Contributions à la modélisation et à la simulation des systèmes complexes

Eric Ramat

Université du Littoral
Côte d'Opale

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

Contributions à la modélisation et à la simulation des systèmes complexes

Habilitation à diriger des recherches

présentée et soutenue publiquement le 11 décembre 2003

pour l'obtention du grade

Habilité à diriger des recherches

(spécialité informatique)

par

Eric RAMAT

Composition du jury

Président :	Henri Basson	LIL – Université du Littoral
Rapporteurs :	Bernard Espinasse	LSIS – Université de Marseille III
	Michel Gourgand	LIMOS - Université de Blaise Pascal – Clermont-Ferrand
	Jean - Pierre Müller	CIRAD – Montpellier
Examinateurs :	Paul Bourgine	CREA – Ecole polytechnique
	François Carlotti	LOB – Station marine d'Arcachon – Université de Bordeaux I
	Norbert Giambiasi	LSIS – Université de Aix-Marseille III
	Philippe Preux	GRAPPA – Université de Lille 3

Laboratoire d'Informatique du Littoral — UPRES JE-2335

Mis en page avec la classe thloria.

Table des matières

Table des figures

Introduction générale

Les activités de recherche qui vont être descrites dans ce document sont à mettre en correspondance avec l'émergence d'un groupe de recherches un peu particulier caractérisés par des domaines de compétences variées (informatique, biologie, physique, mathématiques, ...), des statuts divers (des jeunes maîtres de conférences, des jeunes professeurs, des ATER et des étudiants en thèse, en DEA et en deuxième cycle) et des localisations géographiques dispersées (Calais, Wimereux, Marseille, Rennes, Clermont-Ferrand, Arcachon, Paris, Pau, ...). L'aventure a commencé en 1997 avec mon arrivée sur un poste d'ATER à l'Université du Littoral et ma rencontre avec Philippe Preux, qui, à l'époque, était Directeur du Laboratoire d'Informatique du Littoral, Christophe Cambier, jeune maître de conférences et Yvan Lagadeuc, futur professeur, de la Station Marine de Wimereux. Avec l'arrivée de Christophe Cambier, un premier contact avait été établi entre le LIL et la Station Marine de Wimereux pour répondre à une question d'écologie marine liée à l'impact du facteur comportemental d'un organisme zooplanctonique et du niveau d'hétérogénéité spatiale de la nourriture de cet organisme sur les bilans de production de biomasse. La question semble au premier abord complexe mais Christophe Cambier et Philippe Preux vont proposer une modélisation centrée individus qui se trouve être la thématique développée dans la thèse de Christophe Cambier [CAM94]. Ce type de modélisation permet de prendre en compte les aspects comportementaux des individus et les échelles impliquées dans les processus biologiques et physiques. En 1997, un premier modèle était en cours de développement. Malheureusement (pour le projet), à cette même époque, Christophe Cambier a demandé son détachement à l'IRD et a quitté le laboratoire pour l'Afrique. Dès mon arrivée, mon action était donc claire : poursuivre le projet. De plus, la politique du Laboratoire s'orientait vers le développement d'activités de recherche liées aux questions d'environnement. Ces questions restent aujourd'hui l'une des grandes orientations de l'Université et de la Région Nord-Pas de Calais. Afin de planter complé-

tement le décor, il faut noter que je venais tout juste de finir ma thèse (juin 1997) sur le thème de la gestion de l'incertitude dans la gestion de projets complexes à contraintes de ressources[RAM97] ce qui, vous l'avouerez, est loin d'être connexe à l'activité que l'on m'a demandé de développer. On peut alors réellement parler de reconversion thématique. Il ne faut pas pour autant croire que je regrette quoi que ce soit et le futur démontrera que mes travaux passés et mon domaine d'origine, les systèmes industriels, seront aussi une source d'inspiration.

La première phase du projet donnera naissance en 1998 à une première modélisation centrée individus et aux premières interprétations de la dynamique du système. Ces résultats seront l'occasion d'intégrer progressivement les groupes de travail en écologie marine (le GdR Manche dès 1998 puis le PNEC en 2000). A l'époque, le groupe de chercheurs travaillant sur cette modélisation est relativement réduit puisqu'il faut compter deux informaticiens (Philippe Preux et moi-même) et deux biologistes (Yvan Lagadeuc et Laurent Seuront, étudiant en thèse). Etant donné ce faible effectif, il faut alors compter sur des étudiants de deuxième cycle participant au travers de projets de fin d'études au développement de certaines parties de la plateforme VLE, notre outil de modélisation et de simulation de systèmes complexes. La participation d'étudiants de deuxième cycle reste encore vraie aujourd'hui mais plus pour des raisons pédagogiques liées à la découverte du monde de la recherche par nos étudiants de la filière professionnelle. En 1998, le Laboratoire d'Informatique du Littoral et surtout les formations en informatique à l'Université du Littoral (hormis l'IUT Informatique du Littoral) ne sont qu'à leurs débuts (un embryon d'option en IUP). Malgré le contexte difficile, le noyau dur de l'équipe va se former et sera complété par l'arrivée en thèse de Raphaël Duboz en 2000 au LIL. Raphaël Duboz résume à lui tout seul la physionomie du futur groupe : étudiant de biologie ayant préparé un DESS d'informatique puis un DEA océanologie pour finir en thèse d'informatique. La pluridisciplinarité sera une constante importante dans la constitution et la vie du groupe.

Parallèlement à ce projet de modélisation en biologie marine, plusieurs pistes seront tentées et donneront lieu à des prototypes afin d'argumenter des propositions de projets dans le cadre de programmes nationaux qui n'aboutiront pas : un modèle en géologie avec le Laboratoire de géologie et sédimentologie de l'Université de Lille1 pour étudier l'intérêt des modèles agents dans le domaine de la géologie et un modèle de systèmes flexibles de production avec mon ancien laboratoire (Laboratoire d'Informatique de Tours). Deux projets d'ACI dont l'un où j'étais le porteur ont fait l'objet d'une acceptation lors de la première phase de selection mais qui n'ont malheureusement pas passé le dernier filtre. Néanmoins, nous allons voir que nous ne sommes pas laissés arrêter par ces deux contretemps.

Après deux années de travail sur la modélisation centrée individus de systèmes complexes (surtout les écosystèmes marins), la validation des modèles, l'interaction de nos modèles avec d'autres modèles, nous avons dégagé trois orientations de recherche qui resteront jusqu'à aujourd'hui nos priorités :

- l'étude du couplage de modèles dans un contexte d'hétérogénéité des formalismes et d'interaction entre des processus multi-échelles,

– la formalisation du couplage de modèles et le développement d'outils d'aide au couplage,
– la construction de langages formels pour la spécification de systèmes spatio-temporels à l'aide du paradigme agent et centré individus.

Ces orientations vont déboucher sur le sujet de thèse de Raphaël Duboz mais surtout sur l'élargissement de l'équipe et son intégration à des groupes de travail au plan national. La thèse de Raphaël est financée par la région Nord-Pas de Calais conformément à ses priorités de recherche. Nous rejoignons les thématiques liées à l'environnement. Après notre intégration au GdR Manche, grâce à Yvan Lagadeuc, l'équipe va s'orienter vers le PNEC (Plan National d'Etudes Côtières) suite à la disparition du GdR Manche. Nous avons intégré le PNEC suite à l'appel à projets réitéré chaque année. Le PNEC propose deux modes d'actions : les actions attachées à un chantier (un chantier étant un lieu géographique, le Golfe de Gascogne, par exemple, où l'on étudie un écosystème particulier) et les actions thématiques transversales (les ART). Les activités que l'on mène sur l'impact des processus à micro-échelle et individuels sur le fonctionnement à plus grande échelle seront qualifiées de transversales et nous intégrerons alors une ART.
Depuis trois ans, l'équipe se structure en fonction des actions que l'on propose au PNEC. Au départ, Yvan Lagadeuc était le responsable des actions puis m'a transmis progressivement le flambeau depuis ces deux dernières années. Les questions que l'on aborde maintenant, nous ont conduit à intégrer de nouvelles personnes apportant soit une problématique de modélisation soit une approche complémentaire soit un cas d'étude nécessitant une nouvelle approche telle que celle que l'on développe au sein de l'équipe. Je ne citerai ici que les principales personnes : Jean-Christophe Poggiale, mathématicien du Centre d'Océanologie de Marseille, Ovide Arino, mathématicien de l'Université de Pau, François Carlotti, biologiste marin de la station marine d'Arcachon. Aujourd'hui, je suis co-reponsable avec Yvan Lagadeuc et François Carlotti de deux activités au sein de l'action "Hiérarchisation de processus et transferts d'échelles dans le milieu marin par une approche de modélisation dynamique" pilotée par Jean-Christophe Poggiale. Ces activités sont financées depuis deux ans. A l'heure actuelle, nous sommes en phase de discussion au sein du PNEC pour reconstruire de nouvelles ART en fonction des projets émergents et des nouveaux orientations de la recherche dans le domaine de la biologie marine.

Pourquoi mettre autant en avant cette activité de recherche en écologie marine ? La réponse est simple : c'est notre inépuisable source de questionnement. Lorsque l'on évoque le terme de systèmes complexes, n'y-a-t-il pas de meilleur exemple que les systèmes naturels ? Afin de profiter au maximum de notre expérience en modélisation de systèmes complexes et en conception d'outils de simulation, nous avons naturellement mené en parallèle une réflexion de fond sur l'activité de modélisation centrée individus pour les systèmes spatio-temporels et le couplage de modèles hétérogènes. Cette réflexion s'inscrit pleinement dans deux groupes de travail : MIMOSA du GdR I^3 piloté par Jean-Pierre Müller et GTIMMS rattaché au GdR I^3 et au tout nouveau GdR MACS piloté par Michèle Chabrol et Nasser Mebarki.
Le premier groupe de travail né en 2001 sous l'impulsion de Jean-Pierre Müller réunit des chercheurs intéressés par la construction d'un *framework* pour la modélisation et la simulation centrées individus de systèmes spatio-temporels. Cette initiative est motivée

par une volonté d'unification des approches centrées individus dans le domaine. Le travail passe dans un premier temps par la mise à plat des concepts, du vocabulaire utilisé par chacun. Il donnera lieu à plusieurs versions d'un modèle d'ontologie. On retrouve principalement dans ce groupe des chercheurs du monde agent. L'action de notre équipe a été inscrite dès la création du groupe et nous nous intéressons à la question de la spécification XML des éléments du *framework* et de la spécification des expériences toute en participant à la réflexion globale sur l'ontologie. Une fois encore, l'intérêt pour notre équipe est de confronter nos idées issues de notre propre expérience aux idées développées dans les autres laboratoires (CIRAD, IRD, LIRMM, LISC, LIP6, ...). Ce groupe sera pour nous l'occasion de se rapprocher du LISC du Cemagref de Clermont-Ferrand qui développe des outils informatiques proches des nôtres. Une collaboration active existe depuis entre nos deux équipes.
Le deuxième groupe de travail GTIMMS a aussi été crée en 2001 sur l'initiative de Manadou Traoré du LIMOS de Clermont-Ferrand. L'objectif est de créer un groupe de réflexion en informatique sur la multi-modélisation et la simulation. Le groupe de pilotage dont je fais parti, est alors constitué de chercheurs issus de domaines d'applications très divers (systèmes industriels, automatique, réseaux, micro-électronique, ... et systèmes naturels). Après trois années de fonctionnement, on peut affirmer que, pour notre équipe, cette participation à la réflexion est complémentaire à celle menée au sein du groupe MIMOSA et permet d'étendre le débat à d'autres méthodologies, à d'autres approches et à d'autres domaines. De plus, ce fut l'occasion de croiser Norbert Giambiasi et de ce fait d'entrer dans la communauté de la spécification formelle de modèles à base d'événements discrets. Cette rencontre fut pour nous une révélation et nous permettra d'entrevoir les perspectives possibles des travaux engagés dans ce domaine depuis les années 70. En terme de bilan, GTIMMS par l'intermédiaire de Nasser Mebarki, organise la prochaine session de MOSIM à Nantes et une Action Spécifique dirigée par Norbert Giambiasi a été crée pour poursuivre la réflexion sur la conception d'une théorie de la simulation. Cette action a pour but de définir les actions à mener en France et en Europe en terme de recherche en simulation. Naturellement, l'équipe s'attachera dans l'avenir à participer à cette action dont elle fait partie.

Le rapprochement de notre équipe, du LISC (de Guillaume Deffuant) et du LSIS (de Norbert Giambiasi) a permis de créer une équipe projet multi-laboratoires financée par le département STIC du CNRS. L'objectif est de réfléchir sur un formalisme de spécifications DEVS des systèmes multi-agents et de l'utilisation des concepts agents pour la conception d'environnements de simulation DEVS et G-DEVS. Cette équipe est en cours de mise en place et l'arrivée de nouveaux étudiants en thèse (Gauthier Quesnel en particulier) devra permettre de développer ces objectifs.

L'avenir de l'équipe est maintenant sur des rails. Nous disposons d'un ancrage solide dans les communautés scientifiques de la modélisation et la simulation agent et centrée individus, et de la modélisation et de la simulation au sens large. Nous sommes intégrés et nous participons activement aux principaux groupes de réflexion. L'équipe fait parti de plusieurs projets reconnus et financés. Nous participons aux comités de programmes de conférences nationales et internationales dans le domaine de la modélisation et de la

simulation. L'ouverture en 2002 d'un DEA commun entre le LIL et le LASL (Laboratoire d'Analyses des Systèmes du Littoral) à l'Université du Littoral nous a permis pour la première fois d'accueillir des étudiants de DEA pour travailler sur notre problématique. Cet accueil d'étudiants a débouché dès 2003 à l'inscription d'un étudiant en thèse dans l'équipe. De plus, l'équipe a été renforcée par Jean-Christophe Soulié qui a été recruté en 2003. Il vient compléter l'équipe et son expertise en modélisation et simulation multi-agents apporte un plus au groupe.

Après ce rapide historique de la construction de l'équipe, nous allons nous attacher à présenter plus précisement nos principaux résultats. La première partie fera le tour du domaine en question : la modélisation et la simulation de systèmes complexes. Ce tour d'horizon sera l'occasion de présenter les fondements de la spécification formelle de modèles. Les concepts de schéma d'expérimentation, de modèle, de paradigme, de formalisme, de multi-formalismes et de couplage seront défini et nous permettrons de définir les questions fondamentales que l'on se pose. Nous mettrons en évidence l'aspect fondamental de la notion de paradigme et de formalisme dans le cadre de la multi-modélisation. Nous ouvrirons alors le débat sur le couplage de modèles afin d'introduire DEVS et les extensions qui nous intéressent dans le cadre de la modélisation multi-agents. DEVS posera le cadre formel de la modélisation de systèmes dynmiques et nous exposerons à cette occasion notre proposition d'extension de DEVS pour les agents.
Dans le deuxième chapitre, nous poursuivrons par l'exposé de notre *framework* qui est le résultat de nos divers interactions avec les groupes de travail dans lesquels on travaille. Dans un premier temps, le *framework* sera défini de manière abstraite et sera mis en perspective par rapport aux travaux existants dans le domaine. Cette mise en perspective mettra en évidence les élements intéressants à développer dans le domaine ce qui nous conduira à développer notre exposé vers la spécification de modèles et d'expériences et leur expression en XML. Les concepts de modèle et d'expérience tels qu'on les aura défini, nous conduirons à présenter notre intégration de DEVS dans les spécifications. De plus, nous mettrons l'accent sur la résolution de l'expression formel du couplage de modèles mais aussi du temps et de l'espace. Ces deux derniers aspects seront des exemples de résultats provenants des réflexions du groupe MIMOSA. Après un exposé un peu technique lié à la spécification et à XML, nous placerons tous ces éléments dans notre plate-forme de modélisation et de simulation VLE. Nous ferons une rapide présentation des différentes versions qui ont suivi l'évolution de nos idées.
Nous finirons ce document par un résumé des principaux résultats de nos travaux dans le domaine des écosystèmes marins entre autres. Les écosystèmes marins ont été et restent notre principal domaine d'applications. Le caractère pluridisplinaire du domaine est source de modélisations multiples où les notions de transfert d'échelle, de couplage dynamique et de points de vue multiples sont omniprésents. Nous présenterons donc notre cadre d'étude privilégié : la compréhension de la dynamique du système "sels nutritifs - phytoplancton - zooplancton". Nous chercherons à répondre aux questions suivantes :

- le zooplancton (et en particulier, les organismes de la famille des copépodes) ont-ils des comportements actifs de nutrition ?
- quel est l'impact de la distribution spatiale de la nourriture (le phytoplancton) sur l'ingestion du copépode ?

- comment construire des modèles centrés individus du système "phytoplancton - zooplancton" prenant en compte les problèmes d'hétérogénéité spatiale ?
- comment coupler des modèles prenant en compte des processus au niveau individu avec des modèles de dynamique de populations ?
- le *framework* défini dans le chapitre 2 est-il viable dans des cas réels ? quel est le retour de la réalité de la modélisation sur des cas concrets sur le *framework* ?
- comment atteindre des simulations rapides pour des systèmes de plus en plus complexes en terme de nombre d'entités et de nombre de types d'interactions ? comment utiliser la puissance des modèles distribués ?
- que peut-on faire pour les systèmes flexibles de production avec la plate-forme VLE ?

Les réponses à ces questions s'inscrivent dans une réflexion plus large sur les concepts de couplage de modèles (couplage faible, couplage fort, couplage dynamique, ...), de transferts d'échelle, de modèles multi-échelles, ... Ces questions sont aussi au cœur des questions théoriques et méthodologiques de la biologie.

En conclusion, nous établirons les perspectives de l'équipe à travers un projet de recherche. Ce projet de recherche s'attachera à montrer les orientations présentes et futures de l'équipe. Ces orientations graviteront autour de cinq points :

- la modélisation centrée individus et la spécification formelle de systèmes spatio-temporels (en particuliers en écologie marine),
- les outils formels pour le couplage de modèles,
- la simulation distribuée à événements discrets et la spécification DEVS,
- le développement d'un serveur de services Web pour la modélisation et la simulation et l'intégration de XML,
- l'étude des méthodes de calcul émergent et d'agrégation de variables pour la modélisation de processus multi-échelles.

Chapitre 1

Modélisaton des systèmes complexes : définitions et problématiques

Résumé

L'objectif de ce premier chapitre est de poser les définitions des concepts manipulés en modélisation des systèmes complexes. On verra donc la définition que l'on a adopté pour les concepts de système, de système complexe, de système spatio-temporel, de paradigme, de formalisme et de multi-formalismes. Ce panorama nous conduire à évoquer les problèmes sous-jacents de la modélisation et en particulier de la multi-modélisation. Nous montrerons les réponses apportées par la littérature au problème de couplage de modèles tant au niveau formel qu'opérationnel. Nous introduirons les approches de spécifications formelles des systèmes dynamiques de Zeigler. Cette introduction nous permettra d'évoquer la modélisation multi-agents dans ce cadre formel et nous montrerons comment certaines extensions de DEVS peuvent apporter un complément de réponse à la formalisation des systèmes multi-agents.

Sommaire

La modélisation est au cœur de la compréhension du monde et de la conception d'objets nouveaux. Elle devient au même titre que la simulation, une science à part entière où les questions restent nombreuses malgré plusieurs dizaines années de recherche dans le domaine. Des théories de la modélisation et des théories de la simulation sont nées et ont mis à la disposition des modélisateurs des outils de spécification formelle, de simulation, ..., des démarches et des méthodes.
Que reste-t-il à conceptualiser, à mettre au point ou à valider ? Pour répondre à cette question, nous allons décomposer l'activité de modélisation et de simulation en plusieurs actions :

- définir les questions que l'on se pose sur le système,
- définir ce qu'est le système, les éléments le composant et leurs relations, en fonction de la question posée,
- établir un plan d'expériences conformément à la question à résoudre,
- adopter un ou plusieurs paradigmes et le ou les formalismes associés pour construire un modèle du système,
- décomposer le système et modéliser chacune des parties avec le formalisme le plus adéquat,
- simuler le modèle en fonction du plan d'expériences,
- analyser les sorties des simulations.

Les trois premières étapes constituent le préalable indispensable à l'activité de modélisation et de simulation. La définition du système à étudier est fonction des questions que l'on se pose mais pour aller plus, il est aussi nécessaire de poser clairement les protocoles expérimentaux avant toute opération de modélisation. Cette remarque peut paraître évidente dans le cadre de l'étude d'un système réel c'est à dire lorsque le chercheur dispose du système réel et que les expériences se déroulent directement sur l'objet d'étude. L'approche par modélisation et simulation est légèremment différente : le système réel est vu par l'intermédiaire d'un modèle. Ce filtre peut faire oublier le but premier de l'activité de modélisation. Il ne faut pas penser que le but est d'offrir le modèle le plus complet et le plus "beau". Il faut construire le meilleur modèle avec le formalisme le plus adapté pour répondre à la question posée. En effet, la modélisation n'est qu'un maillon dans la chaîne de la construction de la Connaissance du monde : on ne fait pas de la modélisation pour la modélisation mais pour comprendre le fonctionnement du Monde et comprendre les impacts de certaines perturbations sur ce Monde. L'exemple qui nous touche directement, de par nos collaborations, c'est le changement climatique : l'augmentation de la température va-t-elle perturber durable les écosystèmes ? Pour répondre à cette question, il faut comprendre les mécanismes gouvernant les écosystèmes et mettre en évidence les processus clés. Dans cette perspective, la modélisation est un outil de représentation du Monde et cette représentation est un objet sur lequel l'expérimentation va se dérouler.
Le système, les questions et le schéma d'expérimentations étant posés, le modélisateur doit procéder à des choix méthodologiques et d'approches qui sont très souvent gouvernés par les paradigmes accessibles au modélisateur. La notion d'accessibilité est synonyme ici de capacité du modélisateur à utiliser un formalisme plutôt qu'un autre. Ce point est important car dans beaucoup de cas, ce sont les connaissances en terme de techniques qui conditionnent le choix. Néanmoins, les paradigmes sont au cœur de l'activité de formalisation des modèles et ne sont donc pas à prendre à la légère. Il est vital de faire le bon choix et de ne pas chercher à adapter son paradigme ou son formalisme préféré au système à modéliser. Chaque formalisme est adapté à une famille de systèmes et chaque famille est caractérisée par un certain nombre de propriétés (déterministe ou stochastique, discret ou continu, ...). Nous tenterons dans ce chapitre de faire

le point sur les formalismes et nous introduirons DEVS comme un *framework* de modélisation dont l'objectif est de fédérer la majorité des formalismes disponibles.
La question de l'adéquation du paradigme et du formalisme au système nous paraît central mais, dans le cadre des systèmes complexes, cela conduit à une reflexion sur l'action de décomposition du système et par conséquence, au couplage des éléments du modèle qui peuvent avoir fait l'objet d'une formalisation de nature différente. Le couplage de modèles fait parti intégrante que nos travaux et forme la problématique centrale de la thèse de Raphaël Duboz. Le couplage peut être perçu de plusieurs manières différentes selon la nature de la décomposition et des relations entre les éléments du modèle. De nombreux travaux existent et notamment ceux de Fishwick [FIS95] et de Zeigler [ZEI73]. Ces travaux seront présentés dans ce chapitre et dans le chapitre 2. Pour notre part, nous nous intéresserons à trois formes de couplage : le couplage faible, le couplage fort et le couplage hiérarchique. Ces aspects seront abordés dans le chapitre 3. Néanmoins, on peut déjà en donner une définition. Le couplage faible définit une relation unidirectionnelle entre deux modèles : un modèle influence un autre modèle. On parle alors de paramétrage. Dans le cas du couplage fort ou de couplage dynamique, les deux modèles sont en interaction. Le couplage hiérarchique quant à lui définit une relation de décomposition. Un modèle est composé d'un ensemble de sous-modèles et les sous-modèles sont interconnectés. Ce que nous tenons à ajouter à ce sujet est que ces relations entre modèles posent des problèmes sémantiques. Par exemple, coupler dynamiquement deux modèles n'engendre-t-il pas dans certains cas des changement d'échelles ? Cette question sera évoquée dans le chapitre 3. Nous présenterons dans ce chapitre les réponses disponibles pour le couplage de modèles et en particulier les travaux sur DEVS.

Il nous reste à évoquer les derniers étages de l'activité de modélisation et de simulation : la simulation et l'analyse. La simulation est une mise en action d'un modèle qui s'accompagne de l'observation du comportement de ce dernier. On peut distinguer deux approches du calcul du comportement d'un modèle. Si le formalisme utilisé est apparenté aux équations différentielles et qu'il existe une méthode de calcul de la solution analytique des équations, alors le comportement peut être directement déduit du comportement mathématique de la solution. Ce cas de figure est un idéal difficile à atteindre dès que le système et donc le modèle est complexe. La deuxième approche est celle de la simulation (ou de la résolution par simulation). La construction d'une simulation se résume très souvent en la construction d'un programme informatique. Ce programme informatique est soit une mise en oeuvre du modèle reposant totalement sur le formalisme soit un développement spécifique. Dans le premier cas, le formalisme est accompagné d'algorithmes de calcul du comportement et le modélisateur construit son simulateur en appliquant directement les algorithmes associés au formalisme. Cette approche n'est pas toujours la plus efficace. Le deuxième point de vue est de considérer le formalisme de modélisation comme uniquement un moyen de formaliser le modèle et le développement du simulateur est indépendant de la formalisation. Cette approche de la simulation a un avantage : il permet de définir un simulateur qui est dans la plupart des cas plus efficace. En revanche, on ne dispose plus alors de spécification formelle de l'implémentation du modèle. Si on désire savoir ce qui a été effectivement mis en oeuvre, il faut se plonger dans le code du simulateur. Ce point de vue n'est pas acceptable et V. Grimm le dénonce dans [GRI99] au sujet de la modélisation centrée individus en écologie. Nous pensons que malheureusement ce constat est vrai pour d'autres démarches de modélisation et dans bien d'autres domaines. Il est donc urgent de lutter comme ce manque de spécification formelle.

En conclusion de cette introduction du premier chapitre, nous pouvons dire que finalement l'activité de modélisation et de simulation reste un grand chantier et que les questions sont encore nombreuses. Pour restreindre un peu notre champ d'actions, on peut dire qu'aujourd'hui l'équipe tente d'apporter des réponses aux problèmes :

- de formalisation des modèles et du couplage de modèles hétérogènes,
- de formalisation des plans d'expériences et des simulateurs,
- de l'expression de ces formalisations,
- de concepts tels que le changement d'échelle, le couplage (faible, fort, ...),
- de démarche de couplage de modèles,
- de l'intégration de la modélisation centrée individus dans le processus de modélisation multi-paradigmes.

Ces différents points vont être abordés au fil du présent document. Dans ce premier chapitre, nous allons posé les définitions et les éléments qui nous ont servi de base dans notre travail. La première partie va tente de définir les concepts manipulés en modélisation et simulation de systèmes et nous enchaînerons sur une introduction à la spécification formelle afin de montrer l'importance de l'adéquation entre le système et le formalisme. La démonstration aboutira sur une présentation de DEVS et de son intérêt dans l'unification des formalismes dans une optique de couplage de modèles. Ce sera l'occasion de montrer que l'on peut adapter DEVS pour spécifier formellement des modèles centrés individus. Nous proposerons en effet un début d'extension de DEVS : agentDEVS.

1.1 Définitions

Même si nous les avons déjà employé dans l'introduction de ce chapitre, il est important de définir clairement les termes suivants : système, système complexe, système spatio-temporels, modèle, niveau de spécification formelle des modèles, paradigme et formalisme. Ces définitions permettent de fixer les idées et surtout sont l'occasion de préciser notre point de vue sur les concepts de l'activité de modélisation et de simulation.

Qu'est-ce qu'un système ?

La notion de système est à associer aux objectifs de l'étude que l'on mène. En effet, étudier un système c'est se poser des questions. Le fait même de se poser des questions définit ce qu'est le système. Un système est donc l'ensemble des éléments concernés par l'interrogation. Dès lors que l'on s'intéresse à chaque chose pour en comprendre le fonctionnement (la dynamique), on doit être capable d'identifier les éléments qui constituent l'objet d'étude. Cette définition intégre l'idée que l'on doit être capable de se limiter aux éléments essentiels et de prendre en compte seulement les éléments succeptibles de répondre à la question posée. Cette opération n'est pas toujours simple et le chercheur doit souvent procéder par tâtonnements.

Qu'est-ce qu'un système complexe ?

La définition précédente nous dit qu'un système est un ensemble d'éléments ou d'entités. Ces entités possèdent des comportements ou des dynamiques propres qui par composition et interactions entre elles forment la dynamique du système. Si on étend cette définition, on peut définir un système complexe comme un ensemble composé d'un grand nombre d'entités en interaction et le comportement global d'un système complexe émerge de l'interaction des entités composant

le système.

En systémique, on dit qu'une des caractéristiques des systèmes complexes est en effet d'être hiérarchisées et que chaque système est composé de sous-systèmes interconnectés et élément du super-système. Ce point de vue forme un courant fort chez les modélisateurs. Il est intéressant de se pencher sur la philosophie des Sciences pour comprendre cet état de fait.

Plusieurs courants de pensées ont vu le jour au courant de l'histoire des Sciences : le globalisme, le totalisme, l'holisme, le réductionnisme, ... Ces courants ont conduit à une perception des systèmes bien différents. En quelques mots, le globalisme et le totalisme ne croit pas nécessaire une vision d'un Monde décomposable en éléments plus petits et surtout ne croit en l'idée que le comportement global émerge des interactions des parties. Le globaliste pense qu'il n'y a pas de limite entre les choses et donc qu'une division du système en éléments n'est pas nécessaire. Un exemple simple permet d'illustrer ce précept. Un martien trouve une calculatrice : pour comprendre ce que c'est, est-il nécessaire qu'il ouvre l'objet pour découvrir qu'elle est composée d'un circuit électronique et qu'il est lui-même composé d'éléments plus petits ? Le globaliste dit non. Ce qui est important c'est la fonction globale de l'objet. La connaissance des parties n'est pas nécessaire voire néfaste à la compréhension.
Le holisme, autre courant de pensées, postule que le Tout précéde ou transcende ses parties avec la célébre définition : "le Tout est plus que la somme de ses parties". Cette vision du Monde implique que le Tout n'est pas seulement la réunion d'éléments mais qu'Il participe au comportement du Tout. Mais quelle est cette chose qui donne corps au Tout ? C'est la question que se pose un réductionniste. Pour lui, toute réalité se réduit en fin de compte à des constituants élémentaires. En biologie, on dira que le vivant se réduit à des molécules et à leurs interactions. Le monde est vu comme une hiérarchie de niveaux ordonnés suivant des échelles de complexité, d'espace et de temps, qui s'emboîtent. Le réductionnisme ontologique est tout à fait compatible avec l'idée que les systèmes complexes puissent possèder des propriétés spécifiques qui n'existeraient pas à un niveau inférieur.
Du point de vue de la connaissance, le réductionnisme espistémologique affirme que la connaissance d'un phénomène ne peut se faire qu'en réduisant ses multiples descriptions à un nombre de plus en plus restreint de principes, lois, théories ou concepts. Le scientifique élimine ce qui ne paraît pas essentiel à sa compréhension, fait un choix de paramètres ou de variables susceptibles afin de déterminer la structure du système qu'il étudie. Il est difficile de procéder autrement sous peine de se voir dépasser par la quantité d'informations.
Par l'introduction de la notion d'interactions entre les parties, les réductionnistes prétendent assimiler le précepte des holistes : "le Tout est plus que la somme des parties". En effet, les interactions auraient la propriété de faire émerger des propriétés au niveau du Tout qui ne sont pas décrites en tant que tel au niveau des parties. Cette idée est la pierre angulaire des approches multi-agents et centrées individus. Pour l'instant, nous ne tenons pas à prendre partie.

Pour conclure sur la notion de systèmes, de systèmes complexes et d'approches de modélisation du Monde, J. L. Lemoigne dans [LEM77] propose le nouveau discours de la Méthode et érige quatre préceptes :

- la pertinence : "... *convenir que tout objet que nous considérerons se définit par rapport aux intentions implicites ou explicites du modélisateur. Ne jamais s'interdire de mettre en doute cette définition si, nos intentions se modifiant, la perception que nous avions de cet objet se modifie* ..." ; J. L. Lemoigne définit ici la notion de point de vue et de la

pertinence par rapport à un point de vue (Que cherchons-nous à montrer ? Quelles sont nos intentions ?) ; Un modèle est pertinent ("vrai") par rapport à nos intentions (hypothèses) ; on rejoint ici la notion de schémas d'expérimentation.
- globalisme : "*... considérer toujours l'objet à connaître par notre intelligence comme une partie immergée et active au sein d'un plus grand Tout. Le percevoir d'abord globalement, dans sa relation fonctionnelle avec son environnement sans se soucier outre mesure d'établir une image fidèle de sa structure interne, dont l'existence et l'unicité ne seront jamais tenues pour acquises ...*"
- télélogique : "*... interpréter l'objet non pas en lui-même, mais par son comportement, sans chercher à expliquer a priori ce comportement par quelque loi impliquée dans une éventuelle structure. Comprendre en revanche ce comportement et les ressources qu'il mobilise par rapport aux projets que, librement, le modélisateur attribue à l'objet ...*"
- agrégativité : "*... convenir que toute représentation est partisane, non pas par oubli du modélisateur, mais délibérémment. Chercher en conséquence quelques recettes susceptibles de guider la sélection d'agrégats tenus pour pertinents et exclure l'illusoire objectivité d'un recensement exhaustif des éléments à considérer ...*"

Nous pensons que ces 4 préceptes sont à méditer et que tout modélisateur devrait avoir en tête ces préceptes fondamentaux. Pour notre part, on aimerait ajouter un cinquième précepte celui dicté par les réductionnistes : "... toute réalité se réduit en fin de compte à des constituants élémentaires ...". Ce dernier précepte ouvre la porte à l'approche par décomposition qui aujourd'hui est l'approche dominante. Néanmoins, il est une fois encore important d'avoir en tête les 4 précédents préceptes.

Qu'est-ce qu'un système dynamique et un système spatio-temporel ?

Les travaux que l'on mène s'inscrivent principalement dans le cadre de systèmes spatio-temporels. La définition générale d'un système ne fait pas apparaître explicitement la notion de temps et encore moins la notion d'espace. La théorie des systèmes distingue deux choses : la structure interne du système en terme d'éléments qui le compose et le comportement. Le comportement peut être vu comme le résultat de la dynamique interne du système. Le système peut alors être représenté comme une boîte noire composée uniquement d'entrées et de sorties. On ne s'intéresse pas à la compréhension de la dynamique interne. On observe seulement les manifestations externes. Si le temps n'est pas exprimé alors les sorties sont mises en correspondance avec les entrées. Si on place X en entrée alors on a Y en sortie. On dispose alors d'une table de correspondance comme dans le cas d'un circuit logique. Nous reviendrons sur ce point de vue dans la hiérarchisation des niveaux de spécifications des systèmes.

Un système dynamique est un système où la dynamique est définie par rapport à une base de temps, un état courant et une fonction d'évolution des états en fonction des entrées du système. Clairement, le temps est une composante forte des systèmes dynamiques. Les changements d'états du système s'inscrivent relativement au temps. On peut alors identifier sur un axe temporel l'évolution de l'état du système. La fonction d'évolution est alors appelée fonction de transition.

Un système spatio-temporel est par définition un système dynamique dans lequel les éléments et les processus sont définis relativement à un espace. L'espace est ici entendu comme l'espace dans lequel des entités peuvent évoluer. Les entités possèdent alors des coordonnées relatives à

un repére absolu. De même, les processus sont descrits dans l'espace des entités. Les écosystèmes sont de parfaits exemples de systèmes spatio-temporels.

Qu'est-ce qu'un modèle ?

L'étude d'un système implique son observation. Or observer un système, c'est avant tout construire (même inconsciemment) un modèle du système observé. Nos connaissances et les outils utilisés pour l'observation impliquent de fait un filtrage de la réalité par un modèle. Lorsque l'on regarde une chaise, notre oeil en capte une image que l'on considère comme la réalité. Cette réalité est une construction physico-chimique d'une image mentale d'un objet réel. Une caméra infra-rouge et une mouche auront sûrement une autre vision de la même chaise.

Un modèle est donc un filtre conditionné par nos connaissances, nos vérités et nos capteurs. Si on restreint le discours à l'activité de modélisation, le modèle est le résultat de l'activité de modélisation. Cette dernière implique un schéma expérimental, un ou des paradigmes et une méthodologie. La notion de paradigme se substitue ici à la notion de vérité dans le cadre général (voir figure 1.1).

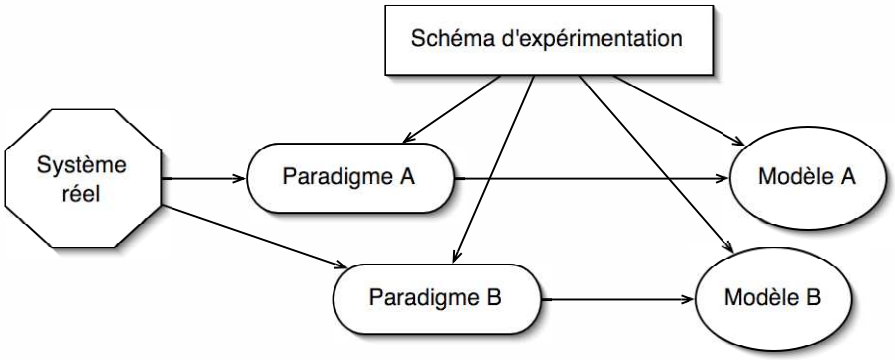


FIG. 1.1 – L'activité de modélisation

Les niveaux de spécifications formelles des modèles

Avant d'aborder les notions de paradigme et de formalisme, attardons nous un instant sur la proposition de B. Zeigler dans [ZEI73]. Un système peut être spécifié formellement et cette spécification dépend d'un certain niveau de description. Cette approche purement formelle permet de décrire un système sans aucune hypothèse liée à un paradigme ou à un formalisme.

Le premier niveau de description d'un système différe peu de la définition de boîtes noires telle que nous l'avons défini précédemment. Les entrées du système sont liées à un intervalle de temps que l'on parlera de segment et noté ω. On définit aussi la notion de segment de sortie ρ ou de trajectoire de sortie qui met en relation un intervalle de temps et la sortie observée. B. Zeigler nomme ce type de spécifications des schémas d'observations. Aucune relation n'est établie entre les entrées et les sorties. Nous sommes ici en présent d'une simple observation d'un phénomène. On recherche aucune interprétation au phénomène.

Si on établit une relation entre les segments d'entrée et les segments de sortie alors on définit le deuxième niveau de spécification formelle d'un système. Ce niveau est appelé observation de la relation d'entrée-sortie. La relation a pour but de construire un ensemble de couples (ω, ρ) sachant que ω et ρ sont observés sur le même intervalle de temps. L'inconvénient majeur de cette spécification est d'être non-déterministe. En effet, pour un même segment d'entrée ω, il peut exister plusieurs segments de sortie ρ. On n'est donc pas capable de prévoir la sortie par simple observation de l'entrée du système.

Il manque effectivement une notion essentielle pour les systèmes dynamiques dans les deux premiers niveaux de spécification : la notion d'état. Il suffit, en effet, d'ajouter la spécification de l'état initial du système pour rendre déterministe la relation d'entrée-sortie. On parle alors d'observation de la fonction d'entrée-sortie.

Le quatrième niveau d'abstractions s'attache à définir la spécification formelle d'un système dynamique. A ce niveau, on va chercher à spécifier le comportement interne du système. Pour cela, on introduit la notion d'état interne et de fonctions de transition.

$$S =< T, X, Y, \Omega, Q, \Delta, \Lambda >$$

où Q est l'ensemble des états du système, Δ la fonction de transition ($\Delta : Q \times \Omega \rightarrow Q$) et Λ la fonction de sortie ($\Lambda : Q \rightarrow Y$). L'ensemble Ω est l'ensemble des segments d'entrée admissibles tandis que X est l'ensemble des valeurs d'entrée possibles. La différence réside dans le fait que Ω définit des couples $(< t_1, t_2 >, x)$ où $x \in X$.
Cette formalisation des systèmes dynamiques permet de décrire le comportement du système en fonction de son état interne et des événements d'entrée. Or, dans cette vision, on considère que le système n'est pas autonome et qu'il n'a donc pas de dynamique propre puisque seuls des événements externes peuvent modifier l'état interne du système. Nous verrons par la suite que B. Zeigler proposera un cadre formel de spécification de systèmes dynamiques généralisant ce niveau d'abstractions en ajoutant une fonction de transition interne. Cette dernière aura pour objectif de modéliser la dynamique interne.

Le cinquième et dernier niveau introduit le concept de couplage de modèles. Le modèle global est alors un ensemble de modèles interconnectés. Ces connexions déterminent les relations entre les modèles et les événements échangés entre les modèles. A ce niveau, on considère que le modélisateur connait la structure interne de son système et qu'il est capable de le décomposer en sous-modèles.

Cette hiérarchisation des niveaux de spécifications formelles des systèmes sera par la suite notre référence dans la plupart de nos choix. Il est intéressant de retenir les idées d'interaction entre modèles à l'aide d'événements, de fonctions de transitions, de fonctions de sortie et de couplage de modèles.

Qu'est-ce qu'un paradigme ?

Sans le préciser lorsque B. Zeigler propose ses niveaux de spécifications, il exploite un paradigme celui de la modélisation à événements discrets. Un paradigme est un cadre de pensée composé par un ensemble d'hypothèses fondamentales, de lois et de moyens sur la base desquels les modèles peuvent se développer.

Dans ce document, nous allons exploiter quelques paradigmes : le paradigme multi-agents, les équations différentielles et les événements discrets, en particulier. Dans la plupart des cas, la notion de paradigme est difficilement dissociable de la notion de formalisme. En effet, les équations différentielles sont à la fois un paradigme et un outil de formalisme mathématique.
En règle générale, le paradigme est utilisé au niveau sémantique (ou méta) tandis que le formalisme est l'outil opérationnel du paradigme. À un paradigme est associé un ou plusieurs formalismes. De plus, on peut concevoir des modèles en s'appuyant sur les concepts d'un paradigme et spécifier le modèle à l'aide d'un formalisme qui n'est pas obligatoirement issu du paradigme. Si on prend l'exemple du paradigme multi-agents, la spécification de la structure du système peut être réalisé à l'aide d'UML (*Unified Modeling Language*) tandis que la dynamique des agents est spécifiée à l'aide de réseaux de Petri ou de règles d'inférences.

Qu'est-ce qu'un formalisme ?

Les définitions précédentes ont posé les bases de la modélisation en définissant ce qu'est un modèle. Nous avons vu aussi qu'il était possible de formaliser un système selon différents niveaux d'abstraction, niveau qui dépend la plupart du temps du niveau de connaissances sur le système. Revenons, par exemple, sur la notion de fonction de transition. Cette fonction a pour objectif de déterminer le nouvel état du système en fonction des événements d'entrée (ou externes). La question qui reste en suspens est comment spécifier cette fonction. Le formalisme est l'outil dédié à cette opération. C'est l'outil d'expression des paradigmes (voir "Qu'est-ce qu'un paradigme ?"). On peut utiliser, par exemple, les automates à états finis pour spécifier les changements d'états. Ce n'est pas la seule possibilité.

Nous détaillerons un peu plus loin le rôle central du formalisme et du paradigme.

Qu'est ce que le multi-formalisme ?

Modéliser un système, c'est construire un objet abstrait représentant la structure et la dynamique du système observé. Pour un même système et pour des modélisateurs différents, l'activité de modélisation va donner naissance obligatoirement à des modèles différents. Essayons de comprendre ce que signifie l'adjectif "différent".

Le premier cas de figure est le "plus simple" : les deux modélisateurs ont le même objectif, ils veulent comprendre la même chose et ils en ont le même point de vue. La seule différence réside dans les outils utilisés. Le choix d'outils différents est très souvent gouverné soit par une méconnaissance des outils existants (et en plus, l'outil que l'on utilise est le sien, tout simplement), soit par la capacité de l'outil choisi pour exprimer la structure ou la dynamique (les deux outils choisis par les modélisateurs ont des capacités équivalentes mais les modélisateurs ont décidé de choisir celui avec lequel ils ont l'habitude de travailler). Dans ce cas, on cherche à vérifier que les deux modélisations sont équivalentes et on parle alors de *mapping*. Le *mapping* cherche à construire une bijection entre les éléments des deux modèles.

Le deuxième cas de figure met en jeu une démarche de hiérarchisation des modèles. Le système est décomposable en sous-systèmes et chaque sous-système fait l'objet d'une modélisation. La notion de décomposition est prise ici au sens large. La décomposition peut aussi s'appliquer sur les éléments de la dynamique du système. Dans ce cas, on peut, par exemple, décomposer un état du système en sous-états. Les formalismes et les paradigmes utilisés pour le système

global (ou pour les états du système global) et les sous-systèmes (ou les sous-états) peuvent être différents. Ce point de vue est développé par P. A. Fishwick dans [FIS95][FIS93]. Il parle alors de raffinement ou d'abstraction dans un cadre de multi-formalisme. Le prérequis à un tel type de multi-formalisme est que les paradigmes et les formalismes utilisés soient compatibles. Prenons un extrait de l'exemple (voir figure 1.2) développé dans [FIS93]. Le système modélisé est une casserole avec de l'eau en train de chauffer et la casserole est posée sur un système de chauffage. Pour le premier niveau de description de la dynamique du système, on peut identifier deux super-états : chaud et froid. Le passage d'un état à un autre est gouverné par l'activation du système de chauffage et la température de l'eau. Le formalisme utilisé pour ce niveau peut être les automates à états finis. L'auteur décompose ensuite ces super-états en sous-états afin de décrire plus précisemment les différents états de l'état "chaud". Le formalisme adopté ne change pas. Nous restons donc dans le cas d'une décomposition hiérarchique. En revanche, l'un des états de l'état "chaud" est à son tour décomposé. Cet état représente l'état "en cours de montée en température" et les conditions de transition sont fonction de la variable "température". Il devient alors intéressant d'utiliser un autre formalisme, ici les équations différentielles, pour répresenter le processus de chauffage et les conséquences sur la variable "température". Quelles sont les implications d'une telle décomposition ? Il faut que le formalisme du sous-état prenne en compte les contraintes du formalisme utilisé au niveau d'abstraction supérieur. Dans l'exemple développé ici, les transitions entrantes sur l'état "en cours de montée en température" doivent être "traduites" dans le formalisme des équations différentielles. Cette traduction détermine les conditions initiales de l'équation différentielle. Réciproquement, les transitions sortantes de l'état sont fonction de la variable décrite par l'équation différentielle.

Les possibilités de couplage sont nombreuses : pilotage d'un état par un réseau de Petri, décomposition d'une transition d'un réseau de Petri en un automate à états finis, ... La décomposition peut aussi s'appliquer à des modélisations structurelles. L'exemple précédent traitait un modèle de comportements. Les règles restent les mêmes : on décompose un élément de la structure en sous-éléments. Si on s'intéresse aux écosystèmes forestiers, on peut identifier l'entité *Forêt* comme un tout à un certain niveau d'abstraction. Une forêt est une composition d'arbres (pour faire simple) qui eux-mêmes sont composés de troncs, de branches, ...

Quelle que soit la forme de modélisation, il faut observer certaines précautions :

- déterminer les contraintes d'un élément d'un formalisme sur les éléments de l'autre formalisme (par exemple, une transition c'est les conditions initiales d'une équations différentielles),
- vérifier la cohérence sémantique,

Nous avons déjà traiter de la première question à travers l'exemple proposé. Le concept de cohérence sémantique est tout aussi important. Il faut, en effet, vérifier la compatibilité sémantique liée, par exemple, aux échelles de temps et d'espace (si le système est spatialisé) ou tout simplement entre les élements du modèle. Ce point sera débattu dans les deux chapitres suivants.

Le concept développé par Fishwick nous semble un peu fort. Nous avons identifié une troisième forme de multi-formalisme. Fishwick centre essentiellement son discours sur la décomposition (ou l'agrégation lorsque l'on modélise le système par une apparoche *bottom-up*) sans proposer un cadre général. Chaque couple de formalismes voire chaque modèle doit faire l'objet d'une étude. Cette étude ayant pour objectif de vérifier l'adéquation des formalismes. D'autre part, Zeigler dans [ZEI95] propose une vision que nous qualifierons de couplage de modèles où les

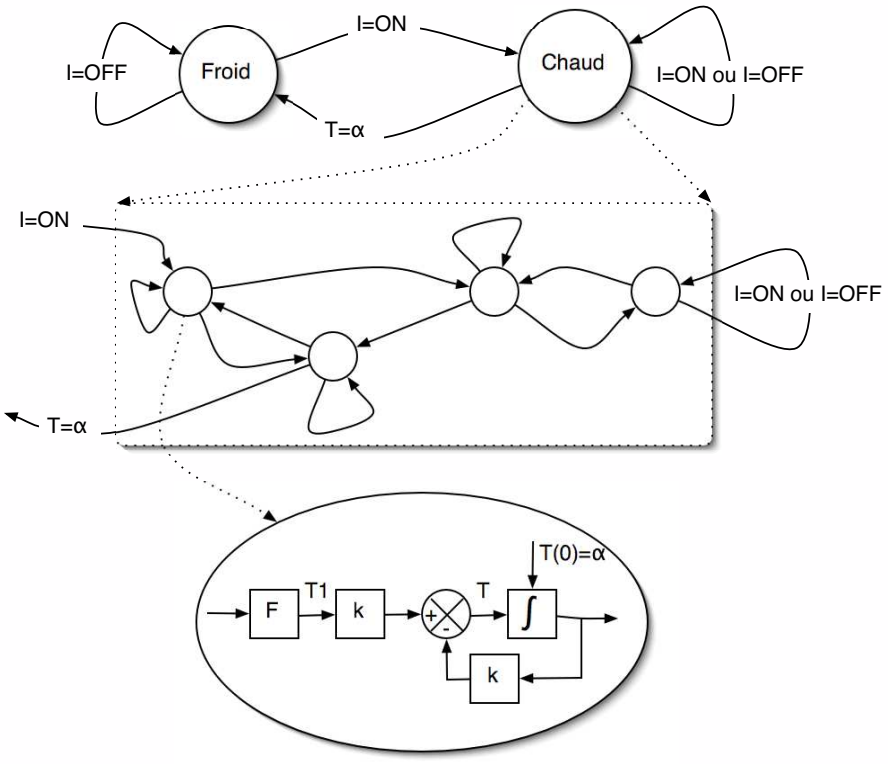


FIG. 1.2 – Exemple de multi-modélisation par décomposition présenté dans [FIS93]

formalismes adoptés pour les différents modèles peuvent être différents. Dans cette vision, Zeigler n'intégre pas l'opération de couplage dans une quelconque démarche (*top-down* ou *bottom-up*). Le couplage fait abstraction de la démarche mais propose en revanche un cadre formel du couplage de modèles. Notre point de vue est donc à la croisée de ces deux courants de pensées auxquels nous ajoutons la notion de cohérence sémantique.

1.2 Formalismes

Traditionnellement, on distingue d'un côté les outils de modélisation de la structure du système (la partie statique ou dynamique comme nous le verrons plus tard) et d'un autre côté les outils de modélisation de la structure dynamique. Le point de vue structurelle décrit les relations statiques qui existent entre les entités du système. Le point de vue dynamique décrit, pour sa part, le comportement des entités du système c'est à dire l'évolution temporelle de l'état du système et de ses parties (les entités). Ces deux points de vue d'un même système a donné naissance à une multitude d'outils. Cette partie va dans un premier temps faire rapidement le point sur ces outils. On essaiera de dégager la philosophie générale de chaque outil en mettant l'accent sur les éléments fondamentaux et originaux. Nous conclurons par une proposition de classification en fonction des caractéristiques du système. Cette proposition a pour objectif d'introduire le deuxième chapitre et de mettre l'accent sur le multi-formalisme et le couplage de modèles.

Cette présentation nous paraît essentielle afin de mieux comprendre les interactions possibles

entre les différents formalismes dans une optique de multi-modélisations. Il est nécessaire de comprendre les éléments fondamentaux des formalismes pour assurer la cohérence sémantique lors du couplage de modèles. Si on se place dans une approche plus abstraite telle que DEVS (voir 1.3), il est tout aussi nécessaire d'être conscient des capacités et des limites des formalismes. Comme nous allons le voir dans la partie 1.3, DEVS et ses extensions auront pour objectif d'encapsuler les formalismes. Or il est nécessaire d'observer certaines précautions et cette encapsulation implique dans la majorité des cas des choix conceptuels. Ces choix ne peuvent pas être pris si les aspects conceptuels des différents formalismes ne sont pas clairs.

1.2.1 Formalismes pour la modélisation structurelle

La modélisation structurelle s'attache à représenter les entités du système et leurs relations. Nous allons développer ici seulement les formalismes qui se proposent de spécifier uniquement la structure du système. Nous verrons par la suite qu'il existe des formalismes où la séparation entre les aspects structurels et comportementaux est difficile. Nous devons dès maintenant donner une définition des concepts d'entité et de relation qui sont à la base de la modélisation structurelle.

Qu'est ce qu'une entité ?

Une entité est un élément d'un système qui possédent un état et un comportement. Cette définition est très proche de celle de système. Quelle est donc la différence ? En effet, une entité est un système mais tout système n'est pas une entité. Un système peut être représenté à l'aide des processus caractéristiques. Dans ce cas, aucune entité n'est décrite. Le système est appréhendé au travers de ces variables caractéristiques et de la dynamique de ces dernières. Pour trancher définitivement la question, on peut dire qu'une entité est un élément discret d'un système. Par exemple, un fluide n'est pas naturellement discret, on choisira donc de le décrire à l'aide d'un système de variables continues. En revanche, un poisson se déplaçant dans le fluide est une entité discrète. On peut en définir les contours.

Qu'est ce qu'une relation ?

Une relation est un lien sémantique entre plusieurs entités. La notion de lien sémantique couvre principalement trois aspects :

- un lien d'accointances ou d'interaction : deux entités se connaissent, elles pourront alors s'échanger des informations
- un lien d'agrégation ou de composition : une entité est formée de l'assemblage de plusieurs entités, une relation hiérarchique est alors établie
- un lien d'héritage : une entité est une généralisation d'une autre entité, les propriétés de l'entité sont formées par l'union des propriétés de l'entité plus abstraite et de ses propres propriétés

Qu'est ce que la modélisation structurelle ?

La modélisation structurelle c'est la description des entités formant le système et leur mise en relation ou c'est la description des variables caractéristiques des processus du système. Cette définition a double sens va donner naissance à deux courants de modélisation : la modélisation

fonctionnelle ou à contraintes et la modélisation déclarative.
La première forme identifie les processus ou les grandes lois gouvernant le système à partir desquels elle identifie les variables caractéristiques. La modélisation comportementale décrit ensuite les relations fonctionnelles entre les variables (par exemple, à l'aide d'équations différentielles). On peut alors se demander si ces relations fonctionnelles ne sont pas elles aussi des relations au sens de liens sémantiques. La question est difficile car les relations fonctionnelles entre variables modélisent des processus qui eux sont typiquement des aspects comportementaux et dynamiques des systèmes.
La deuxième forme a pour objectif tout simplement de décrire les entités et leurs relations. Nous allons donc nous intéresser à cette deuxième forme. Les formalismes dédiés uniquement à la spécification de la structure sont nombreux et souvent très spécialisés au domaine d'applications. Si vous êtes électronicien, vous allez utiliser des formalismes à base de composants logiques, par exemple, pour décrire vos circuits numériques. Les formalismes généraux de cette famille sont rares. Pour notre part, nous allons focaliser sur la modélisation à base objets et d'agents.

Modélisation par objets et par agents

La modélisation objets a fait d'énormes efforts pour normaliser son vocabulaire, ses concepts et ses formalismes. Nous disposons en effet du langage UML (*Unified Modeling Language* – [BOO97][JAC97][RUM97]) qui est aujourd'hui devenu une norme dans la spécification objets. UML est un langage graphique qui nous propose plusieurs types de diagrammes. On peut classifier ces diagrammes en trois catégories : les diagrammes statiques pour la structure, les diagrammes dynamiques et les diagrammes de déploiement. Ces derniers n'ont pas d'intérêts pour les objectifs que l'on s'est fixé. Les diagrammes statiques ou diagrammes de classes et d'objets représentent la structure statique du système. Ils permettent d'identifier les entités et les relations entre les entités. Deux points de vue sont disponibles : le point de vue "objet" où ce sont les entités du système qui sont mises en relation dans une situation donnée et le point de vue "classe" où ce sont les classes d'objets qui sont mises en relation dans un cadre plus général. Les objets sont des réalisations particulières de classes. Une classe est une abstraction d'un ensemble d'objets qui possèdent des caractéristiques identiques en terme d'informations et de comportement.
Le point de vue de la modélisation objets et d'UML est conforme aux définitions proposées précédemment. Les entités sont décrites à l'aide des attributs qui les caractérisent et les relations peuvent être typés afin de respecter les différents types de relations mis en évidence précédemment. L'avantage de ce type de modélisation est qu'il est applicable quel que soit le domaine d'applications. On peut spécifier la structure d'un circuit numérique sans problème.
Le second avantage réside dans le fait qu'il est possible d'avoir plusieurs niveaux d'abstrations : le niveau objet, le niveau classe mais aussi des niveaux meta où les classes représentent des schémas de classes.

Depuis les années 80, le concept objet a évolué vers la notion d'agents (voir une synthèse dans [FER95] et [GRI99]). Un agent est un objet autonome. Contrairement aux objets, un agent possède un comportement autonome. Il est capable de prendre des décisions et d'établir des plans d'actions pour accomplir des activités complexes. Tous les agents n'ont pas ce degré d'autonomie. Il existe un gradiant entre les agents réactifs et les agents cognitifs. Les agents réactifs sont seulement des agents capables de réagir à des stimuli externes en déclenchant des actions. L'approche agents est, comme pour l'approche objets, à la fois un paradigme et un formalisme.

Cette affirmation est particulièrement vraie pour l'approche objets comme nous venons de le voir mais reste partiellement acceptable pour l'approche agents. Il existe des extensions d'UML comme A-UML pour la spécification de systèmes multi-agents mais on s'aperçoit que les choses ne sont pas si simples. La notion d'agents est complexe à cerner dans sa globalité. Avec l'avénement des approches centrées individus et multi-agents dans une multitude de domaines, il est devenu urgent de faire le point sur les concepts. Le groupe MIMOSA s'est attribué cette lourde tâche dans le domaine des systèmes spatio-temporels.

1.2.2 Formalismes pour la modélisation dynamique

La modélisation dynamique a pour objectif de décrire le comportement et les interactions des entités du système ou les contraintes sur les variables caractéristiques du système. Ces deux points de vue s'inscrivent dans deux logiques complémentaires : la description des processus ou la description des lois. Cette partition des approches dessine deux familles de formalismes : les formalismes apparentés aux équations différentielles et les autres. Nous allons faire un petit d'horizon de tous ces formalismes.

Les systèmes d'équations différentielles

Les équations différentielles sont l'outil privilégié des scientifiques. Deux raisons à cet état de fait :

- la puissance d'expression des mathématiques et des outils formels qui les accompagnent ; à partir d'un système d'équations différentielles, on peut, si ce dernier possède de bonnes propriétés, déduire des propriétés de convergence ou de stabilité à l'infini sous certaines conditions ; on peut manipuler le système avec tous ces paramètres et déduire des propriétés en fonction des paramètres
- le niveau de connaissances des outils mathématiques par les scientifiques ; tout le monde sait ce qu'est une équation différentielle ou une équation aux dérivées partielles ; en revanche, tout le monde ne sait pas obligatoirement ce qu'il peut en faire

Comme outil de modélisation, les Mathématiques permettent de mettre en équations les grandes lois de la Nature (équations à équilibre, équations de bilan, ...) et d'en déduire la dynamique temporelle. Il est, en effet, assez facile si les équations ont de bonnes propriétés de déduire l'évolution temporelle des variables composants le système.

Finalement, quels reproches peut-on faire aux Mathématiques ? Le premier est qu'il n'existe pas toujours le bon théorème ou le bon outil pour le système que l'on a construit. Dans ce cas, il n'est plus possible d'en déduire quoi que ce soit sans passer par des outils d'analyse numérique qui procédent par simulation numérique. On rejoint alors les méthodes par simulation. Cet inconvénient n'ote pas le moins du monde les capacités de formalisation des Mathématiques.
Le deuxième reproche concerne la représentation d'entités discrètes. Il est très difficile voire impossible dans l'état actuel des outils mathématiques connus par le commun des mortels de représenter des entités discrètes et de répresenter les processus les mettant en jeu. Les Mathématiques sont dans la grande majorité des cas utilisées pour représenter des lois ou des contraintes entre variables du système. On ne peut pas faire de description directe des processus. Le modélisateur descrit des équations de bilan, par exemple. Ces bilans sont les conséquences de processus qui ne sont pas représentés. Naturellement, cette approche est aussi intéressante. Il n'est pas toujours possible de décrire les processus car ils ne sont pas tout simplement connu. Les Mathématiques accompagne souvent une approche phénoménologique où l'on s'attache à décrire les

conséquences des phénomènes.

Les automates à états finis et à événements finis

Comme nous venons de le dire, les équations différentielles sont très bien adaptées à la description des conséquences de processus. Or, si on s'inscrit dans le "courant de pensées" de Zeigler et de la spécification formelle de systèmes dynamiques, les notions d'états et de fonctions de transitions et de sortie doivent faire parti du formalisme. Les équations différentielles intégrent la notion d'états mais n'explicitent pas les notions de transitions. De plus, l'approche "système dynamique" de Zeigler se focalise sur la description des processus. Les bilans émergent de la dynamique du système. Le prérequis d'une telle approche est que l'on doit possèder la connaissance des processus mis en jeu pour répondre aux problèmes posés.

Les automates à états finis constituent l'outil le plus simple mis à la disposition du modélisateur. Les automates permettent de décrire la dynamique à l'aide des notions d'état et de transition. La structure d'un automate est proche de la structure d'un système dynamique (voir 1.1) :

$$A =< X, S, \Delta >$$

où X et S sont respectivement les entrées du modèle et les états du système. La fonction Δ descrit la fonction de transition en indiquant l'état suivant en fonction de l'état courant et de l'entrée du système. Les entrées sont des symboles mais il est facile de passer de la notion de symbole à celle d'événement et on a alors à notre disposition un outil de spécification à événements discrets. On peut représenter graphiquement un automate à états finis à l'aide d'un graphe où les sommets représentent les états et les arcs les transitions. Les automates à événements finis constituent un point de vue complémentaire où les sommets représentent les événements et les arcs la succession possible des événements. Les arcs portent en général une durée. Cette durée représente le temps qui s'est écoulé entre les deux événements.

Les automates à états finis ou à événements finis sont des outils simplifiés d'outils tels que DEVS que l'on verra par la suite. Ils permettent de spécifier formellement la dynamique du système en prenant en compte ou non explicitement les aspects temporels. La description de la dynamique peut faire partie ou non d'une description structurelle c'est à dire que la dynamique modélisée peut être celle d'une entité ou non. On peut en effet décrire les états du système global sans pour autant décrire le système en terme d'entités.

Il existe de multiple variantes dont les automates non déterministes où les transitions sont exprimées à l'aide de probabilités. Dans ce cas, les changements d'états suivent un modèle stochastique. Cette variante est utilisée dès lors que l'on ne connaît pas les processus sous-jacents aux changements d'états.

Les diagrammes dynamiques d'UML : les *state charts*

La modélisation par objets et UML proposent une extension objets des automates à états finis : les *state charts*. Comme pour la majorité des diagrammes d'UML, il est possible d'utiliser le formalisme selon différents niveaux de détails. Si on le désire, on peut utiliser les *state charts* comme un automate à états finis. Dans sa version complète, le formalisme s'intégre dans une démarche de modélisation objets. Pour prendre un exemple, l'expression des transitions peut se

construire autour des attributs de la classe des entités dont on modélise la dynamique.

Le formalisme des *state charts* est une combinaison des formalismes à base d'états et de transitions. On retrouve :

- des transitions déclenchées sur évenements : si le modèle reçoit un événement (un message dans le vocabulaire objets) provenant d'un autre *state chart* alors la transition est franchie
- des transitions temporisées : la transition est franchie au bout d'un certain temps
- des transitions conditionnelles : si la condition exprimée en fonction du vecteur d'états est vraie alors la transition est franchie

L'ensemble de ces types de transitions sont présentes dans DEVS. Les *state charts* proposent aussi des formalisations de la décomposition d'états en sous-états. Cette idée est défendue dans les travaux de Fishwick. Naturellement, ici la décomposition d'un état donne naissance à un *state chart*. Il n'y a pas d'idée de multi-formalismes.

Les *state charts* proposent une syntaxe relativement riche pour la modélisation de la dynamique d'un système mais la formalisation proposée n'est pas intégrée à un cadre formel de spécifications de systèmes dynamiques. Néanmoins, on peut montrer que si on prend quelques précautions, les *state charts* peuvent offrir un cadre formel. En effet, il est possible de définir une spécification DEVS d'une partie des *state charts* [BOR03]. Cela démontre que les *state charts* sont un candidat précieux et surtout que ces capacités d'expression sont parfaitement intégrées à une démarche de modélisation objets du système.

1.2.3 Formalismes hybrides ou mixtes

La dernière famille de formalismes est constituée de formalismes où la séparation entre structure et comportement est difficile. Nous traiterons que deux exemples : les réseaux de Petri et les automates cellulaires.

Les Réseaux de Petri

Les réseaux de Petri [PER77] sont en quelque sorte une extension des automates à états finis. On y retrouve les notions d'état, rôle joué par le marquage, et de transition. Si les réseaux de Petri sont utilisés dans leur forme primaire, il ressemble comme deux gouttes d'eau aux automates à états finis. La seule différence réside dans le fait que le franchissement des transitions est fonction du marquage (l'état du modèle) et que la description de ces franchissements se fait à l'aide du réseau d'interconnexions des places. La logique de changement d'états est alors guidé par l'algorithme de franchissement des transitions. Les réseaux de Petri nous offre donc juste une autre manière d'exprimer les changements d'états.
Pourquoi l'avoir classé parmi les formalismes hybrides ? Tout simplement du fait que les modélisateurs utilisant les réseaux de Petri affectent une sémantique aux places du réseau. La sémantique peut être liée à l'état d'une entité du système. Si on adopte les réseaux de Petri colorés alors l'état du système devient beaucoup complexe par le fait que les jetons transportent des informations. Ces informations vont conditionner les transitions. Beaucoup de modélisateurs assimilent les jetons à des entités circulant dans le système.

Nous ne développerons pas plus en avant l'argumentaire car nous y reviendrons dans le chapitre 2.

Les automates cellulaires

Le formalisme des automates cellulaires est un exemple à part. Il descrit des systèmes où l'espace est une composante importante. La notion d'espace est prise ici au sens large : espace de variations d'une variable ou espace topologique où des entités sont localisées. Ce formalisme est hybride puisque d'un côté, on décrit la structure spatiale du système à l'aide d'une grille multidimensionnelle de cellules inteconnectées et d'un autre côté, on décrit le comportement local de chaque cellule de l'automate en fonction de son voisinage.

Les automates cellulaires sont un des formalismes les plus utilisés en modélisation d'écosystèmes. Néanmoins, ce formalisme part du postulat que l'espace est discrétisé et non continu, contrairement aux équations aux dérivés partielles. Ces dernières sont des équations différentielles où les processus sont explicités en fonction du temps et de l'espace, et l'espace est considéré continu.

L'espace est une composante présentée dans les modèles de systèmes spatio-temporels. Les automates cellulaires et les équations aux dérivés partielles ne sont pas les seules réponses mais restent les plus couramment utilisées et surtout ces formalismes décrivent explicitement l'espace. On peut construire un modèle de type objet où les entités "embarquent" la description de l'espace. Il est très facile d'introduire dans les caractéristiques de l'objet ses coordonnées. La définition de l'espace ne fait pas alors l'objet d'une description propre. Néanmoins, la dynamique des objets ou des agents tient compte de certains aspects de l'espace. Si l'espace est fermé ou torique, lors du déplacement d'une entité dans l'espace, le processus de déplacement doit garantir le respect de la cohérence des coordonnées.

1.2.4 Une classification des formalismes

Nous venons de présenter quelques exemples de formalismes que l'on peut rencontrer en modélisation de systèmes complexes. Chaque formalisme est spécialisé dans la spécification d'un aspect du système : la description de la structure ou du comportement, la description de l'espace, ... On peut donc établir une classification des formalismes en fonction des aspects à modéliser et des propriétés du système ou d'une partie du système. Cette classification a un seul objectif : montrer qu'il est fondamental de bien choisir son formalisme en fonction de son système. Cette démarche s'inscrit dans un approche par décomposition de son système en sous-systèmes et il est important d'adopter le formalisme le plus adéquat pour la représentation de chaque sous-système.
Cette classification a aussi pour objectif de montrer que certains aspects ne sont pris en compte de la même façon selon les formalismes. Cette différence nous conduira à se poser certaines questions dans le cadre de la multi-modélisation. Un modèle qui manipule des changements d'états continu et un autre de manière discret sont-ils couplages ? Si oui, quelles sont les précautions à prendre ?

Afin d'établir une classification, il est nécessaire de déterminer les propriétés discriminantes des systèmes ou de leur modélisation :

- discret ou continu : cette propriété est à considérer selon plusieurs points de vue (changement d'états, espace, temps, ...)
- les processus modélisés sont-ils déterministes ou stochastiques ?
- l'espace est-il à prendre en compte ?

– le temps a-t-il une importance ?

La première propriété est probablement la plus importante. Pour les systèmes spatio-temporels où l'espace et temps font parti des modèles, le modélisateur doit faire un choix de représentation de l'espace et du temps. Il est bien évident que définir le temps comme une variable discrète ou continue n'implique pas les mêmes outils de modélisation. Typiquement, les automates cellulaires manipulent un temps discret (à chaque pas de temps, l'état de l'automate est calculé en fonction de l'automate à l'instant précédent) et les équations différentielles ordinaires manipulent un temps continu. Par exemple, la température évolue de manière continue dans le temps alors que le nombre de produits en fin de chaîne de production évolue de manière discrète. Les modèles construits autour de ces variables ne seront pas de la même nature et ne mettront pas en jeu les mêmes formalismes.
Comme le propose les automates à états finis, il est possible de spécifier les transitions d'états soit de manière déterministe soit de manière stochastique. Ce choix n'est pas obligatoirement guidé par une propriété de stochasticité du système mais par l'approche adoptée pour la modélisation des processus. Si le niveau de connaissances sur le système n'est pas suffisant, les probabilités modélisent la proportion d'apparition de l'état suivant en fonction de l'état courant. Cette probabilité est déterminée par l'observation du phénomène.

Nous allons donc maintenant dresser une classification des formalismes dédiés à la spécification de la dynamique des systèmes en fonction de trois critères : la manipulation de variables continues ou discrètes, la représentation de l'espace continu ou discret et la présence du temps continu ou discret.

La figure 1.3 nous montrent de manière évidente la multitude des formalismes et l'adéquation aux hypothèses retenues pour le modèle. Dans le cadre de la multi-modélisation, cette diversité de formalismes va poser la question du couplage de modèles : comment coupler deux modèles exprimés dans des formalismes différents ? La présentation de DEVS va nous apporter un éclairage unificateur des formalismes et nous allons montrer comment s'abstraire des formalismes.

1.3 DEVS : intégration de modèles

Comme nous l'avons vu précédemment, on peut définir cinq niveaux de spécifications formelles d'un système dynamique [ZEI73]. Il est bien évidemment que pour répondre au problème d'intégration de modèles ou de couplage de modèles, il faut répondre avant tout à la question : quel est le niveau le plus adéquat pour l'intégration de modèles ? La question sous-jacente est : quel est le niveau d'intégration que l'on désire ? Si le degré d'intégration est faible, on peut se contenter de spécifier tout modèle avec un niveau 0 (la boîte noire). Mais quel est alors le potentiel d'une telle intégration ?

Depuis les années 1970, des travaux formels ont été menés pour développer les fondements théoriques de la modélisation et la simulation des systèmes dynamiques à événements discrets [ZEI73]. B. Zeigler s'attache alors à travailler au niveau systèmes dynamiques et systèmes couplés. DEVS (*Discrete EVent system Specification*) a été introduit comme un formalisme abstrait pour la modélisation à événements discrets et est un formalisme universel. DEVS s'abstrait totalement de la mise en oeuvre des simulateurs mettant en oeuvre la modélisation du système. Néanmoins, DEVS de par sa nature opérationnelle, propose des algorithmes pour le comportement dynamique des modèles. Cet aspect ne sera pas traité dans ce chapitre (voir 2.1.2).

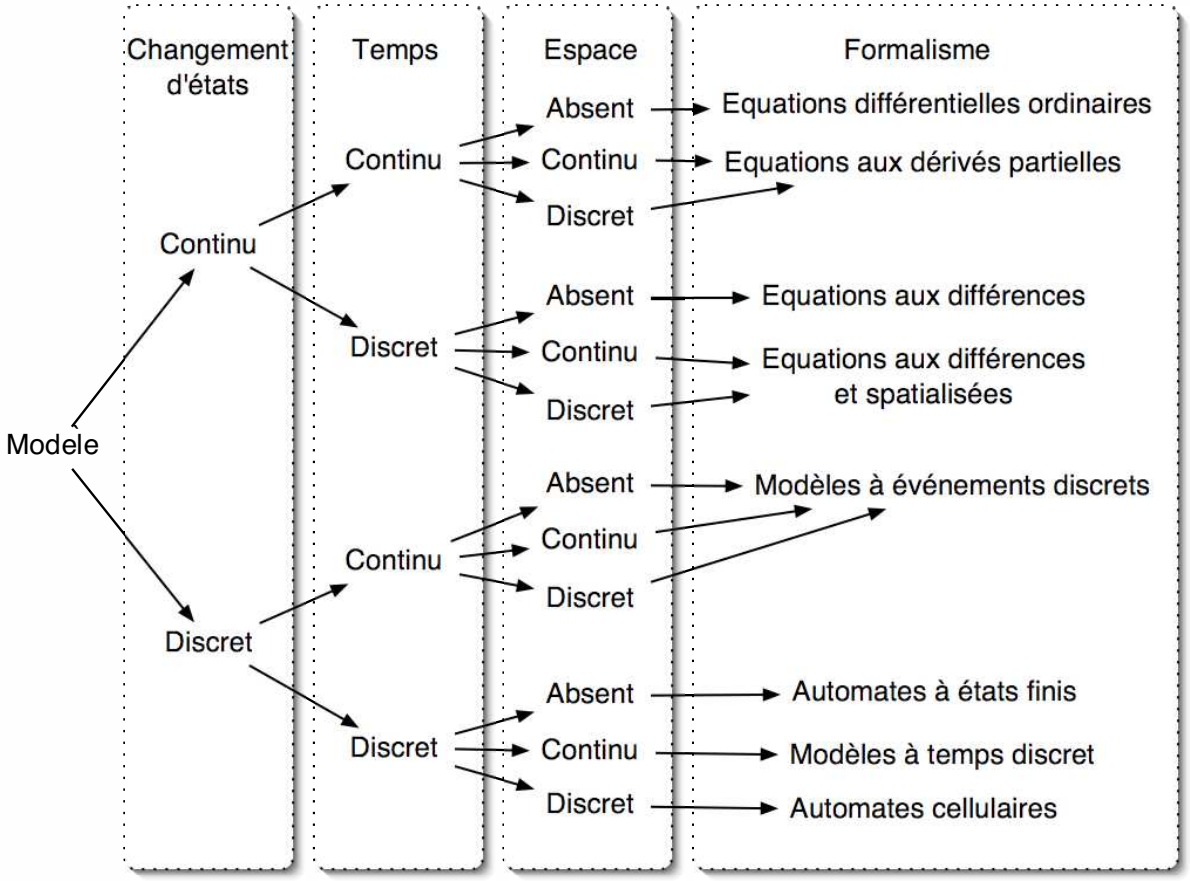


FIG. 1.3 – Classification des formalismes selon l'aspect continu ou discret des variables, du temps et de l'espace

1.3.1 DEVS atomique

Un modèle DEVS atomique possède la structure suivante : DEVS = $(X,Y,S,\delta_{ext},\delta_{int},\delta_{con},\lambda,t_a)$ où

X est l'ensemble des ports et des valeurs d'entrée,

Y l'ensemble des ports et des valeurs de sortie,

S l'ensemble des états du système,

δ_{ext} la fonction de transition externe ,

δ_{int} la fonction de transition interne ,

δ_{con} la fonction de transition "conflit" ,

λ la fonction de sortie,

t_a la fonction d'avancement du temps.

Cette structure est très propre de celle proposée au niveau systèmes dynamiques. La seule différence réside dans la séparation de la fonction de transition en trois fonctions de transition. Un modèle DEVS peut être représenté graphiquement par une boîte avec un ensemble d'entrées

appelées ports d'entrée et de sorties appelées ports de sortie. Les vecteurs d'entrée et sortie sont l'union de tous les ports du modèle.

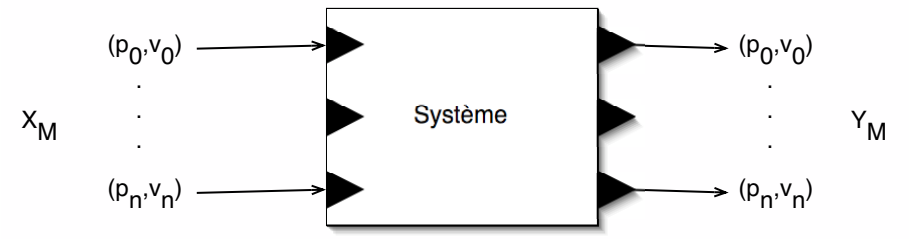


FIG. 1.4 – Représentation graphique d'un modèle atomique

v_i est la valeur prise par un port d'entrée ou de sortie. Cette valeur appartient à l'ensemble des valeurs possibles du port p_i. Un port d'entrée prend une valeur lors de l'émission d'un événement attaché à ce port. Un port de sortie prend une valeur lorsque la fonction de sortie prend une valeur pour ce port.

S est l'ensemble des états du système. L'ensemble des valeurs pris par le vecteur d'états à un instant donné e est appelé état du système. L'ensemble S_T des états totaux du système est :

$$S_T = (s, e)s \in S, o < e < ta(s)$$

e représente le temps écoulé dans l'état s. Ce concept d'état total (s, e) est fondamental car il permet de spécifier un état futur en fonction du temps écoulé dans l'état présent.

$t_a(s_k)$ est le temps pendant lequel le modèle restera dans l'état s_k, si aucun événement externe ne survient. Un état s de S est dit passif si et seulement si sa durée de vie est infinie. Dans ce cas, évidemment, la fonction de transition interne n'est pas définie.

La fonction classique de transition est décomposée en deux fonctions, représentant respectivement les évolutions autonomes et celles dues à des événements externes. La fonction de transition interne, partie autonome de la fonction de transition, est définie par :

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

Elle spécifie les états futurs des états actifs, états dont la durée de vie n'est pas infinie. Ces états correspondent à des régimes transitoires ou des états instables du système. La fonction de transition externe est :

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

La fonction de transition externe δ_{ext} représente la réponse du système aux événements d'entrée. Le système est dans un état (s, e) à un instant t. Un événement externe survient alors la fonction δ_{ext} indique quel est le nouvel état du système en fonction de s.

Remarque : 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 "propre" des évolutions autonomes du modèle, évolutions autonomes représentant des régimes transitoires ou

des états instables du système réel.

La spécification des changements d'états est complétée par la fonction de transition 'conflits' δ_{con} qui permet de spécifier l'état futur dans le cas d'événements simultanés dans le monde du modèle. Cette fonction est présente seulement dans une variante de DEVS qui se nomme *parallelDEVS* [ZEI99].

$$\delta_{con} : S \times X \rightarrow S$$

La fonction de sortie est une application de l'ensemble des états S dans l'ensemble des sorties Y.

$$\lambda : S \rightarrow Y$$

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.

Un système peut être autonome et donc ne recevoir aucun événement 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'états dû exclusivement à l'état interne du système et au temps.
Le système est entré à l'instant t dans l'état s_t. Si aucun événement extérieur survient alors le système changera d'état à $t + ta(s_t)$. La fonction ta donne la durée pendant laquelle le système sera dans un certain état. Si cette durée est nulle alors on l'appellera état transitoire. A l'inverse, si la durée est infinie alors on l'appellera état passif. Un événement extérieur peut survenir exactement à $t + ta(s_t)$. Le nouvel état est alors définie par la fonction de transition δ_{con}. Cette fonction règle les problèmes de conflit entre les transitions dues aux événements extérieurs et les transitions internes.

Illustrons l'évolution d'un modèle DEVS sur un exemple.

A l'état initial, le système est dans l'état s_0 à $t = 0$ (voir figure 1.5). La fonction ta nous indique que pour l'état s_0, le système changera d'état à $t + ta(s_0)$ si aucun événement extérieur ne survient. A $t_1 = t + ta(s_0)$, aucune entrée n'a eu lieu. La fonction de sortie est donc activée et Y prend pour valeur la valeur produite par la fonction λ pour l'état s_0. 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 $t_1 + ta(s_1)$. A l'instant t_2 qui est inférieur à $t_1 + ta(s_1)$, un événement extérieur est placé en entrée. On fait alors appel à la fonction de transition externe pour déterminer le nouvel état. Dans ce cas, la fonction de sortie n'est pas appliquée. Elle est appliquée exclusivement lors de transition interne. A l'instant t_2, le système passe dans l'état $s_3 = \delta_{ext}(s_1)$. Si aucun événement externe n'avait eu lieu, le système serait à $t_1 + ta(s_1)$ dans l'état $s_2 = \delta_{int}(s_1)$. Le système évolue ensuite jusqu'à l'instant t_4 par transition interne. A l'instant t_4, la fonction de transition interne doit être activée mais au même moment un événement extérieur survient. La fonction δ_{con} règle le conflit en indiquant le nouvel état. Ce nouvel état est fonction de l'état courant et de l'événement d'entrée. La fonction de sortie ne prend pas de valeur.

1.3.2 DEVS couplé

Un modèle couplé définit comment est couplé un ensemble de modèles entre eux pour former un nouveau modèle. Il peut lui-même faire parti d'un modèle couplé. On définit alors une

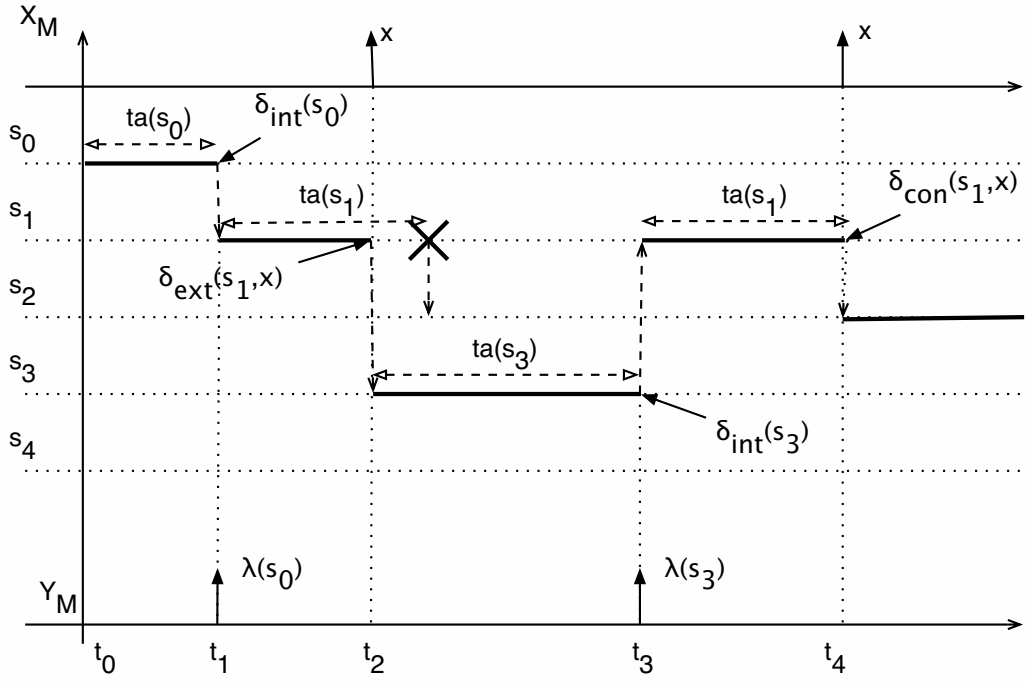


FIG. 1.5 – Exemple de graphe de transitions

construction hiérarchique de modèles. Un modèle couplé comprend les informations suivantes :

- l'ensemble des modèles qui le compose,
- l'ensemble des ports d'entrée qui recevront les événements externes,
- l'ensemble des ports de sortie qui émettront les événements,
- les couplages en ports d'entrée et ports de sortie des modèles composant le modèle couplés.

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 de X_M et Y_M 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 identifiants des modèles intervenants dans le modèle couplé. M_d est un modèle DEVS. Les variables représentant les entrées et les sorties du modèle seront indexées par l'identifiant du modèle. D'où la notation suivante :

$$M_d = (X_d, Y_d, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, t_a)$$

Les entrées et les sorties du modèle couplé noté N sont connectées aux entrées et sorties des modèles composants le modèle couplé. EIC représente la liste de ports d'entrée du modèle couplé $IPorts$ qui sont connectés aux ports d'entrée des sous-modèles $IPorts_d$ avec $d \in D$.

$$EIC = \{((N, a), (d, b)) / a \in IPorts, b \in IPorts_d\}$$

On a la même situation pour les ports de sortie où EOC représente la liste de ports de sortie du modèle couplé $OPorts$ qui sont connectés aux ports de sortie des sous-modèles $OPorts_d$ avec

$d \in D$.

$$EOC = \{((N, a), (d, b))/a \in OPorts, b \in OPorts_d\}$$

A l'intérieur du modèle couplé, les sorties d'un modèle peuvent être couplées aux entrées des autres modèles. Une sortie d'un modèle ne peut pas être couplée à l'une de ses entrées.

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

Prenons un exemple pour illustrer le modèle formel.

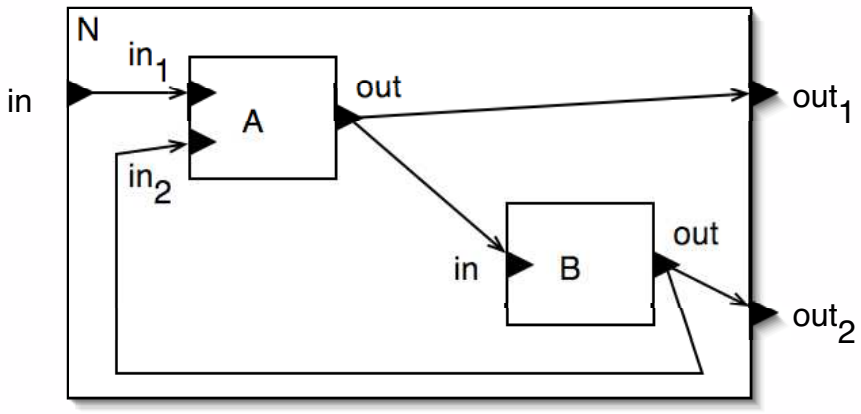


FIG. 1.6 – Représentation graphique d'un modèle couplé

Soit,

$IPortsN = \{in\} et OPortsN = \{out_1, out_2\}$

$D = \{A, B\}$

$EIC = \{((N, in), (A, in_1))\}$

$EOC = \{((A, out), (N, out_1)), ((B, out), (N, out_2))\}$

$IC = \{((A, out), (B, in)), ((B, out), (A, in_2))\}$

Si le modèle couplé fait parti lui-même d'un modèle couplé alors il faut définir les paramètres standards d'un modèle DEVS basic.

Le vecteur d'états du modèle couplé est le n-uple composé des vecteurs d'états des modèles composants le modèle couplé.

$$\underline{S_N} = \{\underline{S_d}/d \in D\}$$

L'ensemble des valeurs de sortie du modèle couplé est définie par :

$$Y = \{(p, v)/p \in OPorts, v \in Y_p\}$$

On peut les exprimer en fonction des valeurs de sortie des modèles composants le modèle couplé en précisant Y_p. Y_p est l'ensemble des valeurs prises par le port de sortie p du modèle couplé.

$$Y_p = \{v/(p', v) \in Y_p, p' \in OPorts_d, d \in D, ((d, p'), (N, p)) \in EOC\}$$

Les valeurs d'un port de sortie sont les valeurs du port qui lui est connecté soit un port de sortie d'un modèle le composant.

$$\delta_{ext_S}(\underline{S_N}, (p, v)) = \{S_{d'}/d' \in D, d \neq d'\} \cup \delta_{ext_d}(\underline{S_d}, (p', v))$$

tel que

$$((N, p), (d, p')) \in EIC$$

1.3.3 DEVS et les agents

Le formalisme DEVS s'intégre parfaitement dans une démarche systémique de modélisation et est basé principalement sur la notion d'événements discrets. D'autre part, il existe les approches agents ou centrées individus qui repose d'une part sur la définition des entités en interaction et d'autre part sur les interactions. La modélisation et la simulation à base d'agents ou d'individus est une avancée majeure dans le domaine des écosystèmes ([BOU94], [HIL96] et [COQ97]) du fait de recentrer l'objet de la modélisation sur les entités du système et leurs interactions. Cette approche offre l'avantage d'être plus pragmatique et plus proche de la réalité du monde que l'on désire modéliser.
Dans notre cas, les écosystèmes marins et en particulier les interactions de type proie-prédateur d'un groupe majeur du zooplancton (les copépodes) sont notre objet d'étude [RAM98]. Cette étude nous a conduit à construire des modèles d'agents réactifs [FER95] dans un environnement 2D discrétisé [RAM98] et dans un environnement 3D continu [DUB01]. Il existe une autre approche possible : la modélisation à événements discrets et à temps continu. Considérer le temps comme continu implique de déterminer la date de changement d'états et d'émission des événements à partir de l'état courant du système ou de l'agent. Les avantages sont multiples : on considère seulement les événements conduisant à des changements d'états et leurs dates d'occurrence. Le temps est continu et l'avancement du temps est non linéaire. On parle ici de "temps construit" par simulation, le temps de simulation est proportionnel au nombre d'évènements et non plus à la durée simulée. De plus, on peut envisager sérieusement de spécifier dans la globalité du système à l'aide de DEVS. Il est important de noter que ce passage aux événements discrets n'est pas toujours simple et qu'il est souvent l'objet d'une refonte des modèles dans le cas d'une spécification complète du système. Nous reviendrons sur ce point dans la partie modélisation d'un écosystème marin du chapitre 3.

1.3.3.1 Une proposition d'un *agentDEVS*

Comme nous venons de le voir, des travaux formels ont été menés [ZEI73], [ZEI95], [ZEI00] qui dérivent des mathématiques des systèmes pour la mise au point du formalisme DEVS (Discrete EVent system Specification) pour la spécification de modèles à évènements discrets et à temps continu. Ces travaux ont aussi conduit à la définition de nombreux formalismes sur la base de DEVS (GDEVS [GIA00], DEV&DESS [ZEI00], Cell-DEVS [WAI01], DS-DEVS [BAR95], ...) ayant pour objectif l'intégration de différents paradigmes de modélisation. Ces différents formalismes englobent notamment les équations différentielles, les systèmes à temps discret, les automates cellulaires, etc... On peut donc ligitime de se poser la question suivante : n'est-il pas possible de définir un agent-DEVS ? Il existe déjà des réponses mais qui reste des réponses partielles ou pour des agents cognitifs (comme DEVS-RAP [HOC02]). Ce constat est bien naturel puisqu'au regard de la diversité des applications utilisant les systèmes multi-agents, 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. Nous partons de la définition générale d'un agent donnée par G. Weiss [WEI99] : "*Agents are autonomous, computational entities that can be viewed as perceiving their environment trough sensors and acting upon their environment through effectors*". Les aspects importants sont ici les notions d'environnement et d'interactions. Pour complèter la définition précédente M. N. Huhngs et M. P. Singh [HUH98] donnent les caractéristiques majeures suivantes pour les systèmes multi-agents :

- chaque agent possède une information incomplète, il est limité dans ses capacités d'action ;
- le contrôle du système est distribué ;
- les données sont décentralisées ;
- les calculs sont asynchrones.

De plus, les deux types d'interactions majeures sont [FER95] :

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

Pour notre part, nous nous intéressons aux systèmes d'agents réactifs situés (ou encore désignés sous le terme de modèle centrés individus). Un système d'agents réactifs situés est un système dynamique où il possible d'ordonner les actions ou activités des agents dans le temps. Pour cette raison, il est possible d'utiliser le formalisme DEVS pour la spécification des systèmes d'agents réactifs. Néanmoins, il ne faut pas perdre de vue le terme *situé* qui est synonyme de représentation de l'espace. Or l'espace ne fait pas parti des fondements de DEVS. D'autre part, le point de vue de DEVS est un point de vue modulaire et hiérarchique ce qui, en revanche, est plutôt en adéquation avec l'approche agent.
À partir des définitions que nous venons de donner et des remarques que nous venons de soulever, nous élaborons un tableau d'équivalence entre le paradigme d'agents réactifs et le formalisme DEVS (voir tableau 1.1). Nous différencions explicitement la structure et le comportement des systèmes d'agents réactifs. Cette approche est celle généralement utilisée pour la formalisation des SMAs.

Dans le tableau 1.1, nous avons exprimé l'architecture du SMA à l'aide de structure DEVS. Le comportement est formalisé par les fonctions de transitions. Ainsi, DEVS fournit une écriture unifiée à la fois de la structure et du comportement.

Le tableau 1.1 traite seulement des agents réactifs. Rien n'est dit sur l'environnement. Il est pourtant fondamental de pouvoir formaliser l'environnement dans un SMA. Nous considérons l'environnement comme non proactif (il n'a pas de comportement individuel). Néanmoins, il doit 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 (comme les phéromones des classiques systèmes à insectes sociaux, de type fourmis, par exemple). Une telle structure peut être formalisée par un modèle DEVS comme 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.
- $S = \{(Idle \cup S_i, E)/i = 1...n\}$ est l'ensemble des états, où $Idle$ est l'état passif et n le nombre de ports d'entrée. Chaque port d'entrée correspond à une question possible pour

	Agent réactifs	***Spécification DEVS***
Structure	Système d'agents réactifs dans son ensemble	Modèle DEVS couplé
	Structure de l'agent	Modèle DEVS atomique ou couplé
	Structure des communications et interactions	Connexions d'entrées-sorties externes
	Récepteurs	Ports d'entrées
	Éffecteurs	Ports de sorties
	Ensemble des états	Ensemble des états
	Supports de communication	Événements
Comportement	Processus internes	Fonctions de transitions internes et fonctions d'avancement du temps
	Réponses aux stimuli	Fonctions de transitions externes
	Émissions de stimuli	Fonctions de sortie

Tab. 1.1 – Passage du paradigme d'agents réactifs vers le formalisme DEVS

l'environnement. E représente l'ensemble des entités passives du système.
- $\delta_{ext} : Idle \times X_i \rightarrow S_i$ correspondent à la réception de la question pour l'environnement. au choix de la réponse par l'environnement.
- $\lambda(S_i) : S_i \rightarrow Y_i$ sont les réponses de l'environnement aux requêtes externes.
- $\delta_{int} : S_i \rightarrow Idle$ est le retour à l'état passif après réponse à une question.
- $ta(Idle) = \infty$, l'environnement est toujours en attente d'une question.
- $ta(S_i) = 0$, les réponses aux questions sont instantanées.

Du fait que $ta(S_i) = 0$, il n'y a qu'une fonction de transition externe appliquée à l'état passif $Idle$. Ceci permet la formalisation d'interactions directes entre les agents et leur environnement. En effet, DEVS ne permet pas l'écriture de messages asynchrones bloquants comme dans un diagramme de séquences UML par exemple. Nous devons le rendre explicite par l'introduction d'un état passif "$Idle$", toujours en attente d'un événement externe, et de $ta(S_i) = 0$.
Le choix de ce que représente S dépend naturellement du système modélisé. Dans le cas d'agents situés, S peut contenir la liste des objets de l'environnement avec leur position. S peut également contenir les limites de l'environnement et, dans le cas où les agents interagissent, la position de tous les agents. Ainsi, c'est l'environnement qui détermine s'il y a interaction ou non. Dans la plupart des SMAs situés, l'environnement est modélisé sous la forme d'une grille qui discrétise l'espace dans lequel les agents évoluent. Cette représentation peut faire l'objet d'une spécification DEVS propre (voir 1.3.3.2) et donc faire partie de S.

On a montré que le formalisme DEVS nous permet de modéliser l'environnement, la dynamique des agents et les interactions entre les agents. Un agent est vu comme un modèle DEVS atomique ou couplé (si la dynamique de l'agent fait l'objet d'une décomposition). L'environnement fédére les informations globales du système et gére les interactions entre les agents. En revanche, le formalisme DEVS nous impose une description totale et statique des modèles composant le modèle global ce qui signifie pour un système multi-agents que tous les agents (et non les classes d'agents) sont formalisés et que leur nombre ne peut évoluer dans le temps. Plus grave encore, les connexions entre les modèles ne peuvent pas évoluées dans le temps : une connexion entre deux ports est définie statiquement. Or le propre d'un système multi-agents est de voir sa structure d'interaction entre les agents changer en fonction de l'état de l'environnement et des agents. De plus, le nombre d'agents peut évoluer.

En conclusion, deux aspects des systèmes à base d'agents réactifs (l'espace et le caractère dynamique des relations entre agents) ne sont pas directement modélisable avec le formalisme DEVS. Nous allons donc montrer comment des extensions telles que Cell-DEVS et DS-DEVS ou des *mapping* DEVS[1] tels que ceux pour les réseaux de Petri ou les automates à états finis peuvent nous y aider.

1.3.3.2 Cell-DEVS

L'extension Cell-DEVS est née de la constatation suivante : de nombreux modèles font intervenir des espaces discrets et utilisent des formalismes tels que les automates cellulaires. En effet, dès lors que l'on doit représenter l'espace deux possibilités sont offertes :

[1]Le terme de *mapping* DEVS signifie que DEVS sans extension est utilisé pour spécifier un autre formalisme.

l'espace est continu ; on définit une origine et un repère par rapport auquel toute entité doit se repérer,

l'espace est discret ; on divise l'espace en régions ; dans la majorité des cas, toute entité sera localisée sur une et une seule région.

Wainer et Giambiasi dans [WAI01] developpent l'extension Cell-DEVS. Cette extension doit pouvoir décrire et simuler des modèles à base d'automates cellulaires multi-dimensionnels et à événements discrets. La dynamique des cellules est temporisée c'est à dire que l'état d'une cellule sera modifiée en fonction de l'état de son voisinage mais il sera connu des cellules voisines qu'après un certain délai. L'idée de base est de fournir un mécanisme simple de définition de la synchronisation des cellules. Comme toute proposition d'extensions, les auteurs offrent à la fois l'extension du formalisme qui se résument à l'ajout de variables supplémentaires et de leur sémantique et le simulateur abstrait. Deux spécifications sont proposées : l'une pour la dynamique des cellules et l'autre pour la dynamique de l'automate complet. Un modèle Cell-DEVS est défini comme un espace composé de cellules qui peuvent être couplées afin de former un espace complet. La sémantique liée aux cellules n'est pas précisée, seule la dynamique du modèle couplé fait l'objet d'une description détaillée. Il est donc possible d'utiliser ce formalisme pour représenter un espace réel (un lieu de déplacement pour des entités) ou un espace plus abstrait (un espace comme un ensemble de lieux abstraits). La remarque concernant le découplage entre le formalisme et la sémantique reste vraie pour maintes formalismes. Nous débattrons de cette question dans le chapitre suivant (voir 2.1.4).
Un modèle Cell-DEVS est basé sur la même structure d'un modèle DEVS classique. Nous allons retrouver les modèles atomiques pour les cellules, les éléments de base d'un réseau, et les modèles couplés pour les réseaux eux-mêmes. Néanmoins, la structure proposée par Zeigler [ZEI99] se voit augmentée d'attributs. Le modèle DEVS d'une cellule est définie par la structure suivante :

$$\text{TDC} = (X,Y,I,S,N,delay,d,\delta_{int},\delta_{ext},tau,\lambda,t_a)$$

où

X est l'ensemble des ports et des valeurs d'entrée,

Y l'ensemble des ports et des valeurs de sortie,

I l'interface de la cellule,

S l'ensemble des états de la cellule,

N l'ensemble des états des cellules voisines,

$delai$ le type de délai utilisé,

d la durée du délai,

δ_{ext} la fonction de transition externe ,

δ_{int} la fonction de transition interne ,

τ la fonction locale de calcul,

λ la fonction de sortie,

ta la fonction d'avancement du temps.

L'interface I de la cellule définit le voisinage de la cellule ainsi que les connexions en terme de ports d'entrée et de sortie entre la cellule et ses voisines. Il y a autant de ports d'entrée que de voisins. La taille du voisin est définie dans l'interface. La fonction de calcul τ modélise la fonction de calcul de l'état de la cellule en fonction de l'état du voisinage. C'est une fonction

booléenne et elle utilise I et N pour effectuer son calcul. L'état de la cellule est effective pour les cellules voisines qu'au bout du délai d'attente d. Ce délai peut être de type transport (le délai représente tout simple le temps de propagation du signal dans la cellule), de type inertiel (le délai est ici relié à l'inertie du système ; il faut un certain niveau d'énergie pour le système change d'état) ou de type signal (le délai est dû au temps de changement d'état ; le fait de passer de signal haut à signal bas, certains systèmes ont besoin d'un certain temps). Le type de délai influence la fonction de sortie. Comme nous l'allons préalablement évoqué, l'état de la cellule est calculée par la fonction τ. Cette dernière met à jour une partie de l'état S de la cellule. En effet, S se décompose en un attribut représentant l'état booléen de la cellule, un autre attribut indiquant la période d'activité et deux attributs stockant les états successifs de la cellule (voir figure 1.7).

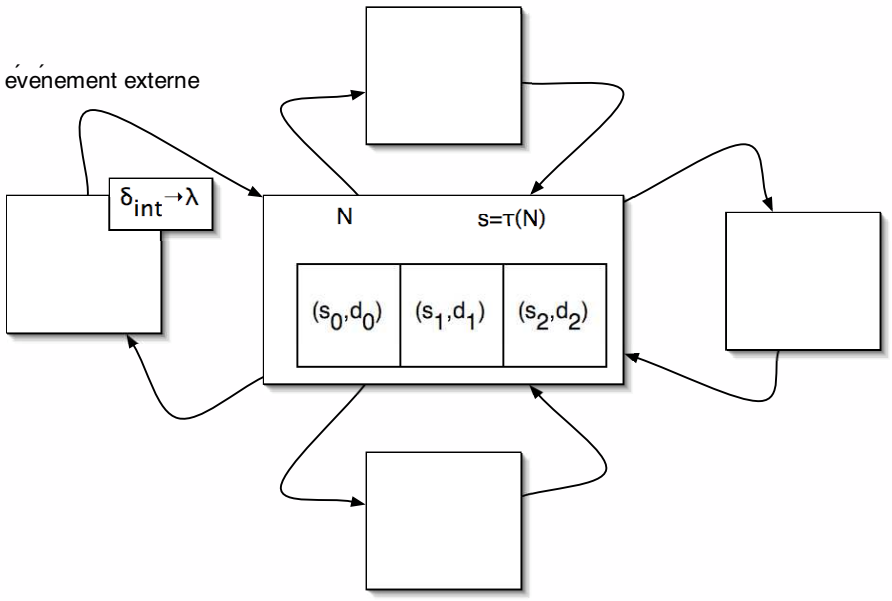


FIG. 1.7 – Une cellule du réseau avec ses connexions aux cellules voisines

La période d'activité de la cellule est proposée sous la forme d'une variable prenant la valeur *active* ou *passive*. Une cellule est dite active si la cellule doit communiqué des états aux cellules voisines. La communication des états s'effectue par l'intermédiaire des ports de sortie et d'entrée des modèles comme pour un modèle DEVS classique. En revanche, l'introduction d'un délai de transmission implique une dynamique bien particulière et nécessite l'introduction d'une file d'attente. Lorsqu'une cellule voisine informe de son changement d'état, la fonction τ est calculée et le nouvel état est alors stocké dans la variable s de S. De plus, et conformément au délai de transmission, ce nouvel état n'est pas transmis immédiatement aux voisins. Il est alors stocké dans une file d'attente avec la valeur du délai. Si rien ne se produit avant la fin du délai, la cellule informe ses voisines qu'elle a changé d'état. En revanche, si une nouveau cellule voisine informe de son changement d'état avant le délai, la cellule change d'état et stocke dans la file d'attente le nouvel état avec le délai. Dès que le premier délai est écoulé, l'état est communiqué ainsi que le second lorsque le second délai est écoulé. Cette dynamique devient complexe dès lors que les délais des différentes cellules ne sont pas identiques. La description qui vient d'être faite est dite de transport. Dans le cas de délai inertiel, l'état transmis au voisinage peut ne pas suivre l'état

de la cellule. Si la cellule revient dans le même état avant le délai inertiel alors le changement d'état n'est pas signalé.
La définition des cellules est complétée par la définition d'un modèle couplé. Le modèle d'une cellule définit l'interface qu'elle offre à l'extérieur mais ne précise pas la forme des connexions. Le rôle du modèle couplé est donc de définir les connexions entre les cellules (à l'aide d'un *pattern* - N, C et B), la taille $\{t_1, ..., t_n\}$ et le nombre de dimension n du réseau et l'interface I avec d'éventuels modèles DEVS (de type Cell-DEVS ou non).

$$GCC = (X_{list}, Y_{list}, I, X, Y, n, \{t_1, ..., t_n\}, N, C, B, Z)$$

X_{list} et Y_{list} définissent la liste des cellules du réseau possédant des ports d'entrée et des ports de sortie non connectés en interne et par conséquent disponibles pour une connexion avec un autre modèle. En terminologie DEVS, cet ensemble est l'ensemble des connexions entre les ports d'entrée et des ports de sortie avec les ports du modèle couplé. Wainer utilise un vocabulaire un peu différent. L'interface I du réseau complète la définition en réunissant au sein d'une même structure les éléments de définition de l'interface du réseau vers l'extérieur. Cette définition intègre l'ensemble Z qui met en relation les ports de sortie appartenant à Y_{list} d'un réseau et les ports d'entrée appartenant à X_{list} d'un autre réseau. X et Y représente l'ensemble des événements d'entrée et de sortie. De manière à simplifier la définition des connexions entre les cellules du réseau (communément noté connexion interne dans les modèles couplés) un *pattern* de voisinage, noté N, est défini. Ce *pattern* spécifie pour toute cellule n'appartenant pas à la bordure B la position relative de ses voisins.

Avec Cell-DEVS, nous disposons d'un outil de spécification DEVS pour les automates cellulaires. Il faut noter que Cell-DEVS ne s'arrête pas à un langage de spécification mais offert aussi les simulateus abstraits afin de préciser le comportement d'un modèle Cell-DEVS. Cette remarque est de nouveau valable comme pour la totalité des extensions de DEVS.

En conclusion, si l'une des parties de votre modèle nécessite une description de type automate cellulaire, il suffit de la spécifier en Cell-DEVS et elle aura la capacité d'être couplé à d'autres modèles compatibles DEVS. On pense naturellement à l'idée d'étendre la spécification de l'environnement dans un système multi-agents. L'environnement est alors un modèle couplé composé de deux modèles : l'un de l'espace représenté par un modèle de type Cell-DEVS et l'autre conforme à la spécification proposée précédemment. La relation entre les deux modèles est en revanche complexe. Le premier point concerne la relation de localisation dans l'espace. Si un agent, un modèle DEVS, est statique c'est à dire qu'il ne change pas sa position, la relation est modélisé par une série de connexions classiques. Ces connexions seront le reflet de l'interaction entre l'espace et l'agent. Si l'agent bouge alors les modèles en relation changent. Un modèle d'agent se connecte dynamiquement au modèle de la cellule sur laquelle il est localisé. Le changement dynamique des connexions n'est pas possible en DEVS. Nous devons faire appel à DS-DEVS (voir 1.3.3.3). La présence de cette capacité de changement est fondamentale.

1.3.3.3 DS-DEVS

L'un des reproches de DEVS mais aussi de bien d'autres formalismes est leur incapacité à changer dynamiquement de structure. Les formalismes peuvent, en général, seulement représenter les changements d'états en fonction des événements d'entrée et de la dynamique interne. Les changements de structures sont alors possibles en les intégrant dans les variables descriptives du

système. On mélange alors les aspects purement comportementaux avec des aspects de structure. Les exemples de changements de structures sont nombreux :

- un nouveau serveur qui apparaît dans un réseau complexe,
- le remplacement d'un composant défectueux par un autre,
- la transformation d'une particule en une autre particule,
- l'apparition et la disparition de produits dans un système de production,
- le changement de modèles dans une simulation en cours d'exécution,
- une entité avec un comportement dans un environnement d'entités communicantes (de type système multi-agents).

Dynamic Structure DEVS a été défini par F. J. Barros dans [BAR95] afin de palier à cette insuffisance. Ce formalisme est basé sur DEVS (comme son nom le laisse supposé) et fournit tous les mécanismes pour le changement dynamique de la structure d'un modèle DEVS. DS-DEVS introduit une nouvelle spécification pour les modèles couplés mais n'effectue aucune modification de spécification des modèles atomiques. Nous allons donc faire le tour de ces modifications afin de comprendre quels sont les apports potentiels pour la formalisation des systèmes multi-agents et centrés individus.

Le changement dynamique de structures est introduit à l'aide de la définition d'un réseau de modèles DEVS atomiques. On peut parler de modèle couplé mais la liste des connexions et la liste des modèles de ce réseau peuvent changent au cours du temps. Ce réseau est défini à l'aide de l'ensemble des modèles atomiques activables Δ et le réseau des modèles actifs χ. Ce réseau est représenté par la structure suivante :

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

Les attributs X_{Δ} et Y_{Δ} représentent l'ensemble des ports d'entrée et des ports de sortie du modèle global. Tous les ports ne sont pas obligatoirement actifs. Le modèle M_{χ} précise de quelle manière la structure du modèle actif change au cours du temps. Par conséquent, ce modèle définit les ports actifs. Le modèle M_{χ} est représenté par la structure suivante :

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

On y retrouve une structure de modèles atomiques classiques afin de respecter la spécification DEVS. Pour comprendre comment est décrit le changement dynamique de la structure, il faut détailler le vecteur d'états S_{χ}. Ce vecteur contient les informations de l'état de la structure du réseau et est défini par le n-uple suivant :

$$s_{\chi} = (X^{\chi}_{\Delta}, Y^{\chi}_{\Delta}, D^{\chi}, \{M^{\chi}_i\}, \{I^{\chi}_i\}, \{Z^{\chi}_{i,j}\}, \Xi^{\chi}, \theta^{\chi})$$

Tout changement de ce n-uple traduit un changement de la structure du réseau des modèles. Prenons un exemple. La valeur X^{χ}_{Δ} définit l'ensemble de ports d'entrée actifs. Cet ensemble est un sous-ensemble de X_{Δ} et s'il est modifié cela signifie que les événements admissibles par le modèle changent. Cette modification est, en général, la conséquence de l'activation ou de la désactivation d'un modèle composant le modèle global actif.
D^{χ} est l'ensemble des identifants des modèles actifs et $\{M^{\chi}_i\}$ l'ensemble des modèles actifs où i appartient à D^{χ}. On peut généraliser la définition en considérant les modèles actifs soit comme des modèles atomiques soit comme des modèles couplés. Les ensembles $\{I^{\chi}_i\}$ et $\{Z^{\chi}_{i,j}\}$ définissent quant à eux les connexions entre les modèles actifs du réseau. Nous ne détaillerons pas ces deux ensembles qui jouent le même rôle que les ensembles EIC, EOC et IC des modèles couplés.

La structure est complétée par une fonction de sélection Ξ^{χ}. Cette fonction gère les problèmes de conflits lorsque plusieurs événements se produisent à la même date et que les destinataires appartiennent au réseau. Comme toute variable d'états, la variable s_{χ} inclu l'attribut θ_{χ} qui permet de définir des variables d'états propres au réseau. Il faut noter que cet attribut n'existe pas dans le cas des modèles couplés. Les modèles couplés définissant seulement un réseau de connexions entre modèles mais n'intégrent pas de dynamique propre.

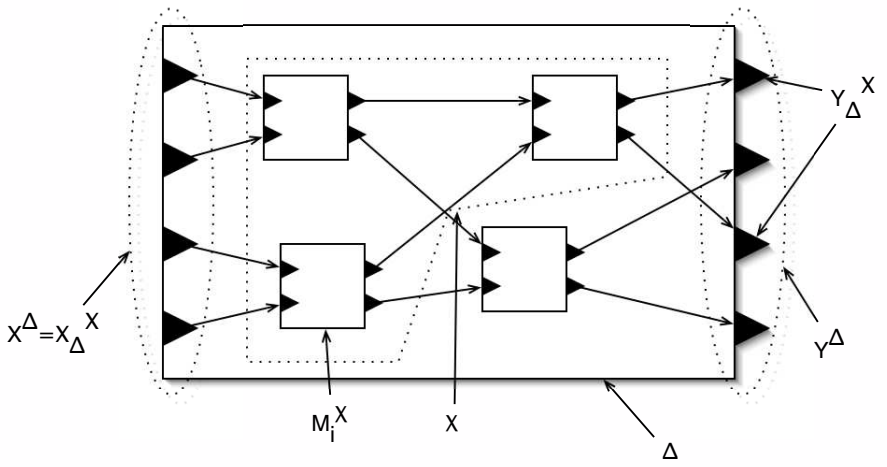


FIG. 1.8 – Représentation graphique d'un modèle DS-DEVS

F. J. Barros démontre dans son article [BAR95] que le formalisme DS-DEVS est conforme aux spécifications DEVS et possède donc la capacité d'être coupler avec tout modèle DEVS. De plus, deux simulateurs abstraits sont proposés. Ces deux simulateurs décrivent les opérations réalisées lors de l'exécution des fonctions de transitions internes et externes du réseau ainsi que l'initialisation des modèles. Quant aux modèles actifs du réseau, ils disposent des simulateurs abstraits des modèles atomiques DEVS (voir 2.9 du chapitre 2). Afin de fixer les idées, nous allons répresenter l'algorithme de traitement des événements externes pour le réseau. Cet algorithme va nous permettre d'illustrer la prise en compte du changement de structures.

Quand reception d'un événement externe : (x, t)
Si $t_l \leq t \leq t_n$ **Alors**
$D^{\alpha} = D^{\chi}$
Envoyer (x, t) à tous les modèles dont une entrée est connectée à l'entrée du modèle global
sollicitée par (x, t)
Attendre jusqu'à ce que tous les modèles aient acquité du traitement de (x,t)
Envoyer un message d'initialisation à tous les modèles appartenant à $D^{\chi} - D^{\alpha}$
Attendre jusqu'à ce que tous les modèles aient acquité du traitement de l'événement d'initialisation
$t_l = t$
$t_n = min\{t_{n,i}/i \in D^{\chi}\}$
Envoyer au simulateur père l'événement $(fait, t_n)$

Sinon "erreur"
Fin Si
Fin Quand

t_l et t_n représentent respectivement la date du dernier événement et la date de fin de l'état courant ($t_n = t_l + ta(s)$). L'algorithme ci-dessus ne met pas directement en évidence le changement de structures. Lorsque l'événement (x, t) est envoyé à tous les modèles connectés à l'entrée sollicitée, le modèle global (le réseau) attend que tous les modèles aient traité le message. Suite à ce traitement, la structure peut alors changée. Dans ce cas, l'algorithme envoie un événement d'initialisation à tous les nouveaux modèles. Si, au contraire, des modèles ont été désactivé alors lors du calcul de la date de fin d'états, seules les dates de fin d'états des modèles actifs sont prises en compte. Les mêmes remarques auraient pu être faites pour le traitement des événements internes (ou de fin d'états).

En conclusion, DS-DEVS nous offre un formalisme et une algorithmique pour les structures dynamiques de modèles couplés. Si on assimile le modèle du réseau à un agent, on peut alors spécifier formellement l'apparition ou la disparition d'agents dans le système. Le modèle DS-DEVS représente alors le système multi-agents. Si le changement de structures concerne les connexions entre les modèles cela peut se traduit dans le monde agent comme une modification du réseau d'accointances de l'agent ou comme un changement de comportement. Dans ce dernier cas, le modèle DS-DEVS ne représente pas le système multi-agents mais un agent et les modèles le composant représente les différentes facettes du comportement de l'agent. L'agent peut "switcher" dynamiquement entre plusieurs comportements.
L'apport de DEVS et de DS-DEVS pour la modélisation multi-agents est de proposer un formalisme rigoureux pour la spécification des systèmes multi-agents. Il existe des réalisations intéressantes [UHR94] [UHR96] [UHR01] pour les agents cognitifs avec James et les agents réactifs [DUB03d]. Néanmoins, il reste encore du travail afin de poser clairement les concepts manipulés par le paradigme multi-agents. Les résultats du groupe MIMOSA sur la construction d'un *framework* de modélisation centrée individus pour les systèmes spatio-temporels devraient nous permettre d'avancer dans le bon sens.

1.4 Conclusion et perspectives

Ce premier chapitre nous a permis de fixer les idées sur les concepts de base de la multi-modélisation et de la spécification formelle de systèmes dynamiques. Il était important de rappeler les notions de système, de système complexe, de schéma expérimental, de paradigme et de formalisme. Il était aussi important de les replacer dans leur contexte épistémologique avec un petit rappel sur l'histoire des Sciences. On oublie trop souvent que l'on est conditionné par notre vision des choses et par nos connaissances. Au travers ce discours, nous avons (enfin nous l'espérons) mis en évidence les problématiques sous-jacentes : l'adéquation du schéma expérimental par rapport aux interrogations sur le système, l'adéquation du formalisme au système et au paradigme, l'intégration d'une démarche au sein de l'activité de modélisation, la cohérence sémantique du couplage de modèles, ... Ces différentes problématiques ont été abordées au travers de diverses définitions pour finalement aboutir à une proposition. Cette proposition s'inscrit à la fois dans les idées de Fishwick et de multi-modélisation et dans les idées de Zeigler et de spécifications formelles de systèmes. La multi-modélisation est vue comme un processus de modélisation marriant des idées de décomposition ou d'agrégation et des idées de couplage

de paradigmes et de formalismes. Les processus de décomposition et d'agrégation sont inhérents à l'approche de modélisation adoptée : *top-down* ou *bottom-up*. Ces approches sont conformes à la vision dominante : le réductionnisme. Le fait d'adopter au sein des processus de décomposition et d'agrégation le postulat du multi-formalisme, on obtient alors la multi-modélisation. Pour être plus juste, il faut alors ajouter tous les problèmes liés à la cohérence du couplage de formalismes par décomposition et le point de vue du *mapping*. L'approche de Zeigler est alors complémentaire puisqu'elle s'abstrait des processus de décomposition et d'agrégation et propose la spécification formelle de systèmes dynamiques et de systèmes couplés avec DEVS. Notre proposition s'inspire d'une règle de bon sens : "il y a de bonnes idées dans les deux propositions, prenons les pour construire notre propre proposition". L'activité de modélisation doit s'inscrire dans une démarche et chaque niveau d'abstractions ou chaque élément du multi-modèles, il faut adopter un paradigme et un formalisme conformes aux propriétés du système ou du sous-système et compatibles formellement et sémantiquement avec les autres éléments du modèle. Cette idée sera défendue tout au long des deux chapitres suivants.

Nous avons conclu ce chapitre sur une ouverture sur les systèmes multi-agents réalisée dans le cadre de la thèse de Raphaël Duboz. Cette ouverture montre les capacités de DEVS à spécifier des systèmes à base d'agents réactifs et introduit l'idée que des formalismes tels que Cell-DEVS de Wainer et DS-DEVS de Barros sont aussi des outils nécessaires à la spécification de modèles centrés individus de systèmes spatio-temporels. Dans le futur, les travaux de Uhrmacher et du groupe MIMOSA seront nos sources d'inspiration afin d'aboutir sur des outils de spécifications formelles de systèmes spatio-temporels dans un démarche de modélisation centrée individus.

Chapitre 2

Des outils formels et opérationnels pour la multi-modélisation

Résumé

Ce deuxième chapitre est un tour d'horizon de nos travaux dans le domaine de la modélisation et de la simulation des systèmes complexes. Après avoir défini notre champ d'actions, la démarche de conception de notre *framework* et le *framework* que nous développons seront présentés. Cette présentation sera l'occasion de préciser ce que pour nous représente un environnement de modélisation et de simulation dans une perspective de multi-modélisation. La notion de multi-modélisation est entendu ici comme la construction de modèles couplés exprimés à l'aide de divers paradigmes et divers formalismes. Le *framework* est hiérarchisé en niveaux d'abstractions (opérationnel, simulation, modèle et sémantique). Chaque niveau est détaillé et mis en perpective vis à vis de l'existant.
Dans une deuxième partie, nous focaliserons notre discours sur la conception d'un langage de spécification formelle des modèles et de leur couplage ainsi que d'un langage de description des expériences. Ces langages sont au coeur de nos travaux et intégrent la majorité des idées que nous développons ou qui sont développés dans les groupes de travail auxquels nous participons. En effet, la conception d'un langage de spécification est l'occasion de poser proprement un certain nombre de termes et de définir une ontologie commune à la communauté des modélisateurs.
La dernière partie de ce chapitre sera consacrée à un survol de l'architecture et de la plate-forme VLE dans son état actuel.

Sommaire

Le thème central de recherche développé a été et reste la modélisation et la simulation des systèmes complexes. Si on s'attarde un peu sur ce titre, on pourrait imaginer que nos travaux de recherche sont des outils et des modèles qui sont destinés à la fois à des modèles mathématiques complexes de particules en physique nucléaire ou à des outils d'optimisation pour la résolution numérique du système d'équations différentielles spatialisées d'un système de satellites artificiels. En réalité, nous nous situons au niveau des outils informatiques orientés vers l'aide à la modélisation et la simulation des systèmes spatio-temporels de type écosystème, par exemple. On entend par outil informatique, toute méthode, tout formalisme ou tout *framework* aidant à la conception et à la mise en oeuvre du processus de modélisation et de simulation. Pour réduire encore notre champ de recherche, nos questions recurrentes sont :

- comment formaliser le plus universellement possible des modèles de systèmes spatio-temporels ? Existe-t-il un formalisme universel ?
- comment coupler deux modèles issus de paradigmes différents ? Comment en gérer la cohérence ?
- comment disposer d'une formalisation "exécutable" ?
- comment faire coopérer ou exécuter dans une même plate-forme des modèles de simulation conçus séparemment ?
- comment s'affranchir des problèmes techniques de simulation ?
- comment réutiliser des modèles déjà existants, validés et reconnus robustes ?

Toutes ces questions sont à l'origine du travail de recherche qui a été mené. Nous allons survoler en deux parties les réalisations et surtout les débuts de réponses à diverses questions. La première partie posera les fondements du *framework* de modélisation et de simulation. Nous synthétiserons alors la plupart des éléments conceptuels exposés dans le chapitre 1. La partie suivante fera le tour de la plate-forme informatique VLE (*Virtual Laboratory Environment*) mise au point à partir du *framework*. Ce sera aussi l'occasion de poser correctement les bases d'un langage unifié de multi-modélisations de systèmes spatio-temporels.

2.1 Un *framework* pour la modélisation et la simulation des systèmes complexes

Il existe une multitude de formalismes, de paradigmes et d'approches de modélisation et face à cette diversité, nous sommes maintenant confrontés au couplage de modèles issus de cette diversité. Notre point de vue d'informaticiens est le suivant : définir un *framework* offrant un cadre directeur de conception de modèles et de mise en oeuvre de simulateurs. Avant de rentrer dans les détails du *framework*, il serait intéressant de connaître la définition d'un *framework* et par conséquent, ses prérogatives. La définition d'un *framework* pourrait être la suivante : un ensemble de règles et d'outils qui structure le processus de fabrication d'un modèle. En génie

logiciel, plusieurs types de framework existent : les *frameworks* projet[1], les *frameworks* de conception et les *frameworks* de développement. Les *frameworks* de conception et de développement sont plus proches de ce que l'on cherche 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 lorsque l'on définit 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*, le *framework* de développement, est de nouveau une source d'inspiration pour nous. 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 la façon de coder. Cette dernière remarque est fondamentale et nous pouvons en déduire la définition de *framework* de modélisation et de simulation : **il consiste à offrir un cadre et des règles de construction de modèles pour le modélisateur afin d'homogénéiser les modèles**.

En modélisation, il faut tout d'abord se poser la question du niveau d'intervention du *framework*. On entend par là : doit-on définir des *frameworks* intervenants au niveau de la structuration des modèles et pour un formalisme donné ou doit-on imposer des règles de construction de plus haut niveau ? Pour être plus explicite : doit-on imposer un formalisme particulier ou une famille de formalismes compatibles [2] pour tous les modèles avec un cadre rigide de formalisation ? Il ne nous paraît pas raisonnable de structurer l'opération de modélisation jusqu'à un niveau de détail trop important. En revanche, il est vital de dégager un *framework* de modélisation unifiant les démarches et les formalismes. Nous nous sommes donc attaqués au problème de *framework* pour les paradigmes et par conséquent pour les formalismes. Ce travail a été initié au début de la thèse de Raphaël Duboz et a fait partie de nos nombreuses discussions au sein des groupes de travail (Mimosa et GTIMMS). Le débat a été étendu à des réunions de travail avec des chercheurs appartenant à d'autres laboratoires (LSIS et LISC). On peut citer à ce titre l'équipe du LISC du Cemagref de Clermont-Ferrand avec lequel nos idées ont muris. Plusieurs publications en témoignent [DUB03a][DUB03c].

Avant de présenter notre *framework*, un petit mot sur un *framework* (voir figure 2.1) proposé par Zeigler et Sarjoughian dans [ZEI00]. La présentation de ce *framework* n'est pas le fruit du hasard, il fait partie des conséquences de nos rencontres et de nos réunions de travail avec des chercheurs extérieurs à notre équipe. Les travaux de Zeigler font partie des bases des travaux développés par l'équipe de Norbert Giambiasi. Le *framework* se compose de six couches : réseau, simulation, modélisation, recherche, décision et collaboration. La couche *réseau* contient tous les éléments physiques de calcul (stations de travail, serveur, ...) et de connexion et d'interconnexion (tous les dispositifs et logiciels pour les réseaux LAN et WAN). La couche *simulation*

[1] il faut cadrer le projet dans ses grandes lignes et se référer à des projets-types.

[2] On entend par formalismes compatibles, tout formalisme que l'on peut coupler sans difficulté (un automate à états finis et une équation différentielle pilotant certains états de l'automate - voir [FIS95])

est une couche logicielle qui a pour objectif d'exécuter les modèles afin d'en dégager leur comportement. Elle intégre les protocoles nécessaires pour les bases de la simulation distribuée (qui sont standardisés dans les spécifications HLA qui seront présentés par la suite), 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 indépendant de l'implémentation de la couche *simulation*. Les trois couches suivantes sont des couches de haut niveau :

- la première et la deuxième ont pour objectif de faciliter l'exploration des modèles,
- la dernière assure la collaboration entre les participants à la construction du modèle.

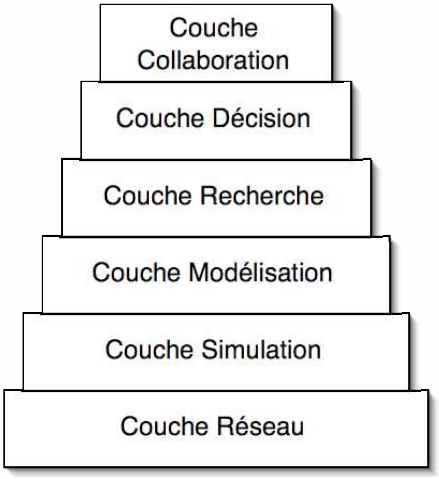


Fig. 2.1 – Framework de Zeigler & Sarjoughian

Notre point de vue est tout à fait conforme dans les grandes lignes à celui de Zeigler et Sarjoughian. Dans une première partie, nous allons présenter notre *framework* basé sur une hiérarchisation de niveaux d'abstrations et sur le principe de bus. Cette présentation sera suivie d'un survol de notre plate-forme logicielle VLE (*Virtual Laboratory Environment*) et de la présentation du langage de spécification des simulations et des modèles qui forme le coeur de notre *framework*.

Le couplage de modèles dans un objectif de simulation répartie et coopérative nous a conduit à organiser notre *framework* selon quatre niveaux d'abstraction (voir figure 2.1) : le niveau opérationnel, le niveau simulation, le niveau modèle et le niveau sémantique. Le niveau opérationnel doit assurer la mise en oeuvre et l'échange d'informations entre les éléments de la simulation dans un environnement réparti. Peu d'hypothèses doivent être faite 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 un moteur de simulation unifié sans spécificité propre à un modèle ou à un autre. La couche modèle doit assurer la spécification des modèles en terme de formalismes et de paradigmes. Une fois encore, cette couche doit être capable de s'abstraire des spécificités des formalismes et des paradigmes afin d'offrir une vue unifiée des modèles. La dernière couche, la couche sémantique, doit permettre la spécification des éléments sémantiques d'un modèle (nature de l'espace, du temps, par exemple).

L'idée retenue consiste à définir pour chaque couche des bus de communication. Un bus est un canal à travers lequel les élements communiquent (les flèches verticales de la figure 2.2). La notion de communication est vue ici du point de vue général. Par exemple, pour la couche modèle, deux modèles communiquent dès lors où ils doivent échanger de l'information (par exemple, des événements). 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 horizontales de la figure 2.2). Par exemple, la simulation doit être conforme à la spécification du modèle. Conceptuellement, le *framework* se structure autour de quatre bus (un bus par couche) : le bus opérationnel, le bus de simulation, le bus des modèles et le bus sémantique. Pour chaque bus, nous devons définir le protocole d'accès au bus. Nous allons passer en revue les différents bus en exposant les solutions possibles, nos choix et les questions qui restent à traiter.

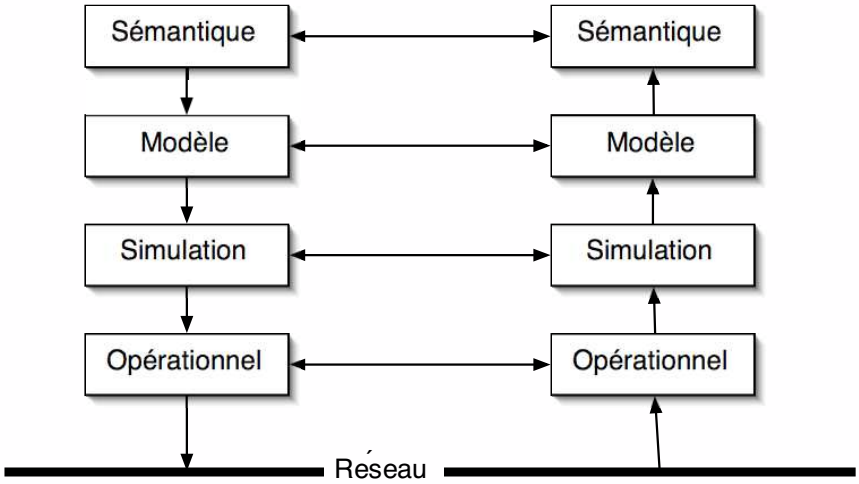


FIG. 2.2 – Hiérarchisation en niveaux d'abstraction

2.1.1 La couche opérationnelle

La couche opérationnelle est probablement la plus étudiée et la plus proléfique en terme de technologies mais aussi la couche la plus hétérogène. Depuis plusieurs années, des groupes d'étudiants de deuxième et troisième cycle d'informatique travaillent sur l'étude des différentes technologies. Cette année encore, des étudiants de maîtrise informatique et de DESS Ingénierie des Systèmes Informatiques Distribués vont s'intéresser à la communication entre les simulateurs mises en oeuvre à l'aide de divers langages (C++, Java, Smalltalk, Python, Fortran, ...). Le but de cette nouvelle étude est d'identifier la meilleure technologie en fonction du choix des langages et des infrastructures. Aussi, afin de mieux comprendre le problème, il faut décomposer un peu cette couche en sous-couches (voir figure 2.4). Celle-ci peut être décomposer en trois couches relativement indépendantes :

- la sous-couche réseau,
- 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, on intégre toutes les technologies réseaux et protocoles mis en oeuvre pour le transport d'informations d'un processeur [3] à un autre. On part donc du postulat que les processeurs sont reliés par un réseau de la famille Ethernet (Ethernet, Fast-Ethernet ou Myrinet) supportant le protocole TCP/IP. La deuxième couche doit offrir 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 [4], Corba, RMI [5], DCOM [6], RPC [7], MPI [8], 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 (voir figure 2.3). Ce protocole repose dans la plupart de ces mises en oeuvre 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).

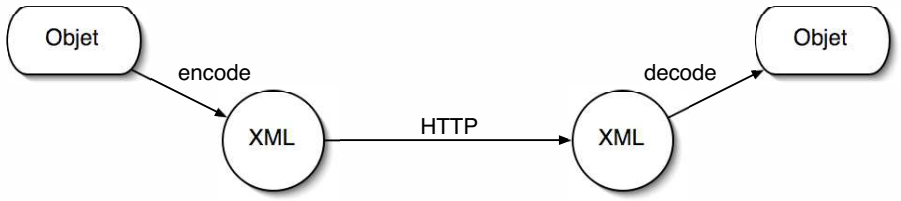


Fig. 2.3 – Principe de SOAP

Avec SOAP, Corba, RMI et DCOM, nous avons à notre disposition des technologies principalement orientées service Web et orientées objets. RPC, MPI ou les sockets sont des bibliothèques de fonctions très proches du niveau réseau. Pour exemple, MPI est utilisé dans les clusters[9] de PC pour la programmation parallèle. Ces bibliothèques sont de bas niveau et pas toujours simples à mettre en oeuvre.

La dernière sous-couche, la sous-couche langage, est tout aussi délicate. Si on fait rapidement le bilan des langages utilisés en programmation de simulateurs, on s'aperçoit que le langage numéro un est Fortran pour tous les modèles de type numérique. Si on s'intéresse aux simulateurs orientés automate cellulaire, individus, ..., les langages que l'on rencontre sont alors Java, C/C++ et Smalltalk. Cette dernière couche peut être vu autrement : au lieu de parler de langage, on pourrait parler de plate-formes de simulation. C'est le point de vue de HLA que l'on

[3] On désigne par processeur tout élément succeptible d'exécuter un programme et de communiquer.
[4] Simple Object Access Protocol
[5] Remote Method Invocation
[6] Distributed Component Object Model
[7] Remote Procedure Call
[8] Message Passing Interface
[9] Un cluster de PC est un ensemble de machines reliés 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.

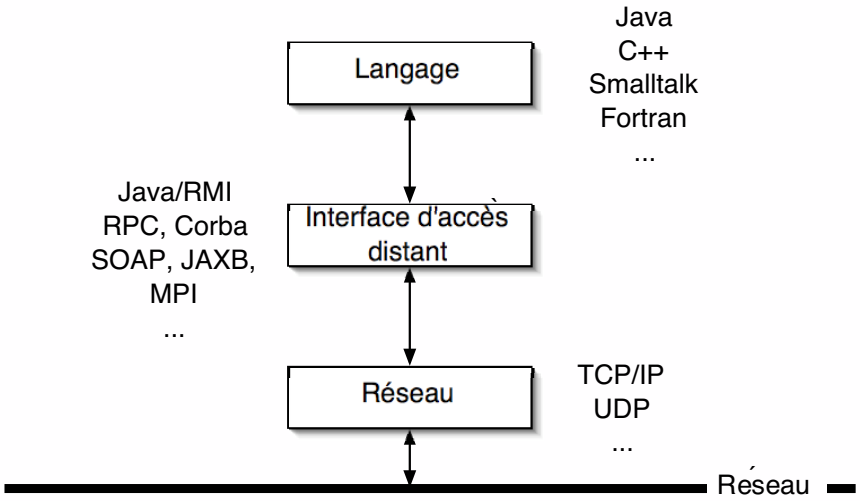


FIG. 2.4 – Décomposition de la couche opérationnelle

présentera par la suite. On ne se soucie pas du langage de programmation mais de l'outil complet. Il est alors nécessaire de se poser la question de son intégration dans un environnement distribué de simulation. Afin de compléter ce point de vue, nous allons présenter HLA.

HLA (High Level Architecture) est une architecture, un *framework* 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 plate-formes de simulation et à des aspects de plus haut niveau tel que l'échange d'évenements 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 (DoD). 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 issue d'un effort de standardisation des architectures de simulation telles que SIMNET (Simulation Network [10][KAN90]) et DIS (Distributed Interactive Simulation [STE94]).
HLA est une architecture et non un environnement logiciel. Elle propose un cadre de travail et un vocabulaire commun. Afin de proposer un environnement logiciel, l'utilisation d'un RTI (RunTime Infrastructure) est nécessaire pour supporter les opérations d'exécution et de coordination des simulateurs. Le RTI fournit un ensemble de services utilisés par les simulateurs (ou fédérés[11]). La définition des services est indépendante des plate-formes et des langages. On ne sera pas étonné de retrouver 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 [12]. Ces services ont pour objectif de coordonner les opérations et les échanges de données

[10]http ://www.stricom.army.mil/PRODUCTS/SIMNET/
[11]Un fédéré est une simulation compatible HLA (*HLA-compliant simulation*). Un fédéré appartient à une fédération.
[12]Une fédération est un ensemble de simulations ou fédérés interopérants.

durant l'exécution de la simulation globale. L'accès à ces services est définie par la spécification de l'interface HLA. Cette interface est une interface fonctionnelle entre le fédéré (la simulation) et le RTI (voir figure 2.5). La connexion au RTI est assurée par des *ambassadors*[13]. 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 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 premier modèle, 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 quant à lui spécifie les échanges de données entre fédérés et les données partagées, et 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.

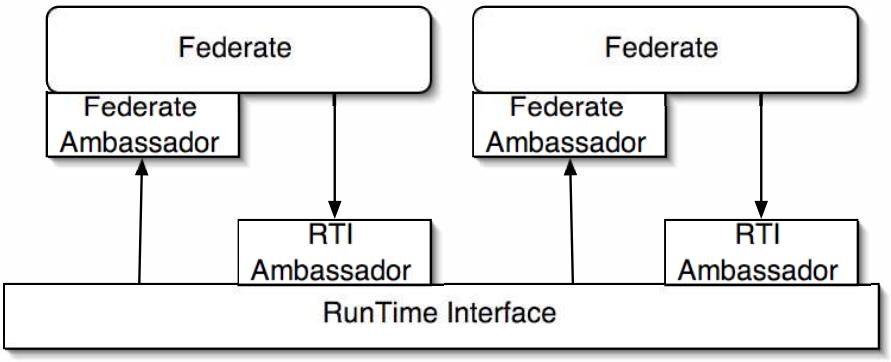


Fig. 2.5 – RunTime Interface

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[14]. Plusieurs techniques sont supportées et parmi ces techniques, on peut citer : les techniques classiques (approches synchrones fortes, approches asynchrones faibles ou fortes -TimeWrap-, ...), le temps discret et le temps réel. 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 ai donné l'autorisation. Cette autorisation est naturellement en adéquation avec la technique adoptée. L'ensemble de ces

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

[14]Un événement ne peut pas arriver dans le passé.[LAM78]

modèles (SOM,FOM et MOM) font l'objet de standardisation IEEE [15]. Il existe de multiples implémentations du RTI dans des divers langages (C++, Java, ...) et pour différentes technologies distribuées (Corba, DCOM, ...).

En conclusion, on peut 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 mais le simulateur est considéré comme une boîte noire. C'est le simulateur qui demande l'autorisation d'avancer son horloge à la fédération et non le contraire c'est à dire ce n'est pas la fédération qui contrôle l'avancement du temps local du simulateur. De plus, les objets internes aux simulateurs et surtout la dynamique interne des modèles ne sont pas pris en charge. Néanmoins, HLA propose une multitude de réponses à des problèmes du niveau opérationnel et d'une partie du niveau simulation.

Notre couche opérationnelle se présente donc sous 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 indépendemment des protocoles et des langages. On peut donc résumer l'infrastructure opérationnelle sous forme d'un schéma (voir figure 2.6).

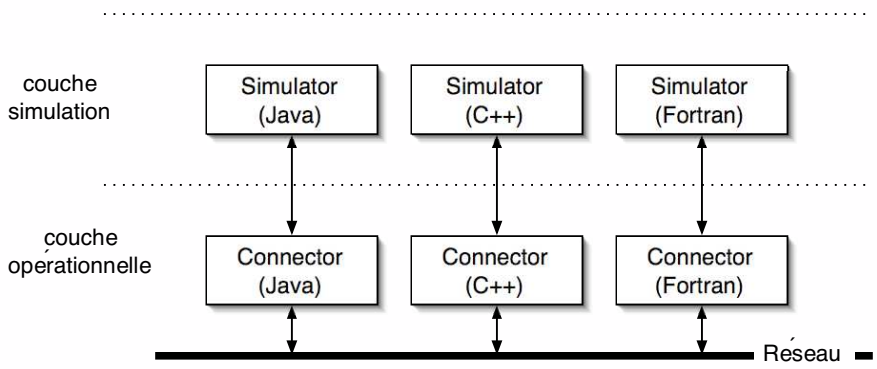


FIG. 2.6 – Infrastructure opérationnelle

L'infrastructure repose principalement sur la notion de connecteur (*Connector*). 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. Le connecteur doit faire le lien entre les simulateurs et l'infrastructure physique de communication. Comme nous l'avons vu au travers HLA, les connecteurs doivent proposer une interface universelle (le RTI). On distingue deux classes de problèmes : celle du langage de programmation utilisée pour les couches supérieures et celle de l'hétérogénéité. L'hétérogénéité est entendue ici comme la diversité des langages de programmation utilisés. En effet, on peut identifier quatre cas de figure[16] :

[15]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

[16]Ces cas de figure ont été étudiés dans le cadre de projets de fin d'années de DESS et dans le cadre d'un stage de DEA

- tous les simulateurs sont implémentés par des langages orientés objets et les langages utilisés proposent une API (Application Programming Interface) SOAP,
- 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 dans un cluster Linux.

Selon les trois premiers cas de figure, on distingue alors trois classes de connecteurs qui doivent être disponibles dans les différents langages (Java, C/C++, Fortran, Smalltalk, ...). Si les simulateurs impliqués dans le modèle global sont programmés en Java, C++ ou Smalltalk par exemple, il est possible de les faire coopérer à l'aide de SOAP[17]. Le connecteur aura pour rôle de transformer les invocations de méthodes et les objets SOAP en invocations et objets Java, C++ ou Smalltalk qui seront alors manipulables 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 les moyens de communication plus rapide mais plus difficile à mettre en place. Avec SOAP, les objets peuvent être transmis via le média de communication (après une phase de traduction en XML assuré 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 sûrement le plus simple de tous. 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 (*Remote Procedure Call*) et en particulier RMI (*Remote Method Invocation*). Nous avons pris pour exemple Java mais la remarque reste vraie pour les autres langages (ONC/RPC[18],[19] pour C++, par exemple).

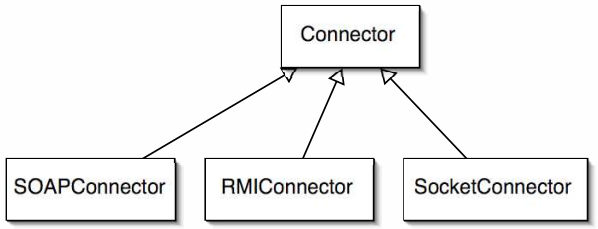


Fig. 2.7 – Les connecteurs

Les technologies SOAP ont fait l'objet d'une étude par un groupe de quatre étudiants de DESS ISIDIS en 2002-2003.

Quant au dernier cas de figure, il est un peu particulier et a été étudié dans le cadre d'un stage de DEA [QUE03a]. Les trois précédentes technologies sont orientées vers les réseaux de type LAN et WAN connectant un ensemble de machines "classiques" sous Linux, Windows, MacOS X, ... Cette dernière technologie est orientée vers les clusters de PC sous Linux. Cette architecture permet de disposer d'un ensemble de processeurs munis de mémoire non partagée

[17]Il existe une alternative à SOAP avec XML/RPC disponible en Java, C++, ...
[18]Il existe aussi une version pour Java
[19]http ://www.plt.rwth-aachen.de/ks/english/oncrpc.html

de type volatile et non volatile. Les processeurs sont aussi accompagnés d'une interface réseau afin d'être connectés entre eux. Le type de réseau est LAN avec des techniques en terme de débit plus élévé 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 légéremment adaptée) et est augmenté de bibliothèques de communication spécifique en l'occurrence dans notre cas MPI (Message Passing Interface). Cette biliothèque est très proche des sockets et permet de mettre en oeuvre 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'exclu pas les autres langages mais interdit leur couplage avec MPI [20].

En conclusion, la couche opérationnelle permet à la couche simulation d'être indépendante de l'infrastructure matérielle et logicielle.

2.1.2 La couche simulation

La couche simulation a pour objectif d'assurer la mise en oeuvre de l'animation des modèles. L'algorithmique des simulateurs est, dans la majorité des cas, spécifiques au formalisme adopté pour la phase de modélisation. Ce constat peut s'avérer être un vrai problème. En effet, il faut alors répondre à la question : comment coupler deux simulateurs dont l'algorithmique est fondamental différent ? Prenons un petit exemple. Nous avons modéliser une partie de notre système 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 par un algorithme d'intégration numérique de type Runge-Kutta et pour le réseau de Petri, nous utilisons tout simplement l'algorithme d'évolution du marquage synchrone[21]. Il paraît assez évident que le couplage de deux simulateurs aussi différents n'est pas simple. Cette difficulté n'est pas la seule oeuvre de la couche simulation. La couche modèle est fortement impliquée.
Deux approches sont alors possibles : soit on recherche un couplage au niveau simulation et modèle soit on recherche un couplage seulement au niveau modèle. Quelles sont les implications des deux approches ? Que signifie s'intéresser seulement au couplage au niveau modèle ? Cette approche est étudiée par Fishwisk sous le terme de *Multi-modeling* [FIS95]. Il considère qu'un modèle peut être composé de plusieurs autres modèles sous forme d'un graphe. De tels modèles sont appelés multi-modèles. Le couplage de modèles est vue soit comme la connexion d'entrées et de sorties de modèles soit comme la décomposition d'un modèle en plusieurs modèles couplés soit comme le raffinement d'un modèle. La décomposition permet de créer des modèles hiérarchiques soit par décomposition soit par agrégation. Des modèles sont agrégés (ou couplés) pour former des modèles de plus haut niveau. Les modèles sont considérés comme des boîtes noires. Dans le cas du raffinement, un élément du comportement d'un modèle est exprimé plus précisemment à l'aide d'un autre modèle. Cette opération définit *in facto* un nouveau niveau d'abstraction. La notion de raffinement travaille au niveau de la spécification de la dynamique. Par exemple, on peut définir un système d'équations différentielles pour expliquer les conditions de changements d'états d'un autre modèle. Ce système d'équations est un raffinement d'un état d'un modèle.
Cette approche du couplage de modèles est intéressant au niveau modèle mais ne répond pas à la question du couplage au niveau simulation. Le modélisateur est obligé de repenser ses

[20] Il existe quelques implémentations Java

[21] Toutes les transitions franchissables sont validées en même temps. Si des conflits existent un tirage aléatoire est effectué.

simulateurs en fonction du multi-modèle construit. Fishwick apporte néanmoins une réponse intéressante pour la démarche de construction de modèles.
Nous nous sommes donc orientés vers les travaux de Zeigler [ZEI73][ZEI99] et le formalisme DEVS. Comme nous l'avons déjà présenté dans le chapitre précédent, DEVS est un formalisme abstrait pour la modélisation à événements discrets. Ce formalisme a la prétention d'offrir à la fois l'encapsulation des autres formalismes et les simulateurs associés. Nous reviendrons sur la notion d'encapsulation dans la couche modèle. Attardons nous sur les simulateurs abstraits. En effet, DEVS propose non seulement une notation pour les modèles mais propose aussi les algorithmes de simulation. Voici les deux algorithmes de base de DEVS : l'un exprime le comportement d'un modèle atomique (voir figure 2.9) et l'autre celui d'un modèle couplé (voir figure 2.10). Ces deux algorithmes tel qu'ils sont présentés font qu'un seul *a priori* qu'ils soient "compatibles" DEVS. Nous approfondirons cette notion après la présentation des deux algorithmes.

Un simulateur abstrait est vu comme une boîte noire (voir figure 2.8) acceptant en entrée trois types d'événements[22] (conformément au formalisme DEVS) :

- initialisation (i,t) : l'état S du modèle est initialisé à l'instant t.
- externe (x,t) : un autre modèle a envoyé un événement à la date t, le modèle était dans son état S depuis t_l ($e = t - t_l$) et devait changer d'état à t_n ($\sigma = t_n - t$). Le modèle va le traiter en fonction de sa fonction de transition externe δ_{ext}.
- interne (*,t) : le modèle a atteint la date de fin de l'état courant S et le modèle va changer d'état selon sa fonction de transition interne δ_{int}.

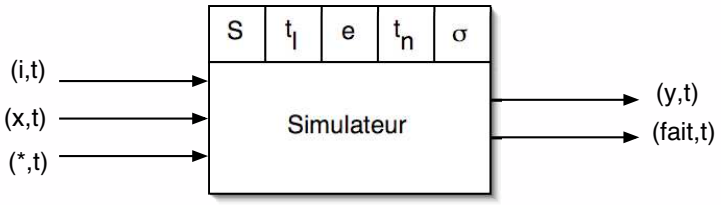


FIG. 2.8 – Le simulateur abstrait d'un modèle atomique

L'algorithme de la figure 2.9 présente les trois phases (initialisation, événement externe et fin d'état) de la dynamique d'un modèle DEVS. L'initialisation fixe l'état initial du modèle et les variables t_l et t_n. La réception d'un événement externe, mis à part la mise à jour des variables e, t_l et t_n, fait appel à la fonction de transition externe δ_{ext}. Cette fonction ainsi que la fonction de transition interne et la fonction de sortie, sont naturellement spécifiques au modèle. La réception d'un événement de fin d'état débute par le calcul de la fonction de sortie λ et se poursuit par l'envoi des événements externes générés par la fonction de sortie au père. La notion de père est synonyme ici de modèle hiérarchiquement supérieur. Si le modèle génére des événements externes, c'est pour un modèle avec lequel il est couplé dans un modèle couplé (le modèle père). Dans cette approche, le modèle père connait les connexions entre les modèles qui le composent et non les modèles le composant. Cette phase se termine par le calcul de la fonction de transition interne afin de déterminer le nouvel état du modèle. Cet algorithme en trois phases constitue la base universelle des simulateurs de modèles atomiques basés sur le formalisme DEVS. Il reste normalement un dernier cas à traiter celui du conflit lorsqu'un événement externe arrive exacte-

[22]Tout événement se produit à une date t

ment au moment de la fin d'un état. Nous ne présentons pas ce cas. Intéressons nous maintenant à l'algorithme d'un modèle couplé.

Quand reception d'un événement d'initialisation (i, t)
  $t_l = t - e$
  $t_n = t_l + ta(s)$
Fin Quand
Quand reception d'un événement externe : (x, t)
  Si $t_l \leq t \leq t_n$ **Alors**
    $e = t - t_l$
    $s = \delta_{ext}(s, e, x)$
    $t_l = t$
    $t_n = t_l + ta(s)$
    Envoyer au simulateur père l'événement $(fait, t_n)$
  Sinon "erreur"
  Fin Si
Fin Quand
Quand reception d'un événement de fin d'état $(*, t)$
  Si $t = t_n$ **Alors**
    $y = \lambda(s)$
    Envoyer au père d'un événement externe (y, t)
    $s = \delta_{int}(s)$
    $t_l = t$
    $t_n = t_l + ta(s)$
    Envoyer au père fin de traitement $(fait, t_n)$
  Sinon "erreur"
  Fin Si
Fin Quand

FIG. 2.9 – Simulateur abstrait d'un modèle atomique

L'algorithme de la dynamique d'un modèle couplé se décompose pour sa part en quatre phases :

- l'initialisation : le modèle couplé transmet l'événement d'initialisation à tous les modèles qui le composent.
- la réception d'un événement externe en entrée : cet événement est envoyé à tous les sous-modèles récepteurs de cet événement[23].
- la réception d'un événement externe provenant d'un sous-modèle : cet événement est envoyé soit aux sous-modèles connectés au modèle qui a émit l'événement (connexion interne - IC) soit au modèle père si le sous-modèle émetteur est connecté à une sortie du modèle couplé (connexion en sortie - OIC).
- la fin de traitement des sous-modèles : lorsqu'un modèle atomique reçoit un événement, il signale à son modèle père qu'il a fini de le traiter ; tant que pour une date donnée, tous les sous-modèles n'ont pas fini, le modèle couplé attend ; si tous les sous-modèles ont fini alors il y a calcul de la date du prochain événement.

[23]Dans le formalisme DEVS, la relation entre un port d'entrée du modèle couplé et un port d'entrée des modèles qui le composent est définie par l'ensemble EIC.

Comme on peut le constater l'algorithme de la dynamique d'un modèle couplé s'articule autour de la notion de relai. On peut dire en effet qu'un modèle couplé est une sorte d'aiguillage d'événements et que cet aiguillage est realisé grâce à la définition des connexions.

Quand reception d'un événement d'initialisation (i, t)
  Envoyer à tous les modèles composant le modèle couplé un événement d'initialisation (i, t)
  $active = \emptyset$
Fin Quand
Quand reception d'un événement externe : (x, t)
  $\forall i \in EIC$
    Envoyer l'événement externe (x, t) à i
    $active = active \cup \{i\}$
Fin Quand
Quand reception d'un événement externe : (y, t) provenant du sous-modèle i
  $\forall (i, j) \in IC$
    Envoyer l'événement externe (y, t) à j
    $active = active \cup \{j\}$
  Si $i \in OIC$ **Alors**
    Envoyer l'événement extern (y, t) au père
  Fin Si
Fin Quand
Quand reception d'un événement de fin d'état $(*, t)$
  I est l'ensemble des sous-modèles tel que leur date de fin d'état est égale à t
  $i^* = select(I)$
  Envoyer un événement de fin d'état $(*, t)$ à i^*
  $active = active \cup \{i^*\}$
Fin Quand
Quand reception d'un événement de fin d'état $(fait, t)$
  Retirer l'expéditeur du message de *active*
  Si $active = \emptyset$ **Alors**
    $t_l = t$
    $t_n = min(M_i.t_n \ \forall i \in D)$
    Envoyer l'événement $(fait, t_n)$ au père
Fin Quand

FIG. 2.10 – Simulateur abstrait d'un modèle couplé

Après cet exposé des simulateurs abstraits, on peut noter un élément intéressant : 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. Quelles sont les contraintes pour le concepteur de simulateurs ? Les algorithmes des simulateurs abstraits reposent pleinement sur la modélisation DEVS ce qui implique *a priori* qu'il est nécessaire de réaliser une spécification complète du modèle en DEVS. Deux approches sont en effet possibles : le *mapping* DEVS ou le *wrapping* DEVS. Le mapping consiste à totalement spécifier le modèle en DEVS et ce quel que soit le formalisme utilisé pour la couche modèle. Les travaux de Jacques et Wainer sont un parfait exemple de *mapping* [JAC02]. Ces travaux s'intéressent à la modélisation à l'aide de réseaux de Petri et Jacques et Wainer proposent

une modélisation 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 est un compromis entre la volonté de disposer d'un noyau unifié de simulation et l'utilisation d'un formalisme quelconque pour la couche modèle. Cette approche est appelée *wrapping*. Cette idée a initialement apparu dans nos réunions de travail avec les membres du LISC. Le simulateur est dans l'obligation de mettre en oeuvre une liste de fonctions qui rentrent dans la logique algorithmique des simulateurs abstraits. Le *wrapper* est donc une sorte d'enveloppe qui encapsule le simulateur et qui est adapté au formalisme utilisé. Le travail du concepteur de simulateurs se résume alors en la construction de ce *wrapper*[24] ou en l'utilisation du wrapper existant du formalisme qu'il a choisi.

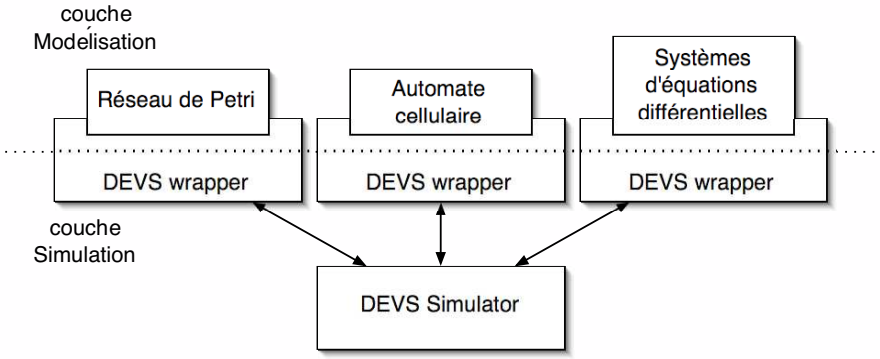


FIG. 2.11 – DEVS wrapper

L'étude des algorithmes des simulateurs abstraits permet d'identifier six fonctions utiles (voir ci-dessous). Il faut ajouter à ces six fonctions une fonction assurant l'analyse du fichier XML qui spécifie le modèle (*parseXML*).

```
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();
public abstract boolean parseXML(Document p_document,Node p_modelNode,Node p_dynamicsNode
```

FIG. 2.12 – Interface du wrapper en Java

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 λ et d'une fonction d'avancement du temps ta. On retrouve dans l'interface DEVS, le *wrapper*, les deux fonctions de transitions sous forme de

[24] On parlera aussi d'interface DEVS

fonctions de traitement des événements externes (*processExternalEvent*) et de fin d'état (*processInternalEvent*). Ces deux fonctions doivent procéder au changement d'état en fonction de l'instance d'événement passée en paramètre. Ils repectent la spécification DEVS :

- $\delta_{int} : S \rightarrow S$
- $\delta_{ext} : S \times X \rightarrow S$

Le paramètre de la fonction *processInternalEvent* ne fait pas parti de la spécification DEVS mais il permet dans notre cas de transporter certaines informations telles que la date d'occurrence de l'événement, par exemple.
Le traitement de l'événement d'initialisation est représenté par une simple fonction (*init*). La fonction de sortie, quant à elle, se traduit par une fonction admettant en paramètre la date courante afin d'estampiller les événements qui seront générés et retourne une liste d'événements. Pour être exhaustif, il faut dire un mot de la notion d'événement. Elle est mise en oeuvre 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 abstraits et représentent l'événement de fin d'état. L'opération de construction d'un événement externe par la fonction de sortie est très simple : elle a juste besoin de connaître le port de sortie sur lequel l'événement doit être émis et la date d'occurence. 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'entête 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 vous disposez d'un *wrapper* DEVS. Il peut être intégrer à une simulation distribuée et le modèle qu'il met en oeuvre pourra être couplé de manière totalement transparente.

Comme nous l'avons vu un simulateur vient s'intégrer à la simulation globale en se connectant sur un bus logiciel défini par l'interface DEVS et la notion de simulateur abstrait. Nous allons maintenant préciser l'architecture du bus. L'architecture repose sur la relation entre un coordinateur et des simulateurs. Contrairement à l'architecture proposée par Zeigler dans [ZEI99] et reprise dans de nombreuses implémentations (DEVS-Java [25], DEVS/C++ [26], CD++ DEVS [JAC02], JDEVS [FIL02a][FIL02b], [PRA96], ...), nous avons assigné aux concepts de simulateur et de coordinateur des rôles légèrement différents.
Un modèle DEVS se présente sous forme d'une hiérarchie de modèles qui peut se représenter sous forme d'un arbre. Les feuilles de cet arbre sont les modèles atomiques et les noeuds les modèles couplés (voir figure 2.13). Les simulateurs abstraits des différents types de modèles sont différents d'où l'approche classique d'associer un simulateur par modèle atomique et d'associer un coordinateur par modèle couplé. Dans notre approche, un simulateur gère une partie de la hiérarchie et ce sous-arbre de modèles atomiques et de modèles couplés regroupe les modèles localisés sur une même machine.

L'architecture de simulation se limite alors à deux niveaux (voir figure 2.14) : le niveau coordinateur et le niveau simulateur. Dans cette optique, le simulateur gère à la fois des modèles atomiques et des modèles couplés. L'algorithme de calcul de la dynamique d'un simulateur est alors la fusion des deux algorithmes présentés précédemment (voir 2.9 et 2.10). Pourquoi cette approche ? En réalité, ce n'est qu'une optimisation. En effet, dans la théorie, les événements externes doivent passer par le modèle père pour déterminer les destinataires (les modèles dont

[25] http ://www.acims.arizona.edu/SOFTWARE/software.shtml#DEVSJAVA
[26] http ://www.acims.arizona.edu/SOFTWARE/software.shtml#DevsC++

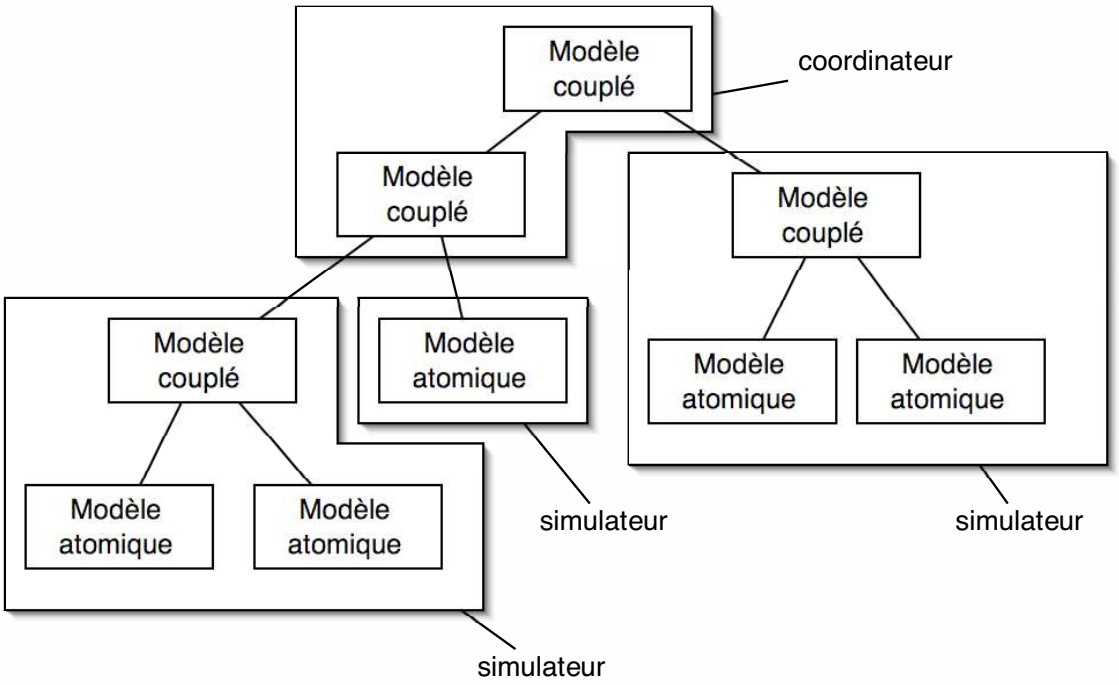


FIG. 2.13 – Association des simulateurs et du coordinateur à la hiérarchie des modèles

un port d'entrée est connecté au port de sortie qui a émit l'événement). Ici, le simulateur connaît l'ensemble de la hiérarchie des modèles et donc leur couplage ce qui permet de déterminer directement les destinataires "locaux". Si le port émetteur de l'évenement est connecté à un port de sortie du modèle couplé de plus haut niveau géré par le simulateur alors il y a appel au coordinateur pour la diffusion de l'événement. Quant au coordinateur, il applique tout simplement l'algorithme des simulateurs abstraits des modèles couplés. La seule différence par rapport au simulateur est qu'il gére la cohérence globale des événements et par conséquent, il garantit la causalité.

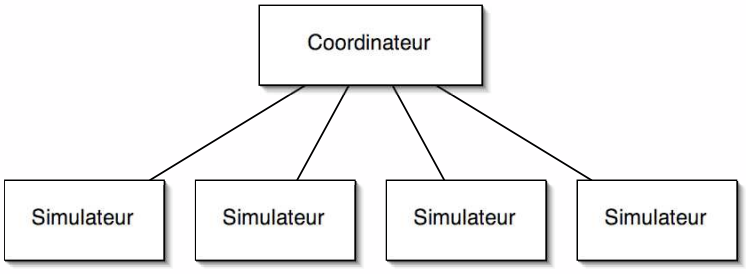


FIG. 2.14 – Architecture des simulateurs et du coordinateur

L'architecture de la couche simulation est résumée dans le diagramme de classes de la figure 2.15. Ce modèle de classes est implémenté en Java et en C++. Les *wrappers* apparaissent sous le terme *Dynamics*.

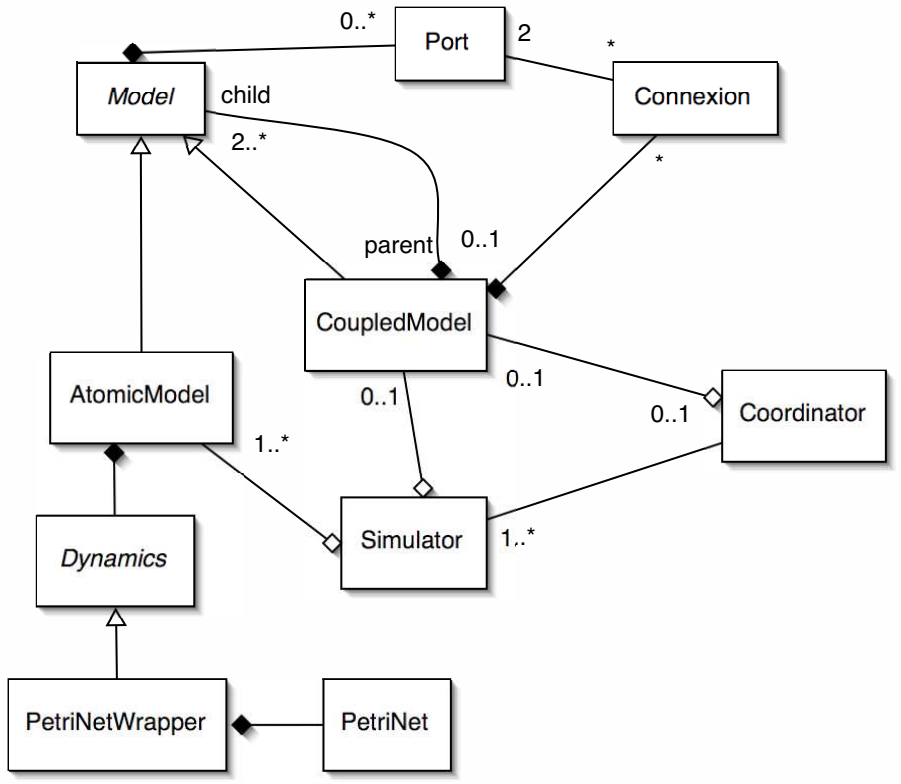


FIG. 2.15 – Modèlisation UML des classes de la couche simulation

Les choix conceptuels qui ont été opéré pour la couche simulation permettent au concepteur de simulateurs de s'affranchir des mécanismes internes des simulateurs. En contre partie, il est obligé d'adopter une certaine logique et surtout de penser ces simulateurs en terme d'événements discrets ou de penser à une interface de communication externe qui s'exprime en terme d'événements. Deux orientations sont possibles : le développement *ex nihilo* d'un simulateur ou l'utilisation d'un formalisme disposant de son interface DEVS. Nous ne développerons ici la première orientation qui implique des aspects techniques (conformité à l'interface DEVS, utilisation de bibliothèques d'objets DEVS, ...). En revanche, nous allons aborder dans la partie suivante, la couche modèle, l'orientation formalisme.

2.1.3 La couche modèle

Le choix de DEVS au niveau simulation des simulateurs abstraits implique l'introduction de DEVS dans la couche modèle. La couche modèle doit comme les autres couches apporter un moyen de communication, ici au niveau des modèles, entre les éléments de la couche. Il est donc nécessaire de définir un bus modèle où les formalismes et paradigmes utilisés pour les modèles sont capables de se coupler conceptuellement. Cette idée est parfaitement développée dans [KIM98] ou dans [ZEI00] sous l'intitulé *DEVS bus* (voir figure 2.16). Avec DEVS-bus, on retrouve aussi les concepts développés pour la couche simulation et en particulier, les *wrappers* sont ici nommés les *converters*.

De nombreux auteurs proposent des spécifications DEVS de formalismes qui menent parfois

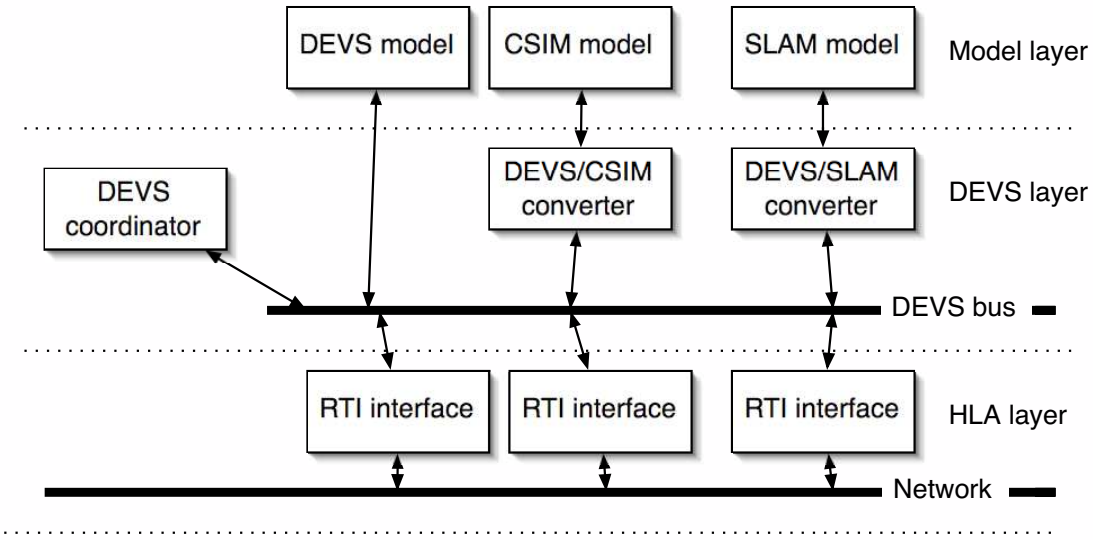


FIG. 2.16 – DEVS bus

à des extensions du formalisme DEVS tout en garantissant une unité conceptuelle. Vangheluwe en fait état dans [VAN00a] et développe son approche $AToM^3$[LAR02] de méta-modélisation et de couplage de formalismes différents[27]. Citons quelques exemples :

- DESS, DTSS, DEV&DESS, Quantized Systems ([ZEI99], [ZEI00a], [KOF01], [BOL02])
- G-DEVS ([GIA00])
- Cell-DEVS ([WAI01]) et DEVS et les automates cellulaires ([VAN00b])
- DS-DEVS ([BAR95])
- ... mais aussi des propositions concernant les state-charts [BOR03], ...

Contrairement à la couche simulation, pour unifier conceptuellement les modèles, il est nécessaire de proposer les spécifications DEVS cités précédemment. On ne peut pas se contenter d'une simple encapsulation du modèle. 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. Dans cette seconde optique, certaines précautions doivent être prises. La première est sûrement la plus importante : définir un comportement en terme d'événements discrets. Certains formalismes tels que les automates à états finis, les réseaux de Petri, les state-charts, ..., ne posent pas de problème de ce point de vue puisqu'intrinsèquement la notion d'état et la notion d'événement font parti du formalisme. En revanche, si on s'intéresse aux équations différentielles, le problème est plus délicat comme le font apparaître des travaux tels que DESS, DEV&DESS et G-DEVS.
Le deuxième point délicat concerne le temps. 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 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. A aucun moment, la notion de temps n'a été évoqué. Afin de développer un peu l'idée, on peut considérer que le temps est discret et prend ses valeurs

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

parmi les entiers positifs. A 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 auxquel il est couplé. Si le réseau de Petri représente un processus temporel, il faut alors adopté une base de temps conforme au processus modélisé.
La troisième et dernière contrainte concerne la différentiation entre la fonction de transition interne et la fonction de transition externe. Cette différentiation fait la spécificité de DEVS et n'est donc pas présente dans les autres formalismes. De toute façon, la contrainte n'est pas très importante puisque l'on peut s'en affranchir en décomposant une transition avec fonction de sortie en une transition externe et une transition interne avec fonction de sortie (voir figure 2.17) en introduisant un état fictif de durée de vie nulle.

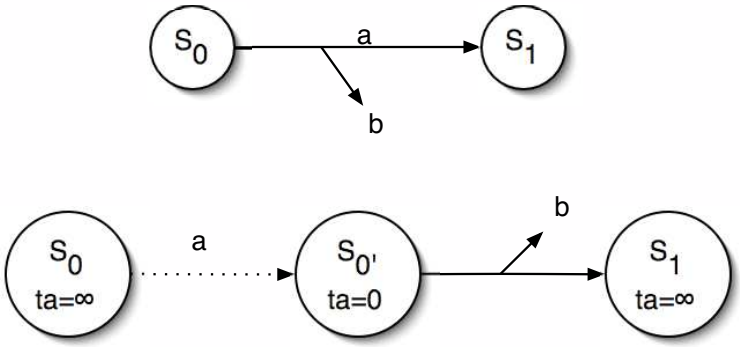


Fig. 2.17 – Décomposition d'une transition

La majorité des formalismes classiques a été spécifié en DEVS. En revanche, nous n'avons pas encore évoqué le paradigme agent et centré individus. Comment spécifier en DEVS un modèle agent ou centré individus ? Nous avons apportés un début de réponse à cette question avec nos travaux sur la spécification DEVS d'agents réactifs [DUB02b] qui sont exposés en 1.3.3.

2.1.4 La couche sémantique

Lorsqu'un modélisateur construit un modèle, ce dernier s'impose un certain nombre d'hypothèses sur le système. Ces hypothèses peuvent être de plusieurs types. Dans le cas des systèmes spatio-temporels, des hypothèses sont souvent admises au niveau de la représentation du temps et de l'espace mais aussi au niveau de la représentation des entités du système. Prenons un exemple dans l'un des domaines auquel on s'intéresse : les écosystèmes. Lorsque l'on construit un modèle de dynamique de populations[28], plusieurs hypothèses peuvent être faites lors de l'utilisation de formalismes telles que les équations différentielles :

– les prédateurs et les proies sont répartis uniformément dans l'espace (cette hypothèse est couramment prise mais elle n'est pas sans conséquence - voir chapitre 3),

[28]Un modèle de dynamique de populations représente l'interaction entre un ensemble de prédateurs et un ensemble de proies.

- les processus impliqués dans la relation prédateur-proie sont exprimés selon la même échelle de temps,
- les paramètres des équations mises en jeu sont dépendants de la nature des prédateurs et des proies alors que les mécanismes décrits sont pour leur part indépendants,
- ...

On pourrait poursuivre l'énumération mais intéréssons nous à ces trois hypothèses et à leurs conséquences. Les deux premières hytpothèses sont liées à la représentation de l'espace et du temps que le modélisateur se fait. L'espace n'est pas représenté dans la plupart des modèles de dynamique de populations pour la simple raison de l'accroissement de la complexité de résolution de tels systèmes d'équations. Cette contrainte implique alors des hypothèses fortes sur la nature de l'espace. De même, le temps est un paramètre important dans les modèles dynamiques au sens large. Il est donc naturel d'intégrer des hypothèses relatives au temps. De plus, dans la perspective d'un couplage avec d'autres modèles ayant leur propre point de vue sur le temps, il est important d'être capable d'expliciter son point de vue sur la question. Dernier point : les paramètres des équations. Lorsque l'on construit un modèle mathématique ou non, le modèle fait apparaître des paramètres. Ces derniers sont, de manière générale, la résultante d'hypothèses sur les entités du système : la vitesse moyenne de capture des proies par les prédateurs ou le taux de conversion d'une proie en prédateur[29]. La vitesse moyenne de capture est un exemple typique. Le fait d'utiliser une valeur moyenne signifie que le modélisateur ne considère pas les variabilités individuelles et que cette valeur moyenne est suffisante pour son modèle ce qui est vrai d'un certain point de vue et dans certaines hypothèses (quelques exemples sont développés dans le chapitre suivant).

On vient de montrer rapidement que les hypothèses sous-jacentes d'un modèle sont primordiales pour comprendre pleinement un modèle. De plus, si on s'inscrit dans une perspective de couplage de modèles, il est nécessaire que les modèles à coupler soient compatibles du point de vue des hypothèses adoptées dans chacun d'eux.

La dernière couche de notre *framework* va donc s'intéresser à spécifier les éléments liés aux hypothèses. Dans l'état actuel de la réflexion, on distingue quatre types d'hypothèses (les "plus simples" à exprimer) :

- 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.

Les deux premiers types ont déjà l'objet d'une illustration. Néanmoins, il faut ajouter que ces deux notions sont fondamentales et qu'une réflexion dans le cadre du projet MIMOSA a été engagé pour définir les différents types de temps que l'on est amené à manipuler. Nous y reviendrons un peu plus tard dans ce chapitre.
Les deux autres types sont liés à la représentativité des variables et des entités du modèle. Lors de la construction d'un modèle, les premiers éléments que le modélisateur définit sont les variables du modèle. Ces variables sont en quelques sortes les fondements du modèle : c'est elles qui définissent l'orientation du modèle. Au travers de ce choix, le modélisateur fixe le domaine d'application de son modèle. Il devient alors intéressant de pouvoir spécifier les variables du modèle ce qui est fait dans la majorité des formalismes mais aussi de spécifier la nature de ces

[29]Lorsqu'un prédateur capture une proie, on considère que cette proie participe à la croissance de la population de prédateurs. Le taux de conversion est la part d'une proie dans cette croissance.

variables vis à vis du temps, de l'espace, du système d'unités choisi et vis à vis des entités du système. Ce dernier aspect est important et permet alors de définir une ontologie du domaine auquel le système appartient et de s'y référer.
La notion d'ontologie n'est pas nouvelle mais elle nous paraît un élément intéressant à intégrer dans notre *framework* afin d'apporter un niveau d'abstraction supplémentaire.

2.2 VLE : Virtual Laboratory Environment

Les travaux de recherche qui ont été menés et qui seront menés sont dirigés vers un seul objectif : proposer un environnement de spécification de modèles et d'expériences, d'exécution de simulations et de traitements statistiques. Il existe une multitude de plate-forme plus ou moins génériques, plus ou moins ouvertes et plus ou moins accessibles. Citons quelques exemples d'outils relatifs à la démarche de modélisation agents qui est notre domaine de prédilection : Swarm [MIN96], Cormas [BOU98], MadKit [GUT00], StartLogo [RES96]... L'ensemble de ces outils proposent à la fois un outil ou un ensemble de fonctionnalités sous forme de bibliothèques mettant en oeuvre un paradigme (en l'occurrence le paradigme multi-agents) et un environnement d'exécution ou de simulation. Ces plate-formes répondent à un besoin celui de mettre en oeuvre un paradigme. Pour les équations différentielles, on peut citer Mathematica ou Stella (outil des biologistes). Toutes ces plate-formes proposent leur propre langage de description des modèles, leur propre moteur de simulation et leur propre environnement d'exécution. Mais comment faire communiquer tout cet ensemble ? Nous avons répondu en partie à cette question dans la partie précédente et nous pouvons résumer notre position par la proposition suivante : la hiérarchisation par niveaux d'abstraction. Certains outils peuvent sans trop de difficulté intégrer cette approche mais ce n'est pas le cas de tous et cela dépend du niveau d'ouverture de la plate-forme.

L'idée que l'on défend est donc celle du laboratoire virtuel : offrir un environnement de travail pour réaliser des expériences via l'outil informatique. L'idéal serait de disposer d'une paillasse virtuelle de chimie, de physique ou de biologie pour étudier une réaction chimique, le comportement d'un organisme, ... Il faut que le scientifique ait l'impression d'être "devant" l'objet qu'il étudie. L'objet est en réalité représenté par un ou plusieurs modèles plus ou moins complexes. Le laboratoire virtuel doit surtout permettre de définir proprement les modèles, les protocoles expérimentaux et les protocoles d'analyse des résultats.

L'étude des plate-formes existantes montre clairement que l'expérimentateur et le modélisateur sont fortemment contraint par l'outil ce qui peut paraître normal lorsque l'on se situe dans une perspective de conception et de validation de modèles. En effet, si on part de rien, pourquoi ne pas utiliser un outil qui répond *a priori* aux besoins du modèle. Si on prend pour exemple Cormas, il répond parfaitement à une majorité de problèmes de modélisation d'écosystèmes. Il dispose d'outils de représentation de l'espace, de description des entités du système et de leur dynamique ... En revanche, si le modélisateur dispose de modèles bien définis et validés qui ne sont pas directement accessibles par la plateforme choisie alors il y a nécessité de transposer son modèle dans la plateforme ce qui n'est pas toujours simple.

Nous avons donc orienté notre réflexion vers la possibilité d'intégrer des modèles hétérogènes pour la simulation de systèmes complexes. L'hétérogénéité peut apparaître à plusieurs niveaux : hétérogénéité des points de vue sur un système donné, des formalismes utilisés, des langages d'implémentation, etc ... Notre but est de proposer une plateforme d'intégration qui permet

une utilisation simple des modèles. Plus précisément, une plateforme de simulation offrant les outils nécessaires à la conception de modèles, l'exécution et l'analyse des résultats de simulation. Nous posons comme postulat de départ la pré-existance de modèles sans exclure la possiblité de concevoir des modèles *ex nihilo*. Pourquoi un tel postulat ? En fait, il y a deux attitudes possibles :

- définir une plateforme de modélisation la plus générique possible qui permet de répondre à une certaine classe de systèmes. C'est l'attitude prise par la plupart des développeurs de plateformes de simulations.
- utiliser des modèles plus spécifiques et permettre leur utilisation combiné (les coupler) pour permettre la simulation de systèmes composites.

Cette deuxième attitude nous paraît intéressante dans le sens où elle autorise la cohabitation au sein d'une même simulation de plusieurs points de vue et/ou formalismes. De plus, nous pourrons autoriser dans le futur l'utilisation de plateformes plus génériques qui permettent une réelle modélisation (au travers de métalangages, par exemple). Nous ne rentrerons pas dans les détails ici de l'intérêt d'une telle pratique.

Virtual Laboratory Environment (VLE) est une plate-forme informatique qui doit offrir les fonctionnalités nécessaires pour la conception, la mise en oeuvre et l'analyse au travers de plans d'expériences structurés et de traitements statistiques de modèles monolithiques et agrégés. Nous entendons donc ici par conception (pour l'instant) le couplage de modèles préexistants pour former des modèles plus complexes. De plus, cette plate-forme doit être accessible via Internet pour permettre une utilisation distante, multi-utilisateurs et sans installation de logiciels spécifiques.

Une telle plate-forme informatique soulève de multiples questions sémantiques, conceptuelles, mathématiques ou opérationnelles, lesquelles sont du domaine de la recherche. Ces questions ont été abordés dans la partie précédente et nous allons présenter les solutions qui ont été choisi pour les différents niveaux du *framework*. Afin de ne pas alourdir le discours, nous allons surtout nous concentrer sur le coeur du *framework* : le langage de spécification de modèles et d'expériences. Les autres niveaux ont fait principalement l'objet d'études techniques que l'on peut retrouver dans les rapports de projets de fin d'étude d'étudiants. Nous concluerons cette partie en présentant les grandes lignes de l'architecture globale de VLE ainsi que des modules actuels disponibles.

2.2.1 Les modèles

Pour la construction du langage de spécifications des modèles, nous nous sommes largement inspirés de la structure DEVS définie par Zeigler [ZEI99] et nous y avons intégré, pour l'instant, une partie des réflexions du groupe de travail Mimosa [30]. Nous reprenons donc les principales définitions apportées par Zeigler, comme les notions de ports d'entrée et de sortie auxquels nous ajoutons des éléments de définition sur les données attachées aux événements. Deux types de ports ont été ajoutés afin de distinguer des événements particuliers : les événements liés à l'initialisation du modèle et les événements liés aux changements d'états. Les ports d'initialisation formalisent les événements dont l'occurrence a lieu seulement à $t = 0$. Si on se réfère à un vocabulaire plus classique, on cherche à différencier les paramètres et les variables du modèle. Les paramètres d'un modèle sont fixés en début de cycle de vie du modèle et ne changent pas de valeur. Ce sont des sortes de constantes. Lors de l'étude d'un modèle, on considérera que deux modèles sont différents à partir du moment où l'un des paramètres du modèle change. Un

[30] Site Web : http ://www-lil.univ-littoral.fr/Mimosa

modèle et ses paramètres forment un tout. En revanche, la notion de variables est à inclure dans la notion de vecteur d'états définie dans une structure DEVS. Les ports d'états sont définis à cet effet. On attache un port d'états à chaque variable d'états que le modélisateur autorisera l'accès ou plus précisement que le programmeur du modèle donnera accès logiciellement. On verra plus tard que ces ports serviront à effectuer les mesures sur le modèle. Les ports d'états ne font pas du tout partie du formalisme DEVS bien au contraire. En effet, ces ports sont des objets purement liés à des problèmes opérationnels. L'expérimentateur a besoin de connaître l'état de son système et donc afin de rester homogène dans le vocabulaire et la notation, nous avons choisi d'appeler ces points d'observation de la dynamique des modèles des ports.

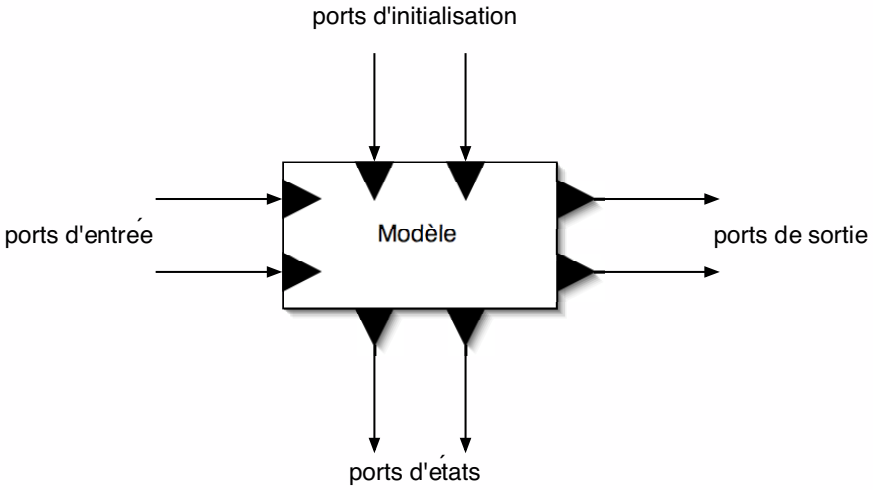


FIG. 2.18 – Représentation graphique d'un modèle

Zeigler dans sa hiérarchisation de la spécification des systèmes a défini cinq niveaux [ZEI99] que nous résumons comme suit :

- le schéma d'observation qui met en relation la base de temps et les entrées-sorties observées,
- la relation d'entrée-sortie qui détermine la relation entre une entrée, une sortie et un intervalle de temps appelé segment,
- la fonction d'entrée-sortie qui introduit la notion d'état initial et établit la relation entre entrée, sortie et segment,
- les systèmes dynamiques qui spécifient la dynamique interne des systèmes,
- et les systèmes couplés.

La spécification de structure DEVS et DEVS couplé s'inscrivent dans les deux derniers niveaux.

Nous rappelons que notre objectif est triple : coupler des modèles, prendre en compte des modèles existants et intégrer les concepts agents. Si on se référe à la hiérarchie de Zeigler, on s'aperçoit que la réponse à notre première exigence s'inscrit dans le niveau 4. Le niveau "Systèmes couplés" impliquent la spécification des connexions reliant les différents modèles. Mais quel est le type de ces connexions ? En DEVS, les connexions relient deux ports (un port de sortie et un port d'entrée) et les modèles connectés échangent des événements. Les événements modélisent les changements d'états internes du modèle qui génére l'événement. Le modèle qui

reçoit l'événement change d'état. Les changements d'états sont décrits dans les fonctions de transition externe dans le cas d'un événement provenant d'un autre modèle. La spécification DEVS d'un modèle atomique intégre la définition des états et des fonctions de transitions. Nous est-il possible de le faire pour notre approche des modèles ? Une multitude d'articles ([ZEI99], [ZEI00a], [KOF01], [BOL02] et [GIA00]) montrent qu'une majorité de formalismes peut être encapsulé dans DEVS. Le problème n'est plus que de construire une représentation DEVS du modèle que l'on désire coupler.

A l'heure actuelle, le langage de spécification d'un modèle est donc une adaptation du formalisme DEVS. Pourquoi ? Comme il a été montré, le formalisme DEVS permet de spécifier la dynamique des systèmes et surtout d'offrir une vision unifiée des systèmes et par conséquent, d'offrir la possibilité de coupler différents modèles. Notre langage est une adaptation parce que notre objectif est clairement la spécification de modèles de systèmes spatio-temporels où aucun *a priori* n'est fait sur le formalisme utilisé (équations différentielles, automates cellulaires, modèles centré individus, réseaux de Petri, ...). La notion de temps est explicitement définie dans le formalisme DEVS et l'encapsulation de certains formalismes est disponible. En revanche, l'espace ne fait pas partie intégrante de DEVS sauf dans son extension Cell-DEVS [WAI01]. De plus, étant très attachés à l'approche centrée individus, il est donc nécessaire d'étudier son encapsulation dans DEVS. Tout cela a donné naissance au langage MLMC (Markup Language for Model Coupling) pour la partie spécification des modèles. Ce travail s'inscrit pleinement dans la thèse de Raphaël Duboz.

Après avoir présenté les objectifs et les contraintes que l'on s'est fixé et les concepts fondamentaux de notre approche de la spécification de modèles, nous allons présenter avec exactitude la syntaxe XML utilisée pour la définition des modèles atomiques et couplés. Cette partie sera l'occasion de faire le point concrètement sur ce qu'est un modèle et quelle est la logique à suivre pour définir un modèle.

Cette partie a pour objectif de définir avec exactitude la syntaxe XML utilisée pour la définition d'un modèle atomique puis couplé. La définition d'un modèle est divisée en deux parties. Une première partie, purement formelle, liée au formalisme DEVS et une seconde liée aux aspects informatiques (localisation des modèles, par exemple).

2.2.1.1 Les modèles atomiques

La définition d'un modèle atomique (ou couplé) est encapsulé dans la balise <MODEL>. L'attribut TYPE permet d'indiquer si le modèle est atomique ou couplé. Tout modèle peut être qualifié d'autonome. Un modèle est autonome s'il existe une entité informatique (un composant, par exemple) qui le met en oeuvre. Le fait qu'un modèle ne soit pas autonome signifie qu'il ne pourra pas être réutilisé dans un autre modèle couplé. Plus conceptuellement, un modèle couplé composé de modèles non autonomes signifie que la description qu'il en est fait est une vue conceptuelle. Le modélisateur a décidé de faire une description hiérarchique de son modèle mais de réaliser une mise en oeuvre informatique monolithique. Les sous-modèles n'existent pas de manière autonome pour être réutilisés. Les deux derniers paramètres de la balise MODEL identifient :

– la localisation du modèle ou plutôt la localisation du composant informatique mettant en oeuvre le simulateur ; cette location peut être locale ou distante d'où l'utilisation de la

notion d'URL définie par le W3C[31].
- le type de protocole offert par le module pour que le noyau de simulation puisse le charger et communiquer.

Les paramètres URL et PROTOCOL n'ont pas lieu d'être si le modèle n'est pas autonome. Nous ne détaillerons pas la balise PROTOCOL. Nous tenons juste à ajouter qu'elle permet de faire le lien avec la couche opérationnelle du *framework*.

```
<MODEL NAME="Copepod" TYPE="atomic|coupled" USEALONE="yes|no" URL="location"
       PROTOCOL="JavaClass|JavaRMI|DynamicLibrary|SOAP|...">
<DESCRIPTION>...</DESCRIPTION>
<TIME>...</TIME>
<SPACE>...</SPACE>
        ...
</MODEL>
```

La balise DESCRIPTION délimite une zone de texte libre pour la description informelle d'un modèle. Elle permettra de renseigner l'utilisateur sur la nature exacte du modèle (le système modélisé, le formalisme utilisé, les hypothèses prisent, les conditions d'utilisation, ...). Deux autres balises sont disponibles dans l'entête d'un modèle : la balise TIME et la balise SPACE. Ces deux balises permettent de définir le référentiel spatial et temporel adopté par le modèle. La balise SPACE n'est pas obligatoire si le modèle ne fait pas intervenir la notion d'espace. Le détail de ces balises est présenté par la suite.

Du point de vue macroscopique, un modèle est vu comme un ensemble de ports d'entrée et de ports de sortie. Ces ports sont les récepteurs et les émetteurs des événements estampillés (i.e. un événement se produit à une date). Cette notion est totalement en accord avec le formalisme DEVS. Nous allons ajouter aux ports d'entrée et de sortie des ports d'initialisation (ou de paramétrage) et des ports d'états. Les ports d'initialisation ont pour objectif de fixer les conditions initiales du système représenté par le modèle. Ces ports sont renseignés au début du cycle de vie du modèle. La notion d'événement n'est donc pas primordiale mais pour rester homogène nous les avons qualifié de ports. Ce choix est une manière de spécifier l'état initial du modèle. On aurait aussi pu choisir de définir la valeur prise par le vecteur d'état à $t = 0$. Dans ce cas, il aurait fallu spécifier le vecteur d'états. La non spécification du vecteur d'états s'inscrit dans notre volonté de considérer un modèle dans une boîte noire dont l'interface est compatible DEVS. On ne s'interdit pas dans le futur d'intégrer la spécification du vecteur d'états et des fonctions de transition. Pour l'instant, on n'en voit pas l'utilité.
Les ports d'états émettent un événement dès lors qu'une variable d'états du modèle change (par transition interne).

Syntaxiquement, nous retrouvons 4 balises contenues dans la balise <MODEL> permettant d'exprimer les différents ports.

```
<MODEL ...>
  ...
  <INIT>
    <PORT NAME=""> ... </PORT>
  </INIT>
```

[31]Site Web : http ://www.w3c.org

```
  <IN>
    <PORT NAME=""> ... </PORT>
  </IN>
  <OUT>
    <PORT NAME=""> ... </PORT>
  </OUT>
  <STATE>
    <PORT NAME=""> ... </PORT>
  </STATE>
  ...
</MODEL>
```

Chaque port, quelle que soit sa nature (initialisation, entrée, sortie ou variable d'état) est susceptible de recevoir ou d'émettre des événements portant des informations structurées. Une partie de nos travaux s'est penchée sur la spécification des informations transportées par les événements. C'est un élément important dans la gestion du couplage de modèles. Si un modèle émet un évenement portant une information quantitative (concentration de poissons par unité de volume, par exemple) et que cet événement soit déclencheur d'un changement d'états dans un autre modèle, il est important de vérifier que le deuxième modèle s'attend bien à recevoir cette information quantitative et non une autre (le nombre total de poissons du milieu, par exemple). Cette vision des choses doit permettre de valider la cohérence des couplages en terme d'événements et de données échangées. Il est alors possible de mettre en jeu des stratégies de conversion. Prenons un exemple simple concernant nos poissons. Le premier modèle manipule un nombre de poissons plongés dans un espace bien défini (dimension, taille, ...). Si le deuxième modèle a besoin d'une concentration, rien de plus simple. Il suffit de diviser le nombre de poissons par le volume total de l'espace en faisant bien attention aux unités. Ce type de démarche n'est possible qu'à une condition : les données et les modèles doivent possèder les informations nécessaires. C'est tout l'objectif de la balise <DATA>. Les informations sont définies dans la balise <DATA>. Si les données attachées aux ports sont complexes et sémantiquement dissociables alors la partie <DATA> est répétée. Par exemple, un nombre de poissons et la hauteur des vagues sont dissociables sémantiquement. En revanche, l'âge moyen, l'âge minimal et l'âge maximal des poissons sont associables. Dans le cas de données associables, on peut définir des agrégats. Un agrégat possède alors la même partie sémantique.

```
<PORT ...>
  <DATA NAME="">
    <METADATA>...</METADATA>
    <UNIT ... />
    <TYPE ... />
    <CONTENT>...
  </DATA>
  <DATA NAME=""> ... </DATA>
  ...
</PORT>

<PORT ...>
  <DATA NAME="">
    <METADATA>...</METADATA>
    <TIME-REF ... />
```

```
    <SPACE-REF ... />
        <AGGREGAT NAME="">
      <UNIT>...
      <TYPE>...
      <CONTENT>...
        </AGGREGAT>
        <AGGREGAT NAME=""> ... </AGGREGAT>
        ...
  </DATA>
</PORT ...>
```

Le typage des données attachées aux événements est accessible en partie par les balises <METADATA>, <UNIT> et <TYPE>. Trois niveaux de typage sont accessibles : un niveau sémantique, un niveau unitaire (l'unité de la donnée) et un niveau informatique. Le typage sémantique offre la possibilité d'adjoindre du sens à une donnée. La donnée représente plus que sa représentation informatique ou son unité aux yeux du modélisateur. L'information sémantique est de deux ordres : le sens propre de la donnée (une distance, une concentration, un nombre d'entités, ...) et sa relation avec le temps et l'espace. Par exemple, si en entrée d'un modèle, un événement "transporte" une information de type "âge moyen des entités", cette quantité peut représenter diverses choses. Dans ce cas, il devient intéressant d'accompagner la donnée d'une information sémantique : l'âge moyen de quoi ? quelle unité de temps est employée ? ... Dans notre exemple, l'âge moyen peut être exprimé en jours et représenter l'âge moyen d'une population de thons d'une certaine famille. Il paraît évident que si on injecte cette information dans un modèle de dynamique de populations construit spécifiquement pour les crevettes, il y a un problème conceptuel. En revanche, on pourra injecter l'information dans un modèle de dynamique de populations plus général qui ne fait aucun *a priori* sur la nature des individus de la population. Ce type de modèles existe : le modèle Lokta-Voltera en est un exemple [LOK25][VOL26]. Ces informations forment la base minimale pour la mise en oeuvre de la couche sémantique conformément au *framework*.
Décryptons la balise <METADATA>. Elle spécifie tous les éléments liés à la sémantique de la donnée : la nature discrète ou continue, la référence temporelle et spatiale, le sens de la donnée et l'objet sur lequel s'applique la donnée. Une donnée peut être discrète ou continue c'est à dire que l'ensemble des valeurs prises par cette donnée est discret ou continu. La spécification du temps et de l'espace est facultative. La balise <SEMANTICS> est quant à elle obligatoire.
Une donnée est dite spatialisée si elle est indéxée par l'espace. Dans le cas d'un espace discret, par exemple, la donnée sera probablement représentée par une matrice où les indices de la matrice correspondront aux indices des cellules de l'espace. Si une donnée est spatialisée, la balise <SPACE> est alors présente. Pour simplifier, il sera possible de faire référence à l'espace descrit au niveau du modèle [32].
Une donnée est dite temporelle si elle est indexée par le temps. La donnée est alors une série de valeurs estampillées par le temps. La balise <TIME> se chargera de la définition du temps. Comme pour l'espace, on pourra faire référence au temps défini au niveau du modèle.

```
<METADATA TYPE="discrete||continous">
  <TIME> ... <TIME />
```

[32] <SPACE MODEL="yes|no" ...> ... </SPACE>

```
  <SPACE> ... <SPACE />
  <SEMANTICS CLASS="" URL="">
        <OBJECT CLASS="x" />
  </SEMANTICS>
</METADATA>
```

Le champ CLASS de la balise SEMANTICS spécifie la classe d'appartenance sémantique de l'information. L'idée originelle a été présenté dans [DUB02d] et elle a été développé plus en avant depuis. On peut citer comme classe : durée, cardinalité, concentration, masse volumique, intensité, volume, aire, vitesse, accélération... La balise <OBJECT> permet de préciser l'objet sur lequel la donnée porte. De manière générale, une donnée est synonyme de mesure sur un objet. Cette remarque est fondamentale. Notre idée est d'offrir la possibilité de spécifier la nature des objets manipulés dans le modèle afin de pouvoir coupler divers modèles. La spécification des objets peut aller plus loin en définissant des hiérarchies. Dans ce cas, un modèle mettant en jeu des objets d'une classe A est "couplable" à un autre modèle B dont les objets ont une classe d'appartenance hiérarchiquement supérieure. Prenons un exemple. On développe un modèle de dynamique de populations où l'objet d'étude est une entité sans aucune autre précision. D'autre part, on développe un modèle d'ingestion (vitesse de capture des proies par un prédateur) qui s'applique spécifiquement sur une espèce. Les informations qui sont susceptibles d'être partagées entre les deux modèles sont compatibles car il existe une relation hiérarchique (d'héritage) entre les entités du premier modèle et celles du second. L'attribut URL de la balise <SEMANTICS> permet de référencer le fichier XML contenant la définition de la hiérarchie de classes. Une syntaxe simple nous permet de décrire les relations d'héritage d'un graphe d'héritage.

```
<CLASS NAME="x">
        <INHERITANCE CLASS="x" />...
        <AGREGATION CLASS="x" CARDINALITY="y">
</CLASS>
```

La notion d'agrégation à ajouter à la notion de classe permet d'indiquer que l'instance d'une classe A est équivalente à un ensemble d'instances d'une autre classe B. Par exemple, une population est un ensemble d'individus. La définition des classes et de leurs relations est placée dans un fichier XML externe du fichier de spécification d'un modèle.

La balise <METADATA> permet de spécifier la partie sémantique des données ce qui permet de la situer dans un contexte particulier du domaine de modélisation considéré (les écosystèmes, par exemple). Dans la perspective de validation du couplage des modèles au niveau sémantique, il paraît nécessaire de définir des ontologies propres au domaine de modélisation. Si on reste dans le domaine des écosystèmes et en particulier des écosystèmes marins, il est possible de construire rapidement un début d'ontologie (voir Figure 2.19 pour un exemple).

D'autre part, nous avons à notre disposition les travaux de taxomonie qui classifient toute entité (animale, végétale, ...) d'un écosystème et qui nous permettent de construire facilement des hiérarchies (un exemple avec les petites bêtes qui seront présentées dans la dernière partie de ce chapitre : les copépodes[33] - voir Figure 2.20). Cette classification peut être rattachée à la classe *Individu* de l'extrait d'ontologie.

[33]Les copépodes forment une sous-classe dans laquelle est définie environ 210 familles, 2280 genres et plus de 14000 espèces.

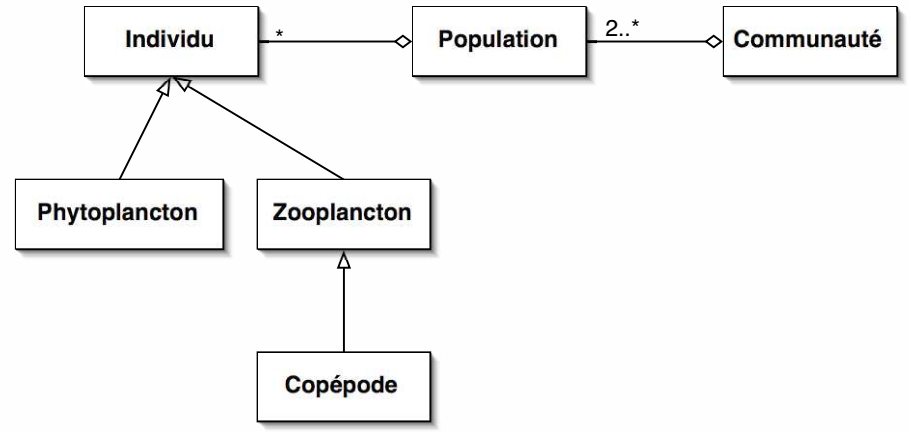


FIG. 2.19 – Extrait d'un exemple d'ontologie pour les écosystèmes

La structure de taxonomie [34] nous sert ici de méta-modèle (voir Figure 2.21).

Ce travail est à mener au sein de chaque domaine tout en cherchant à fédérer le vocabulaire de domaines proches. Une ontologie doit être dynamique et restée ouverte aux modifications. L'approche adoptée avec XML rempli parfaitement cet objectif.

Après la partie sémantique, nous allons nous intéresser à des aspects beaucoup plus simples. Le typage unitaire s'exprime dans le système MKSA (Mètre, Kilogramme, Seconde, Ampère). Il faut ajouter au système MKSA quelques autres unités pour être totalement complet : sans unité, ind. (individu, entité, particule) et lux. Une donnée peut être sans unité (ou sans dimension) ou être une cardinalité (un nombre d'entités). La syntaxe adoptée est simple : le typage unitaire est obtenu par le produit de plusieurs unités élémentaires élevées à une puissance.

```
<UNIT>
  <TERM CLASS="meter|kilogram|second|..." POWER="x">
  ...
</UNIT>
```

Le troisième niveau de typage spécifie le type informatique utilisé pour le codage de la donnée mais aussi le domaine de définition de la donnée ($[0, +\infty[$ ou $]0, 1] \cup]5, +\infty[$, par exemple) et le caractère discret ou continu. Ces informations sont indispensables et sont disponibles dans la balise <DATA>. Quatre types de base sont offerts (entier, réel, booléen et chaîne de caractères). Ces types de base peuvent être utilisés pour construire des structures dimensionnées (tableau, matrice, matrice 3D, ...). La définition d'un domaine de définition est représentée par l'union d'intervalles ouverts ou fermés. La balise <DOMAIN> n'est pas obligatoire. Si elle n'est pas présente alors le domaine est déduit du type informatique (integer = Z et real = R). Si la donnée est de type entier ou réel, elle peut être aussi discrète ou continue. Si elle est discrète, la spécification

[34] règne - embranchement - sous-embranchement - classe - sous-classe - ordre - sous-ordre - infra-ordre - super-famille - famille - sous-famille - tribe - genre - espèce - variété

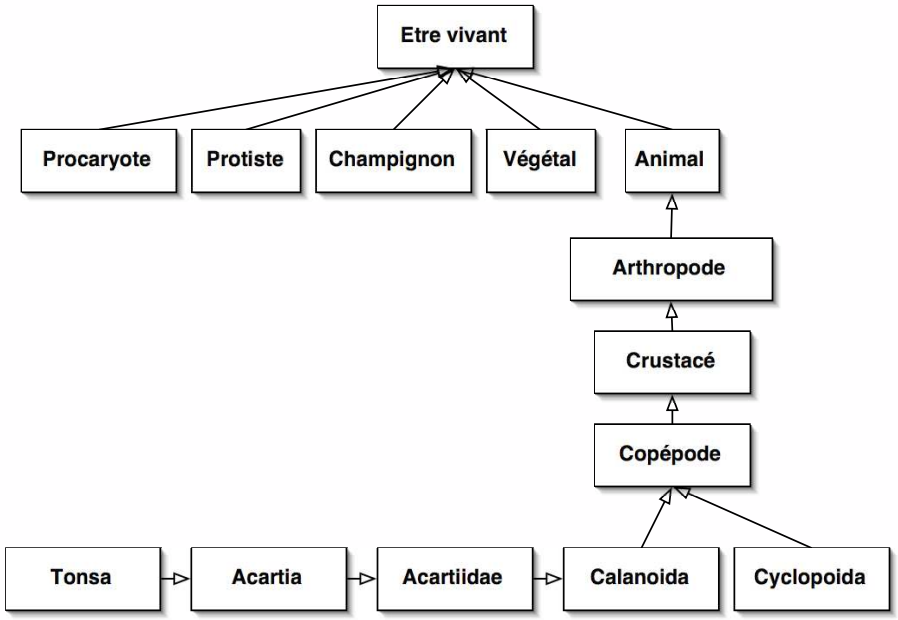


FIG. 2.20 – Taxonomie du Copépode

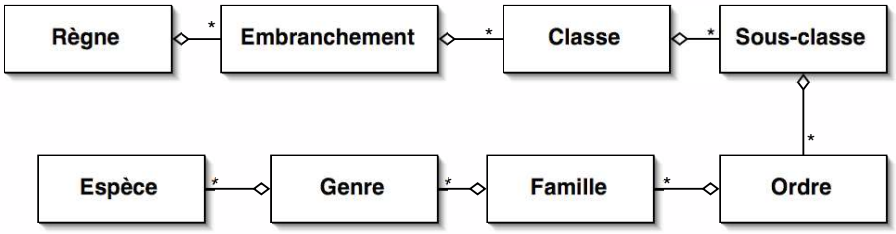


FIG. 2.21 – Une structure simplifiée de taxonomie

du pas de discrétisation est obligatoire. Par définition, si la donnée est de type entier, elle est discrète mais avec un pas de discrétisation de 1. On peut alors modifier ce pas avec l'attribut STEP.

```
<TYPE CLASS="integer|real|boolean|string" DISCRETE="yes|no" STEP="n"
      DIMENSION="0|1|2|3|..." SIZE= SIZE=(cst|*[,cst|*,...])>
<DOMAIN>
<INTERVAL LOWER-BOUND="n|-infinity" TYPE-LOWER-BOUND="open|close"
          UPPER-BOUND="m|+infinity" TYPE-UPPER-BOUND="open|close" />
...
</DOMAIN>
</TYPE>
```

Afin d'aller plus loin dans les possibilités de transport de données par les événements, il est envisager de définir des fonctions. Conceptuellement, deux approches sont possibles : une fonction est un modèle au même titre que le modèle qui désire gérer la fonction ou le modèle

gère une fonction totalement inconnue. Si la fonction est vue comme un modèle, il est probable que le concepteur du modèle sache spécifier totalement la fonction à gèrer en incluant des paramètres. Le modèle ne fait alors que paramètrer la fonction. Nous somme en présence d'un couplage classique : les sorties d'un modèle sont connectés aux entrées du modèle représentant la fonction en question. Les évenements d'entrée transportent les valeurs des paramètres de la fonction. En revanche, si la fonction est totalement inconnue, il est impossible de construire un modèle paramétrique de la fonction. Dans ce cas, un événement peut transporter une fonction. Seul le nombre de variables de la fonction et leurs domaines de variations peuvent être spécifié ($f(X,Y) : R \times R \rightarrow R$, par exemple). La syntaxe des types informatiques est étendue. La notion d'unité et de sémantique ne sont pas concernées. L'unité d'une fonction est tout simplement l'unité des évaluations de la fonction.

```
<TYPE CLASS="integer|real|boolean|string|function" DISCRETE="yes|no" STEP="n"
      DIMENSION="0|1|2|3|..." SIZE= SIZE=(cst|*[,cst|*,...])>
<DOMAIN> ... </DOMAIN>
<VARIABLE NAME="" CLASS="integer|real|boolean" DISCRETE="yes|no" STEP="n">
<DOMAIN> ... </DOMAIN>
</VARIABLE>
...
</TYPE>
```

Dans le cas du port d'initialisation, il est possible de spécifier directement dans le modèle une valeur par défaut par la balise <CONTENT> de la balise <DATA>. La structure de <CONTENT> est fonction du type de la donnée. Dans le cas d'une donnée simple (de dimension 0), la valeur est indiquée entre la marque de début et de fin de la balise <CONTENT>. Dans le cas d'une donnée dimensionnée (vecteur, matrice...), un ensemble de balises est nécessaire pour structurer les données. Si la donnée est agrégée alors la balise <AGREGAT> est utilisée pour séparer les différents champs de la donnée.

```
<DATA>
  <TYPE CLASS="real">
  <CONTENT>2.5</CONTENT>
</DATA>
```

```
<DATA>
  <TYPE CLASS="real" DIMENSION="2" SIZE="(2,2)">
  <CONTENT>
    <L2>
      <L1><L0>3.5</L0><L0>-8.1</L0></L1>
      <L1><L0>0.9</L0><L0>7.2</L0></L1>
        </L2>
  </CONTENT>
</DATA>
```

2.2.1.2 La dynamique

La syntaxe qui a été jusqu'ici développée propose une description de niveau 0 (cf. Hiérarchisation des spécifications de systèmes de Zeigler), une intégration des aspects spatio-temporelles et

une description avancée des données attachées aux modèles. La spécification DEVS va plus loin en intégrant la description des états du système, les fonctions de transition interne et externe, la fonction de sortie et la fonction d'avancement du temps. La question que l'on doit se poser ici est : quel est l'intérêt de spécifier formellement la dynamique du système ? La réponse dépend naturellement de l'objectif que l'on se fixe : soit on désire formaliser la dynamique du système afin de communiquer la modélisation de son système sans ambiguïté, soit on ne cherche qu'à produire un outil de spécification du couplage et de vérification de la cohérence des couplages.

Dans le premier cas, on est confronté à plusieurs problèmes qui peuvent se résumer avec la question suivante : comment spécifier formellement la dynamique d'un système quel que soit le paradigme ou le formalisme utilisé ? On rappelle que l'on ne fait aucun *a priori* sur le modèle que l'on cherche à utiliser. Dans un certain nombre de cas, la réponse existe. Si le paradigme est l'automate cellulaire alors il existe Cell-DEVS [WAI01]. Si le modèle est décrit à l'aide d'automates à états finis ou de réseaux de Petri, la spécification est quasi-immédiate. En revanche, qu'en est-il de paradigme moins structuré tel que les modéles centrés individus ? Une réponse a été présentée dans le chapitre 1 avec Agent-DEVS. Il reste donc à spécifier la syntaxe XML de la description de la dynamique.

Avant de décrire cette syntaxe, un mot sur le deuxième cas. Si l'on cherche à développer un outil de simulation de modèles couplés sans s'intéresser à la spécification formelle de la dynamique des modèles composants le système alors il n'y a rien à ajouter à la syntaxe XML proposée jusqu'ici. Ce que l'on peut alors regretter c'est que l'on n'a pas répondu à l'objectif d'amélioration de communication de nos modèles. Il restera toujours un flou sur la dynamique du système. Nous avons donc fait un double choix.

La dynamique d'un modèle est décrit autour de la balise DYNAMICS contenu dans la balise MODEL.

```
<MODEL ...>
  <DESCRIPTION>...</DESCRIPTION>
  <INIT> ... </INIT>
  <IN> ... </IN>
  <OUT> ... </OUT>
  <STATE> ... </STATE>
  <DYNAMICS> ... </DYNAMICS>
</MODEL>
```

Spécifier la dynamique est une bonne chose mais deux approches sont possibles. On résumera nos deux approches par les concepts de *mapping* et de *wrapping*. Le concept de *mapping* signifie que l'on cherche à trouver une correpondance totale entre le paradigme d'origine et DEVS. Prenons l'exemple des réseaux de Petri [PER77]. Jacques et Wainer propose dans [JAC02] le *mapping* des réseaux de Petri dans DEVS. Chaque élément du formalisme (place et transition) est spécifié totalement en DEVS et un réseau de Petri est alors un modèle couplé composé de modèles de place et de modèles de transition. On dispose alors d'un *framework* DEVS de spécification de réseaux de Petri. Dans le cas d'un *framework*, la structure DEVS n'est pas remise en cause (aucune extension n'est appliquée). DEVS est utilisé tel qu'il est pour spécifier un autre formalisme (ici, les réseaux de Petri).
La *wrapping* est une autre approche : on ne cherche pas spécifier tous les éléments du formalisme

mais on garantit que la dynamique du formalisme considéré est conceptuellement compatible DEVS. Du point de vue opérationnel, le module offrant la spécification et la simulation des réseaux de Petri devra répondre à une interface DEVS standardisée. Ce point a été développé dans la présentation du *framework* VLE. Le formalisme est mis dans une enveloppe DEVS mais les composants du formalisme ne sont spécifiés en DEVS.
On constate que les deux approches n'offrent pas les mêmes avantages. Le *mapping* pousse l'aspect formel jusqu'au bout : tout est formalisé en DEVS. La difficulté réside alors dans le fait de pouvoir définir une syntaxe claire, simple et complète. On s'aperçoit rapidement lorsque l'on réalise des modèles DEVS que la spécification des fonctions de transition est complexe. Il faut très rapidement utiliser des notations mathématiques complexes. Le langage a imaginé devient alors compliqué à mettre au point. De plus, l'objectif du "tout formel" est de faire des preuves ce qui n'est pour l'instant notre objectif. De son côté, *le wrapping* est beaucoup plus simple à spécifier puisque l'on ne spécifie rien de plus. Il suffit d'indiquer la nature du formalisme utilisé et de considèrer que le module en charge de la simulation est *DEVS-compliant*. En revanche, la spécification ne possède aucune indication formelle sur la dynamique.
Afin de trouver un compromis, nous avons tout de même décidé d'intégrer la balise <DYNAMICS> au modèle. Cette balise indique le formalisme utilisé afin de connaître le module à invoquer pour la prise en charge de la manipulation de la spécification et de la simulation. De plus, on autorise l'insertion de la spécification de la dynamique avec une syntaxe spécifique au formalisme mais toujours de type XML. La description de la dynamique peut se localiser dans un fichier externe afin de ne pas alourdir le fichier de description du modèle. Le troisième attribut de la balise <DYNAMICS> est l'ouverture vers une spécification de type *mapping*.

```
<DYNAMICS FORMALISM="FSA|petriNet|..." URL="" TYPE="wrapping|mapping">
...
</DYNAMICS>
```

Prenons l'exemple des réseaux de Petri évoqué auparavant. Un réseau de Petri est composé de places, de transitions, d'un réseau de connexions entre les places et les transitions et un marquage initial. Nous avons ajouté à cette définition les notions de places et transitions génératrices de jetons. Ces places et transitions font le lien avec les ports d'entrée du réseau. Lorsqu'un événement externe arrive sur un port d'entrée du réseau, il est interprété comme un ajout de jetons dans le réseau. Symétriquement, certaines transitions et places terminales sont associées à des ports de sortie. Lors de l'arrivée d'un jeton dans la place ou l'activation de la transition, un événement de sortie est généré sur le port associé (voir figure 2.22). Nous considérons ici un réseau de Petri simple, si l'on s'intéresse aux réseaux de Petri colorés, on peut imaginer que les informations attachées aux événements sont ceux portés par le jeton coloré.

```
<DYNAMICS FORMALISM="petriNet" TYPE="wrapping">
<PARAMETER EVOLUTION="byStep|whileAlive" TIME-STEP="n"/>
<PLACES>
<PLACE NAME="" TYPE="init|input|ouput|internal" PORT-NAME="modelPortName"
       TOKEN-NUMBER="n" />
...
</PLACES>
<TRANSITIONS>
<TRANSITION NAME="" TYPE="input|ouput|internal" PORT-NAME="modelPortName"
```

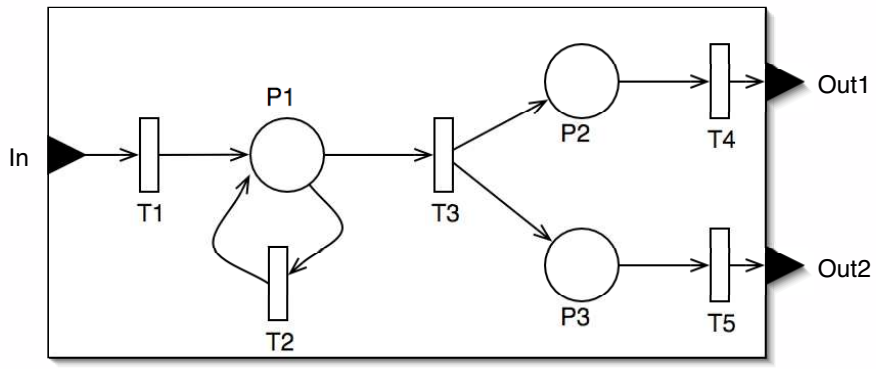


FIG. 2.22 – *Wrapping* d'un réseau de Petri

```
            TOKEN-NUMBER="n" />
...
</TRANSITIONS>
<CONNECTIONS>
<CONNECTION PLACE-NAME="" TRANSITION-NAME="" TYPE="pre|post"
            TOKEN-NUMBER="n" />
...
</CONNECTIONS>
<INITIAL-MARKING>
<MARKING PLACE-NAME="" TOKEN-NUMBER="n" />
...
</MARKING>
</DYNAMICS>
```

La définition d'un réseau de Petri (RdP) se compose de 5 balises (<PARAMETER>, <PLACE>, <TRANSITION>, <CONNECTIONS> et <INITIAL-MARKING>). Nous évoquerons la première balise par la suite. Les deux suivantes posent la définition des deux éléments de base d'un RdP les places et les transitions. L'attribut TYPE permet d'indiquer si la place ou la transition est associée à un port d'entrée (*input*) ou à un port de sortie (*output*) ou si elle est associée à rien (*internal*). Si la place ou la transition est associée à un port, une réference sur le port doit être indiquée. Le port en question est un port défini au niveau de modèle. Une connexion relie une place à une transition (*pre*) ou une transition à une place (*post*). Le nombre de jetons consommés ou produits par la transition figure au niveau de la connexion. La dernière balise ne fait qu'indiquer le nombre de jetons mis dans les places à l'état initial. Il faut noter que cet état initial peut être défini grâce aux ports d'initialisation du modèle. Il suffit d'attacher une place à un port d'initialisation du modèle (*init* de l'attribut TYPE de la balise PLACE).
Pour que l'exemple soit complet, il faut discuter des contraintes sur les algorithmes utilisés au sein du module de mise en oeuvre des réseaux de Petri afin de respecter les spécifications DEVS. Une structure DEVS est composée de quatre fonctions : la fonction de transition interne δ_{int}, la fonction de transition externe δ_{ext}, la fonction de sortie λ et la fonction d'avancement du temps ta. La fonction de transition interne se résume en algorithme de franchissement des transitions franchissables. Il faut juste prendre une précaution : il faut dans un premier temps calculer l'ensemble des transitions franchissables en fonction de l'état courant du réseau puis effectuer

uniquement le franchissement de cet ensemble. Si on ne prend pas cette précaution, l'algorithme s'interromprera seulement lorsque le réseau ne sera plus vivant [35]. On peut néanmoins imaginer que certains modèles désirent appliquer cette logique. Dans ce cas, il faut positionner l'attribut EVOLUTION dans la balise <PARAMETER> à *whileAlive*. La fonction de transition externe injecte des jetons en fonction des événements d'entrée et des associations entre les ports d'entrée et les places et transitions. Le nombre de jetons est fixé par le attribut TOKEN-NUMBER des balises <PLACE> et <TRANSITION> lorsque l'attribut TYPE est fixé à *input* ou *output* ou *init*. La fonction de sortie traduit les arrivées de jetons dans les places ou les franchissements de transitions en événements de sortie. La philosophie DEVS est encore ici respectée puisque la fonction de sortie n'est activée que sur des transitions internes. Le modélisateur n'ayant pas la possibilité de qualifier une place ou une transition en entrée et en sortie en même temps, il n'est alors pas possible d'activer la fonction de sortie sur une transition externe. Il reste plus que la fonction d'avancement du temps *ta*. Ce point est un peu plus déliquat. En effet, dans un cadre purement formel, un réseau de Petri ne manipule pas de temps (sauf dans le cas de réseaux de Petri temporisé). La solution de facilité est de fixer une durée associée à chaque itération de l'algorithme de franchissement quelle que soit la forme d'algorithme choisie (*byStep* ou *whileAlive*). L'attribut TIME-STEP de la balise <PARAMETER> est défini à cet effet. La valeur attribuée à ce pas de temps n'est pas à fixer à la légère. Il faut vérifier la cohérence temporelle du modèle global surtout en cas de couplage.

Le petit travail exposé ici n'est pas compliqué à réaliser pour une majorité de formalismes. On peut mettre à part les équations différentielles où la traduction en evénements de sortie doit passer par la définition de seuils (DEV&DESS [ZEI99]).
Le travail est en cours pour l'encapsulation de la première version de VLE qui proposait la modélisation et la simulation multi-agents (voir la partie Architecture). Ce travail n'est pas incompatible avec la construction du langage de spécification agent-DEVS comme nous l'avons montré précédemment. Agent-DEVS s'inscrit dans la logique du *mapping*.

2.2.1.3 Les modèles couplés

La déclaration d'un modèle couplé diffère syntaxiquement peu de la déclaration d'un modèle atomique. Seul l'attribut TYPE de la balise MODEL est positionné à *coupled* et une nouvelle balise CONNECTION fait son apparition. Cette dernière permet d'indiquer quels sont les ports connectés ensemble. Les ports en question peuvent être des ports des modèles composants le modèle couplé ou des ports du modèle couplé.

Le corps de la balise <MODEL> se décompose de la manière suivante :
- la déclaration des ports du modèle couplé,
- la définition complète des sous-modèles ou la référence à un modèle défini auparavant,
- la déclaration des connexions.

```
<MODEL NAME="N" TYPE="coupled">
<INIT> .. </INIT>
<IN> ... </IN>
<OUT> ... </OUT>
<STATE> ... </STATE>
<SUBMODELS>
```

[35] Un RdP est dit vivant s'il existe au moins une transition franchissable.

```
        <MODEL NAME="A" TYPE="atomic"> ... </MODEL>
        <MODEL NAME="B" TYPE="atomic"> ... </MODEL>
</SUBMODELS>
<CONNECTIONS>
        <CONNECTION> ... </CONNECTION>
        ...
</CONNECTIONS>
</MODEL>
```

Il existe quatre formes de connexion selon la nature du port :
- la connexion d'un port de sortie d'un sous-modèle à un port d'entrée d'un autre sous-modèle,
- la connexion d'un port d'entrée du modèle couplé à un port d'entrée d'un sous-modèle,
- la connexion d'un port de sortie d'un sous-modèle à un port de sortie du modèle couplé,
- la connexion d'un port d'initialisation du modèle couplé à un port d'initialisation d'un sous-modèle

La première forme est qualifiée d'interne (*internal*), la deuxième et la troisième d'externe (*external*) et la dernière d'initialisation (*init*). Les ports d'états ne font pas l'objet de connexion. Tout port d'état est accessible à l'expérimentateur.

```
<CONNECTION TYPE="internal">
        <ORIGIN MODEL="A" PORT="out">
        <DESTINATION MODEL="B" PORT="in">
</CONNECTION>
```

2.2.1.4 Le temps

Comme nous l'avons présenté dans la description des modèles, plusieurs éléments font référence au temps : le modèle possède un référentiel temporel et les données attachées aux événements peuvent aussi faire réference au temps. Qu'entend-t-on par référentiel temporel ? Tout changement d'états d'un système dynamique est ordonné selon le temps. Il est donc fondamental de définir quelle est la base temporelle utilisée pour référencer temporellement les changements d'états. En effet, il est indispensable de savoir si les dates associées aux événements générés en sortie d'un modèle se situent sur une échelle temporelle de type seconde ou année. Cette information est fondamentale pour la gestion du couplage de modèles.

De manière théorique, le temps est défini par un ensemble T et une relation d'ordre sur les éléments de T que l'on notera $<$. L'ensemble T représente des instants ou des lieux sur l'axe temporel. La relation $<$ représente la relation d'ordre qui existe entre les instants. Cette relation est transitive[36], anti-reflexive[37] et anti-symétrique[38].

Dans notre spécification, nous avons retenu les trois types de temps définis au sein du groupe de réflexion MIMOSA, le temps comme :
- un ensemble d'instants sans aucune relation d'ordre,
- un ensemble d'instants disposant d'une ou plusieurs relations d'ordre,

[36] si $t_1 < t_2 < t_3$ alors $t_1 < t_3$
[37] si $t_1 < t_2$ alors $t_2 < t_1$ est impossible
[38] si $t_1 < t_2$ alors $t_2 < t_1$ est faux

– un ensemble d'instants définit par une base de temps (type, unité, intervalle et pas).

Le premier type de temps ne vérifie pas la définition du temps énoncé. On définit des instants mais on ne définit pas de relation d'ordre entre ces instants. Néanmoins, il est intéressant de disposer de ce type de temps pour des modèles où le temps n'est pas important mais dans lesquels des instants sont identifiables. Dans ce cas, aux instants identifiés, on peut exprimer l'état du système. Entre ces instants, le système est dans un état indéfini. Le deuxième et troisième type de temps respectent la définition du temps. La seule différence réside dans la manière de l'exprimer. Dans le deuxième cas que l'on appelera temps ordinal, tous les instants sont nommés et une relation d'ordre est définie explicitement. Dans le troisième cas que l'on nommera temps cardinal, les instants sont déduits de l'expression d'une base temporelle. Cette base définit l'origine du temps (l'instant initial), la borne supérieure du temps, l'unité du temps (tout simplement, seconde, minute, ...) et si le temps est continu ou discret. Si le temps est continu alors le nombre d'instants est infini et un instant est représenté par une valeur réelle. Si le temps est discret, la base de temps est complétée par un pas temporel. Les instants sont alors des multiples de ce pas par rapport à l'origine du temps. Le temps est alors défini en compréhension. L'origine du temps et la borne supérieure du temps forment un intervalle temporel ou segment temporel.

Ces trois types de temps sont spécifiées en XML de la manière suivante. Si le temps est un ensemble d'instants sans relation d'ordre, deux syntaxes sont possibles. On peut définir les instants en extension (tous les instants sont définis) ou en compréhension. Dans le premier cas, les instants sont énumérés à l'aide la balise TIME-SPAN. Dans la version en compréhension, deux possibilités sont encore offertes : soit les instants sont totalement anonymes (NAME=”*”) soit les instants sont nommés par le nom qui précéde les crochets et sont indexés par des entiers pris sur l'intervalle [n,m]. Les instants anonymes représentent la spécification limite : le modélisateur ne sait rien sur le temps. Dans ce dernier cas, l'ordre intrinsèque des entiers ne définit pas d'ordre. Les entiers ne servent qu'à nommer les instants.

```
<TIME TYPE="set">
  <TIME-SPANS>
    <TIME-SPAN NAME="" />
    ...
  </TIME-SPANS>
</TIME>

<TIME TYPE="set">
  <TIME-SPANS>
    <TIME-SPAN NAME="*|name[]" BEGIN="n" END="m"/>
    ...
  </TIME-SPANS>
</TIME>
```

La balise <ORDER> contenue dans la balise <TIME> spécifie la relation d'ordre qui s'applique sur les instants définis par l'une des deux syntaxes précédentes. Selon, la syntaxe utilisée pour la définition des instants, la relation d'ordre peut être spécifiée par une énumération ordonnée des instants ou en indiquant que la relation d'ordre utilise tout simplement d'indexation des instants. Dans l'exemple ci-dessous, la première relation signifie que *name1* précéde *name2*.

```
<ORDER>
  <RELATION SEQUENCE="name1,name2,..." />
```

```
  <RELATION SEQUENCE="index" />
</ORDER>
```

Les différentes balises présentées jusqu'ici permettent de définir le temps comme un ensemble d'instants sans relation d'ordre et le temps ordinal. Les balises <TIME-SPANS> et <ORDER> sont utilisables ensemble.

Le temps cardinal fait appel à une troisième balise qui s'utilise seule. La balise <TIME-BASE> est contenue dans la balise <TIME>. Si le type de base de temps est discret alors tous les attributs doivent apparaître dans la définition. Les attributs BEGIN et END définissent le domaine de validité du modèle qui fait référence à cette base. Ce domaine peut être fermé ou ouvert (END="*"). L'attribut BEGIN fixe l'origine du temps.

```
<TIME TYPE="cardinal">
  <TIME-BASE TYPE="dicrete|continuous" UNIT="second|minute|hour|..."
             BEGIN="n" END="m|*" STEP="p" />
</TIME>
```

Afin de fixer totalement les idées sur cette notion de temps, prenons un exemple. Si on a construit un modèle à événements discrets où le temps est considéré continu, la définition du référentiel temporel du modèle est simple :

```
<TIME TYPE="cardinal">
  <TIME-BASE TYPE="continuous" UNIT="second" BEGIN="0" END="*"/>
</TIME>
```

Nous avons donc dû prendre une décision sur l'unité utilisé (ici, la seconde) et l'intervalle de temps dit valide. Le domaine de validité du modèle est ouvert et son origine est fixée à 0. Lorsqu'un domaine est fermé, cela signifie que le modèle n'est plus considéré valide au delà de la borne supérieure du domaine. Lors de la définition d'une expérience, cette information sera contrôlée.

2.2.1.5 L'espace

Les systèmes auxquels on s'intéresse sont les systèmes spatio-temporels et hormis le temps, l'espace doit être spécifié. On ne s'interdit pas de concevoir des modèles sans l'espace. Dans ce cas, l'espace ne fera pas parti de la spécification du modèle.
La question de l'espace est un élément qui a fait l'objet d'une spécification formelle dans le groupe MIMOSA. Nous en proposons ici une spécification XML. L'espace peut prendre trois formes :

- un ensemble de lieux sans relation de position des uns par rapport aux autres,
- un ensemble de lieux avec une relation de voisinage,
- un ensemble de lieux muni d'une métrique.

La notion de lieu proposée ici peut être interprétée différemment selon le modèle. La première interprétation consiste à dire qu'un lieu est une zone délimitée ou un point dans un espace mais quel type d'espace ? Dans les écosystèmes, la notion d'espace représente l'endroit où évoluent les entités (poissons, oiseaux, fourmis, ...). Dans un autre cas, l'espace peut être le domaine dans lequel les entités évoluent. Ce domaine pouvant être défini, par exemple, par l'appartenance des entités à un groupe ou par le fait qu'un attribut ou un ensemble d'attributs d'une entité

possèdent des valeurs caractérisant une classe d'appartenance. Pour rester dans le domaine des écosystèmes, on peut construire un modèle de dynamique de populations d'animaux où les animaux sont regroupés en classes d'âge. Tout animal appartient à une et une seule classe d'âge. La dynamique du système va faire évoluer les animaux de classe en classe.
La première forme d'espace a pour objectif de représenter ce type d'espace : espace réel ou espace d'appartenance. Les lieux sont sans relation. La syntaxe adoptée est la suivante :

```
<SPACE NAME="" TYPE="set" CLASS="className">
<PLACES>
<PLACE NAME="name1" />
<PLACE NAME="name2" />
...\\
</PLACES>\\
</SPACE>
```

```
<SPACE NAME="" TYPE="set">
<PLACES>
<PLACE NAME="*|name[][][]..." BEGIN="n1,n2,n3,..." END="m1,m2,m3,..."/>
...
</PLACES>
</SPACE>
```

Trois syntaxes sont possibles : l'énumération de tous les lieux, l'énumération en compréhension avec indexation des lieux et l'énumération anonyme. Ce dernier cas indique que la notion d'espace est présente dans le modèle mais qu'aucune spécification détaillée n'est possible. L'énumération en compréhension permet de construire des lieux dont les noms sont indéxés. L'indexation peut être multidimensionnel afin de représenter des espaces de dimension quelconque si la notion de dimension à un sens dans le modèle. L'attribut CLASS disponible dans la balise <SPACE> permet de préciser le type d'espace défini (réel ou d'appartenance). Si l'espace est entendu au sens espace réel, cet attribut prend la valeur "real" sinon sa valeur est libre. Par exemple, si l'espace est un espace d'âge, l'attribut prendra la valeur "age".
Quel que soit le type d'espace utilisé par le modèle (réel ou d'appartenance), on peut définir si nécessaire une ou plusieurs relations de voisinage. La balise <NEIGHBOURHOOD> contenue dans la balise <SPACE> permet l'énumération des relations de voisinage. Le type de l'espace est alors "topological" au lieu de "set". Plusieurs expressions sont possibles pour la définition d'une relation de voisinage.

```
<NEIGHBOURHOOD>
<NEIGHBOUR PLACE="name1" LINK="name2,name3,..." />
...
</NEIGHBOURHOOD>
```

La première possibilité consiste à préciser l'ensemble des relations de voisinage. Dans l'exemple, le lieu *name1* a pour voisin tous les lieux appartenant à la liste contenu dans l'attribut LINK. Cette expression est très fastidieuse mais indispensable si les lieux ont été défini en extension.
L'autre possibilité consiste à définir une matrice de voisinage qui s'applique à l'ensemble des lieux. La matrice de voisinage n'est applicable que si les lieux ont été défini en compréhension.

```
<NEIGHBOURHOOD>
<NEIGHBOUR TYPE="Von Neumann|Moore|...|any">
```

```
<LINK VALUE="0,1" />
<LINK VALUE="-1,1" />
...
</NEIGHBOUR>
</NEIGHBOURHOOD>
```

De multiples voisinages en 2 dimensions existent dans la littérature (Von Neumann[39], Moore[40], Moore étendu [41], ...), il est donc possible de ne préciser que le type dans ce cas ou d'indiquer la matrice de voisinage s'il est quelconque (TYPE="any"). Dans ce cas, il faut la définir à l'aide de la balise <LINK>. Les coordonnées indiquées dans l'attribut VALUE de la balise <LINK> expriment les coordonnées relatives du lieu voisin[42]. Les coordonnées "0,1" en 2 dimensions localisent le voisin Sud.

Le dernier type d'espace est qualifié de métrique. Une métrique composée d'un référentiel (repère et axes) et une distance sont disponibles. Toute entité sera alors repérée grâce à cette métrique.

```
<SPACE NAME="" TYPE="metric">
<REFERENTIAL TYPE="discrete|continuous" DIMENSION="n">
<AXIS ID="" MIN="" MAX="" STEP="" />
...
</REFERENTIAL>
<DISTANCE TYPE="euclidian|..." UNIT="meter|millimeter|..." />
</SPACE>
```

La balise <REFERENTIAL> permet d'indiquer si l'espace est discret ou continu et la dimension (1, 2, 3 ou plus) de l'espace. La dimension permet de connaître le nombre d'axes à définir. On distingue deux types d'espace : discret ou continu. Dans les précédents cas, les espaces sont obligatoirement discret de par leur définition. Pour les espaces métriques, la distinction est possible. La définition des axes tient compte de l'aspect discret et continu. L'origine d'un axe est par convention 0. Les attributs MIN et MAX fixent les limites de validité des coordonnées des entités localisées dans cet espace. L'attribut STEP définit les coordonnées possibles lorsque l'espace est discret. L'espace discret est une voie pour définir des grilles à 2, 3 ou n dimensions. Les coordonnées discrètes sont alors les indices des cellules formant la grille.
Il ne reste plus qu'un élément pour définir totalement un espace métrique : la distance utilisée. La balise <DISTANCE> est prévue à cet effet et permet d'indiquer le type de distance choisie (euclidienne ou autres). On associe à la distance l'unité de mesure adoptée.

2.2.1.6 Perspectives et conclusion

La spécification de modèles est au coeur de nos travaux. Le *framework* et le langage de spécification proposés constituent le premier niveau d'un outil de modélisation et de simulation de systèmes complexes. La perspective de la multi-modélisation et du couplage de modèles nous a conduit à concevoir le *framework* et le langage de spécification selon une certaine orientation.

[39] on considère seulement les voisins Nord/Sud/Est/Ouest.
[40] on ajoute les diagonales par rapport à Von Neumann.
[41] la distance est supérieure à 1.
[42] C'est l'approche utilisée dans la définition des structures Cell-DEVS

Plusieurs hypothèses ont été placé : pré-existance des modèles, formalismes hétérogènes, simulateurs distribués, modèles spatio-temporels, modèles discrets ou continus, ... Tous ces facteurs ont été des éléments favorisants la complexité mais aussi la richesse de l'outil.

Plusieurs questions restent en suspens. Si l'on désire spécifier totalement le modèle en DEVS ou à l'aide d'un langage de spécification héritant des propriétés formelles de DEVS, il faut réfléchir aux implications et aux extensions à apporter au langage actuel. Actuellement, des travaux sont en cours dans le groupe de standardisation de DEVS piloté par G. Wainer. Les questions liées au contrôle sémantique sont aussi à explorer. Certaines informations ont été prise en compte mais quel est leur degré de pertinence. La mise en relief de certains éléments des modèles par leur relation avec l'espace et le temps sont à préciser et à développer. Ces relations sont aussi à intégrer au processus de validation sémantique. Plusieurs études sont à mener pour concevoir et développer des *wrappers* en particulier pour les équations différentielles et les *state charts* d'UML. Le *state charts* sont probablement une piste intéressante à suivre pour l'intégration de formalismes à base d'objets et d'agents.

2.2.2 Les expériences

Une expérience : c'est quoi ? C'est placer le système en cours d'étude dans des conditions bien définies (et donc sous des hypothèses expérimentales bien précises) et c'est observer son évolution en cours du temps par l'intermédiaire de mesures. Deux approches peuvent être ensuite menées : on change les conditions expérimentales (température, luminosité, ...) et on observe les variations de comportements du système ; on change le système et on le place dans les mêmes conditions que le système précédent afin de comparer le comportement des deux systèmes. La comparaison des comportements d'un système ou des plusieurs systèmes conduit à déployer de multiples stratégies statistiques. Ces stratégies répondent par exemple au problème d'étude de l'influence de la variation d'un ou plusieurs paramètres sur le comportement d'un système. Les valeurs des paramètres définissent les conditions expérimentales.

Nous avons défini dans la partie précédente ce qu'était un modèle et nous avons développés le formalisme utilisé pour spécifier en terme d'initialisation, d'entrée, de sortie et d'états un système. Nous devons maintenant définir par analogie avec la notion d'expérience sur un système réel, les notions de conditions expérimentales et de mesures.

Les conditions expérimentales fixent au niveau du modèle 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 être spécifiées.

Selon la politique d'exploration du modèle, les conditions expérimentales doivent être définies comme des ensembles de valeurs à parcourir pour une 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).

La notion de politique d'exploration est une notion clé. Elle est pour nous synonyme de plan d'expériences. Imposer un schéma conducteur et un langage de spécification de ces plans d'expé-

riences est très important. En effet, nous négligeons trop souvent cette partie en ne posant pas proprement les objectifs d'une expérience et les conditions expérimentales. Il existe des règles de base essentielles à suivre afin de mener correctement une expérience sinon que pouvons nous dire des résultats, quels droits avons nous de les comparer ... Les modèles qui sont manipulés sont souvent très complexes, le nombre de paramètres est très important et la correlation entre paramètres est non négligeable. Il faut donc manipuler les modèles et les expériences avec précaution et surtout avec méthode. De plus, une forte critique existe concernant les modèles de type centrés individus ou agents où une partie de la communauté scientifique ne porte qu'un crédit très relatif car ces modèles sont mal formalisés et beaucoup de modélisateurs ne posent pas clairement les conditions expérimentales [GRI99].

La définition d'une expérience inclus l'activation de points de mesure sur les variables d'états. En effet, les modèles offrent la lecture des variables d'états par les ports d'états. L'utilisateur indique alors tout simplement quelles sont les variables d'états qu'il désire observer. A partir de ces mesures, on doit pouvoir appliquer des opérateurs statistiques (moyenne, par exemple) afin de synthétiser des indicateurs du comportement du modèle.

La spécification d'une expérience est accessible de nouveau via un fichier XML. Une expérience peut être la mise en oeuvre de plusieurs simulations mettant en jeu un ou plusieurs modèles. Dans le cas de plusieurs modèles, cela signifie que la simulation porte sur plusieurs modèles que l'on va comparer. Si un seul modèle est impliqué dans la simulation, on cherche juste alors à étudier son comportement.
Chaque modèle fait l'objet de mesures. Chaque mesure est attachée à un port d'états du modèle. Attention, si le modèle concerné est un modèle couplé, il faut spécifier le nom complet du modèle. La syntaxe utilisée est la suivante. Le port p du sous-modèle m_1 du modèle m_0 est désigné par $m_0 : m_1 : p$.

Voici un résumé de la syntaxe XML.

```
<EXPERIMENT NAME="">
  <SIMULATION REPLICA="">
        <MODEL NAME="">
          <CONDITIONS>
            <CONDITION PORT="">
            </CONDITION>
            ...
      </CONDITIONS>
          <MEASURES>
            <MEASURE PORT="" TYPE="frequence|event" [FREQUENCE="x"]>
                </MEASURE>
                ...
      </MEASURES>
        </MODEL>
        ...
  </SIMULATION>
</EXPERIMENT>
```

Les conditions expérimentales sont décrites par les balises CONDITION. Une balise CONDITION spécifie le domaine de variation des données d'un port d'initialisation. Nous rappellons

que le concept de port d'initialisation peut être assimilé à un paramètre du modèle. Il est aussi possible de spécifier des contraintes sur certaines valeurs de ports d'initialisation. On contraint alors l'ensemble des valeurs possibles par une relation mettant en jeu les valeurs prises par un ou plusieurs autres ports. On peut aussi combiner des contraintes et des ensembles de valeurs (ensemble ou intervalle). Dans ce cas, on utilisera toutes les valeurs de l'ensemble respectant la contrainte. Dans l'exemple suivant, le port p_0 du modèle M_0 prendra des valeurs prises dans l'ensemble $1, 2, 3, 4, 5$ sous la contrainte qu'elles soient deux fois supérieures à la valeur prise par le port p_1 du modèle M_0.

```
<CONDITION PORT="M0:p">
        <SET>
                <ITEM VALUE="1">
                <ITEM VALUE="2">
                <ITEM VALUE="3">
                <ITEM VALUE="4">
                <ITEM VALUE="5">
        </SET>
        <CONTRAINST>
                <GREATER THAN>
                        <MULT>
                                <CONST VALUE="2">
                                <VAR PORT="M0:p1">
                        </MULT>
                <EQUAL TO>
        </CONTRAINST>
</CONDITION>
```

Attention, l'utilisation des ensembles et des contraintes conduit de manière générale à la résolution de systèmes d'inéquations. Une contrainte sans ensemble est obligatoirement une contrainte de type égalité.

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).

La lecture des mesures fait l'objet d'une politique d'échantillonnage. Cette politique définit la manière d'acquisition des mesures : les mesures sont réalisées selon une certaine fréquence ou selon l'arrivée des mesures (aux changements de valeurs des variables d'états attachées aux mesures). On parlera dans le premier cas de mesure active : on ne s'intéresse pas aux événements de changement d'états ; selon une certaine fréquence, on lit la valeur de la variable d'état (on effectue une mesure). Dans le deuxième cas, on parlera de mesure passive : chaque mesure est datée par la date d'occurrence du changement de la variable d'états. On peut définir des ensembles de points de mesure réunis de manière virtuelle et font donc l'objet d'une acquisition au même moment.

Si le modèle intègre une partie stochastique (autrement dit, si deux simulations dans les mêmes conditions expérimentales conduisent à des réponses différentes), il est important d'effec-

tuer des réplicats. Un ensemble de réplicats est un ensemble de simulation dont les paramètres ont les mêmes valeurs.

Le langage de spécification des expériences joue le même rôle que le langage de spécification des modèles, il permet de définir formellement les plans d'expériences. Il constitue donc une brique de base du système. La description d'un plan d'expériences est le point d'entrée du processus de simulation. A partir de son expression, le moteur de simulations détermine les expériences à réaliser conformément au plan, active les simulateurs et leur indique les conditions initiales de la simulation qu'ils ont à mener.

En conclusion, le langage de spécification des expériences peut paraître simple mais il a le mérite d'exister et de répondre au problème des plans d'expériences. De plus, il s'intégre parfaitement à la logique de spécification des modèles. Nous allons maintenant voir comment ces divers langages de spécification sont manipulés dans la plate-forme VLE.

2.2.3 Architecture

Cette partie va faire le point sur l'architecture globale de VLE (*Virtual Laboratory Environment*) et sur les briques existantes. Nous considérons à ce niveau que les questions de spécifications sont réglés. VLE a été conçu comme une plate-forme de conception, de déploiement et de traitement pour la modélisation et la simulation des systèmes complexes. Une première version de VLE, en 1998, a été conçu et développé dans un objectif plus restreint puisqu'il s'agissait de répondre aux besoins de modélisation et de simulation d'un écosystème particulier. De plus, l'idée qui avait motivé le développement d'une plate-forme était de disposer d'un environnement de conception de modèles multi-agents adapté à notre problématique. Nous allons donc présenter la première version de VLE puis nous nous intéresserons à la version actuelle. Cette dernière a pour prétention d'atteindre les objectifs précédemment énoncés.

2.2.3.1 VLE : environnement de modélisation et de simulation multi-agents

VLE a été développé afin d'obtenir une plate-forme disposant des outils de conception et de développement d'un laboratoire virtuel d'où son nom *Virtual Laboratoty Environment* et des expériences virtuelles qui peuvent être réalisées avec un laboratoire virtuel. Cela nécessite un outil permettant de décrire et de construire un modèle du système complexe que l'on veut étudier par la simulation, de définir l'expérience que l'on veut mener et d'analyser les résultats des simulations. Dans les faits, nous nous intéressons à la simulation des écosystèmes et, plus spécifiquement, nous nous sommes focalisés sur l'étude d'un écosystème marin particulier (voir chapitre 3). Nous ne discuterons pas ici de l'utilité de la modélisation multi-agents et des laboratoires virtuels pour les écologistes. Il est absolument clair qu'un seul paradigme ne peut rendre compte du tout mais que les différents paradigmes doivent être intégrés. Cette remarque donnera naissance aux travaux de thèse de Raphaël Duboz.

En bref, voici une liste des points clés caractérisant la conception et le développement de VLE :

- dédié à l'étude, la modélisation et la simulation des systèmes naturels ; nous avons réduit le champ d'application de la plate-forme,
- basé sur les agents (la dénomination "approche centrée individus" est privilégiée par les écologistes [GRI99]) ; l'agent est la brique de base mais sa spécification peut faire intervenir des paradigmes divers (systèmes à base de règles, réseaux de Petri, équations

différentielles...), il est alors nécessaire d'étudier le couplage de ces formalismes [CUB97],
- orienté expérimentateur, l'utilisateur d'un tel système est typiquement un non-informaticien, il doit interagir avec VLE via une interface graphique,
- simulations réalistes et réalisées dans un environnement virtuel ce qui implique l'optimisation du noyau de simulation et l'utilisation de techniques distribuées (distribution automatique de la simulation sur un réseau de stations, par exemple).

Le noyau de simulation de VLE a été écrit en C++ et l'interface en Java. De fait, on couple la puissance du C++ pour les calculs et l'universalité du Java pour l'interface graphique. Entre 1998 et 2000, nous avons réalisé le développement de VLE avec l'aide d'étudiants de deuxième cycle de l'IUP GMI de Calais (voir annexe) ce qui a permis de disposer d'une version qui nous a conduit à des résultats intéressants (voir chapitre 3) et à plusieurs publications [RAM98][SEU99b].

Nous allons maintenant présenter VLE dans ses grandes lignes et nous allons focaliser notre présentation sur les différents types d'agents.

Agents réactifs

Dans VLE, les agents peuvent représenter soit l'environnement (cellules interconnectées), soit des objets actifs (objets qui ont un comportement) soit des objets passifs (tels que la nourriture qui est située mais n'a pas de comportement). La Figure 2.23 montre une partie de l'architecture de classes de VLE. Tous les agents de VLE sont réactifs et situés dans l'espace et dans le temps (voir Figure 2.24). Un agent est une extension de la notion d'objet. De ce fait, il possède un ensemble d'attributs (par exemple, taille, poids ...) caractérisant son état interne et met en oeuvre un ensemble d'opérations. Plus spécifiquement, un agent possède des informations telles que sa position dans l'espace et dans le temps. De plus, il est autonome et possède un répertoire de comportements qu'il peut émettre en fonction de son état interne. Ces comportements peuvent changer dans le temps en fonction de son état. Par exemple, le comportement d'un organisme va se modifier au cours de sa maturation. On distingue un sous-ensemble de comportements qui jouent un rôle particulier : la perception. Les facultés de perception de l'environnement font partie des comportements et peuvent changer en fonction de l'état interne.

Nous détaillons maintenant chacune des fonctions d'un agent.

Perception

Les agents perçoivent leur environnement et répondent aux changements qui surviennent. La perception de l'agent est associée à ses capacités à percevoir les agents qui lui sont proches. Chaque agent active ou inhibe une partie de ses sens en fonction de son état interne. La perception joue un rôle prépondérant dans la communication entre agents. Un agent qui perçoit un autre agent, créée un canal de communication entre les deux agents.

Etat

La notion de réactivité s'accompagne de la gestion de l'état de l'agent. A chaque instant t, un agent est dans un et un seul état. Cet état se caractérise par le vecteur de variables d'états de l'agent. On distingue deux types de variables d'états : les attributs et les états comportementaux. Les attributs sont l'ensemble des caractéristiques statiques ou dynamiques de l'agent (volume, couleur, vitesse ...). Nous reviendrons plus tard sur la notion d'état comportemental.

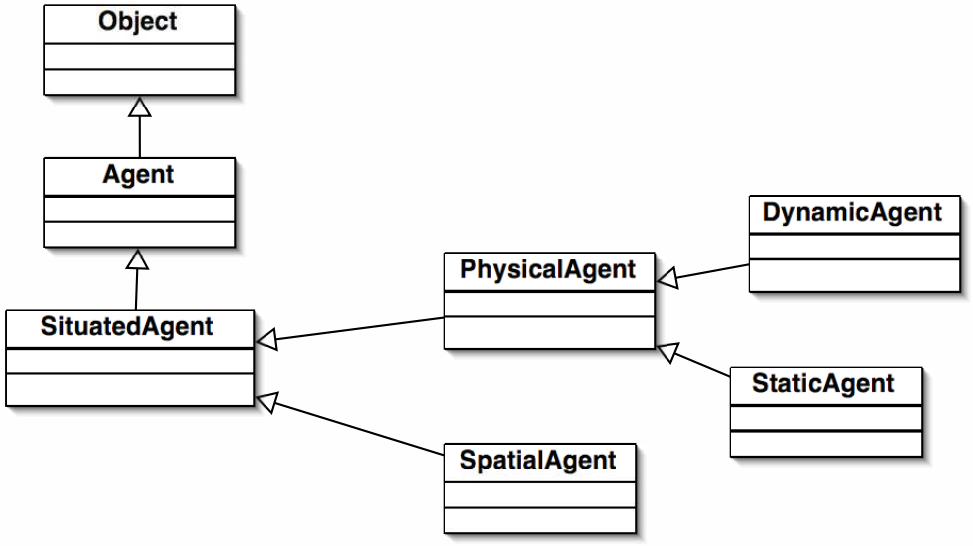


FIG. 2.23 – Une partie de la hiérarchie des classes de VLE

Un changement d'états survient entre t et $t + \Delta t$ (où Δt est le pas de temps de la simulation) sur la perception d'un événement ou sur un processus interne et "continu" (variation discrétisée d'une variable d'états mise en jeu dans le processus, par exemple). Il faut noter que cette première version de VLE était exclusivement à temps discret.

Comportement

Il existe de nombreux formalismes (comme nous l'avons mis en évidence dans le début de ce chapitre) tels que les automates à états finis, les réseaux de Petri et les *state charts*, par exemple. Nous utilisons pour notre part deux formalismes :

- les réseaux de Petri interprétés (*Interpreted Petri Networks*) car ils englobent la plupart des caractéristiques des formalismes nommés précédemment,
- un langage algorithmique accompagné de primitives de haut niveau.

Le comportement d'un agent est donc spécifié par un ensemble de RdP interprétés ou d'algorithmes ou les deux exécutables en parallèle. Intéressons nous aux réseaux de Petri. Un RdP peut être assimilé à un graphe d'états et la présence d'un jeton dans une place indique l'activation de l'état. Considérons un exemple très simple : un agent peut être dans l'un des trois états (vide, plein ou entre les deux). La présence d'un jeton dans la place étiquetée "vide" signifie que l'agent est dans l'état vide (voir Figure 2.25).

Les RdPs interprétés peuvent aussi spécifier des algorithmes qui prennent explicitement en compte le temps. Des actions peuvent être associées aux places ou aux transitions. Ces actions sont exécutées à chaque fois qu'un jeton arrive dans une place ou active une transition. De plus, les transitions sont temporisées et seront donc franchies à l'issue de la temporisation. Dans l'exemple de la figure 2.26, si la condition associée à la transition est valide, le jeton est retiré à la place P1 et après un délai de 5 unités de temps, un nouveau jeton apparaît dans la place P2 et par conséquent déclenche l'exécution des actions associées à la place P2.
Les actions mises à la disposition du modélisateur sont orientées vers les systèmes qui nous intéressent : les systèmes spatio-temporels. Pour résumer, on peut identifier trois classes d'actions :

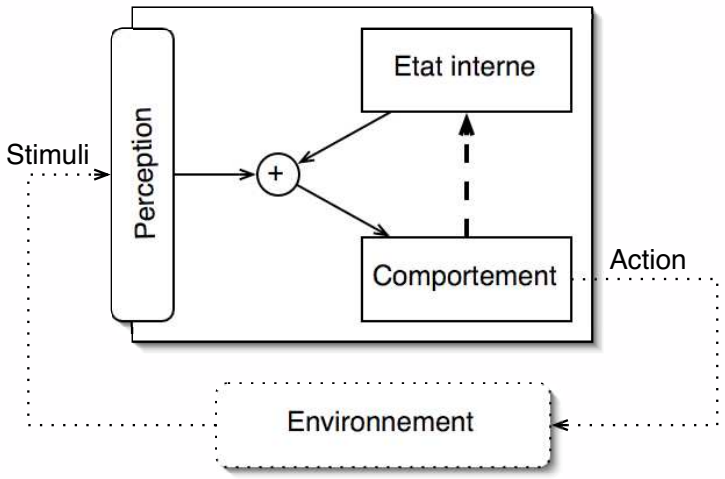


FIG. 2.24 – Agent réactif

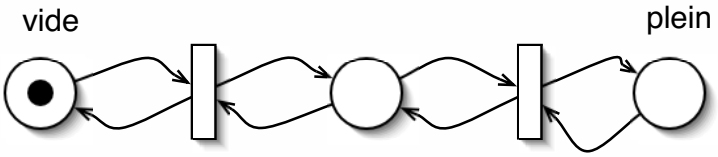


FIG. 2.25 – Un exemple simple de RdP

les actions de perception, les actions de déplacement et les actions de manipulation des attributs des agents. Les actions de perception permettent de créer des canaux de communication entre les agents qui se perçoivent. Ces actions sont aussi impliquées dans plusieurs autres types d'actions. Par exemple, on peut combiner une action de perception avec une action de déplacement. On peut demander à un agent dynamique de se déplacer vers un agent qu'il perçoit. Nous n'en dirons pas plus. En conclusion, on peut dire que les primitives offertes par VLE sont spécialisées dans la description de comportements d'agents réactifs.

Avec ce type de RdPs interprétés, nous spécifions la même chose qu'avec des grafcets. L'introduction des temporisations dans le modèle des comportements est un moyen de spécifier les processus suivant une certaine échelle de temps. La définition concurrente des RdPs interprétés d'un agent permet de spécifier des processus d'échelles de temps différentes. Les RdPs interprétés doivent respecter les conditions suivantes :

- les événements ou points de synchronisation ne peuvent avoir lieu au mieux tous les pas de temps du modèle,
- à chaque instant t, chaque comportement (ou processus) de l'agent doit traiter les événements en un temps fini et indique la date à laquelle ce comportement (ou processus) redeviendra actif.

Deux cas de figure doivent être considérés :

- les actions issues du comportement des agents ont une durée inférieure ou égale à Δt (incluant les actions instantanées). Dans ce cas, les actions sont exécutées et le processus

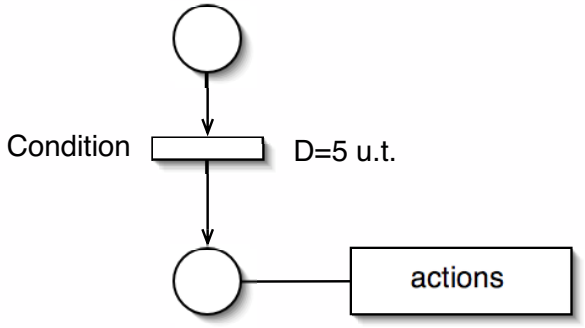


FIG. 2.26 – Transitions temporisées et actions

est prêt à traiter les événements au prochain pas de temps,
- dans le cas contraire, l'action a une durée supérieure à Δt.

Pour ce dernier cas, on peut alors envisager deux possibilités :
- l'action se décompose en sous-actions de durée égale à Δt (on se ramène alors au premier cas),
- l'action peut être effectuée à la date t et le processus devient inactivable pendant la durée réelle de l'action sans pour autant affecter le fonctionnement global du système.

Cette seconde possibilité peut être illustrée par l'exemple suivant. Dans [RAM98], nous avons modélisé le comportement de déplacement et de prédation du copépode (voir chapitre 3). Lorsque le copépode capture une cellule de phytoplancton, une phase de manipulation d'une durée de 10 u. t. est nécessaire pour que le copépode ingère totalement la cellule. Tant que le copépode n'a pas complètement ingéré la cellule, le copépode ne peut pas capturer une autre cellule. Le processus de capture/manipulation est codé par un et un seul RdP interprété. Celui-ci est donc inactivable le temps de la manipulation d'une cellule de phytoplancton.

On peut, néanmoins, envisager une troisième possibilité : les actions continues de durée supérieure à $?\Delta t$, non décomposables et ne respectant pas la deuxième possibilité. Nous ne traitons pas ce cas de figure.

Communication

Les agents construisent dynamiquement leur réseau d'accointance via leur perception. Ils peuvent aussi percevoir l'existence de certains agents sans utiliser leur capacité de perception. On utilise alors les notions d'association et de lien entre les agents. Un agent peut communiquer seulement avec les agent qu'il connaît. La communication est basée sur l'envoi de messages synchrones aux agents qu'un agent perçoit ou qu'un agent connaît. A chaque itération de la simulation, chaque agent actif détermine l'ensemble des agents qu'il perçoit et avec lesquels il est susceptible de communiquer.

Agents spatiaux

De multiple mises en oeuvre de systèmes multi-agents [DEU97][HRA97][LEP96] pour la simulation d'écosystèmes modélisent l'environnement comme une grille $n \times m$ d'agents (ou de cellules) interconnectés. Chaque agent est connecté à ses 4, 6 ou 8 voisines dans un espace à 2

dimensions (voir figure 2.27).

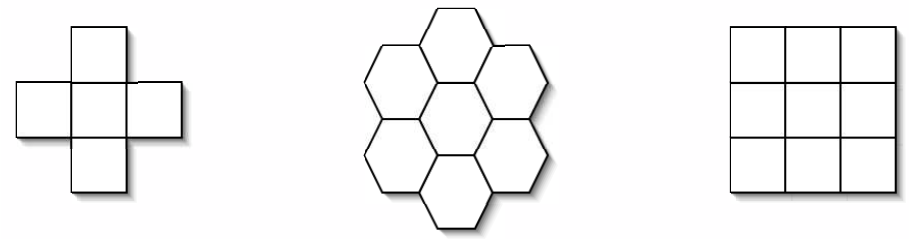


FIG. 2.27 – Type de voisinage

Dans notre approche, la grille est composée d'agents spatiaux qui peuvent être de natures différentes (ou de classes d'appartenance différentes). Le degré de connectivité est 8 et l'environnement est torique (fermé sur lui-même) ou non. Un agent spatial est un agent réactif. Il peut donc posséder un comportement propre en fonction de stimuli reçus de ses voisins ou des agents "localisés sur lui".

Les agents spatiaux forment en quelque sorte l'environnement de nos agents physiques. Ils sont considérés comme des agents à part entière et disposent de leur propre dynamique. La conséquence de cette approche est que les agents spatiaux sont des super automates cellulaires et pour aller dans ce sens, nous avons intégré dans les agents spatiaux des mécanismes d'agrégation en entités de plus haut niveau. Avant de définir de manière précise ce que nous appelons agrégation, voici la définition communément admise [DEL91] : "une agrégation est un regroupement de différentes entités en une nouvelle entité de niveau supérieure". Par exemple, une forêt est une agrégation d'arbres. Une forêt est conceptuellement une entité de haut niveau. L'idée est de représenter des entités émergentes et des systèmes multi-échelles.

Le regroupement d'entités peut être perçu autrement. Dans le cas de l'agrégation, les entités de niveau élémentaire sont toujours présentes après l'opération d'agrégation. La fusion est une autre approche : le regroupement de plusieurs entités donne naissance à une nouvelle entité de niveau supérieure ou non et les agents élémentaires disparaissent. [CAM97][SER98] en fournissent un exemple : les agents "élémentaires" sont des boules d'eau qui se déplacent sous l'effet de la gravité sur un paysage. Le paysage est représenté par une grille de cellules interconnectées. Chaque cellule est caractérisée par sa pente et son orientation afin de calculer la direction d'une boule d'eau. Lorsqu'elles se rencontrent ou lorsqu'elles arrivent dans un minimum local du paysage, les boules d'eau fusionnent pour former des mares, des ravines, des rivières ...

Dans VLE, l'agrégat va représenter un ensemble d'agents homogènes à l'aide d'une et une seule entité de plus haut niveau ou non. Un ensemble d'agents est dit homogène si tous les agents sont issus de la même famille d'agents et si leurs vecteurs d'états sont égaux ou suivent une même loi de distribution. Nous nous situons donc à mi-chemin entre la notion d'agrégation et de fusion. En effet, les agrégats, tels que nous les définissons, représentent un ensemble d'agents situés homogènes mais comme pour la fusion, les agents le constituant disparaissent. Cette disparition n'est que technique. Conceptuellement, les agents élémentaires existent toujours.

Pour chaque agent spatial de VLE, nous pouvons spécifier un ensemble de conditions sous

lesquelles l'agent peut devenir ou rejoindre un agrégat. On parle alors d'états agrégeables. Ces conditions se caractérisent soit par les valeurs prises par les attributs, soit par l'état des RdPs interprétés de l'agent soit par les deux. Certaines contraintes peuvent être relâchées afin que deux agents s'agrègent si leur état est proche. Par exemple, si un agent est "presque vide", il peut rejoindre un agrégat d'agents vides. Lorsque le comportement de l'agent conduit à l'un de ces états agrégeables, l'agent scrute son voisinage pour voir s'il existe un agrégat d'agents dont l'état est identique au sien. Dans ce cas, il s'agrége et disparaît de l'ensemble des agents actifs. Dans le cas contraire, l'agent constitue à lui seul un agrégat.

Nous nous sommes principalement intéressés à l'agrégation des agents spatiaux qui constituent l'environnement et qui sont connectés entre eux. Un agrégat spatial est défini par un ensemble d'objets géométriques (cellule élémentaire, ligne, colonne et rectangle) localisés dans l'espace et par une frontière. La frontière appartient à l'agrégat (voir figure 2.28).

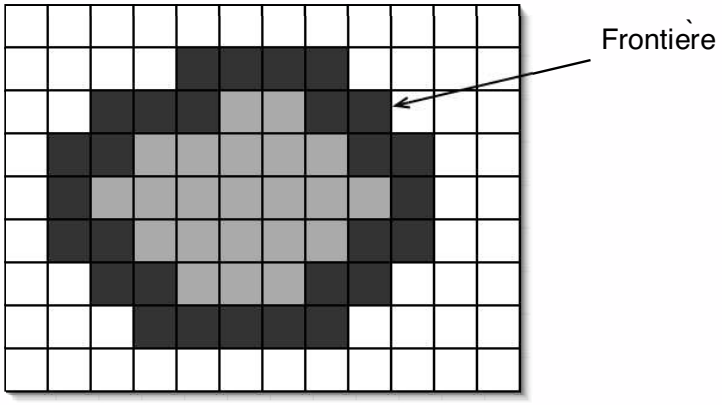


FIG. 2.28 – Un exemple d'agrégat spatial

L'opération d'agrégation est basée sur des règles d'assemblage d'objets géométriques. Ces règles cherchent à minimiser le nombre d'objets géométriques à chaque agrégation d'agent spatial élémentaire (voir figure 2.29) ou désagrégation en agent spatial élémentaire.

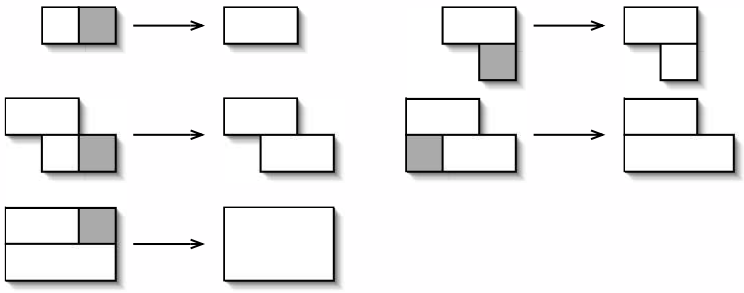


FIG. 2.29 – Exemple d'agrégation d'agents spatiaux (l'agent grisé est l'agent agrégé à l'instant t)

Dès lors que l'agent s'est agrégé, il n'existe plus. Or il doit rester sensible aux événements. Il est facile de répondre à cette première exigence puisque l'agrégat est constitué uniquement

d'agents de la même famille et dans le même état. On établit donc pour chaque état dit agrégeable une liste d'événements provoquant la désagrégation. Autrement dit, si l'agrégat reçoit un message (tout message est envoyé à un agent localisé) et que ce message modifie, par exemple, l'état de l'agent individuel qui aurait dû le recevoir alors l'agent individuel redevient actif et traite le message (voir figure 2.30).

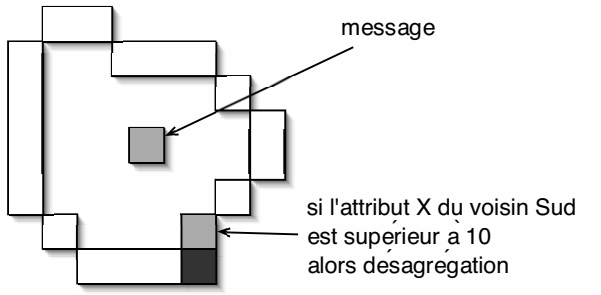


FIG. 2.30 – Exemples de désagrégation par réception de message et par condition à la frontière

La désagrégation est aussi possible lorsqu'un agent à la périphérie de l'agrégat voit ses conditions de voisinage changer (voir figure 2.30). La vérification des conditions de désagrégation nécessite que l'on définisse la frontière de l'agrégat (voir figure 2.28).

Cette approche a été développé dans le cadre d'une collaboration avec le Laboratoire de Géologie de Lille 1. L'idée était de montrer les capacités des approches agents et nous avons pris pour exemple l'infiltration de l'eau dans le sol (voir figure 2.31).

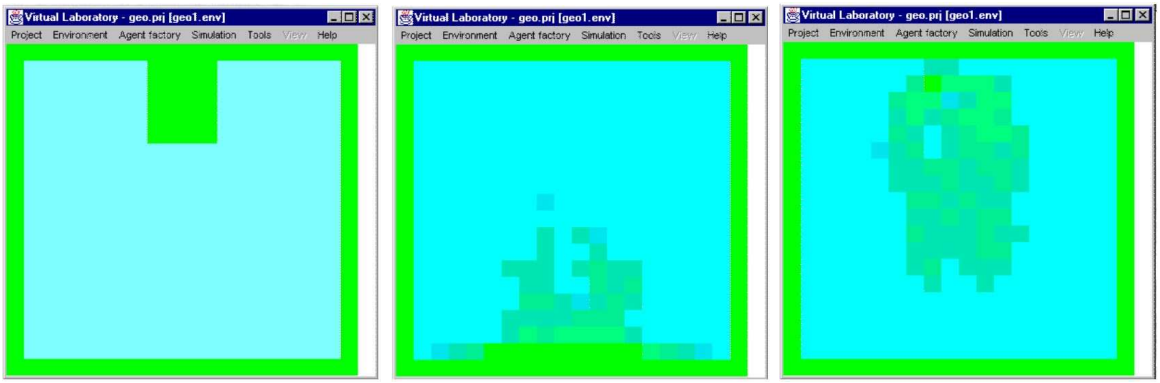


FIG. 2.31 – Les différentes phases de la simulation : état initial, désagrégation et agrégation ; La zone sombre modélise un agrégat de cellules saturées en eau (cette zone a été arbitrairement positionnée en haut afin que l'eau se propage vers le bas). Le reste du système est composé de cellules vides (zone claire). Le contour, quant à lui, se compose de cellules imperméables.

Pour conclure sur la première version de VLE, nous allons présenter son architecture globale. VLE se divise en quatre modules (voir figure 2.32) : le module de définition des modèles et des

expériences, le moteur de simulation, le moteur d'animation graphique et le module d'analyse. Le premier module se charge de l'interface de définition des modèles et des expériences. La première version de VLE ne générait pas un fichier au format XML mais dans un format propriétaire. Le moteur de simulation écrit en C++ calcule le comportement du modèle mis en jeu dans l'expérience. Un moteur d'animation permet de visualiser en temps réel d'évolution des entités dans l'environnement. Plusieurs plugins développés par des étudiants permettent de visualiser la trajectoire d'une entité dans un espace à deux dimensions. Des images sont présentées dans le chapitre 3. Le dernier module a pour objectif de traiter les sorties des expériences.

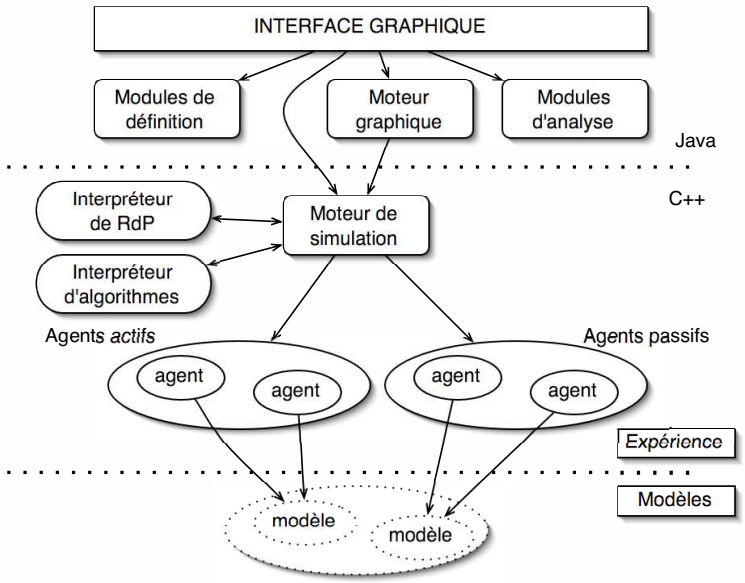


FIG. 2.32 – Architecture globale de VLE "version 98"

Depuis 1998, de multiples extensions ont été apporté à la plate-forme de base : le passage au temps continu et à l'espace continu par exemple. Nous ne développerons pas ici ses extensions. En revanche, nous tenons à souligner que toutes ses nouvelles capacités n'ont pas remise en cause l'architecture globale de VLE. Nous avons tout simplement étendu la bibliothèque de classes et de primitives.

2.2.3.2 VLE : environnement de multi-modélisation et de simulation de systèmes complexes

A partir de l'année 2000 et avec le démarrage de la thèse de Raphaël Duboz, VLE a été réorienté vers un environnement de conception de modèles multi-formalismes et de simulations. L'unité de base devient alors un projet. Un projet est constitué d'un groupe de modélisateurs qui conçoivent, développent et analysent des modèles multi-formalismes couplés. De plus, les modèles sont distribués. L'un des membres du groupe joue le rôle de responsable du modèle global. L'analyse du modèle passe par la constitution de plans d'expériences. Conformément à notre point de vue sur la multi-modélisation, chaque membre du groupe peut participer à la construction du modèle global et en particulier, peut apporter une partie du modèle. L'architecture (voir 2.33) se dessine autour d'un serveur de services Web dont les objectifs sont multiples :

– la gestion de la spécification en ligne des modèles et de leur couplage,

- la gestion de la spécification et de l'exécution des plans d'expériences,
- la gestion de la cohérence du travail collaboratif et du dépôt des modèles,
- offrir des outils de représentation des traces de simulation et d'analyse du comportement des modèles.

L'objectif premier est la spécification de modèles atomiques et couplés. L'interface Web doit offrir les outils nécessaires à la définition des modèles conformément à la syntaxe XML définie précédemment. L'idée sous-jacente est à la fois de proposer aux utilisateurs du système de déposer leurs modèles mais aussi de proposer aux autres utilisateurs une vue standardisée de leur modèle. Cette vue doit permettre la réutilisation par couplage des modèles par d'autres projets. Le langage de spécification de modèles est intégré à part entière dans la plate-forme afin de répondre à cette exigence.
Le modélisateur peut alors venir se "servir" dans la base de modèles et des simulateurs associés pour construire son propre modèle. L'idée est défendable dans le cas de projets multi-laboratoires et pluridisciplinaires ou dans une optique de comparaisons de modèles. La notion de plans d'expériences intègre cette idée. La construction de modèles couplés est alors pris en charge par la plate-forme. Les modules d'aide au couplage et de contrôle de cohérence sont alors activés. L'autre aspect important de l'interface est la possibilité de spécifier un modèle atomique. Comme nous l'avons évoqué lors de la présentation du langage de spécification des modèles, trois alternatives sont possibles pour la spécification de la dynamique des modèles :

- rien n'est spécifié, le modèle est considéré comme une boîte noire,
- la dynamique est spécifiée dans le formalisme choisi par le modélisateur,
- le modélisateur désire proposer une spécification DEVS ou une spécification dans un langage héritant de DEVS.

Dans le premier cas, la plate-forme a peu à offrir. L'utilisateur peut utiliser les fonctionnalités de gestion des plans d'expériences. Dans le cas du formalisme dédié, la plate-forme doit offrir un module de prise en charge du formalisme. On entend par là que l'on doit disposer d'une interface de spécification propre au formalisme choisi. L'architecture de la plate-forme est fortement modulaire et pour répondre à ces exigences, la plate-forme offre la possibilité d'ajouter à volonté des plugins de spécification de la dynamique. Ces plugins doivent proposer une interface Web pour la description du modèle à l'aide du formalisme pris en charge, un simulateur *DEVS-compliant* du formalisme[43] et une application XML chargée de la manipulation du fichier XML de spécification de la dynamique. Les plugins font l'objet d'un cahier des charges strict en terme d'interface fonctionnelle.
La spécification DEVS ne fait pas pour l'instant partie de la plate-forme mais nous ne devons pas l'ignorer.

Le deuxième objectif de la plate-forme est d'offrir un environnement de simulation de modèles couplés et distribués. Nous avons déjà évoqué cet aspect dans le *framework*. Nous résumerons notre approche par le schéma de la figure 2.33.

Dès lors que l'utilisateur a spécifie son modèle, des plans d'expériences peuvent être défini. En fonction de la description et de la politique de mise à disposition du simulateur associé, la plate-forme gènère les fichiers de spécification d'expériences et met en place l'infrastructure de communication afin de réaliser les expériences. L'infrastrcuture se base sur les concepts exposés dans le *framework* ce qui explique l'architecture proposée sur la figure 2.33.

[43]Le module de simulation d'un formalisme doit se conformer au *framework* défini en première partie de ce chapitre.

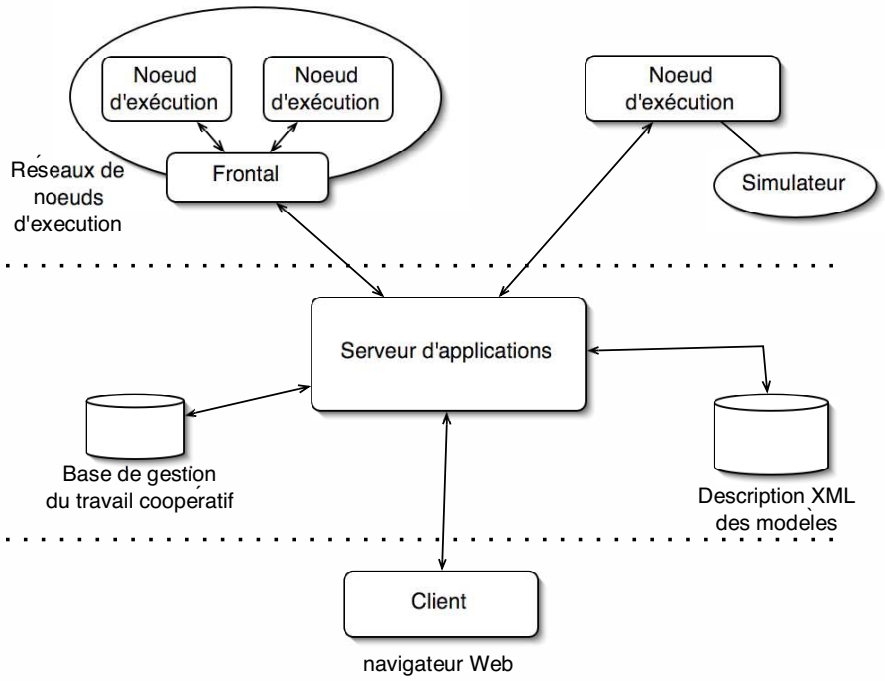


FIG. 2.33 – Architecture globale de VLE "version 2000"

En conclusion, VLE "version 2000" est une généralisation de la plate-forme de 1998. Elle intégre tous les concepts introduits par le *framework* et les langages de spécification. Dans la pratique, le chemin est encore long. Il reste à développer plusieurs modules : des connecteurs pour des clusters, des wrappers pour certains formalismes, des afficheurs de traces de simulation, des outils d'analyse statistiques, ...

2.3 Conlusions et perspectives

Ce deuxième chapitre a été consacré à l'exposé de notre cadre conceptuel de construction (*framework*) de multi-modèles : VLE (*Virtual Laboratory Environment*). VLE a été présenté selon deux angles de vue qui émergent deux pistes parallèles que l'on explore : la spécification formelle de multi-modèles et la modélisation centrée individus et multi-agents. La permière piste nous a conduit à présenter une hiérarchie de niveaux conceptuels pour la multi-modélisation inspirée des travaux de Zeigler & Sarjoughian. Ce découpage met en évidence quatre niveaux : le niveau opérationnel, le niveau simulation, le niveau modèle et le niveau sémantique. Chaque niveau fait l'objet d'un positionnement par rapport aux activités actuelles de recherche dans le domaine ce qui nous a conduit à proposer pour certains niveaux des nouvelles approches. Les niveaux en question sont principalement les niveaux simulation et modèle. Nous évoquons juste la question de la sémantique.
Ayant défini ces différents niveaux, nous nous sommes orientés vers une proposition de formalisation des modèles et des modèles couplés. Cette proposition se décompose en deux parties : la définition des éléments des modèles (structure, dynamique, interaction, ...) et leur spécification en XML. La définition des éléments fut l'occasion de poser clairement les concepts sous-jacents.

Par exemple, il était important de se poser la question du temps : quelle est le temps que l'on manipule au sein d'un modèle ? quels sont les éléments qui sont dépendants du temps ? ... Nous avons aussi fait le point sur l'intégration de l'expression de la dynamique du système dans une optique de multi-formalismes. Deux approches a été dégagées : le "tout DEVS", le *mapping* et le *DEVS-compliant*, le *wrapping*. Ces deux approches permettent de spécifier la dynamique du système avec soit une démarche totalement DEVS soit en encapsulant le formalisme utilisé dans une interface DEVS. Cette encapsulation permet de rendre le formalisme compatible avec DEVS au prix d'une réflexion sur le couplage un formalisme avec DEVS. La réflexion se résume principalement à adapter le formalisme au paradigme à événements discrets.
La construction du *framework* a été réalisé en parallèle de nos actions de modélisations en biologie marine. La deuxième partie de ce chapitre s'est donc intéressé à la présentation des deux versions de plate-formes qui ont été développé pour répondre à nos besoins de modélisations. La première fut purement orientée multi-agents et modélisations de systèmes spatio-temporels. Tandis que la deuxième qui est en cours de développement, est une migration de la première sur la multi-modélisation et la simulation distribuée. La deuxième version de VLE intégre la modélisation multi-agents de systèmes spatio-temporels. Grâce à l'ajout de la spécification XML du couplage de modèles, VLE permet de définir des modèles couplés à des modèles multi-agents. De plus, les simulations peuvent être distribuées.

Les perspectives sont nombreuses. La première consiste à poursuivre nos développements et nos études des *wrappers* DEVS pour des formalismes tels que les équations différentielles, les *state-charts*, ... Notre objectif final étant de mettre à disposition un environnement de services Web pour la spécification de multi-modèles à base de multi-formalismes et pour la simulation distribuée.
Les aspects sémantiques ont été jusqu'à présent seulement évoqués. Il est nécessaire d'explorer cette piste pour concevoir un outil d'aide au couplage. De plus, nous devons poursuivre notre réflexion sur les plans d'expériences. Une première syntaxe XML a été proposé mais il faut prendre en compte des concepts tels que l'analyse de sensibilité, la validation de modèles, le calibration de modèles, ldots

Chapitre 3

Les applications : l'écosystème marin et les systèmes industriels

Résumé

Les théories, les concepts et les outils développés sont l'émergence de réflexions fondées sur notre expérience de modélisateurs de systèmes naturels. Dans ce chapitre, nous allons présenter nos actions en terme de modélisation d'un écosystème marin, d'un système théorique de diffusion de particules et d'un système de production. Nous montrerons comment ces trois systèmes ont été pour nous une source de questionnements et d'applications de nos approches. En effet, les notions de couplages dynamiques de modèles, de paramétrages de modèles par calcul émergent, de spécifications formelles à événements discrets et d'utilisation de la plate-forme VLE seront explicités sur des modèles réels et artificiels.

Sommaire

Les théories, les concepts et les outils développés pour VLE ont été testés et validés, pour une grande partie, dans le cadre d'un travail pluridisciplinaire s'intéressant à la compréhension de la dynamique du système "sels nutritifs / phytoplanctons / zooplanctons". Ce cadre de travail a été et est encore une grande source de questionnements qui permet de dégager des méthodes et des démarches dans le domaine de la modélisation et de la simulation des systèmes complexes. Le va-et-vient entre plate-forme (ou *framework*) et cas d'étude reste prépondérant dans l'approche

de nos travaux. Ce principe reste à nos yeux essentiel à toute démarche scientifique et permet un enrichissement croisé entre les chercheurs en écologie marine et les chercheurs en informatique que nous sommes.
Ce travail est le fruit d'une collaboration de cinq ans qui à l'origine a été initié par Philippe Preux et Christophe Cambier [CAM94] [CAM97]. L'objectif originel de ce travail était d'identifier les règles comportementales du copépode (voir figure 3.1) et les influences de l'environnement sur son comportement en utilisant des approches centrées individus. Ayant repris le flambeau dès 1998, nous avons élaborés le premier modèle centré individus du système. Comme nous allons le voir par la suite, ce premier modèle était une simplification du système, cependant qui nous a permis de valider l'approche centrée individus sur ce type de systèmes [RAM98] et d'obtenir des premiers résultats intéressants pour la biologie [SEU99b]. Au fur et à mesure que nos travaux ont progressé, nous avons intégrés les actions du PNEC (Plan National d'Etudes Côtières) et plus particulièrement l'ART2[1]. Cette intégration nous a permis d'étendre les problématiques abordées et surtout de développer nos outils et nos méthodes de construction de modèles multi-échelles et multi-formalismes. Ces développements vont nous conduire jusqu'à l'intégration de DEVS dans nos modèles, et à l'obtention de résultats dans le domaine de la biologie théorique de plus en plus intéressants [DUB02a][DUB03b].

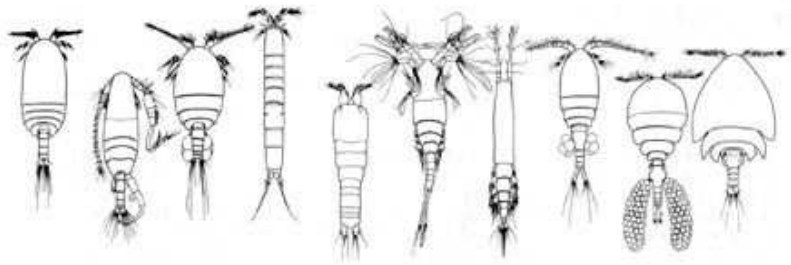


FIG. 3.1 – Quelques copépodes

Nous décrirons dans ce chapitre les modèles développés (le modèle en deux dimensions mais aussi le modèle en 3 dimensions et une ouverture sur un modèle distribué) et nous chercherons à dégager des réflexions générales sur les approches en biologie théorique et sur la méthodologie de couplage de modèles dans un cadre réel.
Les travaux en biologie marine représentent une partie importante du travail réalisé en collaboration avec des non-informaticiens. Un deuxième domaine a été abordé : les systèmes industriels. Nous montrerons à cette occassion l'apport de VLE pour la modélisation et la simulation des systèmes industriels.

[1]ART2 : Action de Recherche Thématique dont le thème central est "Dynamique des populations : structures hydrodynamiques et cycles biologiques"

3.1 L'écosystème marin et le modèle "Copépode"

L'objectif du premier projet de collaboration entre le Laboratoire d'Informatique du Littoral et la Station Marine de Wimereux est d'identifier les règles comportementales du copépode (voir figure 3.1) et les influences de l'environnement sur son comportement : l'hypothèse est que le copépode a un comportement actif de recherche de nourriture (les cellules de phytoplancton). Pour cela, nous nous étions proposés de croiser l'observation sur le vivant, les résultats de modèles mathématiques et la modélisation par agents. La distribution du phytoplancton est fortement hétérogène dans le milieu. L'un des premiers constats consiste à confirmer que cette hétérogénéité a une influence sur le système. Les résultats actuels [CAP96] montrent que cette hétérogénéité influe sur le bilan énergétique du copépode. En mesurant la quantité d'azote absorbée lors de la nutrition, les observations montrent que le rendement du copépode (énergie dépensée/énergie ingérée) varie en fonction du type de distribution de la nourriture [CAP96][BUN93] et de l'adaptation de son comportement aux conditions environnementales [TIS90]. En effet, un milieu perturbé jusqu'à un certain niveau, est propice aux échanges, et favorise le taux de rencontre du copépode avec les particules de phytoplancton et donc augmente son rendement. Il reste maintenant à montrer l'influence du comportement du copépode dans de telles conditions. L'étude de ce comportement est difficilement compatible avec une approche mathématique, comme nous allons le voir. Aussi, nous nous sommes donc dirigés vers les approches centrées individus.

3.1.1 Le modèle 2D

Les travaux de recherche ont débutés en 1998 avec la publication des premiers résultats [RAM98][SEU99b]. L'approche adoptée dans ce premier modèle est classique et relativement simple. L'espace est reduit à deux dimensions bien que l'environnement, la masse d'eau, soit clairement à trois dimensions. Cette simplification a peu d'influence sur les résultats. On montrera plus tard que le passage à 3 dimensions ne démentira pas les résultats obtenus dans ce premier modèle. L'espace est discrétisé et représenté par une grille de cellules. Dans chacune des cellules de la grille, une quantité de cellules de phytoplancton est présente et le copépode est attaché à une et une seule cellule à chaque instant. L'avantage de cette représentation du phytoplancton est de représenter le phytoplancton de manière discrète. En effet, dans la majorité des modèles, le phytoplancton est représenté par un gradiant de concentration dans le milieu. Or, dès lors que le copépode est individualisé c'est à dire qu'il est représenté en tant qu'individu avec ses coordonnées, son volume et ses caractéristiques individuelles, il est plus simple d'adopter la même représentation pour les entités avec lesquelles il va intéragir. De plus, le phytoplancton est par nature une entité spatialisée et individualisée et la description des interactions est d'autant simplifiée. Il faut noter que l'individualisation des entités est chose facile en modélisation centrée individus mais est loin d'être simple et manipulable en Mathématiques. Le deuxième avantage est que la distribution du phytoplancton est fortement hétérogène dans le milieu. Dans notre approche, il suffit de générer des distributions dans les cellules de la grille qui respectent le modèle de distribution. Laurent Seuront, étudiant en thèse en 1998 et membre de notre groupe de travail, a montré dans ses travaux que la distribution du phytoplancton peut se caractériser par des lois de distribution multifractale [SEU99a][SEU96]. Pour compléter le modèle, il ne reste alors plus qu'à définir les règles comportememtales du copépode.

Avant de développer le modèle, arrêtons nous un instant sur les modèles existants en 1998 afin de montrer les limites des approches mathématiques sur la modélisation des interactions. Le copépode (voir figure 3.1), petit organisme marin appartenant à la famille des zooplanctons,

est représenté la plupart du temps par des modèles de type "boîte noire" ou en compartiments (voir figure 3.2) ou mathématiques (voir figure 3.3) et principalement à base d'équations différentielles. Les outils mathématiques sont les outils privilégiés des modélisateurs en biologie. Ce choix se justifie par les capacités d'expression des Mathématiques et par la capacité d'étude formelle des modèles (convergence, cycle limite, ...). Ces modèles cherchent à décrire chaque processus intervenant dans la vie de l'organisme en identifiant les flux d'entrée, les flux de sortie et les fonctions de transfert.

Attardons nous sur le processus d'ingestion des proies dans le cas du phytoplancton (cf. Figure 3.2). Ce processus caractérise parfaitement le lieu des interactions entre les proies (le phytoplancton) et le prédateur (le copépode). 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) que l'on exprime en équivalent azote dans le modèle présenté ici, ou en déchets (pelotes fécales). Cette transformation est continue : à chaque Δt où $\Delta t \to 0$, une quantité Δq transite (cette quantité est proportionnelle à la quantité stockée dans l'estomac). L'énergie utilisable est soit consommée (métabolisme, digestion ou nage) soit stockée (pour la production d'oeufs chez les femelles, par exemple). Quant aux déchets, ils sont éjectés.

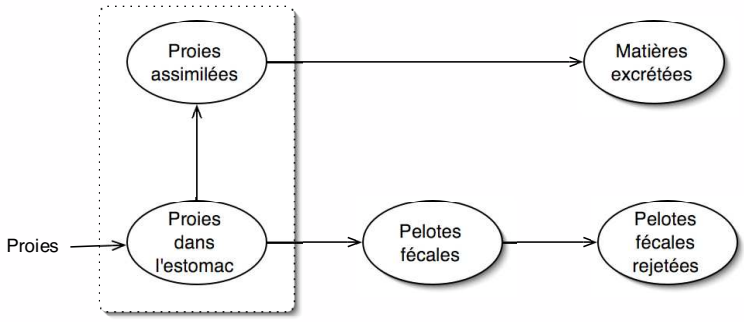


FIG. 3.2 – Modèle en boîte du processus d'ingestion

Il existe des modèles analytiques qui ont pour ambition de représenter, au mieux, les processus biologiques qui régissent les copépodes. [CAP96] propose un modèle (voir figure 3.3) 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.

Prenons, à titre d'illustration, l'équation 3.1 qui décrit mathématiquement la variation du nombre de proies dans l'estomac de l'animal (X_1). Cette variable dépend de trois autres variables :

- le taux d'ingestion I qui mesure le nombre de proies rencontrées par le copépode en fonction de sa satiété et du niveau de turbulence du milieu,
- le taux d'assimilation A, proportion d'azote apportée par les proies qui devient utilisable,
- le taux de formation des pelotes fécales, proportion d'azote apportée par les proies qui n'est pas utilisée mais évacuée en pelotes fécales.

Les deux dernières variables modélisent, *a priori* et en fonction des connaissances actuelles, la digestion. La variable I écrit le processus de capture du copépode en fonction de sa satiété et de l'environnement. La capture fait intervenir la notion d'interaction entre deux entités spatialisées mais comment le modéliser à l'aide d'équations ? Il existe deux approches : une approche phéno-

$$\frac{dX_1}{dt} = I - A - F \tag{3.1}$$

$$\frac{dX_2}{dt} = A - \frac{C}{M_N} \tag{3.2}$$

$$\frac{dX_3}{dt} = F - G \tag{3.3}$$

$$\frac{dX_4}{dt} = G \tag{3.4}$$

$$\frac{dX_5}{dt} = C_{st} + C_{sda} + C_{sw} \tag{3.5}$$

FIG. 3.3 – Modèle analytique du processus d'ingestion

ménologique où on va chercher à construire une équation sans expliciter les mécanismes mis en œuvre et une approche mécaniste où les processus sont décrits à l'aide des mécanismes. Dans le modèle proposé, l'auteur utilise une approche phénoménologique et décompose la variable I où les différents paramètres vont avoir comme objectif de modéliser l'apport des phénomènes dans le processus global (ici, l'ingestion).

$$I = (\beta_{behaviour} + \beta_{turbulence} N_p F_A) \tag{3.6}$$

I est alors directement liée à la capacité de rencontre du copépode. Dans le modèle analytique proposé, cette capacité de rencontre est fonction de quatre paramètres :

- $\beta_{behaviour}$ la contribution du "comportement", celle-ci est fonction du rayon de perception du copépode, du rayon 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, selon le mode de prédation (en suspension ou en embuscade), la densité des proies dans le milieu.

Ne pouvant décrire de manière mécaniste l'interaction proie-prédateur, les modèles mathématiques doivent passer par des paramètres souvent complexes et par la construction d'une relation. Cette construction est très souvent empirique. Les paramètres ont pour rôle de tenir compte de l'influence du processus d'interaction. Ici, les paramètres $\beta_{behaviour}$ et $\beta_{turbulence}$ ont la charge de représenter l'influence du comportement et de la turbulence sur l'efficacité de capture du copépode. Le modélisateur doit alors construire des fonctions pour évaluer ces paramètres. Dans le cas de la contribution du comportement, le rayon de perception, la taille des proies et la différence de vitesse entre proie et prédateur sont pris en compte dans une savante combinaison.

L'idée que l'on cherche à défendre est : n'existe-t-il pas une méthode qui permettrait de construire ou de valider les formes analytiques de certains paramètres par l'utilisation d'un ou de plusieurs paradigmes adéquats pour modéliser les processus en jeu ?

X_1	Nombre de proies dans l'estomac
X_2	Nombre de proies assimilées
X_3	Nombre de pelotes fécales
X_4	Nombre de pelotes fécales évacuées
X_5	Energie dépensée exprimée en azote
I	Taux d'ingestion
A	Taux d'assimilation
F	Taux de formation de pelotes fécales
C	Energie dépensée par unité de temps
C_a	Pourcentage de proies assimilées par rapport aux proies capturées
T_g	Temps de transit (dépendant du nombre de proies dans l'estomac)
M_N	Masse en azote du copépode
G	Nombre de pelotes fécales évacuées par unité de temps
C_{st}	Energie dépensée pour le métabolisme par unité de temps
C_{sda}	Energie dépensée pour l'ingestion par unité de temps
C_{sw}	Energie dépensée pour le déplacement par unité de temps (nage + saut)

FIG. 3.4 – Définition des variables et constantes du modèle analytique de la figure 3.3

C'est cette idée émergente à l'époque qui va guider nos pas. Nous n'allons pas toucher la partie digestion du modèle mathématique. En revanche, notre modèle centré individus va se focaliser sur la représentation mécaniste du processus de capture. Nous considérons que le copépode adopte deux comportements distincts : une nage orientée à la recherche de nourriture et des sauts aléatoires. Ces comportements ont une influence directe sur le processus d'ingestion des cellules de phytoplancton. Nous allons donc nous intéresser exclusivement à ce processus et laisser de côté la partie digestion qui, néanmoins, n'est pas à négliger pour obtenir un modèle complet.

Le système se compose de trois entités : la masse d'eau, les cellules de phytoplancton et les copépodes. Dans un premier temps, nous ne considérons qu'un seul copépode à la fois. Cette restriction n'est naturellement pas due à une quelconque limite du simulateur mais induite par les conditions expérimentales *in vivo*. De plus, nous ne désirons pas introduire des phénomènes de compétition entre copépodes. Nous limitons ainsi consciemment la complexité du système. De plus, il est plus facile de construire des interprétations sur un modèle qui *a priori* paraît simple que de vouloir tout intégre dans le modèle et donc de manipuler un modèle trop complexe pour comprendre les interactions entre les éléments du modèle. La masse d'eau constitue l'environnement (ou le milieu) dans lequel évoluent les autres entités. La taille du copépode (1 mm) sert de longueur de base pour la discrétisation du milieu. Le milieu est considéré, pour l'instant, en deux dimensions et découpé en cellules d'1 mm^2. La quantité de phytoplanctons

est très importante (de 10 phytoplanctons par litre à 10^8 phytoplanctons par litre soit au maximum 10^4 phytoplanctons par cellule). Il n'est donc pas raisonnable pour l'instant de modéliser chaque phytoplancton par une entité discrète. La solution retenue consiste à définir, au niveau des cellules, une propriété "Nombre de phytoplanctons". On délègue la gestion de la nourriture aux cellules spatiales c'est-à-dire à l'environnement. Quant au copépode, il est représenté par un agent dynamique dont on va décrire la dynamique.

Comme mentionné dans le cadre général, les cellules de phytoplancton ne sont pas distribuées de manière quelconque dans l'espace. Dans la plupart des modèles existants, la question de la distribution du phytoplancton est simplifiée et on considère que les cellules de phytoplancton sont uniformément réparties dans l'espace. Cette hypothèse est en effet très souvent retenue. Or, il a été montré [SEU99][SEU96] que c'est loin d'être le cas. Pour y pallier, il faut introduire dans les modèles mathématiques des fonctions spatialisées de gradiant de concentration ce qui implique un niveau tel de complexité que la résolution analytique des équations devient alors impossible (dans l'état actuel des Mathématiques). Dans notre approche de modélisation, le problème est nettement plus simple : il suffit d'affecter un nombre de cellules de phytoplancton dans les cellules de notre grille représentant l'espace. Ce nombre est généré par un algorithme de génération de nombres pseudo-aléatoires. Si toutes les précautions sont prises concernant ces algorithmes[2], on peut obtenir facilement des distributions qu'il suffit d'injecter dans nos modèles. Laurent Seuront a montré que les distributions en question sont des distributions multi-fractales. Nous n'en ferons pas la démonstration ici ce qui nous intéresse avant tout, c'est l'algorithme de génération aléatoire et le paramètrage de cette loi. Nous avons donc été obligé avec l'aide François Schmitt de construire l'algorithme de génération multi-fractale (seule une version à une dimension était disponible à l'époque).

La simulation est discrète. Nous allons donc faire évoluer les entités du modèle selon un pas de temps constant. Celui-ci est fixé par la durée correspondant au temps nécessaire à la plus petite action, c'est-à-dire la manipulation d'une cellule de phytoplancton par le copépode soit $\frac{1}{20}$ s pour l'espèce de copépode et l'espèce de cellule de phytoplancton considéré.

La plate-forme VLE proposait en 1998 qu'une seule possibilité de spécification de la dynamique des agents : les réseaux de Petri interprétés avec une bibliothèque d'instructions orientées systèmes spatio-temporels. La dynamique de déplacement du copépode a donc été modélisé à l'aide d'un réseau de Petri interprété. Ce réseau de Petri se divise en quatre parties :

- dès que l'on atteint la fin d'un cycle de 75 unités de temps[3] (u.t.), le copépode effectue un saut sans considérer ce qu'il l'entoure,
- pendant 20 u.t. (le temps de traverser une cellule de la grille), le copépode explore l'espace où il se trouve et si de la nourriture s'y trouve, il peut, à chaque unité de temps, capturer une cellule de phytoplancton,
- en revanche, s'il n'y a pas de nourriture, il continue à nager pour atteindre la cellule suivante,
- au bout des 20 u.t. nécessaires à la traversée d'une cellule, le copépode choisit une nouvelle cellule à explorer et s'y rend.

Intéressons-nous à la phase active de capture de nourriture. A l'exception des sauts aléa-

[2]Il faut bien étudier les propriétés (longueur de la séquence, périodicité, degré d'indépendance des sous-séquences, rapidité, ...) que l'on a besoin avant de choisir l'algorithme de génération aléatoire.

[3]Les données adoptées ici sont des valeurs moyennes et sont issues de [CAP96].

toires, le copépode nage et parcourt une cellule toutes les 20 u.t. Lorsqu'il a atteint ce délai, il change de cellule. Ce changement est fonction de la stratégie que l'on teste. Dans notre cas, la cellule disposant de plus de nourriture aura plus de chance d'être choisie. Localement, le copépode capture les cellules de phytoplancton. S'il n'a pas encore tout consommé, on vérifie s'il a "envie" de manger (fonction de satiété). En effet, le copépode diminue la quantité de nourriture qu'il absorbe en fonction de son niveau de satiété, lui-même directement lié, pour l'instant, au nombre de cellules de phytoplancton présentes dans l'estomac.

Ce premier modèle est une traduction mécaniste du modèle mathématique de Capparoy et Carlotti. On y retrouve des valeurs moyennes issues de bilans et des processus aléatoires, telle que celle de la satiété, par exemple. La même remarque est valable pour la partie digestion. Les éléments nouveaux sont issus de la prise en compte de l'individualité, de l'hétérogénéité du milieu et du comportement au niveau de la nage. Il faut aussi souligner que la construction des agents est nettement plus simple et fait intervenir un nombre réduit de paramètres par rapport au modèle mathématique. Du point de vue validation, trois pistes ont été suivies : la comparaison qualitative des trajectoires simulées et des trajectoires *in vivo*, l'analyse fréquentielle des trajectoires simulées et *in vivo* et la comparaison avec les résultats des modèles mathématiques.

Quels sont les grands résultats de ce premier projet et donc de ce premier modèle ?

Nous avons défini deux types de copépode en fonction de leur stratégie de nage : purement aléatoire ou orientée. La première stratégie s'apparente à une marche aléatoire. L'environnement se compose d'une grille 2D de 2500 cellules carrées (50x50). Chaque cellule est connectée à ces 8 voisines. Les cellules de phytoplancton sont réparties soit par patches suivant une distribution multifractale (voir figure 3.5 - tâches plus ou moins grises) soit uniformément. Dans le deuxième cas, les cellules de phytoplancton sont réparties selon une loi uniforme. Dans les deux cas, la densité moyenne est identique (2 cellules de phytoplancton par cellule de la grille). A l'aide des variables définies au niveau des entités Copépode, on mesure à chaque pas de la simulation : l'énergie, exprimée en pg[4] d'azote, contenue dans l'estomac (*gutEnergy*), l'énergie utilisable (*energy*), le nombre de cellules de phytoplancton capturées (*cellNumber*) et quatre variables du modèle analytique (X_3, X_4, T_g et C_a).

Nous disposons d'un outil de visualisation de trajectoires qui nous permet de tracer le chemin parcouru par une entité durant la simulation. En superposant la trajectoire du copépode étudié et la distribution des cellules de phytoplancton, on montre que dans le cas d'une distribution des cellules de phytoplancton par patches, la stratégie de la nage orientée est plus efficace (voir Fig. 3.5). Dans le cas de la nage orientée, le copépode passe de patch en patch. Si on superpose les courbes représentant la quantité de nourriture dans l'estomac du copépode en fonction du temps et selon la stratégie de nage (voir figure 3.6), il apparaît très rapidement que la nage orientée est favorable à l'alimentation du copépode du point de vue énergétique.

Si le copépode est plongé dans un champ homogène (distribué selon une loi uniforme), la stratégie de nage n'a aucune influence sur l'alimentation puisqu'il trouvera en probabilité de la nourriture dans toutes les directions en même quantité. En revanche, dans un champ hétérogène, la stratégie aléatoire conduit le copépode à tomber par hasard sur un patch de cellules de phytoplancton et surtout d'en sortir sans chercher à en profiter. Donc cette stratégie est moins efficace (voir figure 3.6).

[4]picogramme

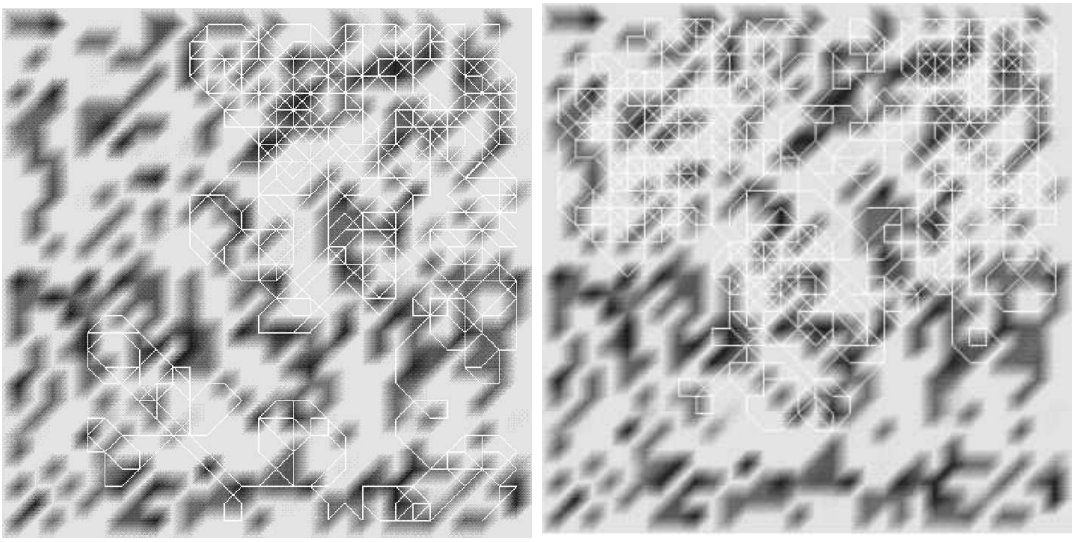


FIG. 3.5 – Trajectoire du copépode (nage orientée - gauche / nage aléatoire - droite)

En conclusion, on retrouve les principaux résultats énoncés dans [CAP96] pour une configuration de champ uniforme et de nage aléatoire. L. Seuront a montré dans sa thèse que les copépodes, dans la stratégie orientée, adoptent des trajectoires modélisables par des lois multifractales. Hormis ces résultats importants pour la modélisation en biologie, nous avons pu, grâce à ce premier projet, mettre en œuvre une technique alternative pour la modélisation en biologie théorique et surtout montrer que l'on est capable de retrouver des résultats similaires aux modèles mathématiques. Ces premiers résultats montrent aussi qu'à partir une description mécaniste des processus d'interactions proies-prédateurs, il est possible de faire émerger des grandes lois. En effet, les courbes de la figure 3.6 peuvent être ajustés par des courbes très connues en biologie et en physique et qui représentent en effet des processus similaires à ceux décrit par le modèle. Nous mettrons en évidence cet aspect avec le modèle 3D.

A l'issue de ce travail de recherche à la fois informatique et biologie, les questions restent nombreuses. Du point de vue informatique, trois types de problèmatiques s'offraient à nous : la formalisation des modèles de systèmes spatio-temporels, la formalisation du couplage de modèles et l'extension des capacités de modélisation de VLE. Les deux premiers points vont être partiellement atteint par l'étude de DEVS et de ses extensions. Quant au dernier, il est nécessaire de franchir la limite des deux dimensions en passant à trois dimensions, d'offrir le temps continu et l'espace continu, d'introduire dans VLE le multi-formalisme et le couplage de modèles.
Du point de vue de la biologie théorique, les réponses offertes par les premiers résultats issus du modèle 2D discret ont permis de croire que l'on pouvait :

- aller plus loin dans les détails des processus individuels,
- introduire des mécanismes liés à la turbulence,
- déduire par émergence des lois intégrants des mécanismes individuels,
- ...

Les modèles centrés individus étaient naturellement déjà connu en 1998 mais aucun modèle en biologie marine à petite échelle existait. De plus, la prise en compte de la relation proie-prédateur et de l'hétérogénéité spatiale et la possibilité de faire émerger des lois permettaient de croire que

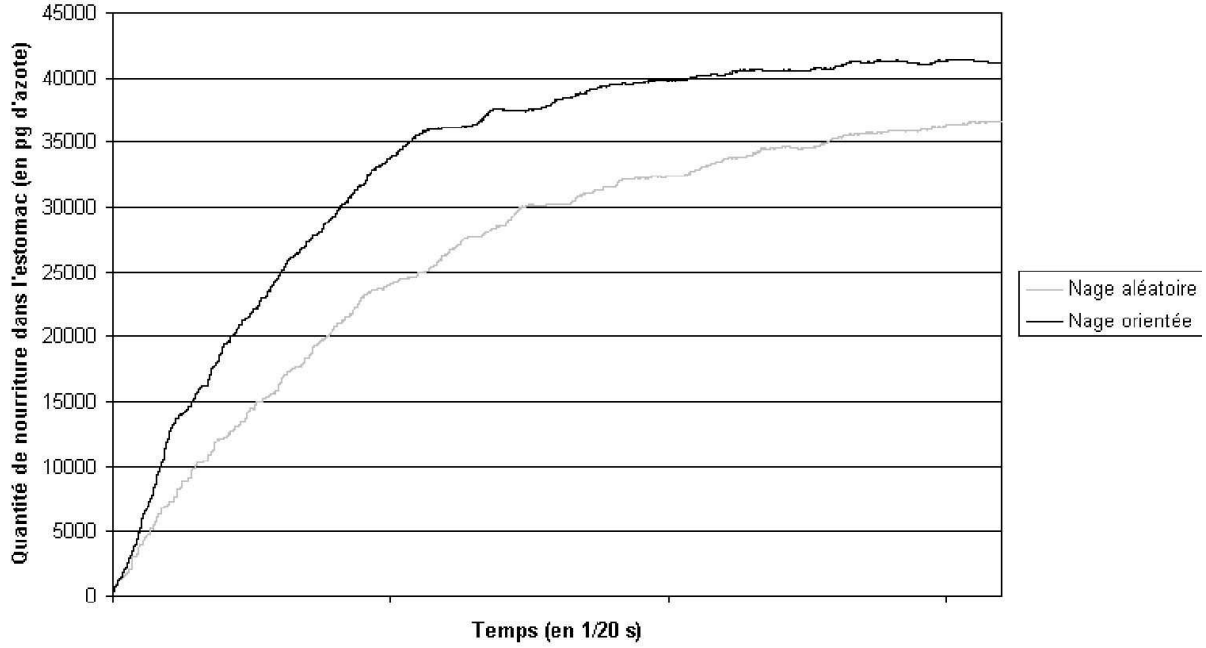


FIG. 3.6 – Quantité de nourriture (en pg d'azote) dans l'estomac du copépode en fonction du temps

l'on pourrait intégrer ces lois issues des petites échelles dans des modèles mathématiques de plus grandes échelles tels que les modèles de dynamique de populations.

3.1.2 Couplage de modèles, transfert d'échelle et changement d'échelle

Lorsqu'un observateur désire observer un système, il fixe les grandeurs physiques, biologiques, ..., qu'il désire observer. Or, observer un système implique aussi le choix d'une échelle d'obervation. L'observateur doit alors utiliser les instruments de mesure adéquats. L'échelle d'observation et l'instrument de mesure sont intimement liés. Du point de vue de la modélisation, l'échelle d'observation impose aux modélisateurs des choix de variables (la mesure du modélisateur) et des hypothèses. Dans notre cas, lorsque l'on parle de système, on s'intéresse en réalité aux systèmes spatio-temporels. Dans ce cadre, on distingue deux types d'échelles : l'échelle spatiale et l'échelle temporelle. Les processus observés sont caractérisés par un temps et par une granularité spatiale caractéristique. La construction d'un modèle s'articule alors autour d'une échelle de temps et d'une échelle d'espace. Ces échelles seront caractéristiques du modèle.
Dans notre démarche de couplage de modèles, l'aspect échelle d'observation ne peut pas être ignoré et nous allons montré quelle est la démarche que nous avons développé pour répondre au couplage de deux modèles [DUB03b]. Un premier modèle est le modèle centré individus du copépode. Nous avons opté pour une modélisation à événements discrets avec un espace à 3 dimensions pour des questions d'efficacité et de cohérence des hypothèses sous-jacentes. Le modèle se voit néanmoins légèrement complexifié par l'introduction de plusieurs copépodes en compétition sur le même champ de phytoplanctons. Le deuxième modèle est un modèle à

base d'équations différentielles modélisant la dynamique de deux populations en interaction proie-prédateur. Ces deux modèles sont exprimés à deux échelles de temps différentes. De plus, l'espace fait parti explicitement du premier modèle mais pas du second. Et pour compléter le tableau, les formalismes utilisés sont totalement différents. C'est une fois encore, le cadre idéal pour valider nos idées !
La question que l'on pourrait se poser est "pourquoi vouloir coupler ces deux modèles ?". L'idée défendue ici est la collaboration entre deux modèles afin qu'ils s'enrichissent mutuellement. Illustrons ce concept de collaboration. Le modèle de dynamique de populations décrit l'évolution temporelle des concentrations en prédateurs et en proies du système mais n'explicite pas les relations qui existent entre les prédateurs et les proies au niveau individuel et tout élément ayant un impact sur cette relation. La plupart du temps, des hypothèses simplificatrices se cachent derrière ces équations comme par exemple, l'homogénéité du milieu[5]. Des formulations du modèle de dynamique intégrent naturellement des conditions plus complexes à condition d'y intégrer des relations mathématiques complexes. D'autre part, on peut construire des modèles représentant les interactions individuelles et les éléments influençant ces interactions. L'idée est alors d'enrichir les modèles de dynamique de populations par des modèles au niveau individuel afin d'intégrer les aspects individuels au niveau population. Nous allons donc développé ces idées en montrant comment paramétrer un modèle mathématique par un modèle à base d'agents, comment faire émerger des lois de modélisations centrées individus et comment mettre en œuvre le couplage dynamique.

3.1.2.1 Application à un modéle proie-prédateur en écologie marine

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 [DUB03c]. 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 faisons une synthèse de ce travail et nous montrons que l'utilisation conjointe de 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. Le modèle centré individus utilisé est un modèle en trois dimensions. Une formalisation est proposée par la suite (voir 3.1.3). Les résultats présentées ici sont le fruit des actions inscrit dans le PNEC. Il faut donc associer tous les acteurs de ses actions : J. C. Poggiale, Y. Lagadeuc, F. Carlotti, O. Arino, P. Preux, R. Duboz et moi-même.

Construction et paramétrage de la réponse fonctionnelle à l'aide du modèle agent

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. C'est cette réponse fonctionnelle qui nous a déjà permis de valider le modèle 2D. Traditionnellement, cette fonction que l'on notera $G(.)$ est considérée comme dépendante uniquement de la quantité de proies. Dans la version originale du modèle de Lotka-Volterra de 1925 [LOK25][VOL26] 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

[5]Les proies et les prédateurs sont distribués spatialement uniformément.

trophique est dite "densité-dépendante" et s'écrit (équation 3.7) :

$$G(N) = \frac{\alpha N}{\beta + N} \tag{3.7}$$

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

L'équation de Holling est la première d'une longue série. Ainsi, en élevant les termes β et N au carré, Real [REA77] propose une variante de l'équation de Holling prenant mieux en compte l'impossibilité pour les prédateurs d'augmenter indéfiniment leur efficacité de prédation (*i.e.* leur taux d'ingestion) en fonction de la concentration des proies. En 1989, P. Arditi propose une équation qui prend en compte le nombre de prédateurs. Dans un des ces articles [ARD89], 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 3.8.

$$G(N, P) = \frac{\alpha N}{\beta N + \gamma P} \tag{3.8}$$

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.

La réponse fonctionnelle reflète un ensemble de processus qui existe 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 paramètres de l'expérience, et notamment le niveau d'hétérogénéité des distributions, nous pourrons étudier par la suite l'impact de cette hétérogénéité sur la dynamique d'un système proies-prédateurs classique au niveau population. L'impact de l'hétérogénéité sur les bilans individuels a déjà été mis en évidence par le modèle 2D (voir 3.1.1).

Concentration et hétérogéniété

Nous nous intéressons ici au paramétragecôté 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 centrées individus.

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[6]. Le deuxième paramètre à faire varier est le degré d'hétérogénéité. Or cette variation n'est pas simple à contrôler. Nous avons donc choisi deux techniques de distribution particulières suivantes :

- distribution homogène : la distribution 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'un côté du cube et n le nombre de particules.
- distribution hétérogène : on tire 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, pour chaque particule, nous tirons les coordonnées des particules suivant une loi normale centrée sur un point virtuel tiré au hasard, suivant une loi uniforme. En jouant sur l'écart type de la loi normale, nous jouons sur la dispersion des paquets et ainsi sur l'hétérogénéité de la distribution.

Ces choix sont un compromis. En effet, L. Seuront dans sa thèse a montré que la distribution du phytoplancton était la résultante du processus intermittant qui se modélise parfaitement avec la loi multifractale. Dans le cas du modèle 2D, nous avions pu adapter une version à une dimension de la distribution multifractale à 2 dimensions. La généralisation à trois dimensions n'est pas encore réalisée et s'avère d'ores et déjà complexe. Quelles sont les conséquences de ce compromis (de cette approximation) ? Dans le modèle multifractal, le contrôle de l'hétérogéniété est possible grâce aux 3 paramètres de la loi. Dans notre cas, le contrôle est beaucoup plus indirect et il est difficile de construire la relation entre l'écart-type des noyaux et le degré d'hétérogénéité. Le problème est aussi présent dans la première technique pour les distributions homogènes. Si la concentration est faible et étant donné le caractère discret des distributions, le degré d'hétérogéniété est alors incontrôlable ou inatteignable. De plus, à partir de quel degré d'hétérogéniété, une distribution est dite homogène. Toutes ces questions ne sont que partiellement résolues. Il faut donc être vigilent dans l'interprétation de nos résultats.

[6]L'estimation du taux d'ingestion est délicate car elle repose sur l'hypothèse de concentration constante or cette concentration fluctue au cours du temps sous l'effet de la prédation. Certaines précautions doivent donc être prises.

Un mot sur une façon simple de mesurer l'hétérogénéité d'une distribution. En écologie, une technique possible est la suivante [FRO95] : on divise 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. Nous comptons le nombre de cellules par petits cubes et calculons l'écart type et la moyenne sur l'ensemble des cubes. Plus l'écart type est grand par rapport à la moyenne, 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. Où est la limite entre hétérogénéité et homogénéité ? Considérant notre calcul, et en toute rigueur, seul un écart type nul nous assure d'une parfaite homogénéité. Cependant, nous considérons que pour un degré d'hétérogénéité $\xi < 1$ (écart type inférieur à la moyenne) la distribution est relativement homogène.

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 mesures du taux d'ingestion à concentration constante en faisant varier ξ et le nombre de copépodes. La figure 3.7 nous montre deux exemples de courbes obtenues par simulation.

Nous pouvons découper les courbes de la figure 3.7 en trois parties :

- $0,5 < \xi < 1$:
  Les valeurs de ξ considérées ici correspondent à une distribution homogène. Le taux d'ingestion simulé paraît chaotique avec une tendance à croître.
- $1 \leq \xi \leq \{2; 2,5\}$:
  La distribution devient de plus en plus hétérogène et le taux d'ingestion croît jusqu'à un maximum dépendant du nombre de prédateur.
- $\xi > \{2; 2,5\}$:
  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 3.7 nous montre que l'hétérogénéité n'a pas le même effet sur le taux d'ingestion selon son importance. Dans un premier temps, elle va être bénéfique aux prédateurs en augmentant le taux d'ingestion, puis défavorable. Nous obtenons la même 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.
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). Nous vérifierons cette hypothèse un peu plus tard. À ce niveau, nous pouvons confirmer l'hypothèse d'Arditi d'une relation entre l'hétérogénéité de la distribution des proies est la forme de la réponse fonctionnelle des prédateurs.

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

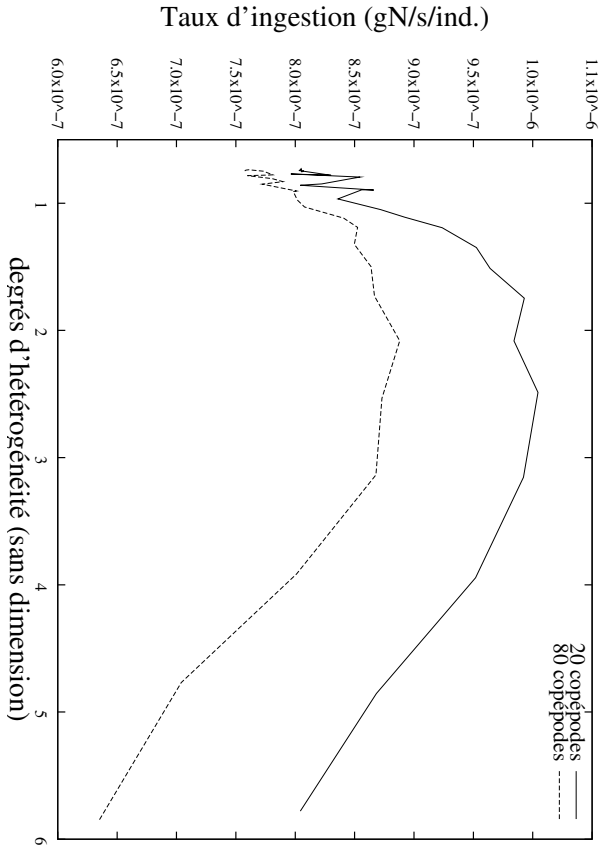


FIG. 3.7 – Variation du taux d'ingestion moyen par individus en fonction du degré d'hétérogénéité de la distribution des proies. Chaque courbe correspond à la moyenne de 30 simulations effectuées pour 20 copépodes (trait plein) et 80 copépodes (pointillés) pour une concentration constante de proies de $6,25.10^{-6} gN/l$.

de la réponse fonctionnelle est donnée par la représentation graphique de la valeur du taux d'ingestion instantané calculée en fonction de la concentration en proies dans le milieu.

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 répéterons chaque simulation 30 fois, ce qui, au regard du théorème centrale limite et sous l'hypothèse d'une distribution normale, tente à diminuer l'erreur sur les estimations des résultats.
La figure 3.8 nous montre la réponse fonctionnelle obtenue avec une distribution homogène des proies. 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 et nous l'avons déjà observé dès nos premiers travaux sur le modèle 2D. La distribution des proies n'est jamais uniforme et, comme nous le voyons avec la

figure 3.9, ceci a des conséquences sur la forme de la réponse fonctionnelle[7].

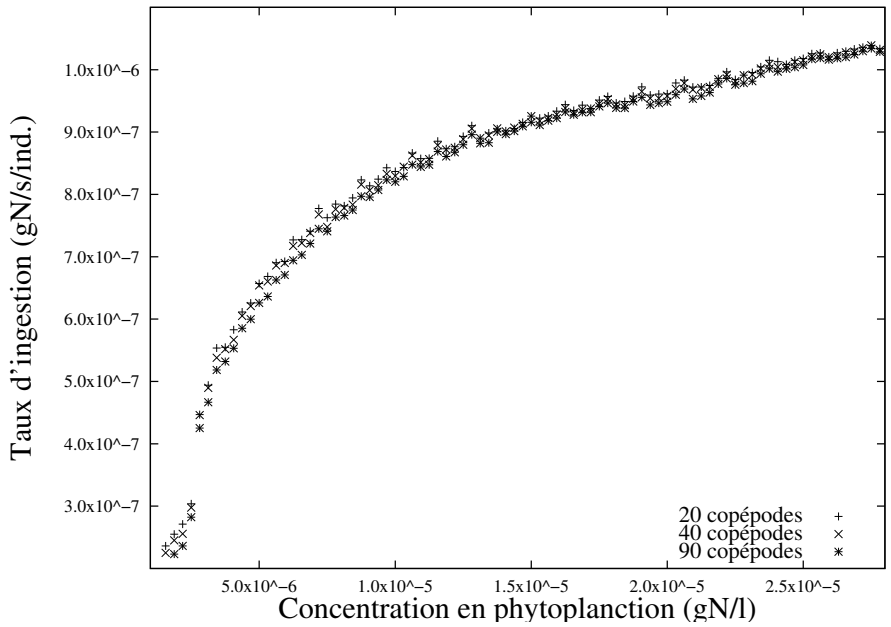


FIG. 3.8 – Réponses fonctionnelles simulées par le modèle agent pour différentes quantités de prédateurs en milieu homogène. Chaque point correspond à la moyenne de 30 simulations. Le taux d'ingestion reste identique avec l'augmentation du nombre de prédateurs. Cette expérience confirme l'hypothèse d'une réponse fonctionnelle non ratio dépendante en milieu homogène.

La figure 3.9 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.

Comme nous allons le montrer par la suite, l'idée finale de ce travail est d'étudier les démarches possibles de couplage de modèles. Avant cela, nous avons montré dans nos plus récents travaux [DUB03b] que notre modélisation mécaniste de la relation prédateur-proie possède un comportement identique à des modèles connus dans la littérature (Holling, Arditi, ...). La question à laquelle nous avons cherché à apporter une réponse est : comment ajuster une réponse fonctionnelle à nos résultats de simulation ? 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 d'espèces étudiées, de conditions expérimentales et dans notre cas, de la nature du modèle produisant cette réponse.

[7]Comme nous l'avons expliqué au paragraphe 3.1.2.1, 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.

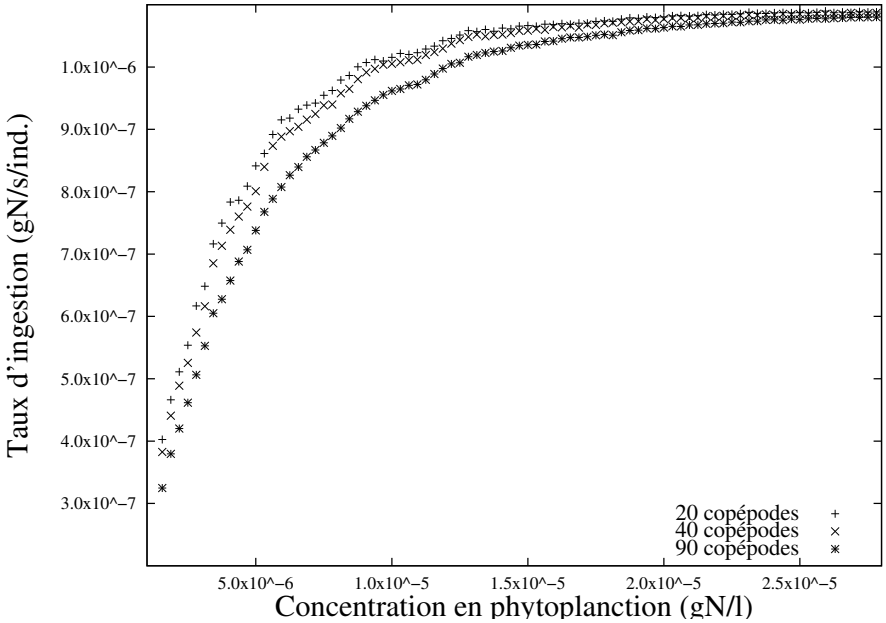


FIG. 3.9 – Réponses fonctionnelles simulées par le modèle IBM pour différentes quantités de prédateurs en milieu hétérogène ($\xi = 1,1$). Chaque point correspond à la moyenne de 30 simulations. La diminution de la vitesse d'accroissement du taux d'ingestion avec l'augmentation du nombre de prédateurs dans le milieu vérifie l'hypothèse d'Arditi d'une réponse fonctionnelle ratio-dépendante en milieu hétérogène.

En résumé, on a montré que les réponses fonctionnelles obtenues comme cela a été décrit dans le paragraphe précédent s'ajuste :

- avec le modèle de Holling pour les conditions homogènes,
- avec le modèle d'Arditi pour les conditions hétérogènes,
- et avec le modèle de Real lorsque les conditions sont hétérogènes et que le taux d'ingestion possède un seuil limite.

La thèse de Raphaël Duboz montre en effet que le modèle de Real (voir l'équation 3.9) est le modèle idéal pour l'ajustement vis à vis de toutes les hypothèses de notre modèle mécaniste (après une légère adaptation afin de tenir compte de la ratio-dépendance et devient de la forme $G(N,P)$). A partir de cette démonstration de l'ajustement de modèles classiques avec notre modèle plongé dans les mêmes conditions expérimentales, nous pouvons sérieusement nous pencher sur le problème du couplage. En effet, notre modèle mécaniste peut être considéré comme équivalent à un modèle mathématique.

$$g(N) = \frac{\alpha N^2}{\beta^2 + N^2} \tag{3.9}$$

où : N est la concentration en proie, α le taux d'ingestion maximum et β la constante de demi saturation[8].

[8] β^2 peut être remplacé par $\gamma^2 P^2 + \mu^2$ dans le cas de la ratio-dépendance

Couplage faible de modèles et transfert d'échelle

Nous proposons maintenant de coupler notre modèle d'agents réactifs avec un système d'équations différentielles ordinaires. Le terme de couplage doit être entendu ici comme le paramétrage d'un modèle par l'émergence des valeurs de paramètres de la simulation d'un autre modèle. De plus, nous allons impliquer deux modèles d'échelles spatio-temporelles différentes. Un tel couplage permet la coexistance de deux niveaux d'organisations dans la même simulation et ainsi d'étudier un transfert d'échelle (*i.e.* les conséquences des propriétés modélisées à l'échelle de l'individu sur une dynamique globale de populations). Cette question est fondamentale en écologie. Nous allons donc commencer par présenter les modèles à coupler puis nous montrerons au travers d'un exemple de simulation une possible utilisation de cette méthode.

Nous considérons un modèle classique de l'interaction proie-prédateur [PAV94] [LOK25] [VOL26] [BRO93]. Ce modèle est formée de deux équations différentielles ordinaires (voir le système 3.10). D'abord l'équation de la dynamique des proies qui se décompose en deux parties. Une première partie correspondant à la croissance 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. La deuxième partie correspondant à la pression de prédation exercée par le prédateur est modélisée par une équation particulière de la réponse fonctionnelle.

$$\frac{dN}{dt} = rN(1 - \frac{N}{K}) - G(N,P)P \quad avec : G(N,P)\ la\ réponse\ fonctionnelle. \tag{3.10}$$

Quant à l'équation décrivant la dynamique des prédateurs (équation 3.11), elle se décompose également en deux parties. Une partie de croissance par consommation des proies basée sur la réponse fonctionnelle pondérée d'un coefficient e de transformation des proies en prédateurs et une partie faisant intervenir un coefficient de mortalité m des prédateurs.

$$\frac{dP}{dt} = eG(N,P) - mP \tag{3.11}$$

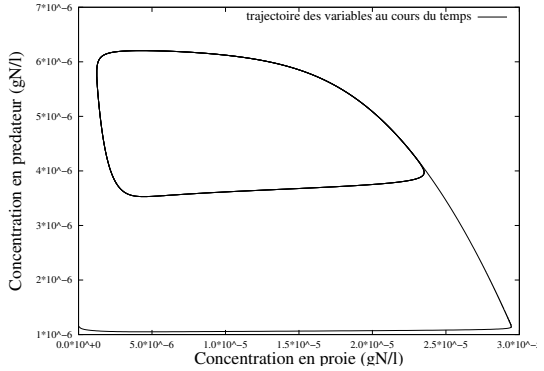
Les dynamiques de tels systèmes peuvent être étudiées analytiquement. Par exemple, si la réponse fonctionnelle est l'équation de Holling (voir l'équation 3.7) alors il apparaît des équilibres dans l'évolution temporelle des variables N et P sous forme de cycles limites stables pour certaines valeurs de paramètres [BRO93]. Il en est de même pour les réponses fonctionnelles ratio-dépendantes [JOS99]. Ces études théoriques permettent notamment de déterminer les valeurs des paramètres des équations du système pour lesquels nous observons des dynamiques particulières.
Considérons les valeurs suivantes de paramètres des équations 3.10 et 3.11 :

- $r = 1j^{-1}$ correspondant à un renouvellement journalier des cellules.
- $k = 3,12.10^{-5}\ gN/l$ $(1,56\ cellules.mm^{-3})$ la concentration maximale en phytoplanctons dans le milieu.
- $e = 0,02$ coefficient de proportionalité sans dimension
- $m = 0,02\ j^{-1}$ correspondant à une durée de vie de cent jours des copépodes.

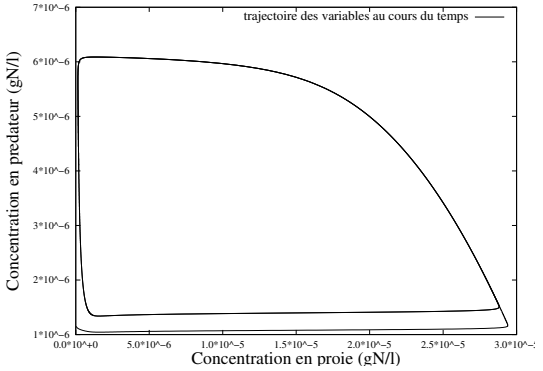
Nous choisissons la réponse fonctionnelle $G(N,P)$ définie par Real (voir l'équation 3.9 avec l'adaptation à la ratio-dépendance). Les valeurs des paramètres de cette équation sont prises pour deux niveaux d'hétérogénéité. Nous définissons deux expériences :

	expérience 1	expérience 2
ξ	$0,9$	$2,9$
α	$1,1.10^{-6}/738.10^{-9}$	$1,1.10^{-6}/738.10^{-9}$
μ	$3,1$	$1,1$
γ	$5,1$	$4,8$

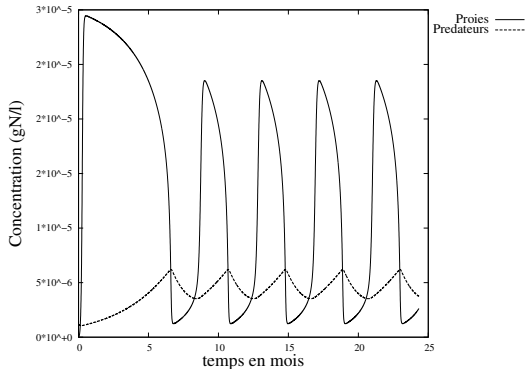
La valeur α est divisée par la masse d'un copépode *Acartia tonsa* adulte [CAP96], ceci afin de conserver la cohérence des unités du système d'équation. Les valeurs de μ et γ sont ainsi définies respectivement pour une distribution plutôt homogène ($\xi = 0,9$) et hétérogène ($\xi = 2,9$) des proies. La figure 3.10 nous montre les portraits de phase (évolution de P en fonction de N) du système formé par les équations 3.10 et 3.11 pour les deux expériences considérées. Pour la simulation d'un tel système, nous avons utilisé le shéma numérique classique de Rung-Kutta d'ordre 4.



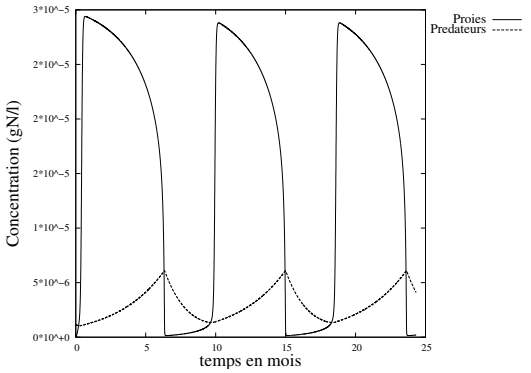
(a) distribution homogène ($\xi = 0,9$)



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



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



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

FIG. 3.10 – Résultats du paramétrage du modèle proie-prédateur à l'aide du modèle agent du copépode. (a) et (b) : Portrait de phase des expériences en milieu homogène ($\xi = 0,9$) et hétérogène ($\xi = 2,9$). (c) et (d) : Evolution au cours du temps des concentrations en proies et en prédateurs.

La figure 3.10 montre l'apparition d'un cycle limite dans les deux expériences. C'est-à-dire que les populations oscillent autour d'une concentration 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. Ceci peut paraître paradoxale. En effet, l'intensité de prédation est plus forte en milieu hétérogène, ce qui ralenti la croissance du phytoplancton. Néanmoins, cette croissance plus faible influence à son tour la croissance des prédateurs. Comme nous pouvons le voir sur la figure 3.10(a) et (b), le maximum de prédateurs 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 à 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. Il est possible d'aller plus loin dans l'exploitation de l'approche avec un couplage des deux modèles au lieu du paramétrage d'un modèle par un autre. Ce couplage se justifie ici par le fait que l'hétérogénéité n'est pas constante, mais est fonction de la concentration. En effet, aux faibles concentrations les cellules sont très dispercées et au contraire très regroupées aux fortes concentrations, ceci implique une hétérogénéité plus grande au faible concentration. Nous ne disposons pas de modèle mathématique qui nous permettrait de formaliser ce phénomène. Comme nous l'avons vu, le modèle agent rend compte de l'effet de l'hétérogénéité et de la ratio-dépendance sur la réponse fonctionnelle. En couplant les deux modèles, nous pouvons pallier à ce manque.

Couplage dynamique des modèles

Nous nous apprétons maintenant à coupler dynamiquement deux modèles de nature très différentes. On parle aussi de couplage fort en opposition au couplage faible synonyme de paramétrage. Le terme de couplage s'entend ici comme l'interaction continue des deux modèles. Le premier modèle effectue le calcul de son prochain état et informe ce second de son nouvel état. Ce dernier calcule son nouvel état en fonction de l'état du premier et informe à son tour le premier modèle de la variation de son état. Le premier peut alors calculer de nouveau son état en tenant compte de l'état du deuxième modèle (voir figure 3.11). Dans notre cas, nous avons montré que le modèle centré individus possède une dynamique équivalente aux modèles classiques rencontrés dans la littérature. Nous avons profité de cet avantage pour considérer le modèle centré individus comme une source de calcul émergent. Le modèle mécaniste est paramétré par les valeurs de N et P du système d'équations différentielles. Ces valeurs sont calculées à chaque pas de temps d'intégration du système d'équations différentielles. En retour, le modèle mécaniste calcule une estimation de la réponse fonctionnelle $G(N, P)$. Cette estimation vient à son tour paramètrer le système d'équations différentielles. Cette intégration met en jeu un échange entre deux modèles de nature différente et d'échelle temporelle différente. Le modèle de populations est basé sur une échelle temporelle caractéristique de l'ordre du jour tandis que le modèle mécaniste capture des phénomènes de l'ordre de la seconde voire de la minute.

De par la nature du couplage, 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 (voir thèse de Raphaël Duboz).

La figure 3.11 présente le système d'équations couplé avec le modèle agent du copépode. Elle est à mettre en correspondance avec la partie droite de la figure 3.12 de la page 113 pour l'aspect opérationnel de la méthode.

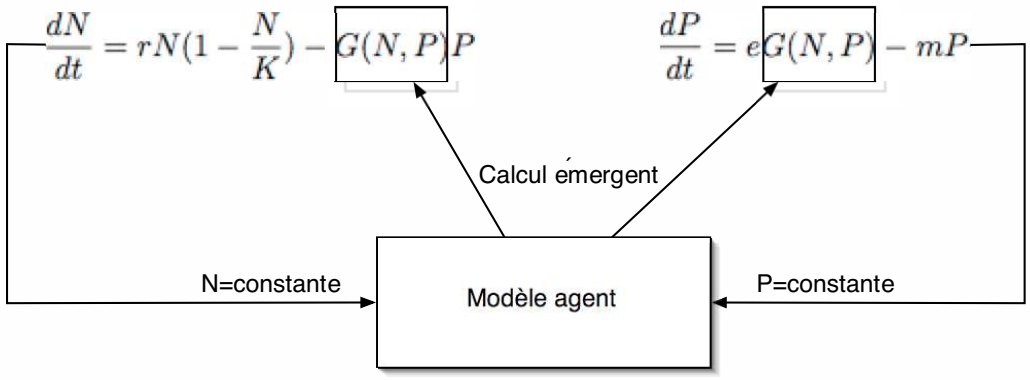


FIG. 3.11 – Schéma du couplage entre le modèle proie-prédateur et le modèle agent du copépode. Le modèle agent "simule" $G(N,P)$ et le modèle proie-prédateur détermine le nombre de copépodes et de cellules de phytoplancton à chaque itération du shéma numérique.

Á chaque pas de temps δt de résolution du shéma numérique, le modèle agrégé fournit le nombre de copépodes et le nombre de cellules de phytoplancton au modèle agent. Celui-ci simule le taux d'ingestion et donne en retour la valeur de $G(N,P)$ au modèle agrégé. Conformément à ce qui a été dit précédemment, le modèle agent simule une durée de $150s$ pour effectuer le calcul. Cette durée est nécessaire afin que l'estimation réalisée capture toutes les caractéristiques du système et que la concentration en proies soit stable. Nous considérons un pas de temps $\delta t = 0,1j$ pour le shéma numérique. Pour chaque δt, la valeur de $G(N,P)$ est constante.

Dans les deux exemples donnés précédemment (paramétrage et couplage), nous avons simulé une même durée dans un même espace pour une même concentration. Cependant, les mêmes méthodes peuvent être appliquées à des échelles différentes. Dans notre exemple, le changement d'échelles ne modifierait pas *a priori* les dynamiques donc la méthode ne présente pas d'intérêt dans ce cas precis. Néanmoins, cet exemple nous a permis d'introduire l'utilisation du couplage entre un modéle centré individus et un modéle 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 caratéristiques des deux modéles.

Nous nous inspirons ici d'une méthode connue en mathématique sous le nom de "méthode d'agrégation de variables" ou de réduction[9] [POG94]. Cette méthode est basée sur un constat simple. Si plusieurs échelles de temps sont mises en jeu, certaines variables sont lentes et d'autres sont rapides. 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 (états d'équilibre) sont considérées comme

[9]Cette méthode est également appelée "méthode de variation de la constante".

des constantes sur le domaine de variation des variables lentes. Ceci suppose que les équations des variables rapides aillent très vite à l'équilibre pour toute variation des variables lentes. Comment appliquer cette méthode dans le cas des modèles centrés individus ? Dans la communauté des système multi-agents, les résultats de simulations (ou traces d'execution) sont considérés comme un épiphénoméne, c'est-à-dire une manifestation extérieure et une conséquence de la simulation. Cet épiphénoméne peut à son tour être modèlisé, nous parlons ici de métamodélisation. De plus, si ce métamodèle 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 leur trace de simulation (ils ont une dynamique équivalente). Cette notion d'équivalence peut être mathématiquement renforcée par le calcul d'un indice de similarité. Dans ce que nous avons exposé précédemment, nous avons déjà illustré cette notion d'émergence lorsque l'on a montré que la trace de simulation du modèle centré individus pouvait être ajustée à certaines formes de réponses fonctionnelles. On peut aller plus loin est construire des modèles probabilistes où le but est de construire des distributions de probabilité en fonction des contraintes du modèle à variables lentes, par exemple. Ce travail est en cours. On cherche à construire une famille de distributions caractéristiques en fonction des niveaux d'hétérogènes. Dès lors que ces distributions sont construites, le modèle centré individus peut alors être remplacé par un générateur aléatoire simulant la distribution qui a émergé du comportement du modèle centré individus.

En utilisant le couplage entre deux mod eles à deux niveaux d'abstractions différents, le principe de séparation des échelles de temps des processus et le calcul émergent, nous proposons une méthode de couplage pour la simulation du transfert d'échelles. Cette méthode se résume en six points :

- modéliser un même système à deux niveaux d'abstractions différents.
- séparer les échelles de temps et d'espace des deux niveaux.
- considérer le modèle de bas niveau comme un laboratoire virtuel sur lequel on peut faire des expériences.
- identifier "expérimentalement" des paramètres ou des fonctions qui émergent du modèle de bas niveau et qui sont présents dans le modéle de haut niveau.
- paramétrer le modèle de haut niveau ou remplacer une partie par le modèle bas niveau.
- contraindre le modèle de bas niveau par le modèle de haut niveau qui impose les conditions initiales.

L'identification de paramètres, le paramétrage de modèle, le remplacement d'une partie d'un modéle et la contrainte d'un modéle par un autre ont été illustré. La figure 3.12 représente l'approche méthodologique citée plus haut dans le cas de l'écologie.

Sur cette figure, 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.

Remarques et réflexions

Le couplage de modèles est une technique trés prometteuse. L'enrichissement mutuel des modèles donne à penser que l'on peut aller plus loin dans l'explication des interactions entre petites et grandes échelles. En particulier, nous pensons qu'il est possible de coupler des modèles dits spatiaux avec des modèles centrés individus. L'exemple qui a été traité prédécemment se situe au niveau de la dynamique de populations mais ne tient pas compte des aspects spatiaux (ce qui n'est pas le cas dans le modéle centré individus). Il existe des modèles à base d'équations

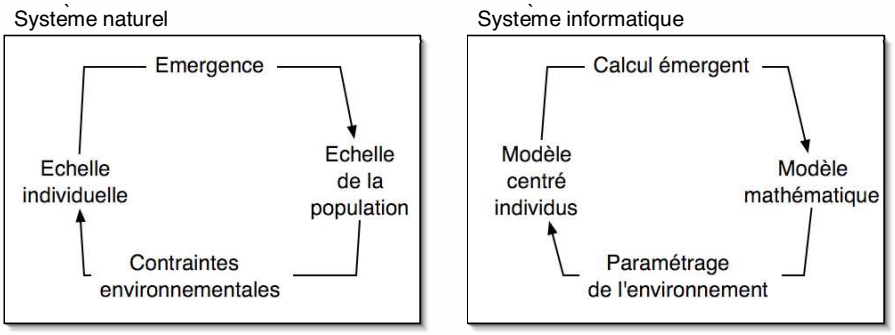


FIG. 3.12 – Approche conceptuelle pour la modélisation du transfert d'échelles entre deux niveaux d'organisation dans les systèmes naturels. La définition de calcul émergent et le principe d'expériences virtuelles sont à la base de cette approche

différentielles partiales intégrant l'espace qui ont pour objectif de représenter la dynamique de la population dans l'espace. Dans ce cas, l'espace est vu comme une grille de cellules et le modèle exprime la dynamique du système dans chaque cellule. Quel peut être l'apport d'un modèle centré individus dans ce type de modélisation ? On peut associer à chaque cellule un modèle centré individus paramétré avec les conditions environnementales et les conditions de population de la zone géographique représentée par la cellule. La condition essentielle à vérifier est que les conditions soient homogénes dans une cellule (concentration en phytoplancton, par exemple). La difficulté réside ici dans le fait qu'il faut autant de modèles centrés individus que de cellules ce qui va probablement poser certains problèmes de performances (voir Vers un modéle distribué - 3.1.4). Dans le cas d'une construction d'une réponse fonctionnelle et donc du remplacement d'une partie du modèle de populations, la question des performances ne se pose pas.

L'intérêt de la méthode est de pouvoir intégrer la variabilité individuelle dans un modèle spatialisé et donc de pouvoir étudier l'influence de la variabilité à l'échelle individuelle sur la dynamique globale. De plus, on remarque que ce couplage procéde à la fois à un changement d'échelles temporelles (comme dans la cas non spatialisé) et à un changement d'échelles spatiales. Les processus sont, en effet, descrits au niveau de l'individu (échelle de l'ordre du millimétre et du centimétre) et de la population (échelle de l'ordre d'un bassin océanique si l'on veut).

Le couplage de modèles peut être une piste pour les modèles de dynamique de populations structurées en classe d'âge. Ce travail a été initié en juin dernier avec François Carlotti. L'idée est d'utiliser la modélisation centrée individus couplée à des modéles mathématiques pour étudier l'impact de la variabilité individuelle et environnementale sur les changements de stade des copépodes. En effet, les différents stades d'évolution du copépode (du nauplii jusqu'à l'adulte) sont caractérisés par des vitesses de déplacement et des distances de perception particulières. Ces trois paramètres influencent le taux d'ingestion et donc la réponse fonctionnelle. 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'hétérogénéité optimaux pour chaque stade de développement, définissant ainsi des habitats favorables.

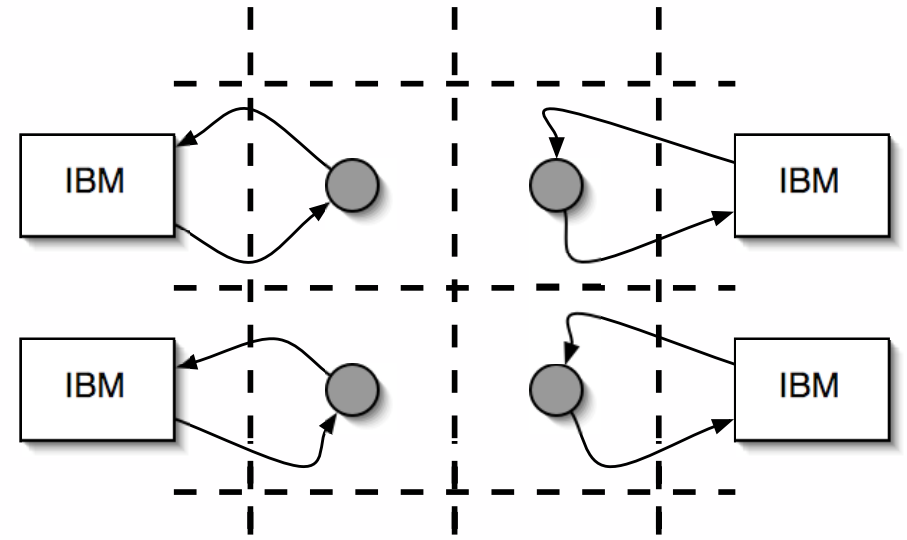


FIG. 3.13 – Schéma du couplage entre un modèle implémenté à l'aide d'un shéma numérique et un modèle centré individus. Pour chaque pas d'espace (cercles gris) et à chaque pas de temps, un couplage bidirectionnel est réalisé.

3.1.2.2 Une autre validation de la démarche

La démarche a été validé sur un autre exemple que l'on peut qualifier de cas d'école : la diffusion de particules animées d'un mouvement brownien. Ce travail est le fruit d'une collaboration avec le LISC du Cemagref de Clermont-Ferrand et a fait l'objet de deux publications [DUB03a][DUB03c]. Ce cas d'école sera aussi retenu pour l'étude d'une implémentation du modèle centré individus en mode distribué (voir 3.1.4). Comme dans le cas du système copépode-phytoplancton, on fait collaborer deux modèles avec leur propre point de vue et leur propre formalisme (centré individus et équations différentielles). Nous reprenons ici la même idée. Le modèle agrégé se réduit à l'equation d'évolution de la concentration des particules dans l'espace. Le modèle centré individus représente l'ensemble des particules et leurs mouvements aléatoires.

L'étude du phénomène de diffusion par la théorie cinétique a établi l'équation de Fick (équation 3.12, modèle agrégé de la diffusion brownienne [FIC55]. On s'interesse ici uniquement à la composante le long de l'axe des abscisses (x) :

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} \tag{3.12}$$

où C 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 3.12 analytiquement, la solution

de cette intégration est :

$$C(x,t) = \frac{C_0 e^{-x^2/4Dt}}{2\sqrt{\pi Dt}} \tag{3.13}$$

où : C_0 est la concentration initiale, c la concentration au temps t 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 temp t donné (voir la courbe en pointillés de la figure 3.15). 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 coefficients variables et à pas constant [**?**], en discrétisant l'espace et le temps. Nous disposons donc maintenant de deux modèles agrégés du phénomène de diffussion : un modèle analytique et un modèle numérique. Construisons à présent le modèle centré individus correspondant.

Le modèle centré individus simulant le phénomène de diffusion est particulierement 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 aléatoirement distribuées sur le plan défini par $x = 0$, séparant le cube en deux moitiés de même volume. La simulation consiste à donner une direction aléatoire 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, on retire 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 événement 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 considéré. 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). Une simple remarque sur ce système : il ressemble énormément au modèle du type copépode lorsque le nombre de phytoplanctons est nul. Le copépode adopte une dynamique de déplacement stochastique. De plus, comme nous l'avons signalé, une complexification du modèle de comportement est toujours possible. Dans ce cas, rien nous interdit d'étendre ce modèle vers celui du copépode. Nous explorons cette idée dans son implémentation en mode distribué.

Le modèle centré individus peut ainsi nous donner une estimation du paramètre de diffusion D du modèle agrégé. Pour cela, nous considérons le modèle centré individus 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, c'est-à-dire la racine carrée de la somme des carrés des distances entre les particules et l'origine de l'axe des x divisée par le nombre de particules. On peut en déduire le coefficient de diffusion D de maniere expérimentale, à l'aide de la relation (équation 3.14) :

$$\overline{\varrho} = \sqrt{2Dt} \quad \Rightarrow \quad D = \frac{1}{2t}\overline{\varrho}^2 \tag{3.14}$$

où $\overline{\varrho}$ est l'écart quadratique moyen, D le coefficient de diffusion et t le temps.

La figure 3.14 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.

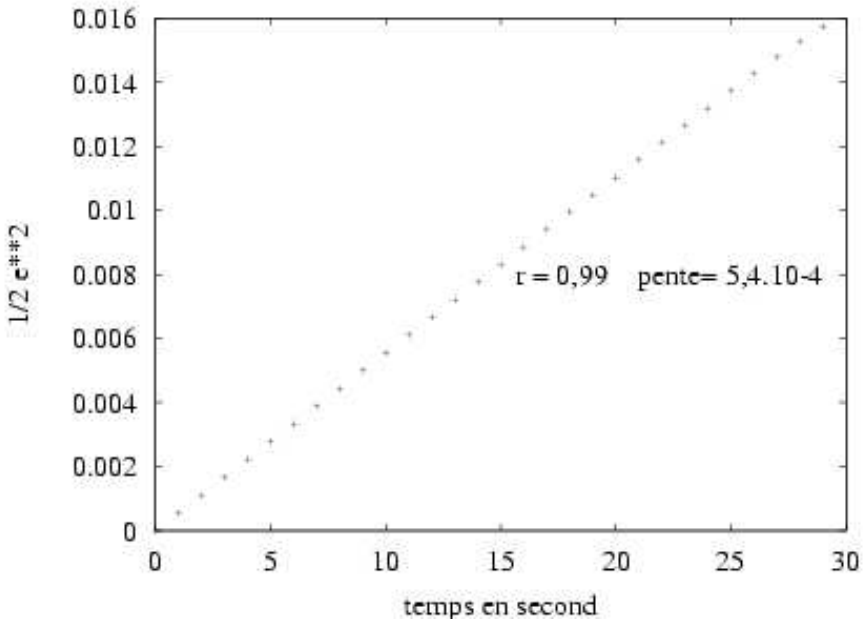


FIG. 3.14 – Écart quadratique moyen $\frac{1}{2}\overline{\varrho}^2$ en fonction du temps. La pente de cette droite correspond au coefficient de diffusion D.

On peut ensuite vérifier que le modèle microscopique des particules a le même comportement que l'équation de Fick (modèle macroscopique). La figure 3.15 permet de comparer qualitativement les deux modèles. La mesure de correlation entre les deux modèles montre que les deux modèles sont significativement proches.

Cette première utilisation de la multi-modélisation permet de déterminer des valeurs particulières de paramètres, ou de fonctions de ces paramètres. Le modèle centré individus est alors consideré 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 3.13) pour une résolution analytique. On parle ici de paramétrage ou de couplage faible. La relation de couplage est unidirectionnelle. Néanmmoins, cette approche apporte l'avantage de pouvoir intégrer des éléments inaccessibles à l'équation différentielle tel que des comportements complexes de particules ou l'interaction entre les particules (répulsion ou rapprochement). Au passage, on démontre l'efficacité des modèles centrés individus en terme d'expressivité. Le revers de la médaille est le temps de simulation qui n'est pas comparable à la résolution de l'équation différentielle. Dans l'approche analytique, une résolution suffit !

Nous allons montrer maintenant qu'il est également possible de paramétrer un schéma numérique en cours de résolution tel que nous avons déjà montré dans le cas du copépode. Dans la

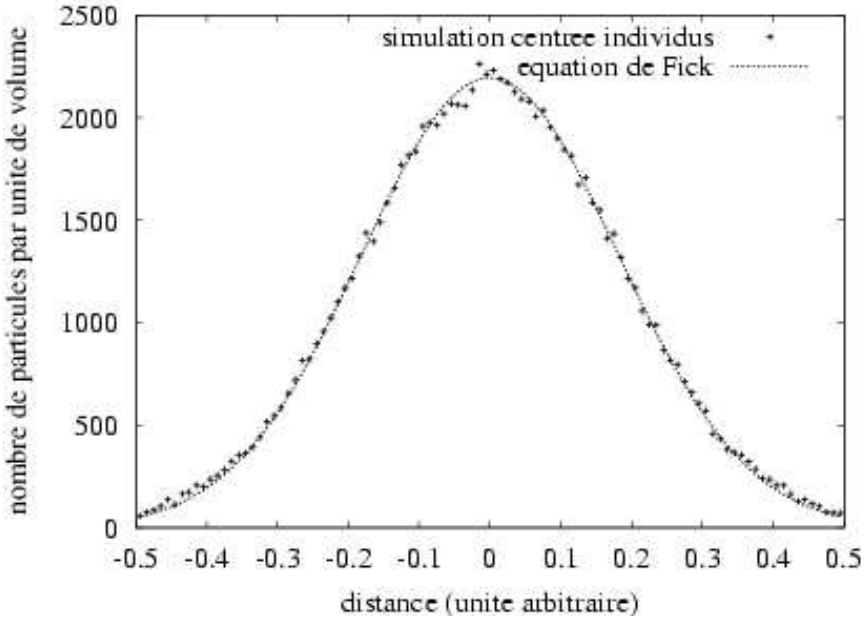


FIG. 3.15 – Simulation centrée individus du phénomène de diffusion pour 10^5 particules (croix) et équation de Fick (ligne pointillée) à $t = 30s$. La simulation et la courbe correspondent presque parfaitement.

plupart des cas, nous ne disposons pas d'une expression de D en fonction de v_{max} qui pourrait être directement utilisée dans l'équation 3.13. Nous simulons donc le phènomene à l'aide du modèle centré individus pour déterminer à chaque itération et pour chaque tranche du cube la valeur des D_i que nous réintroduisons 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. Pour identifier chaque D_i, nous utilisons la méthode présentée au paragraphe précédent, en utilisant un modèle centré individus sur 31 individus seulement pour limiter le temps de calcul. Nous avons verifié que les résultats sont équivalents à ceux obtenus en simulant 10^5 individus.

On peut se demander quel est l'intérêt d'un tel couplage. En effet, nous disposons d'un modèle numérique et d'un modèle centré individus qui simule exactement la même chose, aux mêmes 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 centré individus semble plus expressif que le modèle agrégé.

3.1.3 La formalisation DEVS du modéle Copépode

L'outil privilégié des modélisateurs en biologie ou en physique reste les Mathématiques. La critique principale est le manque de formalisation des modèles centrés individus. Nous nous proposons donc ici de montrer que l'on peut parfaitement formaliser le modèle centrée individus du copépode et on espère que cette approche pourra convaincre les plus réfractaires aux modélisations centrées individus. Nous pensons avoir déjà partiellement montré l'intérêt du couplage et

de l'utilisation de telles modélisations.

Comme nous l'avons montré dans la partie 1.3.3, il est possible de formaliser en DEVS un modèle centré individus ce que l'on a fait pour le modèle qui nous intéresse ici, le modèle Copépode. Le comportement du copépode peut être divisé en quatre modèles DEVS atomiques, qui une fois couplés, formalisent le comportement global du copépode. Le premier modèle spécifie le cycle mouvement-prédation-capture-sédimentation du copépode. Le second traite des rebonds lorsque le copépode atteint les limites de l'espace. Le troisième formalise l'évolution de l'énergie interne et le quatrième la perception des cellules de phytoplancton. Nous allons présenté le modèle global puis développerons seulement le modèle de comportement et le modèle de perception à titre d'illustration.

Le modèle agent du copépode est un modèle DEVS couplé avec la structure suivante :

$$CopepodAgent =< X, Y, D, EIC, EOC, IC >$$

Le modèle DEVS couplé peut être défini formellement par la structure AgentCopepod[10] ou représenté graphiquement (voir figure 3.16). Cette représentation met en évidence le rôle central joué par le modèle de comportement.

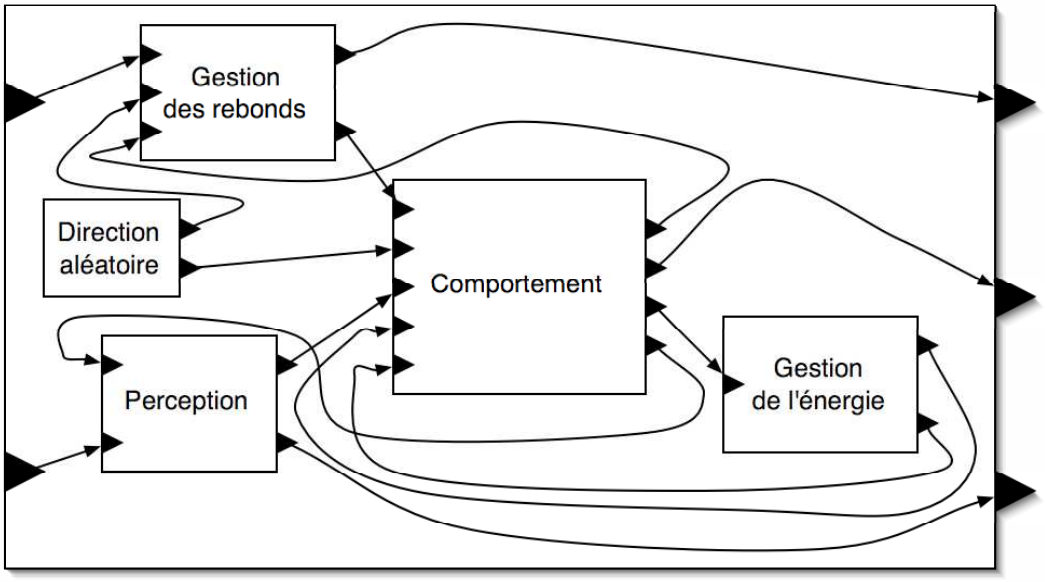


FIG. 3.16 – Représentation de la structure des connexions internes du modèle *AgentCopepod*. On voit le rôle central joué par le modèle de comportement.

Nous devons à présent formaliser les modèles DEVS atomiques en interaction dans le modèle de l'agent. Comme signalé dans l'introduction, nous allons nous intéresser seulement au modèle de comportement et de perception. Le modèle de comportement est un modèle DEVS atomique qui à la structure suivante :

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

[10]Le détail de la structure est présenté dans la thèse de Raphaël Duboz et dans [DUB02b][DUB03d]

où :
- $X = \{ (inact_1, view(A, phase)), (inact_2, bounce(A, \vec{D})), \quad (inact_3, dead), (inact_4, satiety(s)), (ina$
- $Y = \{ (outact_1, view?(P, \vec{D}, phase)), (outcat_2, trajectory(P, \vec{D}, v)), \quad (outcat_3, energy(q)), (outca$
- $ta(S)$ est l'avancement du temps

Les différentes fonctions attachées aux ports d'entrée et de sortie sont décrites plus bas. L'ensemble des états du modèle de comportement est représenté par le n-uplet suivant :

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

Le premier attribut *phase* représente l'état d'activité du copépode. Il peut prendre les valeurs suivantes :

- init : le copépode est initialisé.
- perceive ? : un état inactif transitoire. Le copépode est en attente de la réponse de l'environnement concernant la liste des cellules de phytoplancton qui perçoit ou qu'il va percevoir dans le futur.
- search : le copépode recherche sa nourriture.
- nothing : le copépode perçoit aucune proie.
- hunting : le copépode perçoit une cellule de phytoplancton et se dirige vers elle.
- capture : la nourriture est suffisamment proche, le copépode la capture.
- handle : le copépode manipule et ingère la proie et sédimente (il tombe).
- fall : le copépode sédimente.
- random walk : le copépode n'a pas faim, il se déplace aléatoirement.
- bounce ? : un état inactif transitoire. Le copépode attend sa nouvelle direction dictée par le modèle de rebonds.
- bounce : le copépode effectue un rebond.
- dead : plus aucune activité du copépode, il est mort.

La plupart des phases sont liées au déplacement du copépode. Ceux sont elles qui conditionnent principalement son comportement, c'est-à-dire les séquences d'activation des fonctions de transitions. Les phases "perceive ?" et "bounce ?" représentent des états passifs du copépode. Elles permettent de modéliser des interactions de type question-réponse, comme nous l'avons vu pour le modèle de l'environnement (voir 1.3.3.
L'attribut σ a pour valeur la date d'entrée dans l'état courant. Cet attribut est utilisé lors du calcul de la nouvelle position du copépode sur une transition interne ou externe. Ainsi, l'attribut P a pour valeur les coordonnées $\{x, y, z\}$ du copépode qui sont égales à $P + (\vec{D} \times v \times (t - \sigma))$, v étant la vitesse du copépode et t la date d'occurence du changement d'états. Nous avons donc $e = t - \sigma$ avec e le temps passé dans un état (voir paragraphe 1.3.1).

En fin de phases "init","search","hunting" ou "fall2", c'est-à-dire quand le copépode termine un déplacement, le modèle de comportement génère un événement de sortie adressé au modèle de perception afin de connaître la position de la prochaine proie. C'est le rôle des fonctions de sortie λ suivantes[11] :

- $\lambda(init, 0, P, \vec{D}) = view?(P, \vec{D}, init)$
- $\lambda(search, \sigma, P, \vec{D}) = view?(P, \vec{D}, search)$

[11]Les fonctions de sortie $\lambda : S \rightarrow Y$ sont définies par $\lambda(S) = f(x_1, ..., x_n)$ avec n le nombre de paramètres de la fonction.

- $\lambda(hunting, \sigma, P, \vec{D}) = view?(P, \vec{D}, hunting)$
- $\lambda(fall, \sigma, P, \vec{D}) = view?(P, \vec{D}, fall)$

Après avoir généré la fonction de sortie, le modèle de comportement passe dans un état transitoire *perceive*? en attente de récupérer la réponse venue du modèle de perception. Les fonctions de transitions internes suivantes formalisent ce passage[12].

- $\delta_{int}((init, \sigma, P, \vec{D})) = (perceive?, \sigma + ta(S), P, \vec{D})$
- $\delta_{int}((search, \sigma, P, \vec{D})) = (perceive?, \sigma + ta(S), P' = P + v(t - \sigma)\vec{D}, \vec{D})$
- $\delta_{int}((hunting, \sigma, P, \vec{D})) = (perceive?, \sigma + ta(S), P' = P + v(t - \sigma)\vec{D}, \vec{D})$
- $\delta_{int}((fall, \sigma, P, \vec{D})) = (perceive?, \sigma + ta(S), P' = P + v(t - \sigma)\vec{D}, \vec{D})$

perceive? est une phase bloquante pour le modèle, d'où : $ta(perceive?, \sigma, P, \vec{D}) = \infty$.
Á la réception de la réponse, le modèle de comportement transite vers l'état déterminé par le modèle de perception. Les fonctions de transitions externes suivantes formalisent ces changements d'états[13] :

- $\delta_{ext}((perceive?, \sigma, P, \vec{D}), (view(A, search))) = (search, \sigma, P, \vec{D})$
- $\delta_{ext}((perceive?, \sigma, P, \vec{D}), (view(A, hunting))) = (hunting, t, P, \vec{D}' = \overrightarrow{PA})$ où $\vec{D}'$ est la nouvelle direction du copépode et t la nouvelle date d'entrée dans l'état.
- $\delta_{ext}((perceive?, \sigma, P, \vec{D}), (view(A, capture))) = (capture, t, P, \vec{D})$
- $\delta_{ext}((perceive?, \sigma, P, \vec{D}), (view(nil, nothing))) = (nothing, \sigma, P, \vec{D})$

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

- $ta(search, \sigma, P, \vec{D}) = \|\overrightarrow{PA}\|/v$ avec v la vitesse du copépode.
- $ta(hunting, \sigma, P, \vec{D}) = \|\overrightarrow{PA}\|/v$
- $ta(capture, \sigma, P, \vec{D}) = 0$
- $ta(nothing, \sigma, P, \vec{D}) = 0$

Les deux premières fonctions d'avancement du temps ci-dessus prennent des valeurs qui dépendent du modèle de perception, c'est lui qui joue le rôle d'interface avec l'environnement. C'est donc ce dernier, avec "une touche" d'aléatoire (voir plus bas), qui contraint la dynamique du modèle de comportement du copépode.

Si le copépode perçoit aucune proie. Il passe dans la phase *nothing* de durée nulle, ce qui lui permet, par transition instantanée vers un état d'attente, d'interroger le modèle de gestion des rebonds qui lui communiquera ses nouvelles coordonnées et sa nouvelle direction aux limites de l'espace. Si la phase est égale à *nothing* alors nous avons la fonction de sortie suivante :

- $\lambda(nothing, \sigma, P, \vec{D}) = trajectory(P, \vec{D}, v)$ avec v la vitesse du copépode.
- $ta(nothing, \sigma, P, \vec{D}) = 0$

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

- $\delta_{int}(nothing, \sigma, P, \vec{D}) = (bounce?, \sigma, P, \vec{D})$

[12]Les fonctions de transitions internes $\delta_{int} : S \to S$ sont définies par $\delta_{int}(S) = S$.
[13]Les fonctions de transitions externes $\delta_{ext} : S \times X \to S$ sont définies par $\delta_{ext}(x_1, ..., x_n) = (S)$ avec n le nombre de paramètres de la fonction.

- $ta(bounce?, \sigma, P, \vec{D}) = \infty$
- $\delta_{ext}((bounce?, \sigma, P, \vec{D}), (bounce(A, \vec{D}'))) = (bounce, t, A, \vec{D}')$
- $ta(bounce, \sigma, P, \vec{D}) = 0$
- $\lambda(bounce, \sigma, P, \vec{D}) = view?(P, \vec{D})$
- $\delta_{int}(bounce, \sigma, P, \vec{D}) = (perceive?, \sigma, P, \vec{D})$

Dans sa phase de recherche de proie ou d'atteinte d'une limite de l'espace, le copépode ne perçoit pas la cellule. Il peut donc décider de changer sa direction avant de d'atteindre sa cible. C'est le rôle de l'évènement d'entrée *rand*(). Dans la phase de chasse, il perçoit sa proie et donc 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 *search* et *bounce*? que cela se produit sous l'influence d'un évènement externe comme suit :

- $\delta_{ext}((search, \sigma, P, \vec{D}), rand()) = (perceive?, \sigma, P, \vec{D}')$ avec D' généré aléatoirement
- $\delta_{ext}((bounce?, \sigma, P, \vec{D}), rand()) = (perceive?, \sigma, P, \vec{D}')$ avec D' généré aléatoirement
- $\delta_{ext}((S - \{search, bounce\}), rand()) = (S - \{search, bounce\})$

Le modèle de comportement a deux fonctions de sortie spécifiques pour la capture, elles sont adressées au modèle de gestion de l'énergie et s'écrivent comme suit :

- $\lambda(capture, \sigma, P, \vec{D}) = eat(p)$ indique à l'environnement quelle proie est consommée.
- $\delta_{int}(capture, \sigma, P, \vec{D}) = (handle, t, P, \vec{D}')$ avec t la date d'occurence de l'évènement et $\vec{D}'$ orienté vers le bas.
- $ta(handle, \sigma, P, \vec{D}) = h$, le temps de manipulation d'une proie.
- $\lambda(handle, \sigma, P, \vec{D}) = energy(q)$ avec q la quantité d'énergie absorbée.
- $\delta_{int}(handle, \sigma, P, \vec{D}) = (fall, \sigma + h, P, \vec{D})$
- $ta(fall, \sigma, P, \vec{D}) = t'$ avec t', le temps de chute.

À tout moment, le copépode peut mourir si son énergie passe en dessous d'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.

- $\delta_{ext}(S, dead) = (dead, t, P, \vec{D}),\ \forall S$
- $ta(dead, t, P, \vec{D}) = \infty$

La phase *dead* conduit à l'arrêt total du modèle de comportement.

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 au modèle de comportement un évènement de satiété. Alors le copépode entre dans une phase *random walk* où ses mouvements sont générés de façon aléatoire. La séquence suivante formalise ce passage :

- $\delta_{ext}((fall, \sigma, P, \vec{D}), satiety(true)) = (random\ walk, t, P, \vec{D}')$ où $\vec{D}'$ est généré aléatoirement.
- $ta(random\ walk, \sigma, P, \vec{D}) = rand()$
- $\delta_{int}(random\ walk, \sigma, P, \vec{D}) = (random\ walk, t, P, \vec{D}')$ où $\vec{D}'$ est généré aléatoirement.
- $\delta_{ext}((random\ walk, \sigma, P, \vec{D}), satiety(false)) = (view?, \sigma, P, \vec{D})$

Cette phase *random walk* permet d'éviter l'interrogation du modèle de perception en cas de satiété. Alors, tant que le modèle d'activité n'est pas soumis à une transition externe de type

$satiety(false)$, le déplacement du copépode est aléatoire.

Comme nous avons pu le voir, le modèle de perception joue le rôle d'interface entre l'agent et l'environnement. Il génère les évènements de rencontre des copépodes et des cellules de phytoplancton. À chaque instant, le copépode est localisé dans l'espace et se déplace selon une certaine direction. Le modèle de comportement demande au modèle de perception s'il perçoit ou percevra une cellule sur sa trajectoire courante.

Le modèle de perception est un modèle DEVS atomique avec la structure suivante :

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

où :
- $X = \{\ ((inper_1, view?(P, \vec{D}, \Phi)), (inper_2, list(L)))\ \}$
- $Y = \{\ ((outper_1, list?(P, \vec{D})), (outper_2, view(A, \Phi)))\ \}$
- $S = \{\ \phi, p, \vec{d}\ \}$ où ϕ est le n-uplet $\{idle, question, request, response\}$

Comme nous l'avons vu précédemment, le modèle de comportement interroge le modèle de perception via une transition externe, le modèle de perception passe alors dans un état transitoire qui lui permet de d'interroger le modèle de l'environnement qui lui communique la liste des cellules de phytoplancton susceptible d'être perçue par le copépode. Le modèle de perception peut alors déterminer la prochaine cellule cible du copépode. Cette activité se formalise comme suit :

- $ta(Idle, p, \vec{d}) = \infty$ Le modèle attend une intérogation.
- $\delta_{ext}((Idle, p, \vec{d}), view?(P, \vec{D}, \Phi)) = (question, P, \vec{D})$
- $ta(question, P, \vec{D}) = 0$
- $\lambda(question, P, \vec{D}) = (liste?(P, \vec{D}))$
- $\delta_{int}(question, P, \vec{D}) = (request, P, \vec{D})$
- $ta(request, P, \vec{D}) = \infty$
- $\delta_{ext}((request, P, \vec{D}), list(L)) = (response, P, \vec{D})$
- $ta(response, P, \vec{D}) = 0$
- $\delta_{int}(response, P, \vec{D}) = (Idle)$

Avant la dernière transition interne, la fonction de sortie est retournée en réponse au modèle de comportement. Cette réponse prend deux paramètres. Le premier donne les coordonnées de la cellule cible, le second donne la nouvelle phase Φ' du modèle de comportement. La réponse est totalement dépendante de l'attribut Φ donné par la fonction de transition externe et la géométrie du volume de perception du copépode. Nous exprimons ici le calcul des attributs de la fonction de sortie :

- $\lambda(response) = view(A', hunting)$ si $\exists\ A\ /A \in L,\ \overrightarrow{pA}.\vec{d} \geq 0\ ,\ \|\vec{pA}\| < r$
- où A est la position de la proie, p la position du copépode, r la distance de perception et $A' = p + \vec{d}.(\|\overrightarrow{pA}\| - r')$ et r' la distance de capture.

- $\lambda(response) = view(A', capture)$ si $\exists\ A\ /A \in L,\ \vec{pA}.\vec{d} \geq 0\ ,\ \|\overrightarrow{pA}\| < r'$ avec $A' = P$

- $\lambda(response) = view(A', search)$ si $\exists\ A\ /A \in L\ \overrightarrow{PA}.\vec{d} \geq 0$, c'est-à-dire si la cellule est devant lui. si $l \leq r\ /\ l = |((\overrightarrow{PA} \wedge \vec{d}) \wedge \vec{d}).\overrightarrow{PA}|$, 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 :

– $\lambda(response) = view(nil, nothing)$ Le modèle de perception informe le modèle de comportement qu'il perçoit rien.

Les conditions énoncées ci-dessus sont liées à notre représentation de l'espace continu et à la géométrie du volume de perception. Un quelconque changement de ces conditions ne change pas formellement le comportement du modèle.

Pour terminer la spécification du modèle *CopepodAgent*, il faut dit un mot sur les trois autres modèles atomiques (gestion des rebonds, gestion de l'énergie et direction aléatoire).
Le modèle "Gestion des rebonds" formalise le comportement aux limites du copépode. Ce dernier fait le lien entre la représentation de l'espace disponible dans le modèle de l'environnement et le comportement aux limites du copépode. Le fait d'avoir isolé ce comportement illustre la notion d'encapsulation. Il est facile de changer la politique aux limites sans pour autant modifier le comportement décrit dans le modèle de comportement.
Le modèle atomique "Gestion de l'énergie" représente quant à lui l'ensemble du cycle de l'énergie dans le copépode. Initialement, la proie, la cellule de phytoplancton, est capturée et est stockée dans l'estomac du copépode. Lors de la capture d'une cellule, le modèle de comportement fournit un événement d'entrée dont le paramètre q représente la quantité d'azote contenue dans la proie capturée. Cette quantité peut être spécifique à chaque proie. Dans [CAP96], le cycle de l'énergie est définie à l'aide d'un système d'équations différentielles (voir figure 3.3). C'est ce même système qui a été intégré au modèle 2D. Après l'étude du système, il s'avère que ce système possède une solution analytique. Il est alors simple de construire un modèle de type DESS. Les démonstrations sont disponibles dans [DUB02c] et dans la thèse de Raphaël Duboz.
Le modèle "Direction aléatoire" est sans entrée. Il est un générateur d'événements aléatoires qui vient perturber le copépode si la durée entre deux captures s'avère trop longue. On formalise de cette manière les changements aléatoires de direction observées *in vivo*. De nouveau, la distribution de probabilité intégrée dans le modèle est modifiable à volonté.

Nous avons parcouru une partie du modèle *CopepodAgent*. Ce modèle n'est qu'une brique du modèle global. Il faudrait ajouter le modèle de l'environnement et le modèle global du système. Le modèle de l'environnement est une extension du modèle proposé en 1.3.3. Pour sa part, le modèle global est un modèle couplé composé d'un modèle de l'environnement et d'un ensemble de modèles *CopepodAgent*. Dans le cas présent, la spécification est simple. En revanche, nous avons développé la spécification du couplage entre le modèle centré individus et un modèle DEVS du système d'équations différentielles représentant la dynamique de populations. La spécification est beaucoup plus complexe tout en étant accessible. Nous ne développerons pas ici cette spécification. Néanmoins, que pouvons nous en dire ? La partie "système d'équations différentielles" est modélisable à l'aide de plusieurs extensions de DEVS[14] (DEV&DESS, G-DEVS, ..., ces formalismes ont été évoqué dans le premier chapitre). Le couplage est beaucoup plus sujet à discussion selon le point de vue que l'on a. Nous avons opté un point de vue simple : nous avons deux modèles à coupler au sens DEVS et leurs interactions peuvent se modéliser à l'aide d'évenements. Mais quels sont ces événements ? Si on considère les événements de sortie du modèle de dynamique de populations, ces derniers ont pour objectif de paramètrer le mo-

[14]Ici, nous n'avons pas la chance que le système possède une solution analytique.

dèle centré individus or la nature même du paramètrage pose problème. En effet, il remet en cause la structure du modèle : le nombre de copépodes peut avoir été modifié par le modèle de dynamique de populations ce qui signifie que le nombre de modèles couplés représentant les copépodes est modifié. Le problème serait le même si l'on intégrait un comportement de reproduction dans notre modèle. Il est donc nécessaire de faire appel au formalisme DS-DEVS (voir section 1.3.3.3). Grâce à DS-DEVS, les connexions entre le modèle de l'environnement et les modèles de type *CopepodAgent* peuvent changer dynamiquement en fonction de la contrainte définie par le modèle de dynamique de populations. Pour plus de détails, il suffit de consulter une fois encore la thèse de Raphaël Duboz.

En conclusion, nous pensons que nous avons fait la démonstration de la puissance de DEVS et de ses extensions. La formalisation DEVS d'un système tel que le nôtre est une avancée dans notre recherche de disparition de l'ambiguïté des modèles à base d'agents. Le revers de la médaille est que le travail à fournir reste lourd mais on peut faire l'analogie avec la spécification formelle en génie logiciel. La maintenabilité d'une application passe inévitablement par une phase de spécification et de documentation. Il faut que le monde de la modélisation à base d'agents tende vers cet idéal ... DEVS est peut être une solution.

3.1.4 Vers un modèle distribué

Les perspectives du couplage de modèles sont nombreuses mais nous contraint à réflèchir aux problèmes liés au temps de simulation et à la quantité de mémoire nécessaire. Si on prend l'exemple du couplage de notre modèle centré individus et un modèle spatialisé de dynamique de populations, on se rend compte de deux choses : le nombre d'entités à simuler dans le modèle centré individus pour une cellule spatiale du modèle de dynamique de populations devient prohibitif et le nombre de modèles centrés individus devient lui aussi très important. Il faut noter tout de même que notre approche permet de réduire considérable l'ordre de grandeur des simulateurs[15]. Un travail de DEA, réalisé par G. Quesnel, a donc été initié pour montrer la faisabilité du passage d'un modèle centré individus avec un espace continu et à événements discrets implémenté en mono-processeur à un modèle distribué [QUE03a][QUE03b]. Le modèle distribué doit garantir toutes les bonnes propriétés que l'on verra par la suite et surtout garantir la description de l'espace continu et la formalisation du modèle en événements discrets ou autrement dit il faut respecter la spécification des simulateurs abstraits proposés en DEVS.
Les objectifs à atteindre, en terme de nombre d'entités à simuler, ont été fixé à environ 10^6 phytoplanctons et 10^3 copépodes. Ces chiffres sont à comparer aux simulations mono-processeurs actuelles : environ 10^5 phytoplanctons et 100 copépodes. Ces quantités ont été fixé par expérience et reflète les quantités qui sont échangées entre les deux modèles couplés.
Afin de réduire la complexité du problème, nous avons éliminé les interactions entre phytoplanctons et copépodes, ce qui réduit le système à un ensemble de particules effectuant une marche aléatoire simulant ainsi un mouvement brownien. Nous nous sommes focalisés sur deux éléments : la gestion du déplacement des copépodes (que l'on nommera par la suite particules) et la synchronisation des événements liées à ces déplacements. Cette simplification n'est pas génante à condition de garantir la synchronisation globale du modèle. Ce dernier point est fondamental. En effet, dans le système que l'on cherche à développer, les particules intéragissent et ces interactions nécessitent obligatoirement une synchronisation temporelle. A $t = 0$, chaque particule

[15]Une simulation purement centrée individus pour une zone géographique tel d'un océan est totalement inenvisageable.

est positionnée dans l'espace. Elles se déplacent et se situent donc à chaque instant à un endroit précis dans cet espace. Deux particules peuvent alors intéragir si elle se situe au même instant et dans la même zone (un copépode capture une cellule de phytoplancton situe proche de lui). Cette contrainte n'a pas besoin d'être vérifié dans la majorité des modèles qui sont développées dans la littérature.
En revanche, en simplifiant le modèle, nous avons fait disparaître les calculs complexes qui sont réalisés au sein des particules (calculs liés au comportement et à la dynamique interne des copépodes). Nous avons pallié à ce biais en ajoutant fictivement une charge de calcul au sein des particules. Cette charge de calcul fera l'objet d'une étude afin de déterminer son impact sur les stratégies adoptées.
Le système simplifié possède un précieux avantage : on connaît parfaitement son comportement (voir 3.1.2.2) ce qui nous permettra de valider nos algorithmes. En effet, le système se résume en un ensemble de particules situées dans un espace à trois dimensions et pourvu d'un mouvement brownien. Afin de donner une idée précise, nous avons modélisé ce mouvement de la manière suivante :

- à un instant t, la particule se situe à une position représentée par un triplet (x, y, z) dans un espace à trois dimensions.
- la particule possède une direction $(\overrightarrow{x}, \overrightarrow{y}, \overrightarrow{z})$ dont chacune des composantes est initialisée aléatoirement entre -1 et $+1$.
- la particule possède une vitesse de déplacement générée aléatoirement selon une distribution gaussienne bornée de moyenne 1 d'écart-type 0.2 et de limites 0 et 2. Ces valeurs sont inspirées de la vitesse observée du copépode.
- la particule se déplace pendant un temps Δt généré aléatoirement selon la même distribution que la vitesse.
- à l'issue de Δt, les particules possèdent une nouvelle position et les composantes du mouvement sont regénérées aléatoirement.

Ce modèle nous permet d'identifier les événements du système qui sont tout simplement les instants de changement de direction. Cette approche s'inscrit dans le cadre de la simulation à événements discrets. Entre deux changements, la particule se déplace sur une droite. Si on replace le discours dans le cadre de la simulation à événements discrets, l'avancement du temps est dirigée par les événements. On garantit alors l'approche DEVS.

Il faut maintenant se poser la question : que doit-on distribuer ? L'objectif est d'utiliser plusieurs processeurs interconnectés avec un réseau rapide[16] et chaque processeur doit prendre en charge une partie de la simulation. Plusieurs pistes sont possibles :

- distribuer les entités spatiales et garder une description centralisée de l'espace,
- découper l'espace et garder une description centralisée des entités,
- découper l'espace et distribuer les entités spatiales.

La troisième piste est naturellement la bonne dans notre cas. Il semble beaucoup plus efficace de localiser sur un même processeur les entités qui seront amenées à interagir. Nous effectuons donc un découpage de l'esapce et chaque processeur prend en charge la gestion d'un sous-espace. Si les entités n'ont pas à interagir la première piste est suffisante à condition que les entités connaissent leurs localisations. A partir de ce choix de distribution, nous avons étudié trois techniques de distribution de calcul et leur efficacité.

[16]Dans notre cas, nous avons validé nos approches grâce à un cluster de PC bi-processeurs sous Linux.

La stratégie de distribution adoptée concerne tout d'abord le découpage de l'espace en sous-espaces en y associant les particules appartenant à ces sous-espaces. Cette stratégie a tendance à répartir à la fois la charge de calcul et le besoin en terme de stockage d'information. La question qui se pose alors est la suivante : quel est l'algorithme de simulation distribuée optimal dans notre cas ? Nous rappelons que l'aspect synchronisation globale des évènements, connu sous l'appellation de causalité que nous développerons dans le cadre général, est vital dans l'hypothèse où les particules doivent interagir. En effet, les systèmes physiques obéissent toujours au *principe de la causalité*. Il peut être énoncé de la façon suivante : le futur n'influence jamais le passé. La causalité impose donc un ordre partiel des transitions d'états dans les systèmes physiques. Lamport [LAM78] a défini une méthode basée sur l'utilisation d'estampilles pour permettre aux processus de respecter la causalité. Cet ordre partiel imposé par les systèmes physiques induit un ordre partiel entre les événements du modèle de simulation. En particulier, l'ordre dans lequel le simulateur produit les événements doit être cohérent avec cet ordre partiel, pour que le simulateur soit un juste reflet du système physique simulé. Par exemple, si la transition A du système physique intervient à la date 3 et a une influence sur la transition B intervenant à la date 5, le simulateur doit générer l'événement modélisant la transition A avant celui modélisant la transition B.

La distribution de la simulation sur un système distribué pose alors un problème pour le maintien de ce principe de causalité. Les problèmes sont en plus accentués lorsque les événements sont distribués sur des processus indépendants. En effet, l'échéancier des événements est distribué et chacun des processus évalue localement chaque événement. Il faut alors utiliser différentes approches pour assurer, qu'à tout instant, le principe de la causalité ne sera pas mis en défaut. Trois techniques ont été étudié. Nous les présenterons ici que dans leurs grandes lignes.

La première approche, la plus simple, repose sur le principe suivant : chaque simulateur prend la décision de faire avancer la simulation de son sous–espace uniquement quand il est sûr de pouvoir le faire. Ce principe est connu soous l'appelation : approche synchrone. Dans le cas de l'approche synchrone, un simulateur peut traiter tout événement qui n'a pas de conséquence sur les simulateurs jusqu'à une certaine date appelée barrière de synchronisation. Dans notre cas, nous sommes obligés de considérer que tout événement peut avoir une conséquence d'où l'algorithme adopté. Un processus central gère une horloge globale. Ce processus est appelé coordinateur et donne la main au simulateur qui possède l'événement dont la date d'occurrence est la plus petite globalement. Cette technique s'inspire des travaux de Zeigler et des simulateurs abstraits DEVS[ZEI99]. Dans cette approche, il n'existe aucun parallélisme mais permet une distribution des données. Néanmoins, elle va nous permettre de quantifier le coût engendré par les communications entre les simulateurs puisque globalement le simulateur synchrone possède un comportement identique au simulateur mono–processeur.
La deuxième technique est décrite comme asynchrone pessimiste ou asynchrone faible car elle a pour principe, de faire avancer chaque simulateur d'une manière asynchrone tant qu'aucune erreur ne peut apparaître. Le principe est le suivant : chaque processus possède un échéancier qui fournit la date minimale (date jusqu'à laquelle un simulateur peut traiter les événements sans conséquence sur ces voisins). Celle-ci est envoyée à tous les simulateurs. Si toutes les dates reçues sont supérieures à sa date minimale, le simulateur traite son prochain événement et envoie à tous ses voisins sa date minimale. Le simulateur ne pourra recommencer à dépiler son échéancier uniquement lorsqu'il aura reçu des dates supérieures à son temps courant des autres processus.

La dernière approche est dite optimiste. Chaque processus décide de faire avancer la simulation, sans se préoccuper de l'état des autres. Dans ce cas, le principe de la causalité peut être violé. Pour rétablir ce principe, Jefferson [JEF85] propose une solution : le *Time Warp*. Lorsqu'un processus s'aperçoit que le principe de la causalité n'est plus vérifié, c'est à dire que la date courante de la simulation est postérieure à la date du dernier événement reçu, il annule tous les messages déjà expédiés. Pour cela, il faut gérer un historique des états du système, mais aussi le contenu de tous les messages reçus et expédiés afin de pouvoir avertir les autres processus de faire des annulations. Pour cela, nous utilisons des *anti-messages*, qui possèdent exactement les mêmes caractéristiques que les messages d'origine à l'exception d'un flag.

Les différentes approches ont fait l'objet d'une implémentation dans une architecture matérielle et logicielle offrant les caractéristiques de base d'un système informatique distribué : processeurs interconnectés sans mémoire partagée et bibliothèque de programmation réseau[17]. De plus, nous avons porté une attention toute particulière sur les générateurs aléatoires. En effet, certaines précautions sont à prendre lorsqu'on se situe dans un environnement distribué [ECU98][ECU02]. La première étape fut de valider ces différentes approches dans un contexte simple : nous avons placé toutes les particules au centre de l'espace, l'espace a été découpé suivant l'axe des abscisses x et nous avons mesuré le coefficient de diffusion (voir 3.1.2.2) ainsi que la localisation des particules au cours du temps. Le résultat se présente sous la forme d'une gaussienne (figure 3.17).

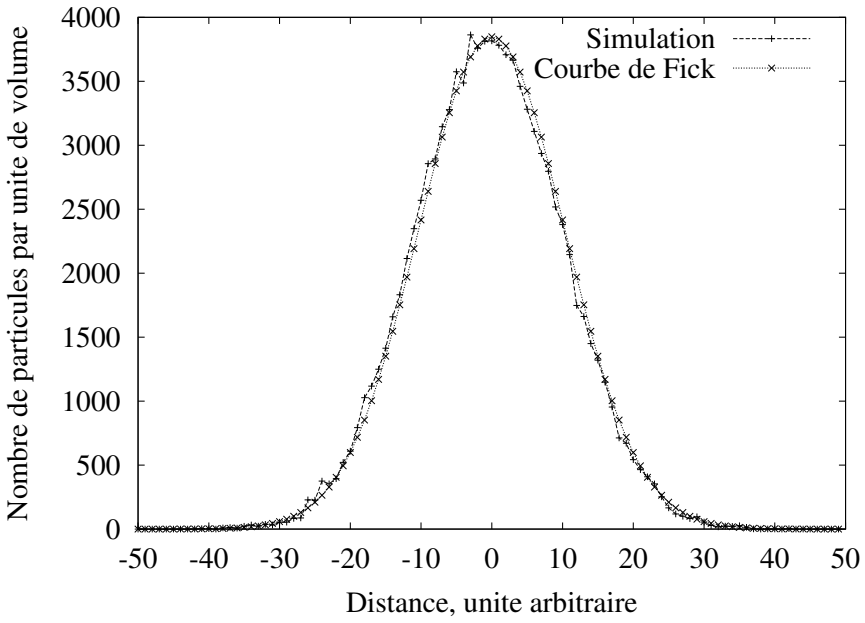


FIG. 3.17 – Courbes de Fick simulé par l'approche synchrone forte et calculé

Le résultat que nous obtenons montre effectivement que notre modèle distribué, dans les cas synchrone fort et faible, ajuste bien la courbe de Fick avec un degré de confiance élévé, validant ainsi notre modèle distribué par rapport au modèle mono-processeur.

[17]Nous avons utilisé MPI - *Message Passing Interface*

La question à laquelle on doit répondre maintenant est : quelle est la meilleure stratégie et sous quelles conditions ? Pour y répondre, la comparaison des performances est réalisée entre la version mono-processeur, qui sert de référence, la stratégie synchrone forte et la stratégie synchrone faible. La stratégie asynchrone ne fait pas partie de ce comparatif. En effet, la charge due au backtracking (i.e. au traitement des anti–messages) s'avère trop coûteux. Ce coût prohibitif s'explique facilement. En effet, notre modèle est fortement communicant. On a montré que la fréquence des synchronisations est très élevée et est proportionnelle au nombre de particules. Cette propriété fait rarement partie des modèles faisant l'objet d'une mise en œuvre distribuée. La figure 3.18 nous montre deux choses. La stratégie synchrone faible est plus coûteuse en temps total de simulation que la version mono-processeur ce qui était prévisible. Le surcoût observé est tout simplement l'œuvre des communications entre le coordinateur et les simulateurs. Il faut noter tout de même que ce coût est relativement faible ce qui nous autorise à dire que cette stratégie est intéressante dans le cas où les données attachées aux particules et aux sous–espaces manipulés par le simulateur ont une taille importante. De plus, le nombre de messages échangés entre le coordinateur et les simulateurs est très élévé en comparaison à d'autres modèles. A chaque changement de direction d'une particule, le simulateur ordonnance un nouvel événement ce qui se traduit par deux messages. On s'aperçoit rapidement que le nombre de messages peut devenir très élevé si le nombre de particules est lui-même élevé. Cette remarque restera vraie quelle que soit l'approche.

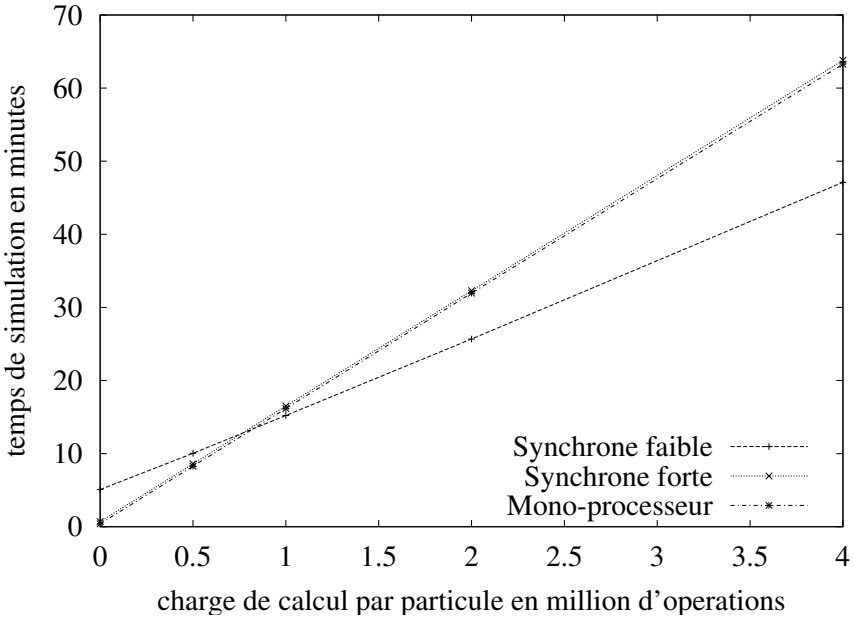


FIG. 3.18 – Simulation de trente mille particules, en faisant varier la charge sur les particules

Le deuxième constat concerne l'évolution du temps de simulation dans le cas synchrone faible. Si la charge de calcul est négligeable voire nulle, il existe néanmoins un coût lié à la communication et à la synchronisation qui est incompressible. Ce qui implique que cette stratégie est moins bonne que notre stratégie de référence (en mono–processeur) pour des charges de calculs inférieurs à 0.7 million d'opérations. Au delà de cette valeur seuil, la stratégie synchrone

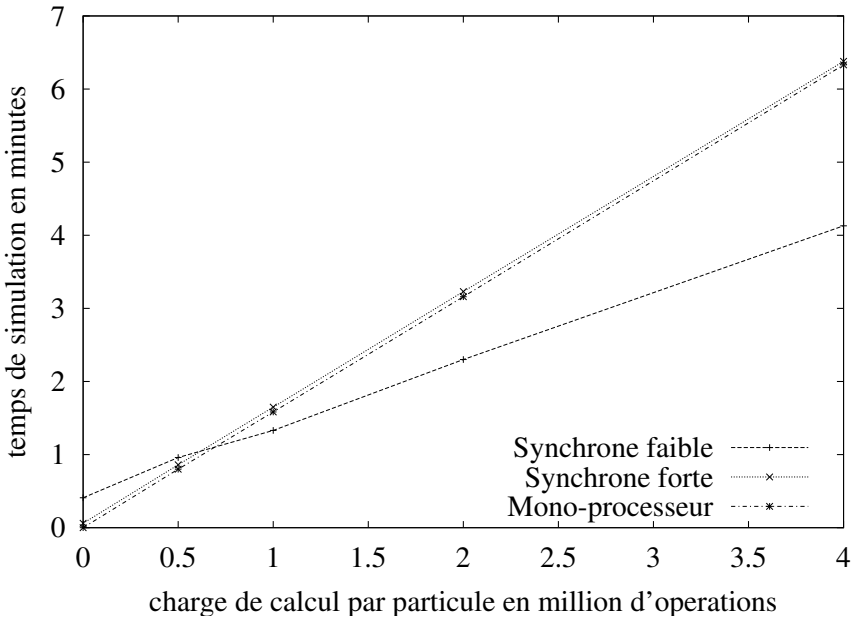


FIG. 3.19 – Simulation de trois mille particules, en faisant varier la charge sur les particules

faible est meilleure. Le constat reste vrai quel que soit le nombre de particules (voir figure 3.19). On observe cependant une variation de ce seuil : plus le nombre de particules est élevé, plus la valeur de ce seuil est élevé.
La figure 3.20 représente la variation du temps total de simulation en fonction de la charge de calcul par particule dans le cas synchrone faible en faisant varier le nombre de simulateurs. Faire varier le nombre de simulateurs consiste tout simplement à diviser l'espace en un nombre plus élevé de sous–espaces. Le passage de 3 à 6 diminue sensiblement le temps de simulation. Mais on observe cette tendance qu'à partir d'un certain seuil de charge. Si on augmente le nombre de simulateurs (dans notre exemple à 9), on s'aperçoit que ce seuil s'éloigne rapidement. L'explication est simple. Si on mesure le nombre total de messages échangés entre les simulateurs et entre le coordinateur et les simulateurs, on observe une augmentation linéaire. En effet, plus le nombre de sous-espaces est élevé plus le nombre de messages est élevé.

En conclusion, les résultats que nous avons obtenus nous permettent de penser que l'utilisation de la technique synchrone faible semble être la meilleure stratégie. En effet, cette stratégie est la seule de nos deux approches permettant de gagner du temps par rapport à la technique mono–processeur. Cependant, il faut prendre en compte, que la technique synchrone faible ne devient intéressante qu'à partir d'une certaine charge de calcul par particule. L'approche synchrone forte n'a comme principal intérêt que l'augmentation du nombre de particules possibles sans gagner de temps avec la technique mono–processeur et la simplicité de son algorithme.
Dans notre système proie–prédateur (copépodes–phytoplanctons), les copépodes réalisent un grand nombre de calculs en interne pour leurs comportements (digestion, perception, ...), mais aussi pour calculer dans l'espace 3D les prochains événements sur lesquels ils pourraient se heurter (calculs de trajectoires, intersections ...). Ces coûts nous font penser que la technique pessimiste est la meilleure pour notre modèle proie–prédateur.
Dans l'exemple du mouvement brownien, nous n'avons pas été confronté à un problème d'équi-

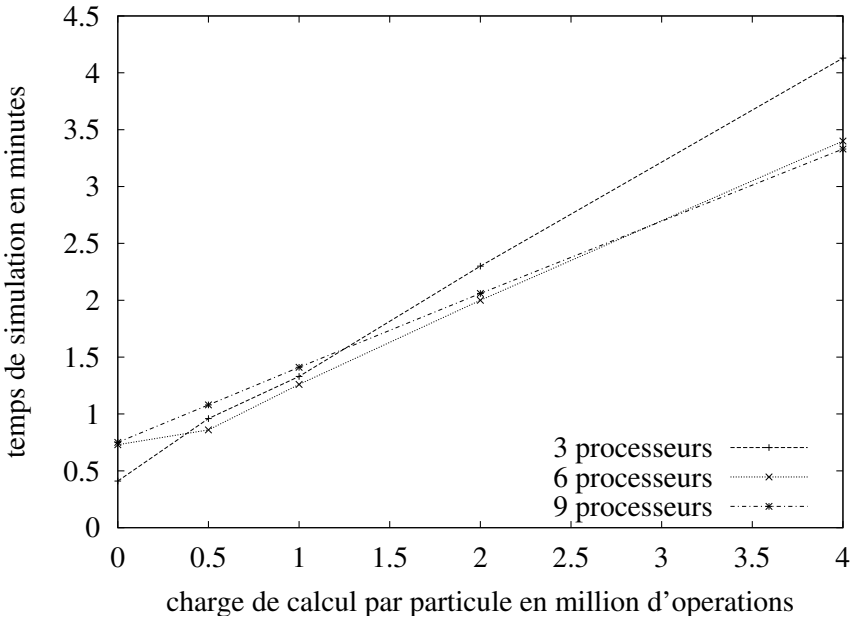


FIG. 3.20 – Approche pessimiste, variation du nombre de processeurs utilisés

libre de charge, car la distribution des particules a tendance à rester uniforme et on sait que la charge est dû au nombre de particules par simulateur. En revanche, dans le cas de la relation du copépode avec les cellules de phytoplancton, il est nécessaire de se poser la question. En effet, les cellules de phytoplancton ont tendance à former des patchs et les copépodes sont alors localisés principalement dans ces patchs. Il existe de nombreux travaux traitant de ce problème [MIG95][FON98]. Il reste néanmois à l'adapter à notre système. On peut tout de même penser qu'une stratégie de découpage dynamique de l'espace en fonction de la localisation des patchs de cellules de phytoplancton soit intéressante à étudier.
Le dernier point à développer concerne l'intégration opérationnelle de l'architecture logicielle et matérielle dans le *framework* proposé dans le chapitre précédent.

3.2 Les autres modèles

Afin d'illustrer et de valider à la plate-forme VLE, nous nous sommes aussi intéréssé à un domaine relativement éloigné des systèmes naturels : les systèmes de production et en particulier les systèmes flexibles de production (SFP). Ce travail est le fruit d'une collaboration bien naturelle avec le Laboratoire d'Informatique de Tours, mon ancien laboratoire. D'un certain point de vue, une chaîne de production est un ensemble d'éléments actifs (robots, tapis roulants, convoyeur, centre d'usinage, fraiseuse, tour ...) et d'éléments passifs (zone de stockage, par exemple). Tout élément actif peut être vu comme une entité qui prend en entrée un ou plusieurs objets, effectue des opérations dessus et le donne à une autre entité. Prenons l'exemple d'un robot. Le robot saisit un objet dans une zone de stockage, le transporte jusqu'à une machine et le dépose sur la machine.

Afin de construire une modélisation d'un tel système, nous avons adopté une approche élabo-

rée dans le cadre d'une notation générique de systèmes flexibles de production (SFP) développée au Laboratoire d'Informatique de Tours [MAR97]. Cette notation permet de décrire un système flexible de production à l'aide d'un graphe dont les sommets sont les ressources de traitement, au sens large, et les arcs les circuits sur lesquels les moyens de transport circulent.

Toute ressource (de traitement ou de transport) du SFP est modélisée par un moyen de stockage composé de trois stocks :

- un stock amont recueillant les entités en entrée avant d'être effectivement traitées ou transportées,
- un stock interne représentant le nombre d'entités traitées en parallèle ou transportée par la ressource,
- un stock aval regroupant les entités après traitement ou transport.

Citons quelques exemples. Une fraiseuse est vue comme un moyen de stockage de capacité 1. En effet, une fraiseuse traite une pièce à la fois. On peut étendre cet exemple en y ajoutant un stock amont et un stock aval si des pièces peuvent être stockées à proximité de la machine. Un chariot est un autre exemple et se représente par un moyen de stockage de capacité égale au nombre de pièces que le chariot peut transporter. En revanche, il n'existe pas de stock amont et de stock aval.

Le modèle développé par le LI va plus loin et définit pour les ressources de traitement trois catégories :

- les ressources de traitement réelles : les entités prises en charge par ces ressources subissent un traitement (usinage, assemblage, contrôle ...)
- les ressources de traitement virtuel : le seul traitement effectué est une action décisionnelle (échange d'entités entre plusieurs manipulateurs, aiguillage ...)
- les méta-ressources : la ressource est un ensemble de ressources décrites comme un SFP (méta-modélisation).

Etant donnée cette notation des SFP, nous allons maintenant montrer comment VLE représente cette notation et ce que VLE peut apporter de plus. Dans un premier temps, un modèle objet de la notation est à construire afin de faire correspondre les éléments de la notation à des classes d'agents spécifiées dans la partie formalisme de VLE. Deux possibilités nous sont offertes : on utilise soit le formalisme agent combiné avec les langages de description de la dynamique (langage algorithmique ou réseaux de Petri interprétés) soit on utilise un autre formalisme disponible dans VLE (par exemple, les réseaux de Petri) et le modèle est alors un couplage de modèles atomiques ou couplés. Nous avons optés pour la première solution.

En VLE, les ressources de transport sont considérées comme les éléments dynamiques du système et héritent donc de la classe *C2DDynamicElementaryPhysicalAgent*. En effet, ils ont pour fonction la prise en charge des produits et leur transport d'une ressource de traitement à une autre. Vis à vis de la structure spatiale, les ressources de transport seront localisées dans un espace 2D continu ou discret et se déplaceront d'un point de l'espace à un autre. Les chemins suivis, modélisés par des arcs entre ressources de traitement (voir Figure 3.21), feront l'objet de données stockées dans les ressources de transport.

Les ressources de traitement (tapis, robot, machine ...) sont modélisées par des éléments dynamiques mais privés de la capacité de se déplacer dans l'espace. Ils héritent donc de la classe *C2DStaticElementaryPhysicalAgent* ou *C2DstaticComplexPhysicalAgent* selon leur taille vis à

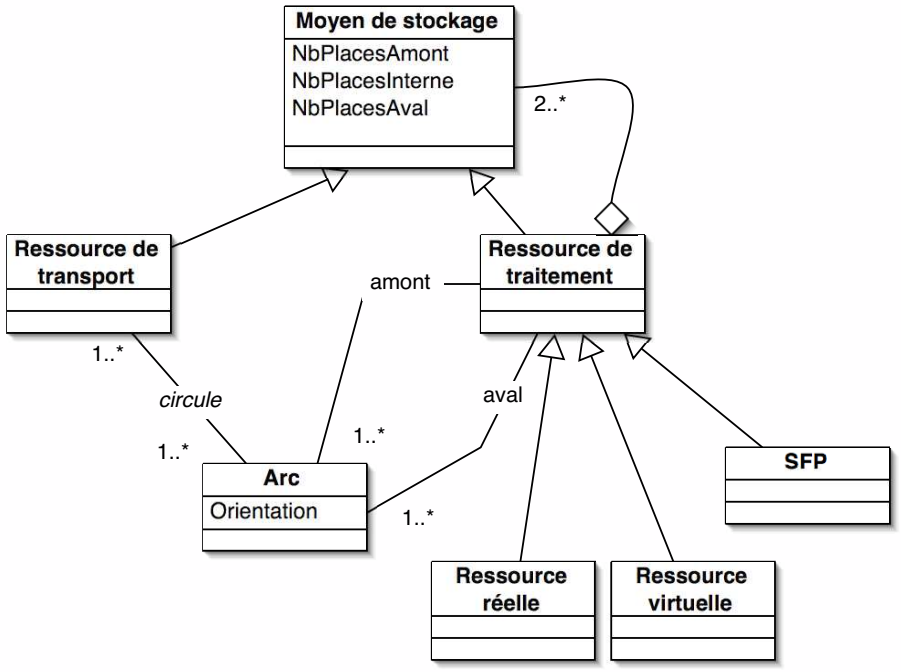


FIG. 3.21 – Diagramme de classes de la notation générique des SFP

vis de la représentation spatiale choisie (voir 2.2.1.5).

Il reste les produits circulant dans le système. Ce sont des éléments passifs qui sont transportés par les ressources de transport et transformés par les ressources de traitement. Ils comportent principalement des informations et n'exercent aucune modification sur l'environnement. Ils seront donc représentés par des instances d'une classe héritant de la classe de base *C2DStaticElementaryPhysicalAgent*.

Les différentes ressources (de traitement et de transport) possèdent une dynamique commune induite par leur structure (stock amont / stock interne / stock aval). Cette dynamique est représentée par des réseaux de Petri interprétés[18] qui assurent les fonctions de transfert d'un stock à un autre. Si on s'intéresse maintenant aux fonctions plus spécifiques des différentes ressources, les ressources de transport ont un réseau de Petri modélisant les déplacements d'un point à un autre en fonction des chemins définis. Il faut noter que ce réseau de Petri peut intégrer des politiques d'évitement de véhicules, de choix entre plusieurs chemins possibles ... autrement dit, la souplesse des réseaux de Petri interprétés nous autorisent à ajouter les "algorithmes" que l'on veut. Quant aux ressources de traitement, leur dynamique exprime les actions de transformation des produits. Ces transformations sont essentiellement des modifications au niveau des attributs des agents représentant les produits.

Afin de fixer les idées, nous allons nous intéresser à un exemple de systèmes flexibles de production qui, pour la clarté de l'exposé, restera simple. Un objet circule dans le système et passe de l'état de produit brut à l'état de produit fini.

[18]Nous avons fait ce choix seulement à titre illustratif.

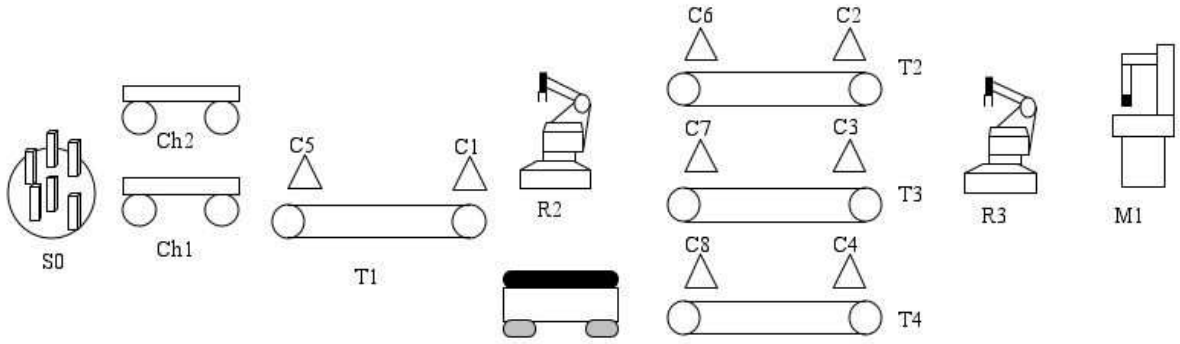


FIG. 3.22 – Représentation schématique d'une chaîne de production

Les produits bruts sont dans un stock initial S_0. Tous les produits bruts ne sont pas identiques. Deux chariots Ch_1 et Ch_2 prennent en charge les produits, les transportent du stock à un tapis roulant et les produits sont déposés sur le tapis roulant T_1. Les temps de transport sont fonction des produits chargés. Le tapis avance de 10 centimètres à chaque fois qu'un produit est posé par le robot en début de tapis. Lorsqu'un produit est arrivé à la fin du tapis un capteur C_1 le détecte et un deuxième robot R_2 vient chercher le produit qu'il dépose sur une balance B_1. Le tapis T_1 ne peut avancer que s'il n'y a pas de produit en fin de tapis soit juste après une prise par le robot R_2. Le capteur C_1 indique s'il y a un produit en fin de tapis T_1. En fonction du poids du produit, R_2 reprend le produit et le dépose sur un tapis. Il y a trois tapis T_3, T_4 et T_5 ; chacun étant réservé pour un poids. A l'extrémité des trois tapis, des capteurs C_2, C_3 et C_4 détectent la présence de produits. Comme pour le tapis T_1, les tapis avancent de 10 centimètres à chaque fois qu'un produit est déposé. Il avance si aucun produit se trouve en fin de tapis. Lorsqu'un produit arrive à l'extrémité d'un tapis, un dernier robot R_3 prend le produit si la machine M_1 est libre et le pose sur M_1. Un opérateur humain retire les produits de la machine lorsque l'opération sur le produit est finie.

Nous venons de décrire le fonctionnement normal de la chaîne de production. Voici quelques compléments :

- si un tapis est plein c'est-à-dire s'il y a des produits au début du tapis et à la fin du tapis, les robots ne peuvent pas déposer de produits. Les tapis acceptent 10 produits maximum.
- un robot ne pourra donc pas déposer sur un tapis si la première place est occupée ; le tapis ne pourra pas avancer s'il y a un produit en fin de tapis.
- si un produit est déposé sur un tapis et que le tapis ne peut pas avancer alors le tapis avancera dès que le produit en fin de tapis sera pris par un robot.
- lorsqu'il se présente simultanément des produits en fin de deux ou trois tapis, le robot R_3 vide en priorité le tapis T_2 puis le tapis T_3 et finalement le tapis T_4. En revanche, lorsqu'un tapis est plein, il devient le plus prioritaire.

Si on s'intéresse à la modélisation complète du tapis roulant T_1, on peut dire que T_1 est une instance de la classe *CTapisRoulant*. *CTapisRoulant* est une sous-classe de la classe de base *C2DStaticComplexPhysicalAgent* (voir Figure 3.23). En effet, un tapis roulant est un élément localisé dans l'espace. Deux options s'offrent à nous : soit on considère chaque emplacement du tapis comme un élément spatial et ils doivent alors être individuellement localisés dans l'espace soit l'espace est décrit à l'aide de lieux sans référentiel spatial. Dans ce dernier cas, le tapis est

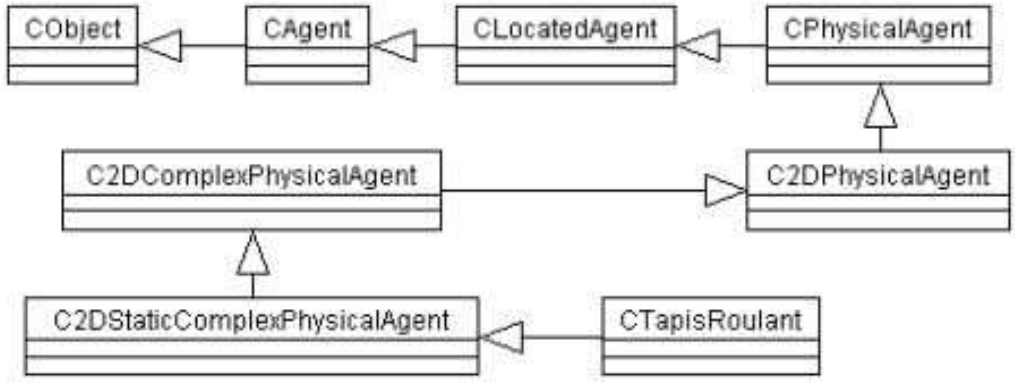


FIG. 3.23 – Graphe d'héritage de la classe CTapisRoulant

localisé à l'aide d'un lieu au sens virtuel (voir 2.2.1.5). Sa seule fonction "dynamique" est de faire avancer les produits au fur et à mesure qu'une ressource de transport désire déposer un produit en début de tapis.
Un tapis roulant est donc une suite d'emplacements pouvant contenir des produits. Pour chacun de ces emplacements, un produit peut y être localisé. La manipulation des produits, ou plus généralement d'objets attachés à un objet de type *C2DStaticComplexPhysicalAgent*, est rendue possible grâce à des fonctions définies dans la classe *C2DStaticComplexPhysicalAgent*. Par exemple, l'évocation de la fonction *move* provoque le déplacement d'un objet dans l'espace. Cette évocation est possible car les instances de la classe *C2DStaticComplexPhysicalAgent* possèdent des références sur les objets (ici des produits) qui leur sont rattachés.
Quelques attributs peuvent être introduits dans la définition à la classe *CTapisRoulant* afin d'observer l'évolution de certaines caractéristiques du système : le nombre de produits présents sur le tapis, un indicateur de saturation du tapis ("le tapis est plein"), par exemple.

A ce stade, il ne nous reste plus que deux éléments à définir : le traitement des messages et la dynamique propre du tapis. En ce qui concerne le premier point, le tapis roulant doit traiter un message. Lorsqu'un produit arrive en début de tapis et qu'il désire occuper le premier emplacement, un message est envoyé au tapis par le chariot par exemple. Ce message est traité si la place d'entrée (*entry*) est libre (voir figure 3.24 et figure 3.25).

VLE autorise l'exécution parallèle de plusieurs réseaux de Petri pour un même agent dans le formalisme agent. Dans notre cas, un réseau est associé à l'avancement du tapis si aucun produit n'est présent en fin de tapis et n autres réseaux à la gestion de la présence d'un produit sur un emplacement (voir figure 3.25). On associe à chaque emplacement du tapis deux places : l'une modélisant l'absence de produit (PL_i), l'autre la présence d'un produit (PO_i). Lorsqu'un produit est déposé en début de tapis (traitement du message *deposeProduit*), un jeton est ajouté à la place *Entry* (instruction *addToken*).

Lorsqu'un jeton est placé dans la place *Entry*, deux cas de figure sont à envisager : soit un produit se situe en fin de tapis ou non. Si aucun produit occupe la fin du tapis (T_2 est validée) alors le jeton est transmis à la place P. Deux actions sont associées à la place P. La première émet un événement qui va déclencher l'avancement du tapis. La deuxième action ajoute un jeton dans la place E_1 pour modéliser le fait que le produit change de place. Une action est exécutée à

```
message deposeProduit(o : CObject)

message CTapisRoulant::deposeProduit(o)
{
  <take,one(o,Produit)>
  <sendMessage,toObject(o,objetPris())>
  <addToken(main,Entry,1)>
}
```

FIG. 3.24 – Exemple de message traité par *CTapisRoulant*

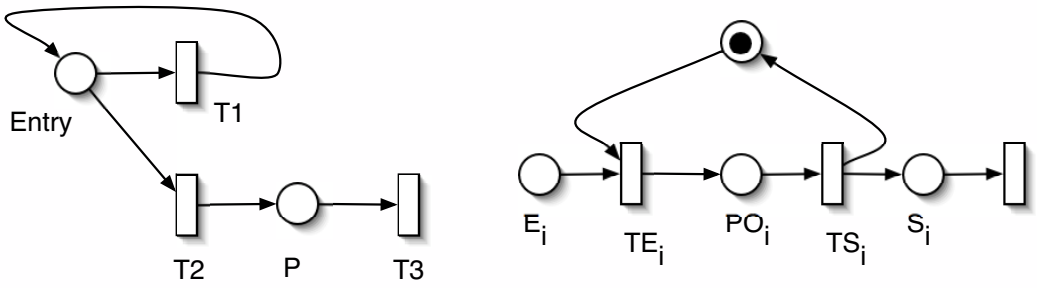


FIG. 3.25 – Réseau de Petri de la classe *CTapisRoulant*

l'arrivée d'un jeton dans une place E_i qui conduit au déplacement du produit sur le tapis roulant. L'événement émis par la place P rend valide les transitions TE_i et TS_i des réseaux de Petri des emplacements. Si les places E_i et PL_i (l'emplacement est libre) possèdent chacun un jeton alors la place PO_i reçoit un jeton (l'emplacement devient occupé). Si la place PO_i possède un jeton (au lieu de PL_i) alors étant donnée que la transition TS_i est franchissable, un jeton est ajouté à la place PL_i (l'emplacement redevient libre) et un jeton est ajouté à la place S_i. L'arrivée d'un jeton en S_i déclenche une action d'ajouter d'un jeton dans la place E_{i+1}. La transition finale n'est pas conditionnée donc le jeton disparaît. Afin de synchroniser le tout, les transitions TE_i sont temporisées d'une durée égale au temps nécessaire au tapis d'avancer d'un pas.

En résumé, le tapis roulant est modélisé par un ensemble de RdP exécutés en parallèle. La génération d'événements et le transfert de jeton d'un RdP à un autre garantissent la synchronisation des déplacements des produits sur le tapis. Les attributs disponibles dans l'objet *CTapisRoulant* permettent un suivi de l'évolution de ce dernier dans le temps.

En conclusion, l'approche orientée objet et agent pour les systèmes de production n'est pas nouvelle [BER99] mais a l'avantage d'être plus naturelle. De plus, les réseaux de Petri interprétés sont utilisés comme principale notation de la dynamique, ce qui permet aux modélisateurs d'utiliser des outils qu'ils ont l'habitude d'employer. Le deuxième avantage de VLE pour les systèmes de production est de proposer de manière native le couplage de modèles. Il est donc possible comme dans le cas des systèmes naturels de coupler un modèle d'atelier avec un modèle de conduite, par exemple, ou avec un modèle d'optimisation. Ce travail en cours est l'un des

thèmes traités dans le groupe de travail GTIMMS[19] dont on fait parti depuis l'origine. Ce groupe rassemble principalement des informaticiens intéressés par la modélisation et la simulation de systèmes complexes.

3.3 Conclusion et perspectives

L'activité de modélisation est à la base de nos activités de recherche. Dès l'origine, nous avons intégré nos réflexions fondamentales dans le cadre de nos actions de modélisation de systèmes réels. Nous sommes persuadés que seuls les allers-retours entre modélisation sur le terrain et réflexions fondamentales peuvent faire avancer les méthodologies pour la modélisation et la simulation de systèmes. Notre système favori a été et reste le système "zooplancton - phytoplancton". Ce système est à la fois complexe au sens du nombre et de la hiérarchisation des processus et complexe au sens des modélisations existantes. Cette complexité permet de dégager les grandes classes de problèmes auxquels on tente d'apporter une réponse : le couplage dynamique de modèles, l'interaction entre des modèles hétérogènes et le transfert d'échelles en sont des exemples.

Nos réflexions nous ont permis de confronter nos approches et nos modèles aux modèles existants et d'entrevoir des réponses aux questions de fond. Il reste encore un long chemin à parcourir et nos travaux actuels nous laissent penser que les pistes qui se sont ouvertes vont nous permettre d'aller plus loin dans les méthodologies de modélisation des systèmes naturels. De plus, certaines actions du PNEC s'ouvrent de plus en plus à notre discours. L'appui de certains mathématiciens et notre effort de formalisation n'y sont pas étrangers.

Nous tenons à souligner en conclusion que les travaux présentés dans ce chapitre sont le résultat d'une interaction entre des informaticiens, des biologistes, des mathématiciens et des physiciens. Et encore une fois, l'emergence de nouvelles idées, de nouvelles méthodes ou de nouveaux outils n'est possible que grâce à cette interaction. En effet, toutes les réunions de travail étaient (et nous l'espérons resteront) le lieu de débats et de confrontations des modèles et des outils de chacuns. Cette interaction pour être fructueuse doit être basée sur la recherche des apports de chacun dans l'évolution des idées. En d'autres termes, il ne faut pas chercher à démontrer que notre outil est le meilleur mais comprendre comment l'interaction et le couplage des différents outils peuvent être bénéfique.

[19] http ://mad3.univ-bpclermont.fr

Chapitre 4

Conclusions et perspectives

Résumé

Ce chapitre fait le bilan de l'action menée jusqu'à présent et trace les perspectives à moyen terme de l'équipe travaillant sur la modélisation et la simulation des systèmes. Les pistes qui vont être suivi sont multiples : la construction du langage de spécification formelle des modèles et de leur couplage, le développement d'une plate-forme informatique disponible sur le Web pour la modélisation et la simulation et l'étude du système "Phytoplancton-Zooplancton". Ces trois pistes sont déjà en cours d'investigation, nous ne faisons que poursuivre nos travaux.

Le travail de recherche mené durant ces dernières cinq années a eu comme fil rouge la modélisation et la simulation des systèmes complexes. Notre activité s'est divisée en cinq volets :

- la construction d'un *framework* de modélisation et de simulation et la définition d'un langage XML de spécification formelle des modèles dans un cadre de multi-modélisation
- le développement d'une plate-forme informatique mettant en oeuvre, dans un premier temps, un environnement de modélisation et de simulation multi-agents pour les systèmes spatio-temporels et dans un deuxième temps, le *framework*
- une formalisation des modèles à base d'agents réactifs
- une étude du système "sels nutritifs – phytoplancton – zooplancton"
- une réflexion sur des problèmes techniques et conceptuelles du couplage de modèles et du transfert d'échelles

Les questions soulevées par ces cinq volets sont nombreuses et restent ouvertes. Notre projet de recherche s'inscrit donc dans la perspective de répondre à certaines de ces questions tout en s'appuyant sur les projets actuels et les collaborations existantes dans les domaines de l'informatique, de l'écologie, des mathématiques et de la physique. Le principal apport attendu à moyen terme concerne le développement d'outils fondamentaux et opérationnels en informatique pour la modélisation et la simulation des systèmes complexes et en particulier en écologie marine. En effet, le projet mené doit permettre de poursuivre les travaux en biologie théorique et d'acquérir une meilleure connaissance de l'écosystème marin prenant en considération la variabilité individuelle et l'hétérogénéité à différentes échelles des processus impliqués. L'écologie marine possède un double avantage : un cadre d'expérimentation des outils et théories dévelop-

pées en informatique et une source de questionnement. L'arrivée d'un nouveau chercheur dans l'équipe, Jean-Christophe Soulié, doit aussi permettre de développer la thématique Modélisation et Simulation des systèmes naturels. Son arrivée nous permet de consolider nos compétences en multi-agents et en modélisation de systèmes complexes[1].

Le projet de recherche se divise en cinq volets interdépendants :
- la modélisation centrée individus et la spécification formelle de systèmes spatio-temporels (en particuliers en écologie marine),
- les outils formels pour le couplage de modèles,
- la simulation distribuée à événements discrets et la spécification DEVS,
- le développement d'un serveur de services Web pour la modélisation et la simulation et l'intégration de XML,
- l'étude des méthodes de calcul émergent et d'agrégation de variables pour la modélisation de processus multi-échelles.

Chacun de ces volets va faire l'objet d'un développement détaillé par la suite afin de mettre en évidence les questions fondamentales qui seront abordées et traitées. Nous tenons à ajouter que l'aspect étude de l'écosystème pélagique marin sera peu développé dans le texte qui suit mais constituera notre fil rouge tout au long du projet.

4.1 La modélisation centrée individus et la spécification formelle de systèmes spatio-temporels

En informatique, les modèles centrés individus permettent de considérer les individus dans la dynamique globale des systèmes et ainsi d'intégrer plusieurs échelles de temps et d'espace [GRI99][BOU94]. De tels modèles s'attachent à décrire le comportement individuel (dans son sens le plus large) des individus et leurs interactions. Les bilans sont alors émergents et peuvent paramétrer des modèles informatiques ou mathématiques de plus haut niveau.
L'approche actuellement privilégiée est une modélisation centrée individus utilisant le formalisme DEVS [ZEI73] et le concept d'événements discrets. DEVS permet en effet d'avoir une approche formelle de la modélisation de systèmes, permet de disposer d'un langage formel pour la spécification des modèles et permet d'encapsuler des formalismes *a priori* hétérogènes (équations différentielles, automates cellulaires ...). Jusqu'à présent, on utilise une variante de DEVS que l'on a définit pour aider à la spécification de systèmes spatio-temporels où un environnement est défini ainsi que l'ensemble d'entités en interaction. D'autre part, il existe une multitude d'extensions de DEVS dont certaines (DS-DEVS, Cell-DEVS, ...) semblent prometteur dans le cadre de la modélisation centrée individus.
A partir de cet état des lieux et du point de vue théorique, on doit aborder plusieurs questions :
- quelle démarche de modélisation adopter dès lors que les processus sont complexes et qu'ils s'intègrent plusieurs échelles ?
- comment étudier l'émergence de propriétés et les intégrer dans les niveaux d'organisation supérieurs ?
- quels outils formels utiliser pour intégrer des modèles hétérogènes en terme de formalisme et de paradigme ? DEVS et ses extensions sont-ils la réponse ?

[1] Jean-Christophe Soulié a participé à un projet de modélisation et d'analyse de la relation zone de pêches et activités de pêches en utilisant des approches agents. Ces travaux de thèses sont des travaux plus fondamentaux sur les systèmes multi-agents multi-environnements [SOU01].

- quel est l'apport du paradigme centré individus en modélisation de systèmes complexes et en particulier en écologie ?
- quel langage ou formalisme utiliser pour présenter formellement et sans ambiguïté un modèle informatique en écologie ?

Les pistes que l'on s'est ouverte pour répondre à ces diverses questions sont multiples. Premièrement, nous participons de manière active à la réflexion nationale sur un *framework* de modélisation centrée individus des systèmes spatio-temporels : le projet MIMOSA associé au GdR I^3. Le groupe de travail a trois objectifs :

- proposer un vocabulaire (une ontologie) pour les concepts manipulés dans le cadre de la modélisation centrée individus des systèmes spatio-temporels,
- proposer un langage de type XML pour la spècification des modèles,
- proposer une ou plusieurs implémentations de simulateurs basés sur le langage de spécification de modèles.

Ce projet est important pour nous puisqu'il doit nous offrir une base solide pour la spécification des systèmes que nous intéressent. De plus, si l'aspect formel est au rendez-vous, il est fort probable que l'on puisse proposer une encapsulation DEVS telle que les réseaux de Petri présentés en exemple dans le chapitre 2. Nous en avons déjà proposé un aperçu avec la spécification du temps et de l'espace. En effet, la structure XML proposée est directement issue des réflexions du groupe MIMOSA. Nous avons aussi montré que la majorité des concepts manipulés dans la partie centrée individus de VLE étaient présents.

Deuxièment, toutes les questions que l'on se pose sont au coeur de l'étude de l'écosystème marin est un excellent cas d'étude et un lieu de questionnement irremplaçable. De plus, les problématiques de modélisation et de compréhension de ce système sont aussi au coeur des projets nationaux (PNEC, en particulier) et internationaux (GLOBEC, par exemple) en environnement. En effet, la compréhension du fonctionnement des écosystèmes marins, et en particulier de l'écosystème côtier, est confrontée à la grande complexité des nombreux processus impliqués à différentes échelles de temps et d'espace. En particulier, la compréhension de l'influence de la turbulence, processus multi-échelle, sur la production de matière organique en est un des principaux enjeux [BRO96][SEU96]. Le chapitre 3 a été l'occasion de faire le bilan des résultats déjà obtenus mais il reste encore une multitude de questions en suspens que l'on exposera dans la dernière partie de ce chapitre.

Troisièment, une réflexion est menée par le groupe GTIMMS (Groupe de Travail en Informatique pour la Multi-Modélisation et la Simulation) dont nous faisons parti depuis l'origine, sur les problèmes liés à la modélisation multiple et à la simulation. Contrairement au projet MIMOSA, les réflexions ne font pas d'*a priori* sur le type de systèmes étudiés. Cette approche du problème permet de confronter les idées développées par des chercheurs d'horizons divers et qui n'ont pas obligatoirement le même point de vue sur les approches de modélisation. Cette "confrontation" des idées est bénéfique et pour preuve, notre orientation vers les travaux formels de Zeigler et DEVS est née de notre rencontre avec Norbert Giambiasi lors des réunions du groupe.

4.2 Le couplage de modèles et le *framework* de modélisation et de simulation de systèmes

La modélisation et la simulation de systèmes complexes mettent en jeu une multitude d'outils, de méthodes et d'approches. Au travers des questions qui se posent, on comprend mieux la complexité de la recherche dans ce domaine et les implications pratiques sur les domaines d'applications (biologie, physique, industriel, ...). Si on essaie de structurer les problématiques, on s'aperçoit que l'on peut identifier plusieurs niveaux d'abstractions (niveau qui ont été identifiés dans le chapitre 2).

Le premier niveau est identifié comme le niveau opérationnel ou de communication qui doit répondre aux problèmes de mise en oeuvre informatique de la simulation (communication entre modèles, échange de données structurées, distribution des traitements, ...). Nous n'aborderons pas du point de vue théorique ce niveau tout simplement parce qu'il existe un éventail très large de solutions. Pour exemple, on peut citer Corba, Java/RMI, SOAP, MPI, ... pour l'aspect communication et distribution. En revanche, le développeur de simulateurs doit faire un choix parmi ces technologies et notre travail consiste à proposer la meilleure solution en fonction de l'architecture. Ce travail a fait l'objet d'une première étude mais il reste à la finaliser en proposer un ensemble d'API cohérentes et fonctionnelles. La première partie de la thèse de Gauthier Quesnel sera consacrée à ce travail.

Au dessus du niveau opérationnel apparaît le niveau simulation des modèles. De même que le niveau opérationnel, les techniques offertes sont nombreuses : simulation à temps discret ou à temps continu, simulation à événements discrets, simulation distribuée ou non, ... Les algorithmes de base de chacune de ces techniques sont connues depuis longtemps. Prenons pour exemple la simulation distribuée à événements discrets, depuis une dizaine d'années des modules et des spécifications ont été définis dans un cadre international et ont donné naissance dans un premier temps à DIS (Distributed Simulation) puis plus récemment à HLA (High Level Architecture). Ces travaux offrent des spécifications détaillées d'implémentation de simulateurs dans des environnements informatiques distribués et hétérogènes et intègrent la majorité des algorithmes connus et reconnus dans le domaine. On peut et on doit donc s'appuyer sur ces spécifications pour la couche simulation. Il faut tout de même noter que tout n'est pas encore réalisé : une implémentation opérationnelle des spécifications HLA sur un cluster avec les technologies MPI n'existe pas à notre connaissance. De plus, HLA ne répond pas à tous les problèmes du niveau simulation. On peut effectivement y trouver des réponses pour tous les aspects génériques des simulations mais la mise en oeuvre d'un simulateur en mode distribué par exemple est un travail très dépendant du modèle considéré. En conclusion, il est nécessaire de poursuivre notre bilan des technologies et de montrer dans quelle mesure ces technologies répondent ou non à nos besoins en simulation de systèmes complexes spatio-temporels.

Nous quittons maintenant les niveaux plutôt opérationnels pour nous intéresser au niveau conceptuel (ou modèle) et sémantique. Ces deux niveaux sont beaucoup plus central à notre travail. Les formalismes et paradigmes utilisés en modélisation sont tout aussi variés que les technologies d'implémentation et de déploiement. Pour notre part, nous nous sommes plutôt intéressés au paradigme agent et centré individus, au formalisme mathématique et DEVS. L'objectif de la couche conceptuelle est d'offrir un langage unifié de spécification de modèles. Comme il existe dans le domaine informatique pour la spécification objet de systèmes au travers d'UML

(*Unified Modeling Language*), il paraît vital de développer le même type de langage pour la modélisation de systèmes complexes. Pourquoi ? La première raison est très simple : pour coupler conceptuellement deux modèles, il faut soit utiliser le même formalisme et le même paradigme pour les deux modèles soit offrir un méta-formalisme encapsulant les formalismes utilisés pour les deux modèles. Ce méta-formalisme doit être générique tout en offrant un certain niveau de spécification commun à l'ensemble des formalismes encapsulés. Hormis le besoin évident de couplage de modèles, le monde de la modélisation a aussi besoin d'un langage pour communiquer. Si on développe un modèle à base d'équations différentielles, il n'y a pas de problème. Le formalisme est suffisamment précis, concis et sans ambiguïté pour être compris par autrui. En revanche, comme le cas de modèles centrés individus, les modèles sont très souvent réduit à un programme informatique. Le seul moyen pour connaître ce que contient le modèle c'est de lire le programme. Or il n'est pas raisonnable de procéder de la sorte. De plus, les résultats ne sont pas toujours reproductible parce que les auteurs du modèle ont utilisés une architecture matérielle particulière ou une bibliothèque de fonctions non communiquée ou un algorithme non spécifié de génération aléatoire, par exemple.
Nos recherches se sont orientées vers DEVS et ses extensions. En effet, DEVS est un formalisme générique de modélisation de systèmes et offre des simulateurs abstraits pour chacune de ses extensions. On peut citer :

- DESS, G-DEVS et DEV&DESS pour la spécification de systèmes à base d'équations différentielles,
- Cell-DEVS pour les automates cellulaires,
- DS DEVS pour les structures dynamiques, ...

La question à laquelle on cherche à répondre est : quelles extensions apportées à DEVS pour les systèmes spatio-temporelles et l'encapsulation du paradigme agent et individus centré ? Nous avons proposés un début de réponse mais nous sommes encore loin d'une extension propre de DEVS. Le travail engagé avec le groupe MIMOSA est probablement une piste sérieuse. D'autre part, ce travail de fond s'accompagne d'une réflexion sur le paradigme centré individus en écologie.

Nous allons aussi poursuivre notre travail sur le développement de *wrappers* pour les différents formalismes couramment utilisés (*state chart*, équations différentielles, automate à états finis, ...). Ce travail est :

- un travail conceptuel de mise en adéquation du formalisme et du paradigme à base d'événements discrets,
- un travail technique de conception de langages XML pour la spécification du formalisme,
- un travail de développement de simulateurs *compliant-DEVS* pour le formalisme.

Le développement d'outils de formalisation s'accompagne de l'utilisation du langage XML et des outils tels que les *parsers* XML, les "translateurs" (XSLT), ... Nous avons présenté dans le chapitre 2 l'état de notre réflexion à ce sujet. L'utilisation de XML offre toute la puissance d'un langage expressif et extensible. Pour preuve, de récents travaux se sont orientés vers l'utilisation de XML pour la représentation de 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 équations différentielles. P. A. Fishwisk [FIS02] introduit MXL basé comme un langage basé sur XML pour la représentation des modèles. Néanmoins, ce dernier reste pour l'instant limité et peu adapté aux besoins de couplage de modèles. Malgrès ces premiers résultats sur XML qui ne vont pas dans notre sens, nous allons poursuivre notre développement de notre langage de spécification XML car XML reste la meilleure solution pour la construction de

langages pour la modélisation et la simulation.

Le dernier niveau que l'on a qualifié de sémantique est un aspect jusqu'à maintenant non abordé dans la modélisation de systèmes complexes. L'idée est la suivante : un modèle est développé soit pour un système donné soit de manière plus générique pour une famille de systèmes. Prenons un exemple simple. Si on considère un modèle simple d'équations différentielles (non spatialisées) d'un système proie-prédateur. Les équations modélisent l'évolution des populations en fonction de leur interaction. Les prédateurs et les proies peuvent être aussi bien des loups et des agneaux ou des orques et des sardines. Ce qui signifie que ce modèle s'applique sur des populations au sens large. Seules les valeurs de paramètres de ces équations seront modifiées. La seule hypothèse à vérifier est que les deux populations interagissent et que l'une joue le rôle de prédateur et l'autre de proie. En revanche, si on considère un modèle de la capture des proies par les prédateurs, le modèle peut être défini pour un couple particulier de proie et de prédateur. Un loup "chasse" différemment qu'un orque ! La question que l'on se pose est alors : ces deux modèles sont-ils "couplables" ? Sémantiquement, la réponse est positive. En effet, un loup ou un orque sont des entités d'une population et ils jouent le rôle de prédateur (de même pour les proies). Si maintenant, le modèle proie-prédateur est plus spécialisé c'est à dire que les équations intègrent des aspects plus spécifiques aux prédateurs ou/et aux proies alors la réponse à la question de la possibilité de coupler les deux modèles est remise en cause. Le travail a engagé sur cet aspect est d'étudier l'intégration d'aspects sémantiques sur les modèles et les éléments manipulés dans les modèles. Ces aspects sémantiques peuvent être de type :

- classe d'appartenance des entités des modèles et hiérarchisation de ces classes d'appartenance,
- relation sémantique entre les entités des modèles,
- définition du référentiel temps/espace des variables du modèle...

Certaines de ces informations ont déjà été introduite dans le langage de spécification mais elles ne font pas l'objet d'une exploitation dans le contrôle sémantique du couplage de modèles. Il reste donc du travail dans cette perspective. De plus, de nombreux travaux existent dans le domaine des systèmes d'information et une étude approfondie de ces travaux devraient nous permettre de continuer à intégrer des aspects sémantiques dans les modèles.

Cette partie du projet se résume en la construction d'un *framework* de modélisation de systèmes complexes avec l'utilisation de XML. Cette construction doit être menée dans une réflexion d'intégration de couches d'abstractions, de couplage de modèles hétérogènes et d'un domaine de modélisation spécifique, les écosystèmes riches en problématiques (processus multi-échelle, transfert d'échelle, hétérogénéité des formalismes et des approches de modélisation, ...).

4.3 Simulation distribuée

Comme nous l'avons fait ressentir dans la précédente partie du projet, nous allons aussi nous intéresser à la simulation distribuée. En effet, un aspect du projet concerne la complexité des processus et la taille des populations d'individus à simuler. Cela va nous conduire à explorer la piste de la simulation distribuée à événements discrets [MIS86]. Un travail de DEA a été réalisé sur ce sujet et a exploré la piste de la simulation distribuée au sein d'un cluster pour un modèle simple d'interactions proie-prédateur. On a cherché à travers cette étude :

- à déduire des algorithmes généraux pour la simulation massive distribuée centrée individus et à événements discrets de systèmes spatio-temporels,
- à montrer l'apport de la simulation distribuée dans le cadre des systèmes spatio-temporels,
- à identifier les conditions d'utilisation (la parallélisation des simulateurs n'est pas obligatoirement la solution).

Ce travail sera donc poursuivi dans le cadre d'une thèse de doctorat qui va débuté et viendra s'intégrer dans les couches opérationnelles et simulation comme un outil complémentaire mais indispensable aux systèmes considérés.
De plus, ce travail sera une partie des activités d'une équipe projet multi-laboratoire que l'on a conçu avec le LSIS de Marseille et le LISC du Cemagref de Clermont-Ferrand. Cette équipe projet multi-laboratoire aura aussi pour objectif de compléter notre réponse concernant la spécification DEVS en étudiant l'impact de G-DEVS et d'un AgentDEVS qui reste à étoffer.

4.4 Développement d'un serveur de services Web pour la modélisation et la simulation

Le quatrième volet du projet de recherche est le volet opérationnel et concerne le développement d'une plate-forme de services Web pour la spécification de modèles hétérogènes et de leur couplage, et pour la spécification de plans d'expériences distribuées, le calibration et la validation des modèles et l'analyse statistique des sorties des modèles. Cet environnement est aussi appelé "laboratoire virtuel". Les modèles sont construits par des équipes de chercheurs issus de disciplines diverses et géographiquement distribués sur la planète. Il est donc nécessaire de prendre en compte ces aspects de travail distant et coopératif.

Actuellement, quelques briques de la plate-forme ont été développés par l'équipe (R. Duboz et moi-même) et surtout des étudiants en projet de fin d'études (de l'IUP GMI et du DESS ISIDIS[2]) intégrant les aspects XML et services pour la spécification des modèles et des expériences. Ces premières briques ont permis de valider l'architecture définie dans le chapitre 2. Mais ce n'est qu'un début. Certains aspects n'ont pas encore fait l'objet d'une mise en oeuvre complète tels que :

- la gestion des modélisateurs et de leurs modèles ;
- les outils graphiques de définition de cette partie des modèles,
- la gestion du travail coopératif[3],
- la gestion des contrôles conceptuels et sémantiques lors du couplage de modèles,

Les travaux à réaliser dans cette partie sont en grande partie des implémentations informatiques de couches de gestion. Le coeur du langage de spécification, des analyseurs, des simulateurs sont réalisés. Nous sommes arrivés dans une phase de fusion des briques développées et dans une phase de réalisation de l'interface utilisateur. Ce travail sera principalement mené par des étudiants en fin de cycle universitaire (Maîtrise et DESS).

[2]DESS Ingénerie des Systèmes Informatiques Distribués

[3]Un projet ACI sur cette question a débuté l'année dernière au sein d'une équipe du laboratoire. Des pistes seront donc sûrement à explorer dans les résultats obtenus par ce projet.

4.5 Le système NPZ : intégration d'échelles et changement d'échelles

Dans le projet de recherche que l'on a mené jusqu'à présent, nous nous proposons d'étudier via la modélisation et la simulation informatique et expérimentalement des processus du vivant se déroulant dans un milieu caractérisé par une forte hétérogénéité spatio-temporelle et des transferts d'échelles. Il s'agissait de comprendre le rôle des processus se déroulant à l'échelle des individus sur la dynamique globale du système. Les modèles ont été de deux types : informatique et mathématique. Nous les avons rendu opérationnels et les avons couplé tout en menant une réflexion plus générale sur les outils de modélisation et de simulation des systèmes complexes. Nous avons obtenus de nombreux résultats présentés dans le chapitre 3 que l'on peut résumer de la manière suivante :

- la description mécaniste des processus au niveau individuel du copépode,
- l'effet de l'hétérogéneéité de la distribution des proies sur la réponse fonctionnelle du copépode,
- l'effet de la réponse fonctionnelle intégrant les aspects individuels à petite échelle sur la dynamique de populations,
- l'émergence de fonctions mathématiques connues des traces de simulation des modèles centrés individus,
- le paramétrage de modèles mathématiques par des modèles centrés individus,
- le couplage dynamique de modèles mathématiques et de modèles centrés individus,

Les résultats se divisent en deux parties : des résultats sur la compréhension du système et des résultats sur le couplage de modèles dans un cas réel.

Il reste naturellement de nombreuses questions et l'intérêt grandissant de la communauté des biologistes pour la modélisation centrée individus et le couplage de modèles ne fait qu'accentuer cette demande.

Sur la compréhension du système, nous avons initié des travaux avec F. Carlotti, J. C. Poggiale et Y. Lagadeuc dans le cadre des actions 2003-2004 du PNEC. Ces travaux doivent nous permettre de comprendre l'impact des processus individuels et des interactions des entités à petit échelle sur la structuration en classes d'âge des populations mais aussi de valider certains modèles mathématiques intégrant des hypothèses sur la structure spatiale des proies.
Le premier point pose directement le problème du changement de stade (i.e. de classe d'âge) et de la mortalité. Ces deux points sont assez mal connu ce qui implique que ce travail devra être un aller-retour en l'expérimentation et la modélisation-simulation.
Le deuxième travail en cours est un travail plus théorique et constitue une suite aux travaux déjà initiés avec l'émergence de fonctions mathématiques et le couplage dynamique avec des modèles mathématiques. Il reste néanmoins à étudier tous les impacts de ces couplages sur la dynamique de populations.

Le dernier volet va poursuivre notre réflexion sur le problème de transfert et changement d'échelle. Deux approches ont été abordées : le calcul émergent [FOR90] jumelé aux approches centrées individus et les techniques d'agrégation de variables utilisées pour la modélisation de certains processus multi-échelles.
Le calcul émergent est définit comme l'émergence d'une cohérence globale générée par des interactions locales. Dans la plupart des cas, le calcul émergent se résume en l'identification de

fonctions impliquant des variables globales du système considéré à partir de la simulation de ce dernier modélisé à l'aide d'une approche de type centrée individus. On peut alors considérer ces fonctions globales et les intégrer dans des modèles de plus haut niveau. Cette approche a été appliqué et on en a montré les potentialités sur le système qui nous intéresse. La réflexion doit maintenant être poussée plus en avant afin de confirmer ces potentialités du couplage dynamique de modèles. De plus, nous avons toujours du mal à démontrer que le calcul émergent permet de faire du changement d'échelle. Nous avons encore des débats passionnés à ce sujet dans le cadre des actions PNEC mais aussi dans le projet MIMOSA.

D'autre part, J. C. Poggiale et Y. Lagadeuc utilisent les techniques d'agrégation de variables pour la modélisation multi-échelle. Ces techniques permettent de réduire les systèmes en ne s'attachant qu'à une description des variables dites globales [BER99a]. De telles techniques s'appliquent parfaitement aux systèmes hiérarchisés comme les écosystèmes. Ils permettent en effet de considérer simultanément une dynamique rapide, par exemple celle de l'individu, et une dynamique plus lente, celle de la population. La principale question sous-jacente est comment coupler dynamiquement ou non cette approche mathématique et l'approche centrée individus intégrant DEVS.

Annexe A

Les activités de recherche

A.1 CV

Eric RAMAT
125, rue des coquelicots
62370 AUDRUICQ

Tél. (bureau) : 03-21-46-36-94 ou 03-21-46-56-73
Tél. (standard) : 03-21-46-36-00 ou 03-21-46-06-80
e-mail : ramat@lil.univ-littoral.fr

Né le 3 avril 1970 à Angoulême (Charente)

- 1997 : Doctorat en Informatique au Laboratoire d'Informatique de l'E3i/Université de Tours (soutenue le 6 juin 1997) - Mention Très Honorable avec Félicitations du jury
- 1994 : Diplôme d'Ingénieurs en Informatique pour l'Industrie à Tours - Mention Très Bien
- 1993 : DEA Automatique et Génie Informatique à Tours - Mention Bien

A.2 Contrats

2003-2004 : Co-responsable de deux actions "Couplage de modèles informatiques et mathématiques pour l'étude de l'interaction phytoplancton - zooplancton en milieu hétérogène : de l'individu à la population" et "Modélisation de la dynamique saisonnière de la communauté zooplanctonique à partir de la structure de taille des organismes" du projet "Hiérarchisation de processus et transferts d'échelles dans le milieu marin par une approche de modélisation dynamique" piloté par Jean-Christophe Poggiale de l'action de recherche thématique 2 du PNEC (Plan National d'Etude Côtière). Equipe pluridisciplinaire de 6 personnes et budget de 6 Keuros.

2001-2003 : Responsable de l'action "Couplage de modèles informatiques et mathématiques pour la modélisation des transferts d'échelle dans un milieu hétérogène : application à l'étude de l'écosystème pélagique côtier" du projet "Hiérarchisation de processus et transferts d'échelles dans le milieu pélagique marin par une approche de modélisation dynamique" de l'action de recherche thématique 2 du PNEC (Plan National d'Etude Côtière). Equipe pluridisciplinaire de

6 personnes et budget de 6 Keuros.

Transferts technologiques : 2 contrats
- 2000 : Société Coramy (Gravelines) : étude, conseil et encadrement d'un stage étudiant financés par l'Anvar pour le développement d'un système commercial basé sur les technologies Web - budget : 30 KF
- 2001 : Société Aria-Services (Calais) : formation et suivi de projets financés par l'Anvar sur le thème intégration des technologies Web - budget : 13 KF

A.3 Groupes de travail

1999-2003 : le groupe GTIMMS[1] (Groupe de Travail en Informatique sur la Multi-Modélisation et la Simulation)
- participation à la création d'un groupe de travail, relevant de la 27ème section, sur la simulation multi-modèles (piloté à l'origine par Mamadou Traoré du Laboratoire d'Informatique de Modélisation et d'Optimisation des Systèmes de l'Université Blaise Pascal - Clermont-Ferrand)
- membre du groupe d'animation
- organisateur de la deuxième réunion de travail en 2000
- le groupe est aujourd'hui intégre au GdR I^3 et au tout nouveau GdR MACS

2001-2003 : le groupe MIMOSA[2]
- piloté par J. P. Müller (Cirad – Montpellier)
- membre du groupe d'animation
- responsable de l'action "Spécifications XML" et co-responsable de l'action "Spécifications des plans d'expériences"
- le groupe est financé par le GdR I^3

2003-2006 : membre et co-fondateur de l'équipe projet multi-laboratoires "Spécifications DEVS, modélisation multi-agents et simulation distribuée" des STIC/CNRS.

2003-2004 : membre actif de l'action spécifique "Vers une théorie de la simulation" pilotée par Norbert Giambiasi.

A.4 Encadrements scientifiques

A.4.1 Participations à l'encadrement de thèses

2003/2006 : Encadrement de la thèse de Gauthier Quesnel : *Framework* d'intégration de modèles hétérogènes pour la modélisation et la simulation de systèmes spatio-temporels

[1] http ://mad3.univ-bpclermont.fr
[2] http ://www-lil.univ-littoral.fr/Mimosa

2000/2003 : Encadrement à 80% de la thèse de Raphaël Duboz (bourse région "Nord-Pas de Calais" et moniteur CIES) dont le thème est "Intégration de modèles hétérogènes pour la modélisation et la simulation de systèmes complexes : application à l'écosystème marin de la Manche orientale" – soutenance prévue le 11 décembre 2003.

A.4.2 Encadrement de DEA

2002/2003 : Encadrement du stage de Gauthier Quesnel sur le thème : "Étude de la dynamique de populations de animaux zooplanctoniques par la modélisation et la simulation d'un grand nombre d'agents réactifs".

A.4.3 Autres encadrements d'étudiants

2003/2004 : Développement de modules complémentaires de VLE (2ème et 3ème année / IUP GMI et DESS ISIDIS) :
- Portail Web pour VLE - suite
- Développement de wrappers pour les *state charts*, les équations différentielles et les automates à états finis
- Modules de simulations distribuées : développement de connecteurs Java, C++, Smalltalk, Pyhton et Fortran

2002/2003 : Recherche bibliographique sur les outils de modélisation et simulation des flux urbains dans le cadre du DEA MOSC ; Développement de modules complémentaires de VLE (2ème et 3ème année / IUP GMI et DESS ISIDIS) :
- Portail Web pour VLE - suite
- Interface Web pour la spécification en XML de multi-modèles - suite
- Modules de simulations distribuées avec RMI et SOAP

2001/2002 : Encadrement du stage de Olaf Duteil de modélisation biologique et des écosystèmes : mise au point d'un modèle informatique de la dynamique de population de copépodes (Maîtrise de biologie des populations - Paris VI)

2001/2002 : Classification automatique à l'aide de chaînes de Markov cachées des trajectoires de copépodes (2ème année d'Ingénieurs - E3i/Tours) Développement de modules complémentaires de VLE (2ème et 3ème année / IUP GMI, DESS ISIDIS et DESS ICC) :
- Portail Web pour VLE
- Interface Web pour la spécification en XML de multi-modèles
- Outils de représentation et d'analyse de données (2ème partie)

2000/2001 : Développement de modules complémentaires de VLE (2ème année / IUP GMI) :
- Editeur graphique de diagrammes dynamiques UML (state charts)
- Intégration d'un moteur d'inférences ordre 0+ dans VLE
- Outils de représentation et d'analyse de données (1ère partie)
- Interface de spécification visuelle des relations entre les objets

1999/2000 : Développement de modules complémentaires de VLE (2ème année / IUP GMI et DESS ICC) :

- Développement d'un système distribué pour la simulation multi-agents (PVM et PM2)
- Fenêtres pour le suivi en temps réel de l'évolution de variables numériques d'une simulation

1998/99 : Visualisation d'un modèle 3D d'une région géologique (3ème année / IUP GMI)

1997/98 : Développement d'un éditeur de réseaux de Petri en Java s'intégrant dans l'environnement " Laboratoire virtuel " (3ème année / IUP GSI option Informatique)

A.5 Collaborations

- Station marine de Wimereux – URA CNRS 1363 / Université de Lille I (Laurent Seuront – Chargé de recherche et Sami Souissi – Maître de conférences)
- Laboratoire des Sciences de l'Information et des Systèmes – UMR-CNRS 6168 (Norbert Gambiassi - Professeur)
- Université de Rennes 1 – UMR CNRS ECOBIO (Yvan Lagadeuc – Professeur)
- Université de Bordeaux – UMR 5805 – EPOC (François Carlotti – Chargé de recherche)
- Centre d'Océanologie de Marseille, Laboratoire d'Océanographie et de Biogéochimie – UMR CNRS 6535 (J. C. Poggiale, HdR)
- Institut de Recherche pour le Développement – Centre de Bondy, UR GEODES (O. Arino, Directeur de Recherche – Professeur)
- CIRAD – Montpellier (J.P. Müller, cadre scientifique)
- LIRMM – Montpellier (J. Ferber, professeur)
- Université de Lille 3 – Grappa – EA 3588 (Philippe Preux – Professeur)
- IFREMER – Brest – Equipe d'économie marine (O. Thébault)

A.6 Diffusion

A.6.1 Séminaires et groupes de travail

- GTIMMS :
  - 1999 : "Modélisation multi-agents des systèmes naturels" – Calais
  - 2003 : "Simulations distribuées d'un système spatio-temporel" – Saint-Etienne
  - 2003 : "Modélisation multi-agents des systèmes spatio-temporels : le *framework* Mimosa" – Marseille
- MIMOSA :
  - 2003 : "XML : interopérabilité des modèles et des plans d'expériences" – Paris
  - 2003 : "DEVS et les agents réactifs" – Paris
- PNEC – Art2 :
  - 2000 : "Hétérogénéité, turbulence et copépode : une approche multi-agent (centrée individu)" – Paris
  - 2002 : "Modélisation centrée individus, couplage de modèles et transfert d'échelles" – Paris (IRD)
  - 2003 : "Couplage faible, couplage fort et transfert d'échelles" – Marseille
- Cemagref :
  - 2002 : "Modélisation multi-agents et à événements discrets : couplage et changement d'échelles" – Clermont-Ferrand

- LIL :
  - 1999 : "Agrégation et désagrégation d'agents" – Calais
  - 1998 : "L'approche multi-agents appliquée à l'océanographie" – Tours
- Groupe Colline du GdR I^3 :
  - 2000 : "Modélisation multi-agents des systèmes naturels" – Rennes
- URECA :
  - 1999 : "Modélisation multi-agents du comportement du copépode" – Lille
- GdR Manche :
  - 1997 : "Les laboratoires virtuels" – Boulogne sur Mer

A.6.2 Formation DEA

2002-2004 : chargé du module "Modélisation et simulation des systèmes naturels" du DEA MOSC (Modélisation et Simulation des Systèmes complexes) de l'Université du Littoral

1998 : intervenant dans le DEA Océanologie de Lille 1 / Paris 6 sur le thème "Modélisation centrée individus pour la biologie"

2004 : chargé d'une partie du module "Simulations, Relation modèle/données" du futur master recherche de Rennes 1 "Modélisation en biologie".

A.7 Publications

Ce premier ensemble de publications concerne directement les travaux effectués au Laboratoire d'Informatique du Littoral. On y retrouve principalement les contributions liées à la modélisation et à la simulation des systèmes complexes.

[DUB04] R. DUBOZ, E. RAMAT et P. PREUX, *Coupling agent based models with differential equation systems : from individual to population scale*, ASLO - Ocean Research - Conference, Honolulu (Hawaï), février 2004.

[RAM03a] E. RAMAT et P. PREUX, *Virtual Laboratory Environment (VLE) : An Software Environment oriented Agent and Object for Modeling and Simulation of Complex Systems*, Journal of Simulation Practice and Theory 11, pp. 45–55, mars 2003.

[DUB03b] R. DUBOZ et E. RAMAT, *Towards the simulation of scale transfert in ecological modelling using computational emergence : Parametrization of classical population model with reactive agent model*, Systems Analysis – Modelling – Simulation, vol. 43, n^o6, pp. 793–814, juin 2003.

[RAM03b] E. RAMAT, *Agent-Based and individual-based modelling and simulation in Ecology*, Conférencier invité, Agent-based simulation 2003, Montpellier (France), avril 2003.

[DUB03a] R. DUBOZ, F. AMBLARD, E. RAMAT, G. DEFFUANT, P. PREUX, *Individual-based model to enrich an aggregate model*, Internationnal Workshop Model To Model, pp. 57–61, Marseille, 2003.

[DEF03] G. DEFFUANT, F. AMBLARD, R. DUBOZ, E. RAMAT, *Une démarche expérimentale pour la simulation individus-centrée*, Rochebrune : Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels, pp. 45–64, janvier 2003.

[DUB02b] R. DUBOZ, E. RAMAT et N. GIAMBIASI, *Utilisation du Formalisme DEVS pour la Spécification de Systèmes d'Agents Réactifs*, JFIADSMA 2002, Lille, 17p., 2002.

[DUB02a] R. DUBOZ et E. RAMAT, *Modélisation et Simulation Objet et à Événements Discrets : Quels apports possibles pour l'écologie théorique ?*, XXIIème Séminaire de la Société Française de Biologie Théorique, Saint-Flour, juin 2002.

[DUB01] R. DUBOZ, E. RAMAT et P. PREUX, *Towards a coupling of continuous and discrete formalism in ecological modelling : influences of the choice of algorithms on results*, European Symposium on Simulation, pp. 481–487, Marseille, 2001.

[RAM01] E. RAMAT et P. PREUX, *Virtual Laboratory Environment (VLE) : un environnement multi-agents et objet pour la modélisation et la simulation de systèmes complexes*, MOSIM, 2001.

[RAM00] E. RAMAT et P. PREUX, *Virtual Laboratory Environment (VLE) : un environnement multi-agents pour la modélisation et la simulation d'écosystèmes*, Session Démonstration, JFIADSMA, 2000.

[SEU00] L. SEURONT, E. RAMAT, P. PREUX et Y. LAGADEUC, *An individual-based approach of zooplankton behavior in microscale phytoplankton patches*, ASLO, 2000.

[SEU99] L. SEURONT, F. SCHMITT, Y. LAGADEUC, E. RAMAT et P. PREUX, *Turbulence intermittency and small-scale phytoplankton patchiness : effects on plankton trophodynamics*, XXIV General Assembly of the European Geophysical Society, The Hague, Pays Bas, 19-23 avril 1999.

[RAM98] E. RAMAT, P. PREUX, L. SEURONT et Y. LAGADEUC, *Modélisation et simulation multi-agents en biologie marine. Etude du comportement du copépode*, Colloque SMAGET, Cemagref, Clermont-ferrand, pp. 35–49, 6-8 octobre 1998.

Le deuxième groupe de publications concernent les travaux effectués au Laboratoire d'Informatique de Tours ou en collaboration avec ses membres.

[MON00] N. MONMARCHE, E. RAMAT, L. DESBARATS et G. VENTURINI, *Probabilistic Search with Genetic Algorithms and Ant Colonies*, OBUPM-2000, 3p., juillet 2000.

[RAM97a] E. RAMAT, C. LENTE et C. TACQUARD, *Modélisation de l'incertitude dans les projets d'innovation : le modèle RAIH*, Journal Européen des Systèmes Automatisés (JESA), vol 31, n¡4, p615-643, juin 1997.

[RAM97b] E. RAMAT, M. SLIMANE et C. LENTE, *Fiabilité et le risque en gestion de projets*, XXIXème journée de statistiques, Carcassonne, p678-681, 26-30 mai 1997.

[LEN97] C. LENTE, E. RAMAT et C. TACQUARD, *Durée et chemins critiques d'un PERT stochastique*, Actes de la première conférence francophone en modélisation et simulation des systèmes de production et de logistique (MOSIM'97), ed. Hermès, p459-466, juin 1997.

[RAM97c] E. RAMAT, G. VENTURINI , C. LENTE et M. SLIMANE, *Solving Multiple Resource Constrained Project Scheduling Problem with Hybrid Genetic Algorithm*, 7th International Conference on Genetic Algorithms, East Lansing (Michigan - USA), p489-496, juillet 1997.

[RAM97d] E. RAMAT, G. VENTURINI, C. LENTE et M. SLIMANE, *Planification de projets à contraintes de ressources multiples : un algorithme génétique basé sur la fréquence des gènes*, 2ème congrès international franco-québécois, Albi, 5-7 septembre 1997.

[VER97] G. VERLEY et E. RAMAT, *Générateur de données et superviseur d'expériences*, MODULAD, p39-60, janvier 1997.

[RAM96a] E. RAMAT, C. LENTE et C. TACQUARD, *Modélisation et planification des*

projets d'innovation : le modèle RAIH, 5ème Congrès International de Génie Industriel, Grenoble, p193-198, avril 1996.

[RAM96b] E. RAMAT, C. LENTE et C. TACQUARD, *Project management and scheduling in a stochastic environment - A step towards innovation project*, 5th International workshop on project management and scheduling (PMS'96), Poznan (Pologne), p194-197, avril 1996.

[TAC96] C. TACQUARD, E. RAMAT, C. LENTE et C. PROUST, *An application of RAIH model : industrial case*, Workshop on " Modeling the Product Realization Process : Innovative methods, computer tools and applications ", Conférence Invitée, Washington, 5p., 6-7 mai 1996.

[RAM96c] E. RAMAT, C. LENTE, M. SLIMANE, C. TACQUARD et G. VENTURINI, *Planification de projets stochastiques par détermination de décalages temporels*, XXVIIIème journée de statistiques, Québec, p622-627, 27-30 mai 1996.

[RAM96d] E. RAMAT, C. LENTE et C. TACQUARD, *Scheduling methods for stochastic projects*, 29th ISATA, Florence (Italie), p285-292, 3-6 juin 1996.

[RAM96e] E. RAMAT, C. LENTE, C. TACQUARD et C. PROUST, *The RAIH model : application on Product Realization Process example*, Workshop on Production planning and control, Mons (Belgique), p178-183, septembre 1996.

[RAM96f] E. RAMAT, C. LENTE, M. SLIMANE, C. TACQUARD et G. VENTURINI, *Stochastic Project Scheduling based on Time Lag*, IEEE/SMC'96, Pékin (Chine), p2916-2921, 14-17 octobre 1996.

Annexe B

Les activités administratives

Depuis janvier 2000, j'assure la fonction de **directeur du département disciplinaire d'informatique**. Les prérogatives de cette fonction sont :

- la conception des parcours de formations en informatique de l'université en relation avec les porteurs de projets,
- le suivi pédagogique des formations attachées au département (un IUP, un DEUST, une Licence Professionnelle et deux DESS) et des enseignements d'informatique dans les formations scientifiques (DEUG MIAS, SM, SV),
- la gestion des relations entre le département et le Président de l'université et les autres départements,
- la gestion administrative du personnel enseignant du département (3 professeurs, 10 maîtres de conférences, 4 PRAG, de 3 à 5 ATER selon les années, 2 moniteurs CIES et une vingtaine de vacataires),
- la promotion des formations en informatique (salons, présentations, ...).

Un petit bilan de mon action durant les 4 dernières années avec l'aide de l'ensemble des membres du département :

- la création de deux DESS informatique (en 1999 et 2000),
- la création de la mention Informatique dans la Licence Sciences et technologies du projet LMD (en 2003),
- le doublement des effectifs en IUP GMI (de 1999 à 2002),
- la création de 2 postes de professeurs et de 2 postes de maîtres de conférences,
- la création du Licence Professionnelle en Informatique de Réseaux et Télécommunications en partenariat avec deux départements de l'IUT de Calais,
- en conclusion, une progression de notre offre et de notre potentiel qui se traduit par un triplement de nos effectifs étudiant,

En résumé, mes autres activités administratives :

depuis février 2003 : **Directeur adjoint du Laboratoire d'Informatique du Littoral (LIL)**.

2000-2004 : Membre du conseil des études et de la vie étudiante (CEVU) de l'Univeristé.

2002-2003 : Membre suppléant de la commission de spécialistes – section 27 de l'ULCO.

2002-2004 : Responsable et président de jury de l'IUP Génie Mathématiques et Informatique.

2002-2003 : Porteur du projet de Licence Sciences et technologies mention Informatique

dans le cadre du passage au système LMD (Licence/Master/Doctorat).
2002-2003 : Encadrement du projet de la discipline informatique pour le passage au système LMD dans le secteur Sciences de l'ULCO. Négociations inter-disciplinaires pour les Licences et Masters.
2002-2003 : Participation au dévéloppement pédagogique de la Licence Professionnelle du département informatique et du département GEII de l'IUT de Calais de l'ULCO.
2000-2004 : Responsable du tutorat en informatique 1er cycle.
2000-2003 : Membre de la commission "Campus numérique" et chef de projet "DESS ICC en ligne".
2001-2002 : Membre de la commission de réflexion sur la création d'un Centre de Ressources Informatique de l'ULCO
1999-2002 : Responsable du DESS Informatique - Compétences complémentaires (habilité en juillet 1999) et de la formation continue en Informatique de l'IUP GMI de Calais (gestion pédagogique du catalogue de formations en informatique inter-entreprises en collaboration avec le CUEEP)
1999 : Porteur du projet d'habilitation du DESS Informatique - Compétences complémentaires
1999 : Participation à l'organisation locale de la conférence internationale Evolution Artificielle 99 (EA'99)
1998/99 : Responsable de la formation continue en Informatique de l'IUP GMI de Calais :
- développement et gestion pédagogique du catalogue de formations en informatique inter-entreprises (40 modules) en collaboration avec le CUEEP
- rédaction du dossier d'habilitation du DESS Informatique Compétences complémentaires
- mise en place de la licence / maîtrise en informatique en formation continue conjointement avec le DUIPS Réseaux et Communications

Annexe C

Les activités d'enseignement

C.1 Activités d'enseignement au DEA Modélisation et simulation des systèmes complexes (MOSC) de l'Université du Littoral - Côte d'Opale

2002/2003 et 2003/2004 : 12 étudiants en 2002 et 10 étudiants en 2003

- Modélisation et simulation des systèmes naturels : 9h de cours (sur les 18h du module ; la deuxième partie est réalisée par J. C. Soulié)

C.2 Activités d'enseignement au DESS Informatique - Compétences complémentaires de l'Université du Littoral - Côte d'Opale

2001/2003 : 16 étudiants et 25 étudiants

- Java : 10h de cours et 35h de travaux pratiques
- Méthode et conception orientée objet (UML) : 10h de cours et 15h de travaux dirigés

2000/2001 : 24 étudiants

- Programmation orientée objets – Java : 10h de cours et 35h de travaux pratiques
- Langage C++ : 10h de cours et 25h de travaux pratiques
- Méthode et conception orientée objet (UML) : 10h de cours et 15h de travaux dirigés

1999/2000 : 24 étudiants

- Java : 10h de cours et 25h de travaux pratiques
- C++ : 10h de cours et 25h de travaux pratiques
- Méthode et conception orientée objet (UML) : 10h de cours, 15h de travaux dirigés et 15h de travaux pratiques
- HTML/PHP/Java Script : 20h de travaux pratiques

C.3 Activités d'enseignement à l'IUP Génie Mathématiques et Informatique de Calais - Université du Littoral - Côte d'Opale

2003/2004[3]

Deuxième année[4] (48 étudiants - 2 groupes de TD/TP)
- Langage Orienté Objets - Java : 18h de cours, 2x36h de travaux pratiques
- Langage C++ : 10h de cours et 2x30h de travaux pratiques

Première année (42 étudiants - 2 groupes de TD/TP)
- Processeur et assembleur : 9h de cours, 2x21h de travaux pratiques

2000/2003

Deuxième année (48 étudiants - 2 groupes de TD/TP)
- Langage Orienté Objet - Java : 18h de cours, 2x36h de travaux pratiques
- HTML/PHP/Java Script[5] : 2x15h de travaux pratiques

Troisième année (46 étudiants - 2 groupes de TD)
- Méthode et conception orientée objet (UML) : 15h de cours, 15h de travaux dirigés et 20h de travaux pratiques[6]

1999/2000

Deuxième année (28 étudiants - 2 groupes de TP)
- Intelligence artificielle : 20h de cours, 20h de travaux dirigés et 2x20h de travaux pratiques

Troisième année (20 étudiants)
- Méthode et conception orientée objet (UML) : 15h de cours, 15h de travaux dirigés et 20h de travaux pratiques

1998/1999

Deuxième année (27 étudiants - 2 groupes de TP)
- Intelligence artificielle : 20h de cours, 20h de travaux dirigés et 2x20h de travaux pratiques
- Théorie des langages et compilation : 20h de cours, 20h de travaux dirigés et 2x20h de travaux pratiques

Troisième année (18 étudiants)
- Méthode et conception orientée objet (UML) : 15h de cours, 15h de travaux dirigés et 20h de travaux pratiques

Formation continue
- Développement avancé en HTML : 14h de cours et de travaux pratiques
- Administration d'un site Internet/Intranet : 2x14h de cours et de travaux pratiques

[3] Suite à l'arrivée de nouveaux enseignants, nous avons recentrés l'activité de chacun sur un domaine plus restreint.

[4] Les enseignements de travaux pratiques sont réalisés avec l'aide d'étudiants de DEA pour les initier à l'enseignement.

[5] Seulement en 2000-2001

[6] Seulement en 2001/2002

Bibliographie

[ARD89] Arditi R. et Ginzburg L. R., *Coupling in Predator-Prey Dynamics : Ratio-Dependence*, Journal of theoritical Biology, pp. 311-326, vol139, 1989.

[BAR95] Barros F. J., *Dynamic Structure Discrete Event System Specification : A New Modeling and Simulation Formalism for Dynamic Strcuture Systems*, Proceedings of the 1995 Winter Simulation Conference, pp. 781-785, 1995.

[BER99] Berio G., Leva A. D., Giolito P. et Vernadat F., *Process and Data Nets : The Conceptual Model of the $M^* - -OBJECT$ Methodology*, IEEE Transactions on Systems, Man, and Cybernetics SMC 29, pp. 104-114, 1999.

[BER99a] Bernstein C., Auger P., Poggiale J. C., *Predator Migration decisions, the Ideal Free Distribution and predator-prey dynamics*, American Naturalist, 1999.

[BOL02] Bolduc J. S. et Vangheluwe H., *Expressing ODE models as DEVS : Quantization approaches*, dans Fernando Barros et Norbert Giambiasi editors, Proceedings of the AIS'2002 Conference (AI, Simulation and Planning in High Autonomy Systems), pp. 163 - 169, Society for Modeling and Simulation International (SCS), Lisbonne, Portugal, 2002.

[BOO97] Booch G., Rumbaugh J. et Jacobson I., *Unified Modeling Language User Guide*, ISBN : 0-201-57168-4, ed. Addison Wesley, 1997.

[BOR03] Borland S. et Vangheluwe H., *Transforming Statecharts to DEVS*, dans A. Bruzzone et Mhamed Itmi editors, Summer Computer Simulation Conference, Student Workshop, pp. S154 - S159, Society for Computer Simulation International (SCS), Montréal, Canada, 2003.

[BOU94] Bousquet F, Cambier C. et Morand P., *Distributed Artificial intelligence and Object-oriented Modelling of a fishery*, Mathl. Comput. Modelling, vol. 20, pp. 97-107, 1994.

[BOU98] Bousquet F., Bakam I., Proton H. et Le Page C., *Cormas : common-pool resources and multi-agent systems*, Lecture Notes in Artificial Intelligence, 1416 :826-838, 1998.

[BRO93] Brown D. et Rothery P., *Models in Biology : Mathematics, Statistics and Computing*, éd. Wiley Publishers, 1993.

[BRO96] Browman H. I., *Predator-prey interaction in the sea : commentaries on the role of turbulence*. Mar. Ecol. Prog. Ser., vol. 139, pp. 301-312, 1996.

[BUN93] Bundy M. H., Gross T. F., Coughlin D. J. et Strickler J. R., *Quantifying copepod searching efficiency using swimming pattern and perceptive ability*, Bulletin of Marine Science, vol 53, n¡1, pp. 15-28, 1993.

[CAM94] Cambier Ch., Simdelta – Un système multi-agents pour simuler la pêche sur le delta central du Niger, Thèse de doctorat, Université de Paris 6, 1994

[CAM97] Cambier Ch., Perrier E., Treuil J-P. et Preux Ph., Action physique et géométrique, Contribution à une réflexion sur l'utilisation des processus physiques, Application RIVAGE, poster, 5ième JFIADSMA, avril 1997

[CAP96] Caparroy P. et Carlotti F., *A model for Acartia tonsa : effect of turbulence and consequences for the related physiological processes*, Journal of Plankton Research, vol.18, no.11, pp.2139-2177, 1996.

[COQ97] Coquillard P. et Hill D., *Modélisation et simulation d'écosystèmes*, Ed. Masson, 273 p., 1997.

[CUB97] Cubert R. M. et Fishwick P. A., *MOOSE : An Object-Oriented Multimodeling and Simulation Application Framework*, Simulation, 1997.

[DEL91] Delobel C., Lécluse C. et Richard P., *Bases de données : des systèmes relationnels aux systèmes à objets*, Inter-Editions, 1991.

[DEU97] Deutschman D. H., Levin S. A., Devine C. et Buttel L. A., *Scaling from Trees to Forests : Analysis of a Complex Simulation Model*, Science Online : A new electronic journal published by AAAS, 1997.

[DUB01] Duboz R. Ramat E. et Preux P., *Towards a coupling of continuous and discrete formalism in ecological modelling : influences of the choice of algorithms on results*, ESS'01, Marseille (France), pp. 481–487, 2001.

[DUB02a] Duboz R. et Ramat E., *Modélisation et Simulation Objet et à Événements Discrets : Quels apports possibles pour l'écologie théorique ?*, XXIIème Séminaire de la Société Française de Biologie Théorique, Saint-Flour, juin 2002.

[DUB02b] Duboz R., Ramat E. et Giambiasi N., *Utilisation du formalisme DEVS pour la spécification de systèmes d'agents réactifs*, JFIADSMA 2002, Lille, pp 99–102, 2002.

[DUB02c] Duboz R., Ramat E. et Giambiasi, *Introduction à la modélisation à évènements discrets : Modélisation d'un système proie-prédateur*, Rapport interne n^0 LIL 2002-2, 32p., 2002.

[DUB02d] Duboz R., *XML for the representation of semantic in model coupling*, AIS'2002 : AI, Simulation and Planning in High Autonomy Systems, pp. 267-270, Lisbonne, Portugal, 2002.

[DUB03a] Duboz R., Amblard F., Ramat E., Deffuant G. et Preux P., *Individual-based model to enrich an aggregate model*, Internationnal Workshop Model To Model, pp. 57–61, Marseille, 2003.

[DUB03b] Duboz R. et Ramat E., *Towards the simulation of scale transfert in ecological modelling using computational emergence : Parametrization of classical population model with reactive agent model*, Systems Analysis Modelling Simulation, Vol. 43, No. 6, pp. 793-814, 2003.

[DUB03c] Duboz R., Ramat E., Amblard F., Deffuant G. et Preux P., *Utiliser les modèles individus-centrés comme laboratoire virtuels pour identifier les paramètres d'un modèle agrégé*, MOSIM, Organisation et conduite d'activités dans l'industrie et les services, Toulouse, pp. 353–357, 2003.

[DUB03d] Duboz R., Ramat E., Giambiasi N. et Preux P., *Using DEVS for the specification of Reactive Agent Systems : Application to a prey-predator model in marine ecology*, Transactions of The Society for Modeling and Simulation International, 20p., soumis, 2003.

[ECU98] L'Ecuyer P., *Random Number Generation*, Chapter 4 of the Handbook on Simulation, Jerry Banks Ed. Wiley, pp. 93–137, 1998.

[ECU02] L'Ecuyer P., Simard R., Chen E. J. et Kelton W. D., *An Objected-Oriented Random-Number Package with Many Long Streams and Substreams*, Operations Research, n_0 50, vol 6, pp. 1073–1075, 2002.

[FER95] Ferber J., *Les systèmes multi-agents, vers une intelligence collective*, InterEditions, 1995.

[FIC55] Fick A., *Über diffusion*, Annalen der physik und chemie, vol. 94, pp. 59-86, 1855.

[FIL02a] Filippi J. B., Chiari. F. et Bisgambiglia P., *Using JDEVS for the modeling and simulation of natural complex systems*, Proceedings of the SCS AIS 2002 conference on simulation in industry, vol 1, p 317, 2002.

[FIL02b] Filippi J. B., Bernardi F. et Delhom M., *The JDEVS environmental modeling and simulation environment*, Proceedings of the the IEMSS 2002 conference on Integrated Assessment and Decision Support, vol3, p. 283, 2002.

[FIS93] Fishwick P. A., *A Simulation Environment for Multimodeling*, Discrete Event Dynamic Systems : Theory and Applications 3, pp. 151-171, 1993.

[FIS95] Fishwick P. A., *Simulation Model Design and Execution : Building Digital Worlds*, Prentice Hall, 448p., 1995.

[FIS02] Fishwick P. A., *Using XML for simulation modeling*, Proceedings of the 2002 Winter Simulation Conference, 2002.

[FON98] Fonlupt C., Marquet P. et Dekeyser J., Data-parallel load balancing strategies", Parallel Computing, n_0 24, p. 1665–1684, 1998.

[FOR90] Forrest S., *Emergent computation : Self organizing, collective and cooperative phenomena in natural and artificial computing networks*, Introduction of the Ninth Annual CNLS Conference, Physica D, vol 42, pp. 1-11, 1990.

[FRO95] Frontier S. et Pinchod-Viale D., *Écosystèmes. Structure-fonctionnement évolution*, éd. Masson, 1995.

[GIA00] Giambiasi N., Escude B., et Ghosh S., *GDEVS : A generalized Discrete Event specification for accurate modeling of dynamic systems*, Transactions of SCS, Vol. 17 No. 3, pp. 120-134, 2000.

[GRI99] Grimm V., *Ten years of individual-based modelling in ecology : what we have learned and what could we learn in the futur*, Ecological Modelling, vol 115, pp. 129-148, 1999.

[GUT00] Gutknecht O. et Ferber J., *The madkit : Agent Platform Architecture*, Agents Workshop on Infrastructure for Multi-Agent Systems, pp. 48-55, 2000.

[HIL96] Hill D., *Object-Oriented Analysis and Simulation*, Ed. Addison-Wesley, 291 p., 1996.

[HIL00] Hilaire V., Koukam A., Gruer P., J.P. Müller J.P. , *Vers une méthodologie formelle de spécification de Systèmes Multi-Agents*, Actes des 8ème Journées Francophones d'Intelligence Artificielles et Systèmes Multi-Agents (JFIADSMA'00), Eds Hermes, pp. 209-221, 2000.

[HOC02] Hocaoglu M. F., Firat C. et Sarjoughian H., *DEVS/RAP : Agent-based simulation*, Proceedings of the Internationnal Conference on AI, Simulation and Planning in High Autonomy Systems, Lisbon, pp. 117-121, 2002.

[HOL59] Holling C. S., *Some characteristics of simple types of predation and parasitism*, Canadian Entomologist Journal, pp 385-398, vol 91, 1959.

[HRA97] Hraber P. T., Jones T. et Forrest S., *The ecology of Echo*, Artificial Life, 3(3), pp. 165-190, 1997.

[HUH98] M. N. Huhngs M. N. et Singh M. P., *Agents and multi-agent systems : themes, approaches and challanges*, Reading in agents, pp. 1-23, 1998.

[JAC97] Jacobson I., Booch G. et Rumbaugh J., *The Objectory Software Development Process*, ISBN : 0-201-57169-2, ed. Addison Wesley, 1997.

[JAC02] Jacques C. et Wainer G. A., *Using the CD++ DEVS Tookit to Develop Petri Nets*, SCS Conference, 2002.

[JEF85] Jefferson D. R., *Virtual time*, ACM Transactions on Programming Languages and Sytems, vol 7(3), pp. 404–425, 1985.

[JOS99] Jost C., Arino O. et Arditi R.,*About deterministic extinction in ratio-dependant predator-prey models*, Bulletin of mathematical Bioloy, vol. 61, pp. 19-32, 1999.

[KAN90] Kanarick C. N., *A technical overview and history of the SIMNET project*,Proceedings of the SCS Multiconference on Distributed Simulation, pp. 104-109, 1990.

[KIM98] Kim Y. J. et Kim T. G., *A heterogeneous simulation framework based on the DEVS bus and the high level architecture*, Proceedings of the 1998 Winter Simulation Conference, 1998.

[KOF01] Kofman E., Lee J. S. , Zeigler B., *DEVS Representation of Differential Equation Systems : Review of Recent Advances*, DEVS Workshop, European Simulation Conference, Marseille, 2001.

[LAM78] Lamport L., *Time, clocks and the ordering of events in a distributed system*, Communications of the Association of the Computing Machinery, 21(7), pp. 558-565, Juillet 1978.

[LAR02] De Lara J. et Vangheluwe H., *Using AToM^3 as a Meta-CASE tool*, 4th International Conference on Enterprise Information Systems (ICEIS), pp. 642 - 649, Ciudad Real, Espagne, 2002.

[LEG97] Legay J. M., *L'expérience et le modèle. Un discours sur la méthode*, INRA, 1997.

[LEM77] Le Moigne J. L., *La théorie du système général - Théorie de la modélisation*, Presses Universitaires de France, 1977.

[LEP96] Le Page C. et Cury P., *How spatial heterogeneity influences population dynamics : Simulations in SEALAB*, Adaptive Behavior, 4(3/4), pp. 249-274, 1996.

[LOK25] Lotka A.J., *Element of physical biology*, Williams and Wikins, Baltimore, 1925.

[MAR97] Martineau P. et Tacquard C., *A generic model to allow an easy construction of the control software for FMS*, Conférence IFAC / IFIP / IMACS / AFCET on control of industrial systems, Belfort, Vol 3/3, pp.456-462, 1997.

[MIG95] Miguet S. et Pierson J. M., *Dynamic load balancing in a parallel particle simulation*, In High Performance Computing Symposium, pp. 420–431, Montréal, Canada, 1995.

[MIN96] Minar N., Burkhart R., Langton C. et Askenazi M., *The SWARM simulation system : a toolkit for building multi-agent simulations*, 1996.

[MIS86] Misra S., *Distributed discrete-event simulation*, ACM Computing Surveys, vol. 18, no. 1, pp. 39–65, 1986.

[PAV94] Pavé A., *Modélisation en Biologie et en Écologie*, éd. Aléas, 1994.

[PER77] Peterson J. L., *Petri Nets.* ACM Computing Surveys, Vol 3, No. 5, pp 221-252, 1977.

[POG94] Poggiale J. C., *Applications des variétés invariantes à la modélisation de l'hétérogénéité en dynamique des populations*, Thèse de doctorat, Université de Bourgogne, Dijon, 1994.

[PRA96] Praehofer H., *An Environment for DEVS-Based Multiformalism Modeling and Simulation in C++*, Proc of Annual Conference on AI, Simulation, and Planning in High Autonomy Systems, San Diego, 1996.

[QUE03a] Quesnel G., *Vers la simulation distribuée et à événements discrets d'entités spatialisées*, Rapport de DEA, Laboratoire d'Informatique du Littoral, 30p. , juin 2003.

[QUE03b] Quesnel G., Nolot F., Duboz R. et Ramat E., *Vers la simulation distribuée et à événements discrets d'entités spatialisées*, Majecstic'03, Marseille, 8p., 2003.

[RAM97] Ramat E., *Modélisation et planification de projets complexes à contraintes de ressources : le modèle RAIH*, Thèse de doctorat, Université de Tours, 222 p., 1997.

[RAM98] Ramat E., Preux P., Seuront L. et Lagadeuc Y., *Modélisation et simulation multi-agents en biologie marine. Etude du comportement du copépode*, Colloque SMAGET, Cemagref, Clermont-ferrand, pp. 35-49, 1998.

[RAM03] Ramat E. et Preux P., *"Virtual laboratory environment" (VLE) : a software environment oriented agent and object for modeling and simulation of complex systems*, Simulation Modelling Practice and Theory, pp. 45-55, vol 11, 2003.

[REA77] Real L. A., *The kinetics of functional response*, American Naturalist, pp. 289-300, vol 111, 1977.

[RES96] Resnick M., *Starlogo : An Environment for decentralized Modeling and decentralized Thinking*, CHI'96 conférence, pp. 11-12, 1996.

[RUM97] Rumbaugh J., Jacobson I. et Booch G., *Unified Modeling Language Reference Manual*, ISBN : 0-201-30998-X, ed. Addison Wesley, 1997.

[SER98] Servat D., Perrier E., Treuil J. P. et Drogoul A., *When Agents Emerge from Agents : Introducing Multi-Scale Viewpoints in Multi-Agent Simulations*, 15 p., MABS'98, Paris, 1998.

[SEU96] Seuront L., Schmitt F., Lagadeuc Y., Schertzer D., Lovejoy S. et Frontier S., *Multifracatl analysis of phytoplankton biomass and temperature in ocean*, Geophys. Res. Lett., 23 (1996), pp. 3591-3594, 1996.

[SEU99] Seuront L., *Hétérogénéité spatio-temporelle et couplage physique-biologie en écologie pélagique : implications sur les flux de carbone. Exemple d'un écosystème côtier à fort hydrodynamisme : la Manche Orientale*, Thèse de doctorat, Université de Lille 1, 1999.

[SEU99a] Seuront L., Schmitt F., Lagadeuc Y., Schertzer D. et Lovejoy S., *Universal multifractal analysis as a tool to characterize multiscale intermittent patterns. Example of phytoplankton distribution in turbulent coastal water*, J. Plankton Res., 1999.

[SEU99b] Seuront L., Schmitt F., Lagadeuc Y., Ramat E. et Preux P., *Turbulence intermittency and small-scale phytoplankton patchiness : effects on plankton trophodynamics*, XXIV General Assembly of the European Geophysical Society, The Hague, Pays Bas, 1999.

[SOU01] Soulié Jean Christophe, *Vers une approche multi-environnements pour les agents*, Thèse de Doctorat, Université de la Réunion, 158 p., 2001.

[STE94] Steering Committee of DIS, *The DIS Vision, A Map to the Future of Distributed Simulation*, Mai 1994.

[TIS90] Tiselius P. et Jonsson P. R., Foraging behaviour of six calanoid copepods : observations and hydrodynamic analysis, Marine ecology progress series, vol. 66, pp. 23-33, 1990.

[UHR94] Uhrmacher A. M. et Arnold R., *Distributing ans Maintaining Knowledge : Agents in Variable Structure Environment*, Proceeding of the 5th Annual Conference on Artificial Intelligence, Simulation and Planning in High Autonomy Systems, pp. 178-184, 1994.

[UHR96] Uhrmacher A. M., *Variable Structure Modelling - Discrete Events in Simulation*, Proceeding of the 6th Annual Conference on Artificial Intelligence, Simulation and Planning in High Autonomy Systems, pp. 133-140, 1996.

[UHR01] Uhrmacher A. M., *Dynamic Structures in Modeling and Simulation - A Reflective Approach*, ACM Transactions on Modeling and Simulation, Vol.11, No.2, pp. 206-232, 2001.

[VAN00a] Vangheluwe H., *DEVS as a common denominator for multi-formalism hybrid systems modelling*. In Andras Varga, editor, IEEE International Symposium on Computer-Aided Control System Design, pages 129-134. IEEE Computer Society Press, Anchorage, 2000.

[VAN00b] Vangheluwe H. et Vansteenkiste G. C., *The cellular automata formalism and its relationship to DEVS*, dans Rik Van Landeghem editor, 14th European Simulation Multiconference (ESM), pp. 800-810, Society for Computer Simulation International (SCS), Ghent, Belgique, 2000.

[VIL01] , Villa F., *Integrating modelling architecture : a declarative framework for multiparadigm, multi-scale ecological modelling*, Ecological Modelling, vol. 137, pp. 23-42, 2001.

[VOL26] Volterra V., *Fluctuations in the abundance of species considered mathematically*, Nature, 118, pp 558-560, 1926.

[WAI01] Wainer G. A. et Giambiasi N., *Application of the Cell-DEVS Paradigm for Cell Spaces Modelling and Simulation*, Simulation 76 :1, pp. 22-39, 2001.

[WEI99] Weis99, Weiss G., *Multiagent Systems. A modern approach to distributed artificial intelligence*, MIT Press, 1999.

[ZEI73] Zeigler B., *Theory of Modeling and Simulation*, Ed. John Wiley, New York, 1973.

[ZEI95] Zeigler B., Song H. S., Kim T. G. et Praehofer H., *DEVS Framework for Modelling, Simulation, Analysis, and Design of Hybrid Systems*, Lecture Notes in Computer Science, 1995.

[ZEI99] Zeigler B, Kim D. et Buckley S, *Distributed supply chain simulation in a DEVS/Corba execution environment*, Proceedings of the 1999 Winter Simulation Conference, 1999.

[ZEI99] Zeigler B, Kim D. et Praehofer H., *Theory of modeling and simulation*, 2nd édition, 1999.

[ZEI00] Zeigler B. et Sarjoughian H. S., *Creating distributed simulation using DEVS M& S environment*, Proceedings of the 2000 Winter Simulation Conference, 2000.

[ZEI00a] Zeigler B., Sarjoughian H. S., et H. Praehofer, *Theory of Quantized Systems : DEVS Simulation of Perceiving Agents*, International Journal Cybernetics and Systems, Vol. 31, No. 6., pp. 611-648, 2000.

Résumé

La compréhension et l'analyse du comportement des systèmes deviennent de plus en plus un enjeu majeur pour la science moderne. Le scientifique a besoin de comprendre les processus de la Nature pour les maîtriser. L'ingénieur doit être capable de concevoir et de valider un nouveau produit en travaillant sur son modèle afin de minimiser les coûts. La complexité des systèmes augmente en fonction des connaissances pour le scientifique et en fonction de la sophistication des technologies pour l'ingénieur. Les outils de modélisation et de simulation doivent donc suivre le mouvement. De nouveaux outils doivent être conçu pour modéliser les processus, les entités et leurs interactions. L'augmentation de la complexité des systèmes fait aussi apparaître de nouvelles pratiques de modélisations et de simulations : la multi-modélisation, le couplage de modèles hétérogènes, la simulation distribuée, ... L'objectif des travaux présentés dans ce rapport est donc de contribuer aux nouvelles pratiques de modélisations et de simulations.
Le *framework* et la plate-forme VLE (*Virtual Laboratory Environment*) sont une réponse. Le *framework* propose un cadre formel de modélisation de modèles hétérogènes et de leur couplage. Cette approche repose en partie sur la proposition de multi-modélisation de Fishwick et sur le *framework* formel DEVS de Zeigler. Nous fusionnons les deux propositions et les enrichissons d'une hiérarchisation en niveaux d'abstrations de l'activité de modélisation et de simulation. Chaque niveau fait l'objet de choix conceptuels et opérationnels. Un langage XML est né du *framework* VLE et autorise la spécification formelle des modèles, de leurs couplages et des plans d'expériences. Ce langage est au cœur de la plate-forme informatique VLE qui est le point d'entrée opérationnel en offrant un serveur de services Web. Elle permet de spécifier formellement les éléments d'un multi-modèle, de gérer le couplage, de mettre en œuvre les schémas expérimentaux à l'aide de simulations distribuées ou non.
La plate-forme VLE est à l'origine et reste un environnement de modélisations et de simulations multi-agents pour les systèmes spatio-temporels. L'introduction du *framework* lui confére des capacités de multi-modélisations et de simulations distribuées. L'aspect multi-agents est central aux applications en biologie marine que l'on développe. Ce domaine d'applications est notre source d'expérimentations et de questionnements. Il nous a permis de construire des modèles multi-agents sur des systèmes spatio-temporels complexes, de réaliser des couplages dynamiques avec des systèmes d'équations différentielles, de mettre en évidence des processus de transfert d'échelles et de calculs émergents, de construire sur des cas concrets des spécifications formelles de type DEVS.

Mots-clés: Multi-modélisation, couplage de modèles, DEVS, spéficiation formelle de systèmes, modélisation multi-agents, simulation à événements discrets, systèmes spatio-temporels, écosystèmes