ELEMENT EXECUTION_NODE EMPTY>

20  <!ELEMENT EXPERIMENTAL_CONDITIONS (CONDITION+)>

    <!ELEMENT CONDITION ((SET | INTERVAL)?, CONSTRAINST?, RANDOM?)>

    <!ELEMENT INTERVAL EMPTY>
25
    <!ELEMENT SET (ITEM+)>

    <!ELEMENT ITEM EMPTY>

30  <!ELEMENT RANDOM EMPTY>

    <!ELEMENT CONSTRAINST (GREATER_THAN*, LOWER_THAN*, EQUAL_TO*,
                           GREATER_EQUAL_THAN*, LOWER_EQUAL_THAN*)>

35  <!ELEMENT GREATER_THAN (MULT | ADD | DIV | MINUS | EXP | SQRT |
                           LOG | SIN | COS)>

    <!ELEMENT LOWER_THAN (MULT | ADD | DIV | MINUS | EXP | SQRT | LOG |
                           SIN | COS)>
40
    <!ELEMENT EQUAL_TO (MULT | ADD | DIV | MINUS | EXP | SQRT | LOG |
                           SIN | COS)>
```

```
    <!ELEMENT GREATER_EQUAL_THAN (MULT | ADD | DIV | MINUS | EXP | SQRT |
45                          LOG | SIN | COS)>

    <!ELEMENT LOWER_EQUAL_THAN (MULT | ADD | DIV | MINUS | EXP | SQRT |
                            LOG | SIN | COS)>

50  <!ELEMENT MULT (CONST*, VAR*)>

    <!ELEMENT ADD (CONST*, VAR*)>

    <!ELEMENT DIV (CONST*, VAR*)>
55
    <!ELEMENT EXP (CONST*, VAR*)>

    <!ELEMENT MINUS (CONST*, VAR*)>

60  <!ELEMENT SQRT (CONST*, VAR*)>

    <!ELEMENT LOG (CONST*, VAR*)>

    <!ELEMENT SIN (CONST*, VAR*)>
65
    <!ELEMENT COS (CONST*, VAR*)>

    <!ELEMENT CONST EMPTY>

70  <!ELEMENT VAR EMPTY>

    <!ELEMENT MEASURES (MEASURE+)>

    <!ELEMENT MEASURE EMPTY>
75
    <!ATTLIST EXPERIMENT    name CDATA #REQUIRED
                            date CDATA #REQUIRED>

    <!ATTLIST MODEL         xmlns:xlink CDATA #FIXED
80                                ’http://www.w3.org/1999/xlink’
                            xlink:type (extended) #FIXED ’extended’>

    <!ATTLIST DESCRIPTION   xlink:type (locator) #FIXED ’locator’
                            xlink:href CDATA #REQUIRED>
85
    <!ATTLIST EXECUTABLE    xlink:type (locator) #FIXED ’locator’
                            xlink:href CDATA #REQUIRED>

    <!ATTLIST OUTPUT_STREAM xlink:type (locator) #FIXED ’locator’
90                          xlink:href CDATA #REQUIRED>

    <!ATTLIST EXECUTION     type (mono | distributed | parralel) #REQUIRED>

    <!ATTLIST EXECUTION_NODE xlink:type (locator) #FIXED ’locator’
95                           xlink:href CDATA #REQUIRED>

    <!ATTLIST EXPERIMENTAL_CONDITIONS replicat CDATA #REQUIRED
```

```
                                apply (sequential | cartesian) #REQUIRED>

<!ATTLIST CONDITION     xmlns:pathtoport CDATA #FIXED
                              'http://www.w3.org/1999/xlink'
                        pathtoport:type (simple) #FIXED 'simple'
                        pathtoport:href CDATA #REQUIRED>

<!ATTLIST INTERVAL      begin CDATA #REQUIRED
                        end CDATA #REQUIRED
                        step CDATA #IMPLIED>

<!ATTLIST RANDOM        function (uniform | normal | lognormal |
                        exponential | poisson | beta | gamma) #REQUIRED
                        average CDATA #IMPLIED
                        standard_deviation CDATA #IMPLIED
                        a CDATA #IMPLIED
                        b CDATA #IMPLIED>

<!ATTLIST ITEM          value CDATA #REQUIRED>

<!ATTLIST CONST         value CDATA #REQUIRED>

<!ATTLIST VAR           pathtoport:type (simple) #FIXED 'simple'
                        pathtoport:href CDATA #REQUIRED>

<!ATTLIST MEASURES      xmlns:pathtoport CDATA #FIXED
                                        'http://www.w3.org/1999/xlink'
                        pathtoport:type (extended) #FIXED 'extended'>

<!ATTLIST MEASURE       pathtoport:type (locator) #FIXED 'locator'
                        pathtoport:href CDATA #REQUIRED
                        frequence CDATA #IMPLIED
                        type (active | passive) #IMPLIED
                        apply CDATA #IMPLIED>
]>
```

B

Modèle mathématique de l'ingestion de proies par le copépode

Le modèle présenté ici a été développé par P. Caparroy [CC96]. Il synthétise les différents modèles développés jusqu'à présent en résumant à l'aide de cinq équations différentielles interdépendantes l'activité métabolique du copépode à partir de l'ingestion de proies. Le modèle est le suivant :

$$\frac{dX_1}{dt} = I - A - F \tag{B.1}$$
$$\frac{dX_2}{dt} = A - \frac{C}{M_N} \tag{B.2}$$
$$\frac{dX_3}{dt} = F - G \tag{B.3}$$
$$\frac{dX_4}{dt} = G \tag{B.4}$$
$$\frac{dX_5}{dt} = C \tag{B.5}$$

Les deux quantités qui nous intéressent plus particulièrement sont décrites par $X1$ et $X2$. La première constitue le chyme alimentaire (*i.e.* la quantité de proies en cours de digestion dans l'estomac) soumis au processus de digestion. La seconde représente la fraction de chyme ayant franchi la barrière intestinale et mise à disposition des processus métaboliques. C'est donc l'énergie disponible pour le copépode. La liste suivante nous donne la signification de l'ensemble des variables du modèle, les concentrations sont exprimées en azote et le temps en seconde :

X_1 concentration en proies dans l'estomac,
X_2 concentration en proies assimilées,
X_3 concentration en pelotes fécales dans le tractus digestif,
X_4 concentration en pelotes fécales évacuées,
X_5 énergie dépensée exprimée en azote,
I taux d'ingestion,
A taux d'assimilation,

F taux de formation de pelotes fécales,
G taux d'égestion,
C taux d'excrétion total,
M_N masse en azote de la proie,

La variable I décrit le processus de capture du copépode en fonction de sa satiété et de l'environnement. I est alors directement liée à la capacité de rencontre du copépode avec ses proies. Dans le modèle original, cette variable est calculée par la fonction mathématique suivante :

$$I = (\beta_{behaviour} + \beta_{turbulence} N_p F_A) \tag{B.6}$$

avec :

- $\beta_{behaviour}$ la contribution du « comportement ». Celle-ci est fonction du rayon de perception du copépode, de la taille des proies et la différence de vitesse de nage entre le copépode et les proies,
- $\beta_{turbulence}$ la contribution de la turbulence,
- N_p la densité des proies dans le milieu,
- F_A une mesure de l'activité de nutrition dépendant de la quantité de nourriture dans l'estomac du copépode et de la densité des proies dans le milieu.

Dans le modèle original, ce sont les expressions de $\beta_{behaviour}$ et $\beta_{turbulence}$ qui modélisaient le « comportement » du copépode. Dans notre travail, nous avons choisi une approche mécaniste en modélisant explicitement l'évènement de rencontre entre proies et prédateurs (voir annexe C pour les implications sur le modèle mathématique). Nous avons choisi de conserver trois des processus modélisés par P. Caparoy : la satiété, la vidange de l'estomac et l'excrétion. Ce sont ces processus qui gouvernent « l'appétit » du copépode. Le tableau B.1 présente les processus modélisés. Les valeurs des différents paramètres sont données par le tableau B.2.

TAB. B.1 – Expressions mathématiques des processus représentés dans le modèle (d'après [CC96]).

Satiété	
Fonction de satiété	$C_s = 1 - \left(\frac{V_p \frac{X_1}{M_N}}{0{,}66 V_i} \right)^2$
Vidange de l'estomac	
Taux d'assimilation	$A = C_a Z_i$
Efficacité d'assimilation	$C_a = 1 - e^{-aT_i}$
Taux de vidange de l'intestin	$Z_i = \frac{X_1}{T_i}$
Temps de transit intestinal	$T_i = \frac{t_{min} t_{max}}{\left(\frac{X_1 V_p}{V_i} (t_{max} - t_{min}) \right) + t_{min}}$
Taux de formation des pelotes fécales	$F = 1 - C_a Z_i$
Taux d'egestion (élimination de matière azotée)	$G = \frac{V_{pf}}{V_p dt}$ si $V_p X_3 < V_{pf}$
	$G = 0$ si $V_p X_3 \geq V_{pf}$
Excretion	
Taux d'excrétion total	$C = C_{st} + C_{sda} + C_n$
Excrétion liée à l'activité métabolique standard	$C_{st} = f_1 W$
Excrétion liée à la digestion et à la croissance	$C_{sda} = f_2 A M_N$
Prédation suspensivor	$C_n = \frac{2 N_m}{C_{cal} R_{ON} V_{mol}} Z(U)$
Coût catabolique de la nage à une vitesse U	$Z(U) = \frac{P(U)}{E_{mec} E_m}$
Puissance mécanique d'un copépode nageant à la vitesse U	$P(U) = \frac{k}{2} \rho^{1-n} L^{-n} U^{3-n} \mu^n$

TAB. B.2 – Valeurs des paramètres du modèle (d'après [CC96]).

Symbole	**Paramètre**	**Valeur**	**Unité**
a	Taux de digestion spécifique du contenu azoté des proies	$4,4.10^{-4}$	s^{-1}
t_{min}	Temps de transit intestinal minimal	2100	s
t_{max}	Temps de transit intestinal maximal	3900	s
V_p	Volume d'une proie	$7,5$	μm^3
V_i	Volume de l'intestin moyen	$3,5.10^{-6}$	cm^3
Z_{pf}	Volume d'une pelote fécale	2.10^{-7}	cm^3
f_1	Coefficient de proportionnalité entre le poids et le métabolisme standard	35.10^{-6}	min^{-1}
f_2	Coefficient d'action dynamique spécifique	$0,28$	sans dimension
W	Poids d'azote de *A. Tonsa*	738	ng
V_p	Volume de la proie	$1,7.10^{-6}$	mm^3
M_N	Masse d'azote de la proie *T. weissflogii*	25	pg
N_m	Masse molaire de l'azote	14	g
C_{cal}	Coeffcient oxycalorifique	$20,3$	$KJ.lO_2^{-1}$
R_{ON}	Rapport atomique Oxygène consommé / Azote ammoniacal excrété	8	sans dimension
V_{mol}	Volume molaire d'un gaz parfait	$22,4$	l
E_{mec}	Efficacité mécanique de la nage d'un copépode	$0,3$	sans dimension
E_n	Efficacité musculaire (métabolique) d'un copépode	$0,25$	sans dimension
k	Coefficient empirique reliant le coefficient de frottement au nombre de Reynolds	$85,2$	sans dimension
ρ	Densité de l'eau de mer	$1,024$	$g.cm^3$
n	Coefficient empirique reliant le coefficient de frottement au nombre de Reynolds	$0,8$	sans dimension
L	Longeur céphalotoracique de *A. Tonsa*	820	μm
U	Vitesse de nage de *A. Tonsa*	$0,2$	$cm.s^{-1}$
μ	Viscosité dynamique de l'eau de mer	119.10^{-4}	$gcm^{-1}s^{-1}$

C

Résolution analytique du modèle d'ingestion

Dans cette annexe, nous présentons la résolution analytique du système d'équations différentielles présenté en annexe B, qui fournit également les valeurs des différents paramètres. Le but de cette résolution est d'éviter l'intégration numérique du système d'équations différentielles afin de conserver une approche entièrement évènementielle de notre modélisation du comportement alimentaire du copépode. En effet, la résolution analytique va nous permettre de calculer à l'aide d'une fonction les dates d'évènements de satiété et de mort du modèle à évènements discrets.

Entrée en état « satiété »

À partir de la fonction de satiété définie par P. Caparroy, nous pouvons définir deux conditions qui activent ou désactivent l'état de satiété comme suit :

$$\begin{cases} s = 0 & si\ \alpha > 0 \\ s = 1 & si\ \alpha \leq 0 \end{cases} \qquad avec \qquad \alpha = 1 - \left(\frac{V_p \frac{X_1}{M_N}}{0,66 V_i} \right) \tag{C.1}$$

Cette équation modélise le fait que si le volume de proies dans l'estomac du copépode dépasse $\frac{2}{3}$ du volume de l'estomac, alors le copépode n'a plus faim ($s = 1$) et réciproquement ($s = 0$). Dans notre modélisation évènementielle, l'ingestion correspond à un évènement de capture d'une cellule de phytoplancton. Ainsi, à chaque capture, X_1 augmente d'une unité de masse en azote d'une cellule de phytoplancton. Cette nouvelle valeur de X_1 nous permet de savoir si le copépode est en état de satiété en remplaçant X_1 dans l'équation C.1.

Sortie de l'état « satiété »

Entre deux évènements de capture, la variable X_1 est dépendante des processus liés à l'ingestion et à la digestion. En observant le système d'équations différentielles (voir annexe B), nous nous apercevons que X_1 ne dépend d'aucune autre équation. Ceci est

vrai si nous retirons le processus d'ingestion (I) de son expression. L'équation B.1 de l'annexe B devient alors :

$$\begin{aligned} \frac{dX_1}{dt} &= -A - F \\ &= -(C_a Z_i + (1 - C_a) Z_i \\ &= -Z_i \\ &= -\frac{X_1}{T_i} \end{aligned}$$

d'où :

$$\frac{dX_1}{dt} = -\frac{X_1}{\frac{t_{min} t_{max}}{\frac{X_1 V_p}{V_i}(t_{max} - t_{min}) + t_{min}}}$$

Nous désirons trouver une expression analytique pour X_1. Posons :
$a = t_{min} t_{max}$
$b = \frac{V_p}{V_i}(t_{max} - t_{min})$
$c = t_{min}$

alors

$$\frac{dX_1}{dt} = -\frac{X_1(X_1 b + c)}{a}$$

L'équation peut se réécrire de la manière suivante :

$$\frac{b}{a} X_1^2(t) + \frac{c}{a} X_1(t) + \dot{X}_1(t) = 0 \tag{C.2}$$

avec $\dot{X}_1$ la dérivée de X_1 par rapport au temps.

Le théorème de Ricatti nous dit que si une équation est de la forme $\dot{y} = A(t)y^2 + B(t)y$ alors, en posant $\omega = (y - y_1)^{-1}$ avec y_1 une solution particulière, il suffit de trouver la solution de l'équation en y de la forme :

$$\dot{\omega} + (B(t) + 2y_1 A(t))\omega = -A(t)$$

Dans notre cas, $y = X_1(t)$, il nous faut donc trouver y_1. Nous pouvons déduire une solution évidente de l'équation C.2 tel que :

$$\dot{X}1 = 0 \qquad si \qquad X_1 = -\frac{c}{b} = y_1$$

En identifiant $A(t)$ et $B(t)$, nous obtenons :

$$\dot{\omega} + \left(\frac{c}{a} - 2\frac{c}{b}\frac{b}{a}\right)\omega = \frac{b}{a} \qquad d'où \qquad \dot{\omega} - \frac{c}{a}\omega + \frac{b}{a} = 0$$

Il existe une solution évidente $\omega(t) = Ke^{-\frac{c}{a}t} + \tau$ où τ est une solution particulière de l'équation en ω tel que $\tau = \frac{b}{c}$. Nous pouvons maintenant écrire la solution analytique de $\dot{X}_1$. Nous savons que :

$$\omega(t) = X_1(t) + \frac{c}{b} \qquad d'où \qquad X_1(t) = -\frac{c}{b} + \frac{1}{\omega(t)}$$

ce qui nous amène à l'expression de $X_1(t)$ suivante :

$$X_1(t) = -\frac{c}{b} + \frac{1}{\frac{b}{c} + Ke^{-\frac{c}{a}t}} \tag{C.3}$$

Nous avons maintenant une expression analytique de $X_1(t)$. Cette fonction ne dépend que du temps et d'une constante K qui correspond à la condition initiale. Pour calculer la valeur de K, nous considérons la valeur de X_1 à $t = 0$ correspondant à l'évènement de capture d'une proie. Ainsi, à chaque évènement de capture, une nouvelle valeur de K est déterminée. La forme de la fonction $X_1(t)$ ne change pas. Nous pouvons donc connaître la valeur de $X_1(t)$ $\forall t$ entre deux évènements de capture. Ainsi, nous pouvons calculer la date à laquelle le copépode cessera d'être en état de satiété si aucun évènement de capture ne survient à l'aide de l'équation suivante :

$$si\ s = 1 \qquad alors \qquad d = -\left[\frac{a}{c}\left(ln\left(\frac{1}{\frac{0.66V_iM_N}{Vp} + \frac{c}{b}}\right) - ln(K)\right)\right] \tag{C.4}$$

où d est la date de sortie de l'état de satiété.

Entrée en état « mort »

Pour calculer la date de mort du copépode, nous devons être capables de connaître la durée nécessaire pour que le niveau d'énergie interne du copépode devienne nul. L'équation B.2 de l'annexe B décrit l'évolution du niveau d'énergie mis à disposition des processus métaboliques du copépode (variable X_2). Connaissant la forme analytique de X_1, nous sommes capables de trouver la forme analytique de X_2. Pour cela, nous devons également être capables de donner la forme analytique de X_5(équation B.5). Les détails des calculs sont décrits dans un rapport interne du laboratoire [DRG02]. Nous donnons ici les résultats en commençant par l'énergie dépensée (X_5) :

$$X_5(t) = -f_2X_1(t) + C_{st}t - f_2\frac{a^2}{b}\frac{1}{a\frac{ab+acKe^{-\frac{c}{a}t}}{bc}}e^{-a\frac{ab+acKe^{-\frac{c}{a}t}}{bc}} - f_2E_i\left(-\frac{a^2b + acKe^{-\frac{c}{a}t}}{bc}\right) + cst \tag{C.5}$$

Connaissant $X_5(t)$, il nous a été possible de définir l'équation qui décrit la variation au cours du temps de l'énergie disponible (X_2) comme suit :

$$\begin{aligned} X_2(t) \;=\; & -(1-f_2)X_1(t) + C_{st}t - (1-f_2)\frac{a^2}{b}\frac{1}{a\frac{ab+acKe^{-\frac{c}{a}t}}{bc}} \\ & e^{-a\frac{ab+acKe^{-\frac{c}{a}t}}{bc}} - (1-f_2)E_i\left(-\frac{a^2b+acKe^{-\frac{c}{a}t}}{bc}\right) + cst \end{aligned} \tag{C.6}$$

Pour calculer la date de mort du copépode, il est nécessaire de trouver t tel que $X_2(t) = 0$. Pour cela, nous avons implémenté un algorithme de recherche dichotomique.

L'équation qui décrit l'évolution de la concentration en pelotes fécales dans le tractus digestif (X_3) et celle qui donne l'évolution de la concentration en pelotes fécales évacuées (X_4) sont ignorées. En effet, elles n'interviennent pas dans le calcul de la date de passage en état de satiété ou en état de mort. Nous en avons déterminé les formes analytiques [DRG02]. Il est donc possible de calculer ces valeurs pour de futures expériences où elles seraient utiles (par exemple pour un couplage avec un modèle de croissance du phytoplancton ou l'excrétion participerait à l'enrichissement en azote du milieu).

D

Simulateurs abstraits pour DEVS

D.1 Simulateur abstrait pour un modèle DEVS atomique

Nous présentons le ici le simulateur abstrait d'un modèle atomique.
Le signe « := » est l'affectation, « `Fonct_Sortie` » correspond à λ, « `Trans_int` » à δ_{int} et « `Trans_ext` » à δ_{ext} dans un modèle DEVS atomique. « // » précède un commentaire (d'après [ZKP00]).

```
DEBUT simulateur devs

    DEBUT déclaration variables
        parent // le coordinateur
        S      // l'ensemble des états
        tl     // la date d'entrée dans l'état s
        tn     // la date du prochain évènement
        devs   // le nom du modèle DEVS associé au simulateur
        e      // le temps passé dans l'état s
               // (s,e) l'état total de ce modèle
        y      // la valeur courante de sortie du modèle
    FIN déclaration variables

    QUAND réception d'un évènement d'initialisation (i,t)
        tl := t
        tn := tl + ta(s)
    FIN QUAND

    QUAND réception d'un évènement de transition externe : (x,t)
        SI tl <= t <= tn ALORS
            e  := t - tl
            s  := Trans_ext(s,e,x) // calcul de la fonction de transition externe
                                   // x désigne le couple (port,valeur)
            tl := t
            tn := tl+ta(s)
        SINON erreur
        FIN SI
    FIN QUAND
```

```
30      QUAND réception d'un évènement de transition interne (*,t)
            SI t = tn ALORS
                y  := Fonct_Sortie(s)
                Envoyer au parent l'évènement de sortie (y,t)
                s  := Trans_int(s)
35              tl := t
                tn := tl+ta(s)
            FIN SI
            SINON erreur
        FIN QUAND
40
    FIN Simulateur devs
```

D.2 Simulateur abstrait pour un modèle DEVS couplé

Nous présentons le simulateur abstrait d'un coordinateur DEVS. Ce simulateur correspond au modèle DEVS couplé. Nous avons adapté l'algorithme décrit dans [ZKP00] puisque nous n'utilisons pas la fonction de sélection. De plus, pour une question de lisibilité, nous avons utilisé les notations formelles au minimum.
« :=» est le signe d'affectation, « //» précède un commentaire (algorithme adapté d'après [ZKP00]).

```
 1  DEBUT coordinateur

        DEBUT déclaration variables
            parent     // le coordinateur parent
 5          D          // ensemble des modèles composants
            IC         // ensemble des connexions internes
            EOC        // ensemble des connexions externes en sortie
            EIC        // ensemble des connexions externes en entrée
            tl         // la date du dernier évènement
10          tn         // la date du prochain évènement
            Echéancier // liste triée des couple (d,tn) d appartient à D
        FIN déclaration variables

        QUAND réception l'évènement d'initialisation (i,t)
15          Envoyer à D l'évènement d'initialisation (i,t)
            Trier Echéancier       // en fonction de tn de d appartient à D
            tl := maximum(tl de d) // d appartient à D
            tn := minimum(tn de d) // d appartient à D
        FIN QUAND
20
        QUAND réception d'un évènement de transition interne (*,t)
```

```
            SI t = tn ALORS
                d = dépiler Echénancier
                Envoyer l'événement (*,t) à d
                Trier Echéancier       // en fonction de tn de d appartient à D
                tl := maximum(tl de d) // d appartient à D
                tn := minimum(tn de d) // d appartient à D
            SINON erreur
            FIN SI
        FIN QUAND

        QUAND réception d'un évènement de transition externe (x,t)
                                       // x désigne le couple (port,valeur) en entrée
            SI tl <= t <= tn ALORS
                POUR d dans EIC associé à x
                    Envoyer l'évènement (x,t) à d
                FIN POUR
                Trier Echéancier       // en fonction de tn de d appartient à D
                tl := maximum(tl de d) // d appartient à D
                tn := minimum(tn de d) // d appartient à D
            SINON erreur
            FIN SI
        FIN QUAND

        QUAND réception d'un évènement de sortie (y,t) du modèle d
                                       // d appartient à D
                                       // y désigne le couple (port,valeur)
                                       // en sortie d'un modèle composant
            POUR i dans EOC associé à y// i appartient à D
               Envoyer l'événement de sortie (y,t) à parent
            FIN POUR

            POUR i dans IC associé à y // i appartient à D
               Envoyer l'événement de transition externe (x,t) au modèle i
            FIN POUR
        FIN QUAND

    FIN coordinateur
```

D.3 Simulateur abstrait pour le coordinateur racine

Le coordinateur racine correspond à la boucle générale de simulation d'un modèle DEVS.
« :=» est le signe d'affectation, « //» précède un commentaire (d'après [ZKP00]).

```
1   DEBUT coordinateur racine

        DEBUT déclaration variables
            t      \\ date courante de la simulation
5           enfant \\ simulateur ou coordinateur subordonné direct
        FIN déclaration de variables

        t := t0    \\ intialisation du temps

10      Envoyer évènement d'initialisation (i,t) à Enfant
        t := tn de Enfant \\ réception de (Prochain,tn)

        TANT QUE t <= durée de la simulation FAIRE
            Envoyer l'évènement de transition interne (*,t) à enfant
15          t := tn de Enfant
        FIN TANT QUE
    FIN  coordinateur racine
```

E

Modèles formels

E.1 Modèle couplé de l'agent copépode

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

1. activité ($Acti$),
2. perception ($Perc$),
3. gestion des rebonds ($Rebo$),
4. gestion de l'énergie ($Ener$),
5. direction aléatoire ($Alea$).

Ce modèle possède la structure suivante :

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

où :

$X = \{\ (incop_1, liste(L)), (incop_2, limites(x, y, z))\ \}$

$Y = \{\ (outcop_1, liste?(P)), (outcop_2, limites?(P, \vec{D})), (outcop_3, eat(p))\ \}$

$D = \{Acti, Perc, Rebo, Ener, Alea\}$

$EIC = \{\ ((Copepode, incop_1), (Perc, inper_1)),$
$\quad ((Copepode, incop_2), (Rebo, inreb_1))\ \}$

$EOC = \{\ ((Perc, outper_2), (Copepode, outcop_1)),$
$\quad ((Rebo, outreb_2), (Copepode, outreb_2)),$
$\quad ((Acti, outact_4), (Copepode, outcop_3))\ \}$

$IC = \{\ ((Perc, outper_1), (Acti, inact_1)),$
$\quad ((Rebo, outreb_1), (Acti, inact_2)),$
$\quad ((Alea, outale_1), (Acti, inact_5)),$
$\quad ((Alea, outale_2), (Reb, inreb_3)),$
$\quad ((Acti, outact_1), (Perc, inper_2)),$
$\quad ((Acti, outact_2), (Rebo, inreb_2)),$
$\quad ((Acti, outact_3), (Ener, inene)),$
$\quad ((Ener, outene_1), (Acti, inact_3)),$

$((Ener, outene_2), (Acti, inact_4))$ }

E.2 Modèle atomique de perception

Le modèle de perception est un modèle DEVS atomique avec la structure suivante :

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

où :
$X =$ { $((inper_1, vu?(P, \vec{D}, \Phi)), (inper_2, liste(L)))$ }
$Y =$ { $((outper_1, liste?(P, \vec{D})), (outper_2, vu(A, \Phi)))$ }
$S = (\phi, p, \vec{d})$
où ϕ est le n-uplet $\{idle, question, requete, reponse\}$

Les fonctions d'avancement du temps sont :

$ta(idle, p, \vec{d}) = \infty$: le modèle attend un évènement.
$\delta_{ext}((Idle, p, \vec{d}), vu?(P, \vec{D}, \Phi)) = (question, P, \vec{D})$
$ta(question, P, \vec{D}) = 0$
$\lambda(question, P, \vec{D}) = (liste?(P, \vec{D}))$
$\delta_{int}(question, P, \vec{D}) = (requete, P, \vec{D})$
$ta(requete, P, \vec{D}) = \infty$
$\delta_{ext}((requete, P, \vec{D}), liste(L)) = (reponse, P, \vec{D})$
$ta(reponse, P, \vec{D}) = 0$
$\delta_{int}(reponse, P, \vec{D}) = (idle, P, \vec{D})$

Nous exprimons ici le calcul des attributs de la fonction de sortie :

$\lambda(reponse) = vu(A', chasse)$ si $\exists\ A\ /A \in L,\ \overrightarrow{PA}.\vec{D} \geq 0\ ,\ \|\overrightarrow{PA}\| < r$
où A est la position de la proie, L une liste de positions de cellules envoyée par l'environnement, P la position du copépode, r la distance de perception, $A' = P + \vec{d}.(\|\overrightarrow{pA}\| - r')$ et r' la distance de capture.

$\lambda(reponse) = vu(A', capture)$ si $\exists\ A\ /A \in L,\ \vec{PA}.\vec{D} \geq 0\ ,\ \|\overrightarrow{PA}\| < r'$ avec $A' = P$, ce qui implique que le modèle d'activité ne modifie pas le position du copépode à la réception de l'évènement $vu(A', capture)$.

$\lambda(reponse) = view(A', cherche)$
si $\exists\ A\ /A \in L\ \overrightarrow{PA}.\vec{D} \geq 0$, c'est-à-dire si la cellule est devant lui.
et si $l \leq r \mid l = |((\overrightarrow{PA} \wedge \vec{D}) \wedge \vec{D}).\overrightarrow{PA}|$,
alors $f = \sqrt{r^2 - ||\overrightarrow{PP'}||^2} - \sqrt{||\overrightarrow{PA}||^2 - ||\overrightarrow{PP'}||^2}$ où P' est la projection de P sur le vecteur $\vec{D}$ et $A' = P + f.\vec{D}$

Si les deux conditions précédentes ne sont pas vérifiées alors :

$\lambda(reponse) = vu(nil, rien)$ Le modèle de perception informe le modèle d'activité qu'il ne perçoit rien.

E.3 Modèle atomique de gestion des rebonds

Le modèle de gestion des rebonds est un modèle atomique DEVS avec la structure suivante :

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

où :
$X = \{\ (inreb1, trajectoire(P, \vec{D}, v)), (inreb2, limite(A)), (inreb3, cancel))\ \}$
$Y = \{\ (outreb1, limites?(P, \vec{D})), (outreb2, rebond(A, \vec{D}))\ \}$
$S = (\Phi, P, \vec{D}, v)$ avec $\Phi = \{Idle, question, requete, reponse\}$
$\delta_{ext}((Idle, P, \vec{D}, v), trajectoire(P, \vec{D}, v)) = (question, P, \vec{D}, v)$
$\delta_{ext}((requete, P, \vec{D}, v), limite(A)) = (reponse, P, \vec{D}, v)$
$\delta_{int}(question, P, \vec{D}, v) = (requete, P, \vec{D}, v)$
$\delta_{int}(reponse, P, \vec{D}, v) = (Idle, P, \vec{D}, v)$
$\lambda(question, P, \vec{D}, v) = limites?(P, \vec{D})$
$\lambda(reponse, P, \vec{D}, v) = (rebond(A, \vec{D'}))$ avec $\vec{D'}$ généré aléatoirement tel que le copépode se dirige dans l'espace et A les coordonnées de l'intersection de la trajectoire du copépode avec un plan limite de l'espace.
$ta(Idle, P, \vec{D}, v) = \infty$ *i.e* attente d'une interogation.
$ta(requete, P, \vec{D}, v) = \infty$ *i.e* attente de la réponse de l'environnement.
$ta(question, P, \vec{D}, v) = 0$ la question à l'environnement est instantanée.
$ta(reponse, P, \vec{D}, v) = f/v$ où v et la vitesse du copépode. La réponse au modèle d'activité est effectuée quand le copépode atteint un bord.
$\delta_{ext}((response, P, \vec{D}, v), cancel) = (Idle, P, \vec{D}, v)$ Le modèle de changement de direction aléatoire vient annuler le rebond.

E.4 Modèle atomique de changement de direction aléatoire

Le modèle de changement de direction aléatoire est un modèle atomique DEVS qui a la structure suivante :

$$Direction\ aleatoire =< Y, S, \delta_{int}, \lambda, ta >$$

où :
$Y = \{(outale_1, alea), (outale_2, cancel)\}$
$S = \{attend, genere\}$
$\lambda(genere) = (outale_1, alea)$
$\lambda(genere) = (outale_2, cancel)$

$\delta_{int}(genere) = (attend)$
$ta(attend) = c$ avec $c = random()$ (tirage aléatoire uniforme) et $0 < c < c_{max}$ où c_{max} est la durée maximale avant un changement de direction.
$\delta_{int}(attend) = (genere)$
$ta(genere) = 0$

E.5 Modèle atomique de gestion de l'énergie

Le modèle de gestion de l'énergie est un modèle DEVS atomique de la forme :

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

où :
$X =$ { $(inene, energie(q))$ }
$Y =$ { $(outene_1, satiete(s)), (outene_2, mort)$ } avec s un booléen.
$S =$ { $(faim, \chi), (nonfaim1, \Lambda), (nonfaim2, \Lambda), (mort, \Lambda)$ }
avec $\Lambda = \{X1, X2, X3, X4, X5\}$
l'ensemble des variables du système d'équations décrit en annexe B.
$\delta_{ext}((faim, \Lambda), energie(q) = (faim, \Lambda)$ si $X1 < \alpha$ avec α le seuil de satiété et $X1 = X1 + q$
$\delta_{ext}((hungry, \Lambda), energie(q)) = ((nonfaim1, \Lambda))$ si $X1 > \alpha$
$\delta_{ext}((nonfaim, \Lambda), energie(q)) = ((nonfaim2, \Lambda))$
$\delta_{int}((nonfaim1, \Lambda)) = ((nonfaim2, \Lambda))$
$\delta_{int}((nonfaim2, \Lambda)) = ((faim, \Lambda))$
$\delta_{int}((faim, \Lambda)) = ((mort, \Lambda))$ si $X2 < \beta$ avec β le seuil d'énergie minimal.
$\lambda((nonfaim1, \Lambda)) = satiete(true)$
$\lambda((nonfaim2, \Lambda) = satiete(false)$
$\lambda((faim, \Lambda)) = (mort)$
$ta((nonfaim1, \Lambda)) = 0$
$ta((mort, \Lambda)) = \infty$
$ta((nonfaim, \Lambda)) = f()$ et $f()$ étant une fonction déduite du système d'équations du budget énergétique du copépode (voir équation C.4 de l'annexe C).
$ta((faim, \Lambda)) = f(X2, \beta)$

E.6 Modèle atomique de l'environnement

Le modèle de l'environnement est un modèle DEVS atomique de la forme :

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

où :
$X =$ { $(inenv_1, liste?(P, \vec{D})), (inenv_2, limite?(P, \vec{D})), (inenv_3, mange(p_i))$ }
$Y =$ { $(outenv_1, liste(L)), (outenv_2, limite(A))$ }
$S =$ { $(\Phi, E, P, \vec{D})$ }

où $\Phi = \{Idle, demandeliste, demandelimites, tueparticule\}$
et $E = \{(E_x, E_y, E_z, H)\}$
avec :
$H = \{(p_i x, p_i y, p_i z, status_i)\}$ l'ensemble des particules de phytoplancton avec leurs coordonnées et leur statut tel que $status = mort, vif$ et $(i = 1...n)$ où n est le nombre de cellules dans l'environnement,
et E_x, E_y, E_z les coordonnées des limites maximum de l'espace.

L'environnement est interrogé par le modèle de perception et le modèle de gestion des rebonds. Le modèle d'activité l'informe également qu'une cellule de phytoplancton a été consommée. Le modèle de l'environnement peut donc subir trois transitions externes telles que[94] :

$\delta_{ext}((Idle, E, P, \vec{D}), liste?(P, \vec{D}) = (demandeliste, E, P, \vec{D})$
$\delta_{ext}((Idle, E, P, \vec{D}), limites?(P, \vec{D})) = (demandelimites, E, P, \vec{D})$
$\delta_{ext}((Idle, E, P, \vec{D}), mange(p_i)) = (tueparticule, E_x, E_y, E_z, H, P, \vec{D})$ où $p_i = (p_i x, p_i y, p_i z)$
et $(p_i x, p_i y, p_i z, status_i) \in H$ avec $status_i = mort$.

Les états définis par les phases $\Phi = \{demandeliste, demandelimites, tueparticule\}$ sont transitoires, d'où :

$ta(demandeliste, E, P, \vec{D}) = 0$
$ta(demandelimites, E, P, \vec{D}) = 0$
$ta(tueparticule, E, P, \vec{D}) = 0$

L'état défini par la phase *tueparticule* n'engendre pas de fonction de sortie. La fonction de transition externe $mange(p_i)$ modifie donc l'état de l'environnement en attribuant la valeur *mort* à l'attribut de la cellule de phytoplancton considérée. Ensuite, l'environnement retourne en phase $Idle$ d'où :

$\delta_{int}(tueparticule, E, P, \vec{D}) = (Idle, E, P, \vec{D})$

Pour les deux autres phases et avant le retour à l'état passif défini par la phase $Idle$ avec $ta(Idle) = \infty$, une fonction de sortie peut être calculée telle que :

$\lambda(demandeliste, E, P, \vec{D}) = liste(L)$ où L est la liste des coordonnées des cellules de phytoplancton susceptibles d'être perçues par le copépode.
$\lambda(demandelimites, E, P, \vec{D}) = limites(A)$
où
$A = (A_x, A_y, A_z)$ sont les coordonnées de l'intersection de la trajectoire du copépode avec les limites de l'espace telles que :
$f = \min_{\{i \in x,y,z\}} (m_i - P_i; M_i - P_i)$ où P_i sont les coordonnées du copépode,

[94] La variable e représentant le temps passé dans l'état n'apparaît pas du fait que le calcul des transitions externes ne l'utilise pas.

m_i, M_i les limites minimum et maximum de l'espace sur les trois axes (x, y, z), et $A = P + f.\vec{D}$

E.7 Modèles atomiques de l'activité et de l'environnement reformalisés

Ici figure la reformalisation des modèles d'activité et de l'environnement du modèle SAR. Cette reformalisation est due au couplage avec un système d'équations différentielles. Le modèle de l'environnement est le même modèle atomique DEVS qu'au paragraphe E.6 tel que :

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

Nous ajoutons à S l'ensemble des variables définissant l'état, trois autres variables :
cell le nombre de cellules de phytoplancton dans l'environnement ;
cop le nombre de copépodes interagissant avec l'environnement ;
date la date d'arrivée de l'évènement externe $calcule(cell, cop)$.

Nous complétons également l'ensemble des phases Φ avec *evalue* et *init*. Nous avons donc maintenant :
$\Phi = \{Idle, demandeliste, demandelimites, tueparticule, init, evalue\}$
À l'arrivée de l'évènement externe $calcule(cell, cop)$[95], le modèle de l'environnement réinitialise son état tel que :

$\delta_{ext}(Idle, E, P, \vec{D}, cell, cop, date), calcul(cell', cop')) = (init, E, P, \vec{D}, cell', cop', t)$ où t est la date d'arrivée de l'évènement externe. Le nombre de cellules contenues dans l'ensemble E devient égal à *cell*. Les coordonnées de chaque cellule sont tirées aléatoirement selon une loi qui dépend de la nature de la distribution spatiale. Les détails de ces distributions sont donnés au paragraphe 5.2.2 page 139.

Après cette transition externe, le nouvel état a un avancement du temps nul tel que :

$ta(init, E, P, \vec{D}, cell, cop, date) = 0$

Ceci lui permet d'effectuer une transition interne instantanée avant laquelle il génère une fonction de sortie, d'où :

$\lambda(init, E, P, \vec{D}, cell, cop, date) = init$
$\delta_{int}(init, E, P, \vec{D}, cell, cop, date) = (Idle, E, P, \vec{D}, cell, cop, date)$

Les fonctions suivantes formalisent la vérification par le modèle de l'environnement que la durée simulée par le SAR n'est pas supérieure à la durée nécessaire pour calculer g :

[95] Nous remplacons ici la notation $calcul(N, P)$ par $calcul(cell, cop)$ pour qu'il n'y ait pas de confusion entre P la position du copépode et *cop*, le nombre de copépodes.

$\delta_{int}(demandeliste, E, P, \vec{D}, cell, cop, date) = (Idle, E, P, \vec{D}, cell, cop, date)$
si $t - date < T$, c'est le comportement défini précédemment sinon :

$\delta_{int}(demandeliste, E, P, \vec{D}, cell, cop, date) = (evaluate, E, P, \vec{D}, cell, cop, date)$
avec t la date de transition et T le temps de simulation du modèle agent.

Si le temps de simulation a été suffisant, alors le modèle de l'environnement émet une fonction de sortie $valeur(g)$. Il effectue cette opération instantanément d'où :

$ta(evalue, E, P, \vec{D}, cell, cop, date) = 0$
$\lambda(evalue, E, P, \vec{D}, cell, cop, date) = valeur(g)$ où $g = cell_d/(cop \times (t - date))$ avec $cell_d$ le nombre de cellules ayant le statut *dead*.
$\delta_{int}(evalue, E, P, \vec{D}, cell, cop, date) = (Idle, E, P, \vec{D}, cell, cop, date)$

Une fois la fonction de sortie émise, le modèle de l'environnement revient dans un état passif. Il peut donc recevoir une nouvelle interrogation d'un copépode. Comme la condition $t - date < T$ sera toujours vraie, alors il émettra autant de valeurs de g qu'il y a de copépodes actifs dans le système. Comme nous l'avons vu avec le modèle NP, seule la dernière valeur sera prise en compte dans le schéma d'intégration numérique. De fait, tous les copépodes se trouvent alors dans un état passif, ainsi que l'environnement ; seule l'arrivée d'un évènement $calcule(N, P)$ peut relancer le SMA. C'est le rôle de l'évènement *init* envoyé à tous les copépodes. Le modèle du copépode est donc sensiblement le même que précédemment (paragraphe 3.5.2 page 71) avec un élément de plus pour l'ensemble des EIC correspondant au port de connexion interne permettant au modèle d'activité de recevoir l'évènement *init* :

$$\begin{aligned} EIC = \{\ & ((Copepode, incop_1), (perception, inper_1)), \\ & ((Copepode, incop_2), (gestion\ rebonds, inbounce_1)), \\ & ((Copepode, incop_3), (activite, inact_6))\ \} \end{aligned}$$

Ainsi le modèle d'activité possède un nouveau port d'entrée, d'où son ensemble X modifié comme suit :

$$\begin{aligned} X = \{\ & (inact_1, vu(A, phase)), (inact_2, bounce(A, \vec{D})), \\ & (inact_3, dead), (inact_4, satiete(s)), (inact_5, alea(), (inact_6, init)\ \} \end{aligned}$$

À la réception de l'évènement *init*, le modèle d'activité des copépodes actifs réinitialise leur état :

$\delta_{ext}(S, init) = (init, P, \vec{D})\ \forall S$

Bibliographie

[AG89] R. Arditi et L. R. Ginzburg. – Coupling in predator-prey dynamics : Ratio-dependence. *Journal of theoritical Biology*, vol. 139, 1989, pp. 311–326.

[AH99] A. Aussem et D. Hill. – Wedding connectionist and algorithmic modelling towards forecasting *Caulerpa taxifolia* development in the north-western mediterranean sea. *Ecological Modelling*, vol. 120, 1999, pp. 225–236.

[AKB+01] H. Afsarmanesh, E.C. Kaletas, A. Benabdelkader, C. Garita et L.O. Hertzberger. – A reference architecture for scientific virtual laboratories. *Future Generation Computer Systems*, vol. 17, 2001, pp. 999–1008.

[Ant02] H. M. Antia. – *Numerical Methods for Scientists and Engineers*. – Birkhäuser, 2002.

[APS80] M. Alcaraz, G.A. Paffenhöfer et J.R. Strickler. – *The Evolution and Ecology of Zooplankton Communities*, chap. Catching the algae : a first account of visual observations on filter feeding calanoids. – New England, Hanover, University Press, 1980.

[AS92] R. Arditi et H. Saïah. – Empirical evidence of the role of heterogeneity in ratio-dependent consumption. *Ecology*, vol. 73, num. 5, 1992, pp. 1544–1551.

[Atl86] H. Atlan. – *À tort et à raison. Intercritique de la science et du mythe*. – Seuil, 1986.

[Bar96] F. Barros. – Dynamic structure discret event system specification : Formalism, abstract simulators and applications. *Transaction of the Society for Computer Simulation*, vol. 13, num. 1, 1996, pp. 35–46.

[BBPP98] F. Bousquet, I. Bakam, H. Proton et C. Le Page. – Cormas : common-pool resources and multi-agent systems. *Lecture Notes in Artificial Intelligence*, vol. 1416, 1998, pp. 826–838.

[Ber68] L. V. Bertalanffy. – *Théories générale des systèmes*. – Dunod, 1968. réedition 1993.

[BGCS93] M.H. Bundy, T.F. Gross, D. J. Coughlin et J. R. Strickler. – Quantifying copepod searching efficiency using swimming pattern and perceptive ability. *Bulletin of Marine Science*, vol. 53, 1993, pp. 15–28.

[BGVS98] M.H. Bundy, T.F. Gross, H.A Vanderploeg et J. R. Strickler. – Perception of inert particules by calanoid copepods : behavioral observations and a numerical model. *Jour. Plank. Resea.*, vol. 20, num. 11, 1998, pp. 2129–2152.

[BR93] D. Brown et P. Rothery. – *Models in Biology : Mathematics, Statistics and Computing*. – New York, Wiley Publishers, 1993.

[Bro94] J.H. Brown. – *Complexity : Metaphors, Models, and Reality*, chap. Complex Ecological Systems. – Addison-Wesley, 1994, *Santa Fe Institute in the Science of Complexity*.

[BV02] J. S. Bolduc et H. Vangheluwe. – Expressing ode models as devs : Quantization approaches. *In : AIS'2002 Conference (AI, Simulation and Planning in High Autonomy Systems)*, éd. par F. Barros et N. Giambiasi. pp. 163–169. – SCS, 2002.

[BV03] S. Borland et H. Vangheluwe. – Transforming Statecharts to DEVS. *In : Summer Computer Simulation Conference, Student Workshop*, éd. par A. Bruzzone et M. Itmi. pp. S154–S159. – SCS, Montréal, Canada, 2003.

[BWE00] J.H. Brown, G.B. West et B.J. Enquist. – *Scaling in Biology*, chap. Patterns and Processes, Causes and Consequences. – Oxford University Press, 2000.

[Cam94] Ch. Cambier. – *Un système multi-agents pour simuler la pêche sur le delta central du Niger*. – Thèse de Doctorat, Université de Paris 6, 1994.

[Cap96] P. Caparroy. – *Rôle de la turbulence dans les interactions entre le zooplancton et ses proies (phytoplancton et microzooplancton). Modélisation des processus, paramétrisation, expériences au laboratoire*. – Thèse de Doctorat, Université Paris VI, 1996.

[CC96] P. Caparroy et F. Carlotti. – A model for *Acartia tonsa* : Effect of turbulence and consequences for the related physiological processes. *Journal of Plankton Research*, vol. 18, num. 11, 1996, pp. 2139–2177.

[CGW00] F. Carlotti, J. Giske et F. Werner. – *ICES Zooplankton Methodology Manual*, chap. 12 - Modeling zooplankton dynamics, pp. 571–644. – Academic Press, 2000.

[CH97] P. Coquillard. et D. R. C. Hill. – *Modélisation et simulation d'écosystèmes : des modèles déterministes aux simulations à évènements discrets*. – Masson, 1997.

[CL90] P. R. Cohen et H. J. Levesque. – Intention in choice with commitment. *Artificial Intelligence*, vol. 42, 1990, pp. 213–261.

[CL96] P. Curry C. Lepage. – How spatial heterogeneity influences population dunamics : Simulations in sealab. *Adaptative Behavior*, vol. 4, 1996, pp. 255–288.

[CN92] F. Carlotti et P. Nival. – Model of copepod growth and development : mouting and mortality in relation to physiological processes during an individual moult cycle. *Mar. Ecol. Prog. Ser.*, vol. 84, 1992, pp. 219–233.

[Col97] Collectif. – *L'ordre du chaos*. – Bibliothèque Pour la Science, 1997.

[CPC98] P. Capparoy, M.T. Pérez et F. Carlotti. – Feeding behaviour of *Centropages typicus* in calm and turbulent conditions. *Mar. Eco. Pro. Ser.*, vol. 168, 1998, pp. 109–118.

[CPM94] G.A.C. Cowan, D. Pines et D. Meltzer (édité par). – *Complexity : Metaphors, Models, and Reality*. – Addison-Wesley, 1994, *Santa Fe Institute in the Science of Complexity*, volume XIX.

[CS89] F. Carlotti et A. Sciandra. – Population dynamics model of *Euterpina acutifrons* (copepoda : Harpacticoida) coupling individual growth and larval development. *Marine Ecology Progress Series*, vol. 56, 1989, pp. 225–242.

[CS02] C. A. Condat et G. J. Sibona. – Diffusion in a model for active brownian motion. *Physica D*, vol. 168-169, 2002, pp. 235–243.

[CV94] R. Cazoulat et B. Victorri. – étude de la dynamique des populations par simulation. *In : Actes du Colloque Chaos et Société*. – Université de Québec à Hull, 1994.

[DADR03] G. Deffuant, F. Amblard, R. Duboz et É. Ramat. – Une démarche expérimentale pour la simulation individus-centré. *In : Le status épistémologique de la simulation. 10^{eme} journées de Rochebrune : Rencontre interdisciplinaires sur les systèmes complexes naturels et artificiels*, pp. 45–64. – Rochebrune, France, 2003.

[DH01] B. Dumont et D.R.C. Hill. – Multi-agent simulation of group foraging in sheep : effects of spatial memory, conspecific attraction and plot size. *Ecological Modelling*, vol. 141, 2001, pp. 201–215.

[DL84] R. Dautray et J.L. Lions. – *Analyse mathématique et calcul numérique pour les sciences et les techniques*. – INSTN CEA, 1984.

[DRG02] R. Duboz, É. Ramat et N. Giambiasi. – *Introduction à la modélisation à évènements discrets - Modélisation d'un système proie prédateur*. – Rapport de recherche RI-LIL-2002-02, Laboratoire d'Informatique du Littoral - ULCO, 2002.

[Dro93] A. Drogoul. – *De la simulation multi-agents à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents*. – Thèse de Doctorat, Université Paris VI, 1993.

[DRP01] R. Duboz, É. Ramat et P. Preux. – Towards a coupling of continuous and discrete formalisms in ecological modelling - influences of the choice of algorithms on results. *In : ESS01 Simulation in Industry*, éd. par N. Giambiasi et C. Frydman, pp. 481–487. – Marseille, 2001.

[DRP03] R. Duboz, E. Ramat et P. Preux. – Scale transfer modeling : Using emergent computation for coupling an ordinary differential equation system with a reactive agent model. *Systems Analysis Modeling & Simulation*, vol. 43, num. 6, 2003, pp. 793–814.

[Dub02] R. Duboz. – Xml for the representation of semantic in model coupling. *In : AIS'2002. AI, Simulation and Planning in High Autonomy Systems*, éd. par F. J. Barros et N. Giambiasi, pp. 267–270. – Lisboa, Portugal, april 7-10, 2002.

[Dup94] J.P. Dupuy. – *Aux origines des sciences cognitives*. – Seuil La découverte, 1994.

[FCB02] J.B. Filippi, F. Chiari et P. Bisgambiglia. – Using j-devs for the modeling and simulation of natural complex systems. *In : SCS AIS 2002 conference on simulation in industry*, éd. par F. Barros et N. Giambiasi, p. 317. – 2002.

[Fer95] J. Ferber. – *Les Systèmes Multi-Agents*. – Inter-Éditions, 1995.

[FF01] A. Fall et J. Fall. – A domain-specific language for models of landscape dynamics. *Ecological Modelling*, vol. 137, 2001, pp. 1–21.

[FG00] J. Ferber et O. Guntknecht. – Pour une sémantique opérationnelle des systèmes multi-agents. *In : JFIADSMA'00*. pp. 39–55. – Hermes, 2000.

[Fia01] Y.E. Fianyo. – *Couplage de modèle à l'aide d'agents : le système OSIRIS*. – Thèse de Doctorat, Université Paris IX-Dauphine, 2001.

[Fic55] A. Fick. – Über diffusion. *Annalen der physik und chemie*, vol. 94, 1855, pp. 59–86.

[Fis95] P.A. Fishwick. – *Simulation Model Desing and Execution*. – Prentice Hall, 1995.

[Fis02] P. A. Fishwick. – Using xml for simulation modeling. *In : Proceedings of the 2002 Winter Simulation Conference*, éd. par E. Yücesan, C.-H. Chen, J. L. Snowdon et J. M. Charnes. – 2002.

[FM96] J. Ferber et J.P. Müler. – Influences reactions : A model of situated multiagent systems. *In : Second international Conference on Multi-Agent Sustems (ICMAS'96)*, pp. 72–79. – 1996.

[For80] J.W. Forrester. – *Principes des Systèmes*. – Presses Universitaires de Lyon, 1980.

[For90] S. Forrest. – Emergent computation : Self organizing, collective and cooperative phenomena in natural and artificial computing networks. *Physica D*, vol. 42, 1990, pp. 1–11. – Introduction of the Ninth Annual CNLS Conference.

[FPV95] S. Frontier et D. Pinchod-Viale. – *Écosystèmes. Structure-fonctionnement évolution*. – Masson, 1995.

[FWE+99] K. D. Forbus, P. B. Whalley, J. O. Everett, L. Ureel, M. Brokski, J. Baher et S. E. Kuehne. – Cyclepad : An articulate virtual laboratory for engineering thermodynamics. *Artificial Intelligence*, vol. 114, 1999, pp. 297–347.

[FWG98] F. Fashe, C. Wissel et V. Grimm. – Reconciling classical and individual-based approaches in theoretical population ecology : A protocol for extracting population parameters from individual-based models. *The American Naturalist*, vol. 152, num. 6, December 1998, pp. 838–851.

[FZ92] P.A. Fishwick et B.P. Zeigler. – A multi-model methodology for qualitative model engineering. *ACM transaction on Modeling and Simulation*, vol. 2, num. 1, 1992, pp. 52–81.

[GD02] T. Garneau et S. Delisle. – Programmation orientée-agent : évaluation d'outil et environnements. *In : Jou. Fran. Intel. Arti. Dictri. et Sys. Mul.*

Age. JFIADSMA'02, éd. par P. Mathieu et J.P. Müller. pp. 111–123. – Hermes, 2002.

[GEG00] N. Giambiasi, B. Escudé et S. Gosh. – A generalized discrete event specification for accurate modelling of dynamic systems. *Transaction of the Society for Computer Similation International*, vol. 17, num. 3, 2000, pp. 120–134.

[GF98] O. Gutknecht et J. Ferber. – Madkit : Organizing heterogeneity in a platform for multiple multi-agents systems. *In : International Conference on Multi-Agents Systems - ICMAS'98*. pp. 128–135. – IEEE, Paris, France, juillet 1998.

[Gri94] V. Grimm. – Mathematical models and understanding in ecology. *Ecological Modelling*, vol. 75/76, 1994, pp. 641–651.

[Gri99] V. Grimm. – Ten years of individual-based modelling in ecology : what we have learned and what could we learn in the future. *Ecological Modelling*, vol. 115, 1999, pp. 129–148.

[Gri02] V. Grimm. – Visual debugging : A way of analysing, understanding and communicating bottom-up simulation models in ecology. *Natural Ressource Modeling*, vol. 15, num. 1, 2002, pp. 23–38.

[GS77] J. Gerritsen et J.R. Strickler. – Enconter probabilities and community structure in zooplankton : A mathematical model. *J. Fish. Bd. Can.*, vol. 34, 1977, pp. 73–82.

[GWAU99] V. Grimm, T. Wyszomirski, D. Aikman et J. Uchmański. – Individual-based modelling and ecological theory : synthesis of a workshop. *Ecological Modelling*, vol. 115, 1999, pp. 275–282.

[Hak91] H. Haken. – *Synergetics*. – New York, Springer, 1991.

[Har68] J.G.K. Harris. – A mathematical model describing the possible behaviour of copepod feeding continuously in a relatively dense randomly distributed population of algal cells. *J. Con. Perm. Int. Explor. Mer*, vol. 32, num. 1, 1968, pp. 83–92.

[HDP88] M. Huston, D. DeAngelis et W. Post. – New computer models unify ecological theory. *Bioscience*, vol. 38, 1988, pp. 682–691.

[HE88] R. Hockney et J. Eastwood. – *Computer simulation using particles*. – 1988, institute of physics édition.

[HFS02] M. F. Hocaoglu, C. Firat et S Sarjoughian. – devs/rap : Agent based simulation. *In : AI and Simulation in Higth Autonomy Systems*, éd. par F. Barros et N. Giambiasi, pp. 117–121. – Lisbon, Portugal, April 2002.

[Hie94] D. Hiebeler. – The swarm simulation system and individual based modeling. *In : Advanced Technology for Natural Resource Management*. – Toronto, 1994.

[Hil96] D.R.C. Hill. – *Object-Oriented Analysis and Simulation*. – Addison-Wesley, 1996.

[Hil00] D.R.C Hill. – *Contribution à la modélisation des systèmes complexes : Application à la simulation d'écosystèmes.* – Habilitation à diriger des recherches, École Doctorale Sciences pour l'Ingénieur de Clermont-Ferrand, décembre 2000.

[HM01] E. R. Harold et W. S. Means. – *XML in a nutshell. Manuel de référence.* – O'Reilly, 2001.

[HMO02] J. Houshuo, C. Meneveau et T.R. Osborn. – The flow field around a freely swimming copepod in steady motion. part 2 : Numerical simulation. *Jour. Plan. Rese.*, vol. 24, num. 3, 2002, pp. 191–213.

[Hoh02] A. Hoheisel. – Model coupling and integration via xml in the m3 simulation. *In : International anvironemental modelling and software society IEMSs*, pp. 611–616. – Lugano, Switzerland, June 24-27 2002.

[Hol59] C.S. Holling. – Some characteristics of simple types of predation and parasitism. *Canadian Entomologist*, vol. 91, 1959, pp. 385–398.

[HOM02a] J. Houshuo, T.R. Osborn et C. Meneveau. – Chemoreception and the deformation of the active space in freely swimming copepods : a numerical study. *Jour. Plan. Rese.*, vol. 24, 2002, pp. 495–510.

[HOM02b] J. Houshuo, T.R. Osborn et C. Meneveau. – The flow field around a freely swimming copepod in steady motion. part 1 : Theroetical analysis. *Jour. Plan. Rese.*, vol. 24, num. 3, 2002, pp. 167–189.

[HS98] M. N. Huhngs et M. P. Singh. – Agents and multi-agent systems : themes, approaches and challenges. *In : reading in agents*, éd. par M. N. Huhngs et M. P. Singh, pp. 1–23. – San francisco CA, USA, 1998.

[Hug02] M. P. Huget. – Une approche d'Agent-uml à la gestion de chaîne logistique. *In : JFIADSMA'02.* pp. 125–137. – Hermes, 2002.

[JAA99] C. Jost, O. Arino et R. Arditi. – About deterministic extinction in ratio-dependant predator-prey models. *Bulletin of mathematical Bioloy*, vol. 61, 1999, pp. 19–32.

[Jos98] C. Jost. – *Comparaison qualitative et quantitative de modèle proie-prédateur à des données chronologique en écologie.* – Thèse de Doctorat, Institut national agronomique Paris-Grignon, 11 décembre 1998.

[Jou01] F. Jouanot. – *DILEMMA : Vers une coopération de systèmes d'information basée sur la médiation sémantique et la fusion d'objets.* – Thèse de Doctorat, Université de Bourgogne, novembre 2001.

[JW02] C. Jacques et G.A. Wainer. – Using the cd++ devs tookit to develop petri nets. *In : SCS Conference.* – 2002.

[Kan90] C.N. Kanarick. – A technical overview and history of the SIMNET project. *In : SCS Multiconference on Distributed Simulation*, pp. 104–109. – 1990.

[KB94] J. Košecká et L. Bogoni. – Application of discrete events systems for modeling and controling robotic agents. *In : International Conference in Robotics and Automation*volume 3. pp. 2557–2562. – IEEE, 1994.

[KFM85] T. Kiorboe et H. Nicolajsen F. Mohlenberg. – Bioenergetics of the planctonik copepod, acartia tonsa : relation between feeding, egg production and respiration, and composition of specific dynamic action. *Mar. Ecol. Prog. Ser.*, vol. 26, 1985, pp. 85–97.

[KK96] J.Y. Kim et T.G. Kim. – A heterogeneous distributed simulation framework based on devs formalism. *In : Sixth Annual Conference On Artificial Intelligence, Simulation and Planning in High Autonomy Systems*, pp. 116–121. – La Jolla, California, USA, 1996.

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

[Kli85] G. Klir. – *Architecture of Systems Problem Solving*. – Plenum press, 1985.

[Kof02] E. Kofman. – A second order approximation for devs simulation of continuous systems. *Journal of the Society for Computer Simulation International*, vol. 78, num. 2, 2002.

[KPJK96] Y. Kwon, H. Park, S. Jung et T. Kim. – Fuzzy-devs Formalism : Concepts Realization and Applications. *In : AI, Simulation and Planning in Hight Autonomy Systems*. – San Diego, USA, 1996.

[Kuh72] T. Kuhn. – *La Structure des révolutions scientifiques*. – Flammarion, 1972. nouvelle trad. par Laure Meyer, 1983.

[L'E98] P. L'Ecuyer. – *Handbook on Simulation*, chap. 4, Random Number Generation, pp. 93–137. – Wiley, 1998.

[L'E01] P. L'Ecuyer. – Software for uniform random number generation : Distinguishing the good and the bad. *In : Winter Simulation Conference*, éd. par IEEE Press, pp. 95–105. – 2001.

[Leg97] J. M. Legay. – *L'expérience et le modèle. Un discours sur la méthode*. – INRA, 1997.

[Lev73] R. Levins. – *Hierarchy theory. The challenge of complex systems*, chap. The limits of complexity, pp. 111–127. – New York, George Braziller, 1973.

[LG02] K. Lerman et A. Galstyan. – *A macroscopic analytical model of collaboration in distributed robotic systems*. – Rapport de recherche, Univ. of Southern California, USA, Information Science Institute, 2002.

[Lin92] S. Lindenberg. – The method of decreasing abstraction. *In : Rational choice theory : advocacy and critique*, éd. par T.Fararo. pp. 3–20. – Sage publications, 1992.

[Lot25] A. J. Lotka. – Elements of physical biology. *Baltimore : Williams & Wilkins Co*, 1925.

[LS99] H. Lorenk et M. Sonnenschein. – Modelling and simulation software to support individual-oriented ecological modelling. *Ecological Modelling*, vol. 115, 1999, pp. 199–216.

[LV97] M. Li et P. Vitanyi. – *An Introduction to Kolmogorov Complexity and Its Applications*. – 1997, springer verlag, 2nd edition édition.

[LV02] J. De Lara et H. Vangheluwe. – Using *atom*^{3} as a meta-case tool. *In : 4th International Conference on Enterprise Information Systems (ICEIS)*, pp. 642–649. – Ciudad Real, Espagne, 2002.

[Mar63] D.W. Marquardt. – An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, num. 2, June 1963, pp. 431–441.

[MC97] T. Maxwell et R. Costanza. – A language for modular spatio-temporal simulation. *Ecological Modelling*, vol. 103, 1997, pp. 105–113.

[Mil00] L. Millischer. – *Modélisation individu centrée des comportements de recherche des navires de pêche*. – Thèse de Doctorat, Ecole Nationale Supérieure Agronomique de Rennes, juin 2000.

[MN98] M. Matsumoto et T. Nishimura. – Mersenne twister : A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, vol. 8, num. 1, 1998, p. 330.

[Mor77] E. Morin. – *La méthode : la Nature de la Nature*. – Seuil, 1977volume 1.

[Mrj97] Mrjean. – Emergence et sma. *In : JFIASMA'97*. pp. 323–342. – AFCET/AFIA, 2-4 avril 1997. Groupe de travail collectif.

[Mül] J.P. Müller. – Méthodologie de conception de systèmes multi-agents de résolution de problèmes par émergence. *In : JFIADSMA 98*. – Hermes, novembre.

[NB66] J. Von Neumann et A.W. Burks. – *Theory of Self-reproducing Automata*. – Urbana, IL, Seuil, 1966, university of illinois press édition.

[NMZ98] R. Nidumolu, N. Menon et B.P. Zeigler. – Object-oriented buisiness process modeling ans simulation : A discrete-event specification (devs) framework. *Sim. Pract. and Theory*, vol. 6, 1998, pp. 531–571.

[Ö89] T.I Ören. – *Knowledge-Based Simulation : Methodology and Application*, chap. Dynamics Templates and Semantic Rules for Simulation Advisters and Certifiers. – Berlin, Springer-Verlag, 1989.

[Ode02] J. Odell. – Objects and agents compared. *Journal of object technology*, vol. 1, num. 1, 2002.

[OPB00] J. Odell, H. V. D. Parunak et B. Bauer. – Extending uml for agents. *In : Workshop on Agent Oriented Information Systems*. – Austin, Texas, july 2000.

[Osb96] T. Osborn. – The role of turbulent diffusion for copepods with fedding current. *J. Plankton Res.*, 1996.

[Paf98] G.A. Paffenhöfer. – On the relation of structure, perception and activity in marine planktonic copepods. *Journal of Marine Systems*, vol. 15, 1998, pp. 457–473.

[Pat73] H.H. Pattee (édité par). – *Hierarchy theory. The challenge of complex systems.* – New York, George Braziller, 1973.

[Pav94] A. Pavé. – *Modélisation en Biologie et en Écologie.* – Aléas, 1994.

[PB01] H.V.D. Parunak et S. Brueckner. – Entropy and self-organization in multi-agent systems. pp. 124–130. – Montreal, Quebec, Canada, may 2001.

[PBLM95] G.A. Paffenhöfer, M.H. Bundy, K.D. Lewis et C. Metz. – Rates of ingestion and their variability between individual calacoid copepods : direct observations. *Jour. Plank. Res.*, vol. 17, 1995, pp. 1573–1585.

[Pog94] J.C. Poggiale. – *Applications des variétés invariantes à la modélisation de l'hétérogénéité en dynamique des populations.* – Thèse de Doctorat, Université de Bourgogne, Dijon, décembre 1994.

[PP86] H.J. Price et G.A. Paffenhöfer. – Capture of small cells by copepod eucalanus elogatus. *Limmnol. Oceanogr.*, vol. 21, 1986, pp. 115–124.

[Pra91] H. Praehofer. – System theoretic formalisms for combined discrete-continuous system simulation. *Int. J. Gen. Sys.*, vol. 19, num. 3, 1991, pp. 219–240.

[PS79] I. Prigogine et I. Stengers. – *La nouvelle alliance.* – Gallimard, 1979.

[PTVF94] W. H. Press, S. A. Teukolsky, W.T. Vetterling et B.P. Flannery. – *Numerical Recipies in C, (FORTRAN, PASCAL), The art of scientific computing.* – Cambridge University Press, 1994, *second edition.*

[QDNR03] G. Quesnel, R. Duboz, F. Nolot et É. Ramat. – Comparaison d'approches de simulations distribuées à évènements discrets d'entités spatialisées. *In : MAJECSTIC'03. 1re conférence des jeunes chercheurs STIC.* – Marseille, 28-30 Octobre 2003.

[Que03] G. Quesnel. – *Vers la simulation massive et distribuée d'agents réactifs situés.* – Rapport de recherche, Université du Littoral Côte d'Opale, 2003. (mémoire de DEA).

[Rai01] Deanna Raineri. – Virtual laboratories enhance traditional undergraduate biology laboratories. *Biochemistery and Molecular Biology Education*, vol. 29, 2001, pp. 160–162.

[Rea77] L.A. Real. – The kinetics of functional response. *American Naturalist*, vol. 111, 1977, pp. 289–300.

[RP03] E. Ramat et P. Preux. – "virtual laboratory environment" (vle) : a software environment oriented agent and object for modeling and simulation of complex systems. *Simulation Modelling Practice and Theory*, vol. 11, 2003, pp. 45–55.

[RPSL98] E. Ramat, P. Preux, L. Seuront et Y. Lagadeuc. – Modélisation et simulation multi-agents en biologie marine : étude du comportement du copépode. *In : Colloque SMAGET*, pp. 35–49. – Clermont-ferrand, 1998.

[Sch97] F. Scheweitzer. – Active brownian particules : Artificial agents in physics. *Lecture Notes in Physics*, vol. 484, 1997, pp. 358–371.

[Ser00] D. Servat. – *Modélisation de dynamiques de flux par agents. Application aux processus de ruissellement, infiltration et érosion.* – Thèse de Doctorat, Université Paris VI, Novembre 2000.

[Sou01] J.C. Soulié. – *Vers une approche multi-environnement pour les agents.* – Thèse de Doctorat, Université de la Réunion, Décembre 2001.

[SPTD98] David Servat, Edith Perrier, Jean-Pierre Treuil et Alexis Drogoul. – Towards virtual experiment laboratories : How multi-agent simulations can cope with multiple scales of analysis and viewpoints. *Lecture Notes in Computer Science*, vol. 1434, 1998.

[Sr95] E. Saiz et T. Kiørboe. – Suspension and predatory feeding of the copepod *Acartia Tonsa*. *Mar. Eco. Pro. Ser.*, vol. 122, 1995, pp. 147–158.

[Sr00] C. Svensen et T. Kiørboe. – Remote prey detection in *Oithona similis* : hydromechanical versus chemical cues. *Jour. Plank. Resea.*, vol. 22, num. 6, 2000, pp. 1155–1166.

[SS01] S. Soussi et B. Syuhei. – The consequences of individual variability in moulting probability and the aggregation of stages for modelling copepod population dynamics. *Journ. Plank. Resea.*, vol. 23, num. 11, 2001, pp. 1279–1296.

[SSL+96] L. Seuront, F. Schmitt, Y. Lagadeuc, D. Schertzer, S. Lovejoy et S. Frontier. – Multifractal analysis of phytoplankton biomass and temperature in ocean. *Geophys. Res. Lett.*, vol. 23, 1996, pp. 3591–3594.

[SSL+99] L. Seuront, F. Schmitt, Y. Lagadeuc, D. Schertzer et S. Lovejoy. – Universal multifractal analysis as a tool to characterize multiscale intermittent pattern. example of phytoplankton distribution in turbulent coastal water. *J. Plankton Res.*, 1999.

[SSL01] L. Seuront, F. Schmitt et Y. Lagadeuc. – Turbulence intermittency, small-scale phytoplankton patchiness and encounter rates in phytoplankton : where do we go from here. *Deep Sea Research*, vol. I, num. 48, 2001, pp. 1199–1215.

[SU01] B. Schattenberg et A.M. Uhrmacher. – Planning Agents in James. *In : Special Issue on Agent in Modelling and Simulation : Exploiting the metaphor*volume 89. pp. 158–173. – IEEE, 2001.

[SYLL02] D. Shin, E.S. Yoon, K.Y. Lee et E.S. Lee. – A web-based, interactive virtual laboratory system for unit operations and process systems engineering education : issues, design and implementation. *Computers & Chemical Engineering*, vol. 26, 2002, pp. 319–330.

[Tan75] J.T. Tanner. – The stability and the intrinsic growth rates of prey and predator populations. *Ecology*, vol. 56, 1975, pp. 855–867.

[TH01] M.K. Traore et D.R.C Hill. – The use of random number generation for stochastic distributed simulation : Application to ecological modeling. *In : 13e european simulation symposium*, éd. par N. giambiasi et C. Frydman. pp. 555–559. – SCS, Marseille, 2001.

[Tr03] J. Titleman et T. Kiørboe. – Motility of copepod nauplii and implication for food encounter. *Marine Ecology Progress Series*, vol. 247, 2003, pp. 123–135.

[UA94] A.M. Uhrmacher et R. Arnold. – Distributed and maintaining knowledge : Agents in variable structure environments. pp. 178–184. – IEEE, Florida, Gainsville, 1994.

[Uch99] J. Uchmański. – What promotes persistence of a single population : an individual-based model. *Ecological Modelling*, vol. 115, 1999, pp. 227–241.

[UFZ01] A.M. Uhrmacher, P.A. Fishwick et B.P. Zeigler. – Scanning the issue. *In : Special Issue on Agent in Modelling and Simulation : Exploiting the metaphor*volume 89. pp. 127–129. – IEEE, 2001.

[Uhr95] A.M. Uhrmacher. – Reasonnign about changing structure. a modelling concept for ecological systems. *Applied Artificial Intelligence*, vol. 9, num. 2, 1995, pp. 157–180.

[Uhr01] A.M. Uhrmacher. – Dynamics structure in modeling and simulation - a reflective approach. *ACM Transaction on Modeling and Simulation*, vol. 11, num. 2, 2001, pp. 206–232.

[US98] A.M. Uhrmacher et B. Schattenberg. – Agents in discrete event simulation. *In : European Simulation Symposium.* – SCS, Nottingham, october 1998.

[Van00] H. Vangheluwe. – Devs as a common denominator for hybrid systems modelling. *In : IEEE International Symposium on Computer-Aided Control System Design*, éd. par A. Varga. pp. 129–134. – IEEE Computer Society Press, Anchorage, Alaska, 2000.

[Var01] F. Varenne. – What does a computer simulation prove ? The case of plant modelling at CIRAD (France). *In : 13th European Simulation Symposium, ESS'01. Simulation in Industry*, éd. par G. Giambiasi et C. Frydman. pp. 549–554. – SCS Europe Bvba, October 2001.

[Var03] F. Varenne. – La simulation conçue comme une expérience concrête. *In : Le status épistémologique de la simulation. 10eme journée de Rochebrune : Rencontre interdisciplinaire sur les systèmes complexes naturels et artificiels.* pp. 299–313. – ENST, 2003.

[Ver89] F. Verla. – *Autonomie et connaissance.* – Seuil, 1989.

[Vil01] F. Villa. – Integrating modelling architecture : a declarative framework for multi-paradigm, multi-scale ecological modelling. *Ecological Modelling*, vol. 137, 2001, pp. 23–42.

[VLM02] H. Vangheluwe, J. Lara et P.J. Mosterman. – An introduction to multi-paradigm modelling and simulation. *In : AIS'2002. Simulation and Planning in High Autonomy Systems*, éd. par F.J. Barros et N. Giambiasi. pp. 9–20. – Society for Modelling and Simulation International, Lisbon, Portugal, April 2002.

[Vol26] V. Volterra. – Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. *Mem. R. Accad. Naz. dei Lincei. Ser. VI*, vol. 2, 1926.

[VV00] H. Vangheluwe et G.C. Vansteenkiste. – The cellular automata formalism and its relationship to DEVS. *In : 14th European Simulation Multiconference (ESM)*, éd. par Rik Van Landeghem. pp. 800–810. – SCS, Ghent, Belgique, 2000.

[Wei48] N. Weiner. – *Cybernetics*. – Paris, Hermann, 1948.

[Wei99] G. Weiss. – *Multiagent Systems. A modern approach to distributed artificial intelligence*. – London, England, MIT Press, 1999.

[WG01] G. A. Wainer et N. Giambiasi. – Application of the cell-devs paradigm for cell spaces modelling and simulation. *Simulation*, vol. 76, num. 1, 2001, pp. 22–39.

[WJ99] J. Waniewski et W. Jedruch. – Individual based modeling and parameter estimation for a lotka-volterra system. *Mathematical Biosciences*, vol. 157, 1999, pp. 23–36.

[Wos88] L. Wos. – *Artificial Intelligence, A Knowlegde Based Approach*, chap. Automated Reasoning. – 1988, pws-kent édition.

[XSDJ03] Y. Xu, D. Sauquet, P. Degoulet et M.-C. Jaulent. – Component-based mediation services for the integration of medical applications. *Artificial Intelligence in Medicine*, vol. 27, 2003, pp. 283–304.

[YOS91] H. Yamazaki, T. Osborn et K. Squires. – Direct numerical simulation of planktonic contact in turbulent flow. *J. Plank. Res.*, vol. 13, num. 3, 1991, pp. 219–241.

[ZC92] B. P. Zeigler et S. D. Chi. – Symbolic discrete event systems specification. *IEEE Trans. on Systems, Man and Cybernetics*, Dec. 1992, pp. 1428–1443.

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

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

[ZL98] B. P. Zeigler et J. Lee. – Theory of quantized systems : formal basis for devs/hla distributed simulation. *In : SPIE*, pp. 49–58. – 1998.

[ZS00] B.P. Zeigler et H.S. Sarjoughian. – Creating distributed simulation using devs m&s environment. *In : 2000 Winter Simulation Conference*. – 2000.

Résumé

L'étude des systèmes complexes est en passe de devenir une science en elle-même, avec l'informatique comme l'un des supports (théoriques et opérationnels) possibles pour son développement. La modélisation et la simulation des systèmes dynamiques s'inscrit dans ce mouvement actuel, en offrant de nombreux paradigmes et de nombreuses techniques pour appréhender la complexité des systèmes artificiels et naturels. Le travail présenté ici participe à ce courant en exposant une recherche sur le couplage et l'interopérabilité des modèles de simulation. Nous proposons d'aborder cette question en trois points :

- l'intégration formelle,
- l'intégration opérationnelle,
- l'intégration pour la simulation multi-échelles.

Pour illustrer notre réflexion, nous avons choisi un système proies-prédateurs emprunté à l'écologie marine (des copépodes se nourrissant de phytoplancton). Cette réflexion nous amène, concernant le premier point, à effectuer un rapprochement entre le paradigme d'agents réactifs situés, intéressant pour la simulation des comportements des prédateurs, et le formalisme DEVS, issu de la théorie des systèmes. Ce dernier permet une formalisation de l'ensemble du modèle couplé, constitué d'un Système Multi-Agents (SMA) et d'un système d'équations différentielles. Pour le deuxième point, nous proposons un *framework* d'intégration de modèles hétérogènes qui se démarque principalement par la notion de connecteur (ou *wrapper*), basé sur l'algorithmique des simulateurs abstraits de DEVS et la description des modèles et de leur couplage *via* une application XML particulière. Pour traiter le troisième point, nous développons d'abord une méthode de couplage pour le transfert de propriétés entre niveaux d'organisation (ou transfert d'échelles). En nous basant sur cette méthode, nous construisons un modèle mathématique à partir de simulations du SMA qui nous permet de le coupler avec un modèle « classique » d'interaction proies-prédateurs. Cette application illustre l'importance de la prise en compte du niveau micro (les individus) sur le niveau macro (les populations) d'un système dynamique particulier.

Mots-clés: intégration, multi-modélisation, multi-formalisme, multi-échelles, DEVS, Systèmes Multi-Agents (SMA), couplage, copépode *Acartia Tonsa*.

Abstract

Complex systems studies are becoming a science on its own. Computer science can be one of the theoretical and operational basis towards this evolution. Modelling and simulation (M& S) of dynamical systems is currently a central activity in a lot of sciences. M& S brings numerous paradigms and methods for the specification and the simulation of complex artificial or natural systems. This work aims at tackling the issue of interoperability between heterogeneous models following three directions :

- formal integration,
- operationnal integration,
- multi-scales integration.

In order to illustrate our investigation, we have chosen a marine prey-predator systems (copepods grazing on phytoplankton). First, we bring together the reactive agent paradigm with the DEVS formal specification langage, coming from systems theory. DEVS enables the complete formal decription of a multi-agents system (MAS), well suited for individual modelling, coupled with a differential equations system (well suited for population modelling). Secondly, we propose a framework for the integration of heterogneneous models. Our framework is mainly based on the concept of "wrapper". It provides us a way to interoperate different models based on DEVS abstract simulators. In this context, we develop a particular XML application for the description of models and models coupling. The last point of our work concerns scale transfers modelling in natural systems. We develop a method to achieve it and illustrate this method with our prey-predator model. The construction of a mathematical model based on simulations from our MAS leads to the coupling of our MAS with a differential equations system. Then, we show that micro-level activity (individuals) has a potentialy strong effect on macro-level dynamics (populations).

Keywords: integration, multimodelling, multiformalism, multiscale, multi-agent systems (MAS), coupling, copepod *Acartia Tonsa*.