UNIVERSITÉ PARIS IX-DAUPHINE
UFR SCIENCES DES ORGANISATIONS

Thèse

Pour l'obtention du titre de

Docteur en Informatique

(arrêté du 30 Mars 1992)

Couplage de modèles à l'aide d'agents : le système OSIRIS

présentée et soutenue par

Yawa Edem FIANYO

le 9 avril 2001 devant le jury composé de:

Directeurs de Thèse:	**Mme Suzanne PINSON** Professeur à l'Université Paris IX Dauphine
	M. Yves DEMAZEAU Chargé de Recherche au CNRS
Rapporteurs:	**M. Jean-Pierre MÜLLER** Professeur à l'Université de Neuchâtel, Suisse
	M. Philippe PREUX Professeur à l'Université du Littoral Côte d'Opale
Examinateurs:	**M. Pascal BOIVIN** Directeur de Recherche à l'IRD
	M. Serge HADDAD Professeur à l'Université Paris IX Dauphine
	M. Jean-Pierre TREUIL Ingénieur de Recherche à l'IRD

Numéro :.....

aux habitants de la boite de sardines :

les pionniers : Ruth Paku et Do Franck Fianyo,
les sociétaires du "cirque des 3 Jeunes Filles" : Enyonam et Sedinam,
les trois derniers "E" : Eyram, Emefa et Eyom.

Remerciements

Jean-Pierre Treuil m'a proposé ce sujet, m'a accueillie au LIA, a suivi mon travail durant ces quatre années de thèse, m'a proposé des idées quand je n'en avais pas, a toujours émis des critiques constructives tout en me laissant une grande liberté d'action dans mon travail, a relu toutes mes pages, et les a corrigées sans concession. Pour tout cela (et pour le reste !), je le remercie infiniment.

Je remercie chaleureusement Suzanne Pinson, pour l'attention qu'elle accorde à tous ses étudiants, même ceux qui, comme moi, sont géographiquement éloignés; pour sa relecture attentive de ma thèse; pour avoir été disponible toutes les fois où celà c'est avéré nécessaire. Yves Demazeau m'a apporté à chacune de nos rencontres des conseils pertinents et m'a beaucoup appris sur les exigences d'un travail de recherche. Je lui en suis très reconnaissante.

Jean-Pierre Müller et Philippe Preux ont accepté d'être les rapporteurs de cette thèse. Je les en remercie tous deux très vivement. Je suis particulièrement sensible à l'honneur que me font deux spécialistes des systèmes multi-agents en participant à mon jury de thèse. Je remercie Serge Haddad, dont j'ai eu la chance de suivre le cours sur la spécification formelle des systèmes informatiques, de me faire l'honneur de participer à ce jury de thèse. Je remercie Pascal Boivin, pédologue, spécialiste des problèmes de salinisation des sols, d'avoir accepté d'être dans ce jury, après avoir été à l'origine de ce travail.

L'IRD en tant qu'employeur m'a donné le temps (ingénieur au centre ORSTOM de Dakar, j'ai obtenu en juin 1997, une mise en disponibilité) et les moyens (j'ai été allocataire de l'IRD pendant 3 ans) de réaliser cette thèse. Je tiens à remercier tous ceux dont les décisions et les actions m'ont été favorables, notamment Philippe Mathieu, représentant de l'IRD à Dakar en 1997 et Jean-Pierre Treuil, alors directeur du LIA. Je tiens également à remercier Pascal Renaud, ancien chef de la MTI; Hervé Chevillotte, ancien responsable de l'ULIS à Dakar.

Au Laboratoire d'Informatique Appliquée de Bondy puis à l'UR GEODES, j'ai bénéficié d'une ambiance de travail éclectique et chaleureuse. Je remercie tous les membres de l'UR, en particulier Gaston Pichon, modélisateur parasitologue, pour son humour, son écoute des autres, sa curiosité, pour tous les livres qu'il m'a fait découvrir; Edith Perrier, notre nouvelle chef d'UR, hydro-pédologue, à qui peu de problèmes résistent; Marie Piron, chercheur statisticienne; Chantal Bernard la spécialiste du web; Kathy Baumont l'indispensable secrétaire. Yunne-Jai Shin, l'agro-halieute brillante et perfectioniste, Sarah Feuillette, la spécialiste de la gestion de l'eau passionnée et idéaliste ont été mes deux compagnes de thèse, je les remercie pour tous les moments partagés. Je remercie Jean-François Delerue, informaticien spécialiste du traitement d'image pour son aide précieuse et ses judicieux conseils sur le comportement à adopter face aux imprimantes récalcitrantes; je remercie Dominique Rémy, informaticien, pour son franc-parler et ses suggestions judicieuses sur mon document; je remercie les mathématiciens M'barek Adioui et Abderrahim El Abdellaoui pour leur gentillesse.

L'équipe de l'atelier informatique du centre IRD de Bondy nous a fait traverser sans aucun problème la foudre, le changement de numérotation IP, le bug de l'an 2000, la tempête et bien d'autres événements. Je remercie Bruno Buisson et Françoise Pelletier pour leur efficacité.

Je remercie Jacqueline Castaing (Paris XIII), Georges de Sablet et Albert Widory (Paris V) avec qui j'ai eu l'opportunité et le plaisir de travailler. A leurs côtés, j'ai beaucoup appris en systèmes, en réseaux et en pédagogie.

Je remercie mes amis et mes proches pour tout.

Table des matières

Chapitre 1

Introduction

Le cadre de cette thèse est relatif au couplage de modèles pour l'étude de systèmes environnementaux. Les systèmes environnementaux, caractérisés par leur complexité [Strosser, 1997; Voinov et al., 1999] nécessitent, pour que leur fonctionnement puisse être correctement apréhendé, la prise en compte des différentes dynamiques qui les font évoluer. C'est l'une des principales raisons qui expliquent l'avènement des études pluridisciplinaires, notamment lorsque le but poursuivi est d'appréhender les conséquences de l'action de l'homme sur ces milieux [Valentin, 1987; Bousquet et al., 1994; Quensière, 1994; Skinner et al., 1997; Hasler et al., 1999]. Ces études conduisent généralement à la construction d'un modèle intégré [Shrestha, 1996; Liao et Tim, 1997; Hasler et al., 1999].

La construction d'un modèle intégré nécessite, de la part de son concepteur, une maîtrise suffisante des notions utilisées dans les modèles de départ, afin de pouvoir les fusionner de façon adéquate dans le modèle résultat. Mais dans un contexte marqué par la spécialisation croissante des disciplines scientifiques, cette intégration forte est de moins en moins possible [Vedeld, 1994; Russell, 1996]. La construction de modèles environnementaux réalistes, qui prennent en compte des connaissances écologiques, agronomiques, économiques parmi d'autres doit alors se réaliser sur d'autres bases.

Dans un tel contexte, le concept d'intégration faible [Russell, 1996] ou de couplage faible [Pouliot, 1999], qui peut être considéré comme un des mécanismes d'application de la notion d'interdisciplinarité [Vedeld, 1994; Rouchier et al., 1998; Irvine et al., 1998], prend tout son intérêt. L'objectif n'est plus de fusionner un ensemble de connaissances mais de permettre la coopération entre modèles provenant de différentes disciplines par la construction de passerelles [Vedeld, 1994]. Il devient nécessaire de réfléchir à de nouveaux outils, de proposer de nouveaux concepts permettant ce nouveau type d'intégration. C'est une des préoccupations les plus actuelles de la recherche en modélisation. Elle se traduit par l'apparition dans la littérature de nouveaux termes : environnement collaboratif virtuel *virtual collaborative environment* [Sky et

Buchal, 1999; Khettry et Sun, 2000]; environnement de modélisation collaboratif *collaborative modeling environment* [Maxwell et Villa, 1998; Sarjoughian et al., 2000] et la proposition de nouveaux standards d'architecture de modélisation [Lutz, 2000; Klein, 2000]. Si la plupart des travaux proposent de nouveaux outils d'intégration [Sky et Buchal, 1999; Khettry et Sun, 2000; Sarjoughian et al., 2000], certains auteurs notent le besoin d'accompagner ces outils de *modes d'emploi* afin de guider le modélisateur dans leur utilisation [Maxwell et Villa, 1998; Pohl et al., 1999]. De tels outils doivent être flexibles [Maxwell et Villa, 1998; Sarjoughian et al., 2000]. En modélisation environnementale, le chercheur ne doit pas être contraint à étudier son système selon le point de vue spécifique de son outil d'intégration, celui-ci doit être le plus ouvert et générique possible afin de conserver au chercheur sa capacité d'innovation [Maxwell et Villa, 1998].

C'est dans cette perspective que nous inscrivons notre travail de thèse. La principale contribution de cette thèse est la proposition d'une architecture de couplage, le système OSIRIS (Outil de Simulation et d'Intégration de processus pour l'étude d'une Région Irriguée du Sénégal) pour permettre le couplage dynamique de modèles conçus indépendamment. L'objectif poursuivi est de permettre à un utilisateur modélisateur de faire fonctionner des modèles numériques distribués sur un réseau, grâce à l'utilisation des concepts proposés, les modèles numériques étant appliqués à un système donné.

Cette thèse est organisée en quatre parties. Dans la première partie, nous présentons le cadre et l'objectif de ce travail. Le chapitre 2 présente la problématique du couplage de modèles. Le chapitre 3 est consacré au contexte qui nous sert de base concrète de réflexion : la modélisation de la dynamique saline dans la région du Ngalenka au Sénégal. Le chapitre 4 illustre à titre comparatif une autre démarche de construction d'un modèle intégré dans un contexte similaire.

La deuxième partie présente les différents concepts sur lesquels sont basés les propositions de l'outil OSIRIS. Les notions de modélisation, simulation et agent sont décrites dans le chapitre 5. Le chapitre 6 présente différentes entreprises de couplage à l'aide d'agents et d'objets.

La troisième partie est consacrée au système OSIRIS. Nous commençons (chapitre 7) par illustrer son utilisation sur un exemple, celui précisemment de la dynamique saline d'une région irriguée. Le chapitre 8 est consacré aux difficultés rencontrées dans le couplage de modèles, difficultés que nous relions à différents types d'hétérogénéité, et à la manière dont OSIRIS veut les résoudre. Le chapitre 9 décrit de façon détaillée et formalisée l'architecture interne d'OSIRIS et son fonctionnement. Les résultats de l'exécution d'OSIRIS sur l'exemple sont présentés à titre illustratif dans le chapitre 10.

Dans la dernière partie, nous discutons de l'intérêt de l'outil OSIRIS. Le chapitre 11 précise les apports de notre approche. Nous présentons enfin dans ce chapitre quelques perspectives de recherche.

Première partie

Etudier la salinité dans la région du fleuve Sénégal, une question de couplage

Chapitre 2

Le couplage de modèles

RÉSUMÉ :

Ce chapitre est consacré à la problématique du couplage de modèles. La notion de couplage de modèles est présentée. Elle fait référence à différents concepts qui sont explicités. Trois exemples représentatifs de différents types de couplage de modèles sont ensuite analysés. La fin du chapitre expose les principaux problèmes posés par une démarche de couplage.

2.1 La problématique du couplage de modèles

De façon générale, le terme de couplage de modèles est utilisé lorsque l'on fait interagir au moins deux modèles qui peuvent par ailleurs fonctionner de façon indépendante.

"Modèle" est un terme générique qui recouvre différentes notions. Par modèle, on désigne toute représentation qui permet de décrire un système dans un formalisme donné afin de mieux comprendre une réalité de départ. En sciences humaines, le formalisme utilisé est très souvent le langage (langue écrite ou parlée) mais les sciences proposent en général des formalismes spécifiques et l'on trouve des modèles mathématiques, des modèles physiques, des modèles biologiques, etc. Ces modèles peuvent être traduits en langage informatique et l'on conservera le terme de modèle pour les logiciels créés qui les mettent en œuvre.

Parmi les différents formalismes permettant de spécifier un problème, les méthodes mathématiques s'inscrivent dans la logique "poser le problème c'est le résoudre" alors que les méthodes de simulation sont utilisées lorsque la complexité du problème nécessite la prise en compte d'aspects que n'autorise pas la spécification mathématique, généralement selon une approche systémique. Les modèles de simulation n'excluent cependant pas la spécification par un formalisme mathématique d'une partie du système. Par exemple, on peut utiliser des formules mathématiques pour spécifier des règles de conservation et d'équilibre, ou pour calculer une valeur objectif évaluée par le modèle de simulation. Il s'agit là d'une forme de couplage.

Une démarche de couplage est généralement adoptée lorsque l'on se trouve dans l'un des deux cas suivants :

- Il peut arriver que le problème à modéliser nécessite la prise en compte de plusieurs niveaux de détails (lorsque le processus étudié est actif sur une région très hétérogène, par exemple) et que le meilleur modèle approchant le système est une association de différents modèles existant déjà. Dans cette démarche de couplage, l'objectif est de coupler différents modèles de processus chargés de résoudre la même question avec des niveaux de détails différents.
- La démarche de couplage peut également être envisagée lorsque la question à laquelle on souhaite répondre est une question globale ou complexe (le processus que l'on veut étudier est la résultante d'un enchevêtrement de processus) dont les différentes composantes ont fait l'objet de modèles spécialisés que l'on souhaite faire coopérer dans un modèle qui les intègre. C'est en général le problème posé par les recherches pluridisciplinaires. La question du climat (comme en général les questions écologiques) est un exemple où une prédiction nécessite la prise en compte de différents domaines de savoirs correspondant à des échelles d'espace et de temps différentes.

De façon classique, la compréhension d'un problème passe par la construction d'un modèle représentant ses caractéristiques principales. La simplification qu'implique la construction du

modèle dépend généralement du niveau de finesse des résultats que l'on souhaite obtenir. L'étude de l'infiltration de l'eau dans une parcelle ne fait pas appel au même modèle que celui que l'on va utiliser pour l'étude de l'infiltration au niveau d'une région. On peut classer les modèles s'attaquant à la même problématique selon leur degré de complexité. Gabrielle [Gabrielle et al., 1997] parle de modèles simples, intermédiaires ou complexes en fonction de la qualité des résultats qu'ils permettent d'obtenir. Les modèles complexes présentent des pas de temps courts au contraire des modèles simples au pas de temps plus long, adaptés aux études portant sur le long terme.

Le couplage de modèles est une démarche *a priori* séduisante : elle vise à réutiliser et valoriser des connaissances déjà validéees, en espérant à la fois un moindre risque d'erreur et une construction plus rapide du modèle de simulation. C'est pourquoi le couplage de modèles est une problématique que l'on rencontre à la fois dans les milieux industriels (où la rapidité de conception de nouveaux produits est une question importante) et dans les milieux de recherche où l'on doit traiter des questions de plus en plus complexes. Les prévisions climatiques, la prévision de risques environnementaux ou la simulation des mécanismes urbains sont des exemples de domaines ou le couplage de modèles est une problématique d'actualité.

2.2 Couplage de modèles : différents types, significations, appellations

2.2.1 Différents types de couplage

2.2.1.1 en fonction du degré

De façon générale, les auteurs parlent de degré de couplage en fonction de la façon dont se déroulent les interactions. Lorsque les modèles demeurent indépendants et que l'échange se fait via des transferts de données, on parle de couplage faible.

Les modèles peuvent être indépendants mais partager la même base de données dans le cas où les modèles d'origine sont construits sur le même système. C'est alors au module d'interface de gérer l'intégrité des données utilisées par les modèles, il s'agit dans ce cas d'un couplage fort.

On parle de couplage intégral ou d'intégration de modèles lorsque les modèles sont modifiés afin d'être adaptés les uns aux autres. Ce couplage correspond à l'écriture d'un super modèle et permet de s'assurer la compatibilité parfaite du couplage.

2.2.1.2 en fonction du niveau et de l'approche

Pouliot [Pouliot, 1999] propose un autre classement en fonction du niveau auquel on décide d'effectuer le couplage.

- le couplage de *finalités* fait référence à la définition des objectifs et à l'utilisation envisagée ;
- le couplage de *méthodes* est guidé par la nature de la théorie appliquée et fait intervenir les contraintes reliées à la discipline étudiée;
- le couplage de *techniques* est dépendant des moyens techniques utilisés pour mettre en œuvre le couplage.

Cette distinction est raffinée par la notion d'approche du couplage, que Pouliot classe en deux catégories : l'approche conservative et l'approche compacte. La combinaison de ces deux dimensions de classement donne le contenu du tableau 2.1.

	Finalités	Méthodes	Techniques
approche conservatrice	exploite chaque spécialité en fonction de leur rôle initial	approche par discipline et souvent analytique. Ramène les spécialités à leurs éléments constitutifs les plus simples	fonctionne de manière indépendante, ne touche pas à la structure interne de chaque spécialité
approche coopérative	répond à un seul et même objectif. Souvent relié à la prise de décision	approche pluridisciplinaire et souvent systémique. Considère le système dans sa totalité, se concentre sur les interactions entre éléments	modifie la structure interne d'une des spécialité afin d'y emboîter l'autre. Peut voir la création d'une nouvelle structure.

TAB. 2.1 – *Types de couplage selon Y.Pouliot. [Pouliot, 1999]*

2.2.2 Différentes significations

Comme le souligne Y. Pouliot, le couplage de modèles peut se faire à différents niveaux. Lorsqu'il est effectué au niveau de l'analyse d'un problème, lorsqu'il consiste à fusionner les formalismes de différents modèles de départ, on réalise de facto un couplage fort qui consiste en l'écriture d'un super modèle. Dans cette catégorie, on peut classer le travail de modélisation de l'écosystème du Delta intérieur du Niger [Quensière, 1994]. Pour atteindre l'objectif de départ qui était de construire un modèle permettant d'aborder des questions complexes du type "comment approcher l'évolution d'un tel système sous la pression démographique, les changements climatiques, l'impact d'aménagements agricoles ou hydraulique, l'impact de changements dans la réglementation", des modélisateurs appartenant à différentes disciplines se sont réunis pour spécifier l'ensemble des processus (hydrologiques, hydrobiologiques, végétatifs, d'exploitation

des ressources, etc) qu'ils ont greffés sous forme de comportement (fonctions de déplacement) sur des entités (pêcheurs, agriculteurs, troupeaux) qui leurs semblaient pertinentes. Ce travail de couplage a été implanté informatiquement sous la forme d'un système multi-couches (chacune regroupant les processus communs à une spécialité scientifique donnée).

Lorsque la spécification d'un modèle a eu lieu dans un langage de programmation, le couplage de modèles va consister à réaliser un couplage de logiciels avec modification des programmes de départ (couplage fort) ou non modification (couplage faible). Dans la suite de ce document, lorsque nous parlons de couplage de modèles, nous nous plaçons dans cette dernière perspective.

2.2.3 Différentes appellations

Il faut enfin signaler que même si l'expression "couplage de modèles" est celle qui se retrouve le plus souvent dans la littérature, le terme d'intégration de modèles est également employé [Boudjlida, 1996], notamment lorsqu'il s'agit de mettre l'accent sur le nombre (plus de deux) de modèles que l'on fusionne. Enfin, on trouve parfois la notion de multi-modélisation [Freire-Junior, 1997; Picavet, 1997], lorsque le couplage consiste à combiner différents formalismes de spécifications.

2.3 Exemples de couplage

2.3.1 Modéliser un problème par couplage de modèles de résolution différente : exemple du couplage Boltzmann / Navier-Stokes

Pour calculer les écoulements de gaz hypersoniques autour d'un objet en phase de réentrée dans l'atmosphère, une équipe de chercheurs [Bourgat et al., 1994] en collaboration avec une entreprise industrielle (Dassault Aviation) a proposé un couplage entre deux modèles : équations de Botzmann/équations de Navier-Stokes.

Le choix d'un tel couplage se justifie par le fait que lors de la réentrée, le corps se trouve dans une situation intermédiaire pour laquelle il faut à la fois prendre en compte une situation de gaz très raréfié (où le modèle de Boltzmann convient) et une situation de gaz plus dense où l'utilisation d'un modèle fluide est plus adéquat.

Les équations de Botzmann résolues numériquement selon un algorithme utilisant une méthode de Monte-Carlo, sont utilisées pour modéliser les alentours proches du corps (espace Ω_B sur figure 2.1) tandis que les équations de Navier-Stokes qui modélisent l'écoulement dans le domaine lointain (espace Ω_{NS} sur figure 2.1) sont résolues numériquement par un algorithme basé sur une méthode d'éléments finis de type fourni par Dassault.

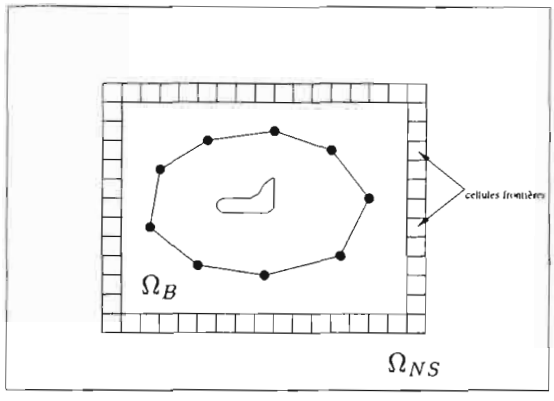


FIG. 2.1 – *Découpage de l'espace du domaine*

Le problème central d'un couplage entre modèles est la gestion des interactions.
Dans le cet exemple, le modèle de Boltzmann utilise un maillage composé de segments alors que les mailles du modèle Navier-Stokes sont triangulaires (cf. figure 2.2). Pour résoudre le problème de l'interface entre les deux modèles, les auteurs ont proposé un algorithme ad hoc pour prendre en compte la frontière entre les deux segmentations.

Pour le code Boltzmann, on génère une couche de cellules frontière. Dans chaque cellule, on génère une distribution uniforme de particule à partir de paramètres récupérés du code Navier-Stokes. Ces paramètres sont attribués à l'isobarycentre de chaque cellule, que lon calcule grâce à l'élément du maillage Navier-Stokes qui contient l'isobarycentre de la cellule considérée.

Pour le code Navier-Stokes, il est nécessaire de calculer les demi-flux sortant du domaine Boltzmann (calcul réalisé par le code Boltzmann). Le code Boltzmann va calculer le nombre de particules qui traversent chaque segment de Ω_B vers Ω_{NS} à partir duquel l'algorithme va calculer le flux sortant en tenant compte de la masse, la vitesse et l'énergie de toutes les particules l'ayant traversé. Le processus de couplage s'établit alors comme suit :

1 Initialisation
   (a) initialiser le problème Navier-Stokes dans le domaine Ω_{NS}
   (b) initialiser le problème Boltzmann dans le domaine Ω_B
2 Itérations en temps
   (a) Boltzmann
      i calculer les conditions aux limites sur la frontière
      ii résoudre N_B itérations en temps dans le domaine Ω_B
   (b) Navier-Stokes
      i résoudre N_{NS} itérations en temps dans le domaine Ω_{NS}

(c) test de convergence : si le test de convergence n'est pas vérifié retourner à l'étape (a) sinon arrêter le processus

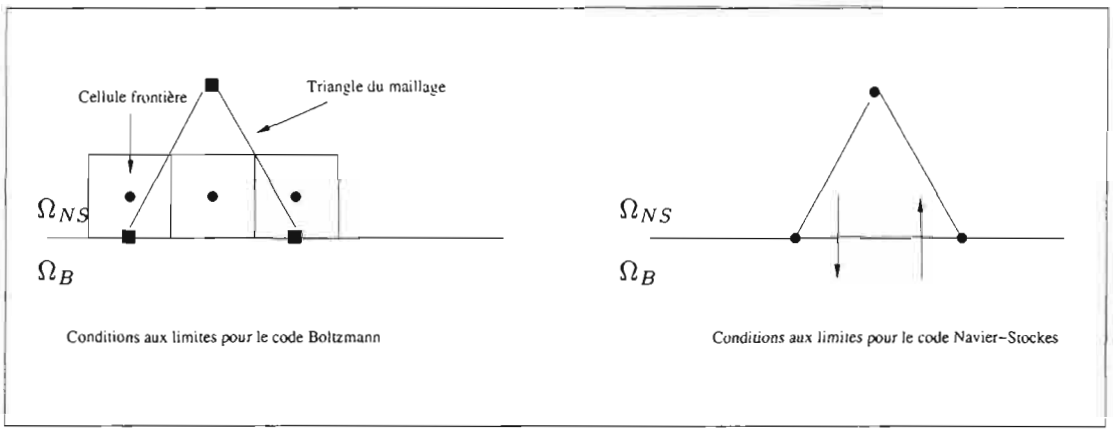


FIG. 2.2 – *Différentes conditions aux limites*

Ce couplage donne des résultats satisfaisants pour les gaz monoatomiques (coûts de calcul raisonnable pour une marge d'erreur spécifiée). Cette démarche a cependant nécessité l'écriture d'un algorithme d'interface assez complexe ainsi que des modifications mineures des deux codes numériques de départ afin de pouvoir faire correspondre leurs paramètres respectifs.

2.3.2 Coupler différents processus par leurs modèles

2.3.2.1 Le projet MOPA

Depuis 1994 et le travail de Bousquet et Cambier [Bousquet et al., 1994] sur le système de pêche dans le Delta central du Niger, de nombreuses recherches pluridisciplinaires ont été initiées (notamment à l'Institut de Recherche de Développement où a été effectué ce travail de thèse) pour une meilleure compréhension des écosystèmes. Certaines ont débouché sur la réalisation d'applications de couplage de modèles. Comme exemple récent de ce genre de démarche on peut citer le projet MOPA (Modélisation de la pêche Artisanale) [LeFur, 1998; LeFur et Bommel, 1998] où pour étudier un système de pêche, il y a eu couplage de deux modèles :

- le premier décrit la dynamique d'une exploitation qui évolue en fonction du comportement des communautés actives (les pêcheurs et les mareyeurs artisans),
- le second modèle représente la dynamique d'une ressource marine donnée (un stock de sardinelles).

Le couplage de ces deux modèles est effectué par l'intermédiaire d'une variable commune : le stock de poissons qui va baisser du fait de l'action de pêche ou augmenter en fonction de la dynamique de la ressource.

2.3.2.2 Le modèle PLM

Le modèle PLM (Patuxent Landscape Model) a été conçu pour étudier les interactions entre les dynamiques physiques et biologiques de la région du Patuxent, conditionnées par le comportement socio-économique de la région [Voinov et al., 1999]. Le couplage du modèle écologique et du modèle socioéconomique a été réalisé par l'intermédiaire d'une représentation spatiale de la région d'étude sous forme de grille, chaque maille de la grille présentant une homogénéité et un niveau de détail permettant aux deux modèles de se communiquer directement des informations de calcul sans transformation supplémentaire. Sur chaque maille de la grille spatiale, une unité du modèle intégré est appliquée, avec des données différant en fonction de la maille considérée (figure 2.3). L'unité de modèle est le résultat de l'opération de couplage effectuée avec l'outil GEM - General Ecosystem Model [Fitz et al., 1996]. Cet outil logiciel permet de connecter différents modèles conçus indépendamment dès lors qu'ils peuvent être manipulés par le logiciel de simulation STELLA. L'environnement spatial de modélisation SME [Maxwell et Costanza, 1995] associé au langage de modélisation modulaire MML ([Maxwell et Costanza, 1997] proposé par Maxwell et Costanza, s'appuyant sur ces différents outils présente alors l'intérêt certain d'offrir au modélisateur un cadre de visualisation graphique facilitant de couplage de modèles. L'intégration de modèles s'effectue via la manipulation d'icônes représentant chacun des modèles que l'on veut coupler. L'application du modèle intégré peut être réalisée par l'utilisation d'un système d'information géographique (SIG), chaque enregistrement de la base de donnée spatiale étant associée à une unité de modèle ce qui favorise une rapide compréhension des résultats de la simulation.

2.4 Les problèmes posés par le couplage

2.4.1 Trouver un mécanisme d'interaction adapté

Les deux premiers exemples ci-dessus (Navier-Stokes/Boltzmann et MOPA)illustrent le principal problème du couplage de modèles qui réside dans le choix d'un mécanisme d'interaction adapté. Deux possibilités se présentent pratiquement:

On peut choisir de modifier le code des modèles de départ afin de faciliter leur interconnexion (cas du couplage Boltzmann/Navier-Stokes). Cela a pour avantage de simplifier le mécanisme d'interaction. Dans ce cas, il y a fusion des modèles de départ dans un super modèle, on est dans le cas d'un couplage fort. L'inconvénient principal d'une telle approche est que l'on passe du temps à modifier les modèles de départ, dont il faut connaître parfaitement l'architecture interne afin de ne pas introduire des incohérences. Ce choix n'est pas

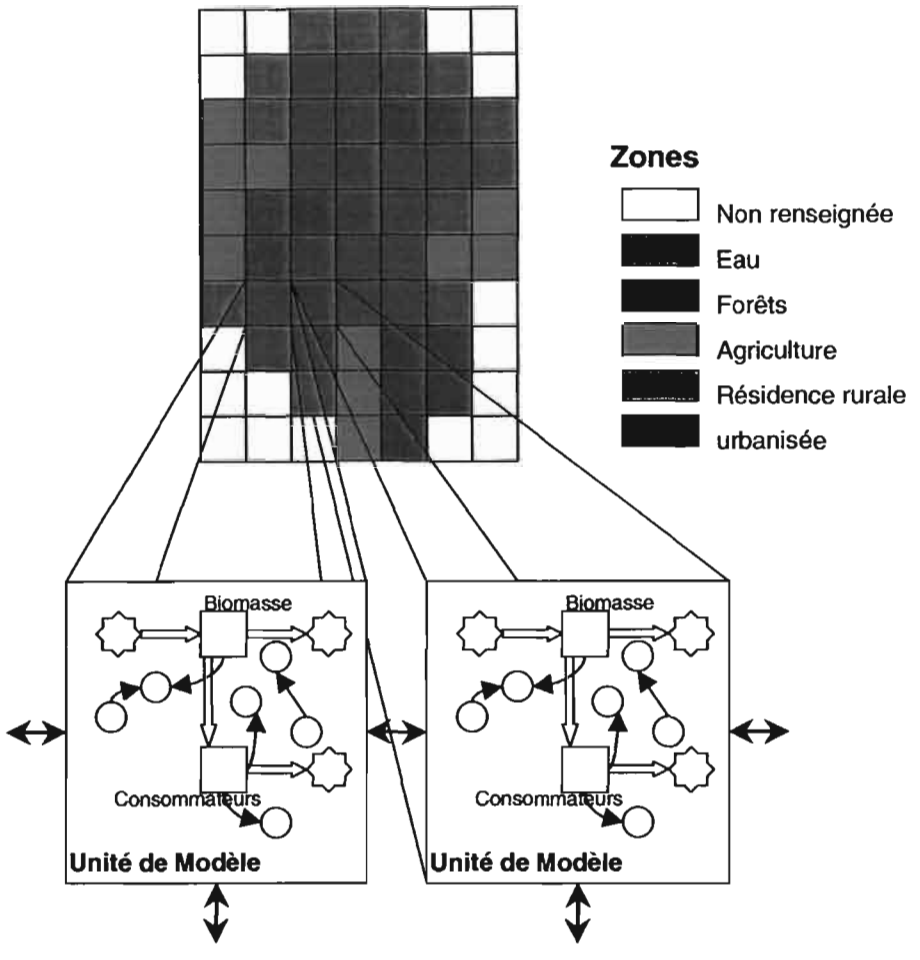


FIG. 2.3 – *Organisation spatiale dans le modèle PLM. L'unité de modèle* Unit Model *[Voinov et al., 1999] est appliquée à chaque cellule de la zone étudiée. Les différences entre cellules se traduisent par différents paramètres appliqués par l'unité de modèle considérée. Les flux hydrologiques connectent les différents unités de modèles.*

toujours possible, lorsque par exemple, l'on se trouve confronté à des modèles développés sur des plateformes différentes.

On peut choisir de ne modifier aucun des modèles de départ mais de construire une interface qui contiendra toutes les méthodes nécessaires au passage des informations entre les modèles à coupler. Dans ce cas, on parle de couplage faible. L'inconvénient majeur est que le module d'interface doit être capable de gérer tous les cas d'interaction. Si les modèles à mettre en relation sont très différents, l'écriture d'un tel module peut demander une somme de travail non négligeable. Il arrive cependant très souvent que l'on soit obligé de choisir une telle option ; il serait alors interessant de disposer d'outils facilitant cette approche.

2.4.2 Faciliter la manipulation des modèles à intégrer

La démarche de couplage retenue dans les deux premiers exemples est la démarche classique adoptée en environnement. Cependant, comme le souligne Maxwell (Maxwell, 1998), elle implique que le modélisateur qui veut coupler des modèles possède la capacité de comprendre et de manipuler des concepts souvent complexes n'appartenant pas à son propre domaine de compétence. Maxwell note également que la structure des modèles complexes caractérisés par des interactions entre multiples composants est très souvent mal explicitée, ce qui limite la compréhension de ces modèles par les utilisateurs et leur capacité à être réutilisés dans des opérations d'intégration. La communauté scientifique manque encore d'outils et de techniques facilitant le couplage de modèles.

Conclusion

Les entreprises de couplage de modèles constituent autant d'opérations d'intégration de savoirs (intégration de modèles mathématiques, informatiques, etc.) dans le but de mieux étudier le système complexe auquel elles se rapportent. Le couplage de modèles se répand dans le domaine environnemental (mais les démarches adoptées restent spécifiques à chaque opération de couplage considérée). Il devient opportun de s'intéresser aux possibles concepts et outils facilitant ce type de démarche. Tel a été l'objet de ce travail.

Chapitre 3

La salinité dans la région Ngalenka

RÉSUMÉ :

Ce chapitre présente un contexte dans lequel l'application d'une démarche de couplage de modèles se révèle intéressante. Il s'agit de l'étude de la dynamique saline d'une région irriguée, plus concrètement, celle du Ngalenka au Sénégal. Les caractéristiques de la région sont présentées, ainsi que celles du phénomène considéré. Le chapitre se termine sur une présentation des différentes connaissances accumulées sur la région du Ngalenka, qui montre que celles-ci sont de nature très diversifiée.

Ce travail a eu pour origine, le souci de scientifiques spécialistes de l'irrigation, d'évaluer l'impact salin de certains aménagements hydro-agricoles et de la mise en place de périmètres irrigués dans une zone du Nord Sénégal. Des connaissances diversifiées ont été accumulées, tant sur les caractéristiques factuelles de cette zone, que sur la manière de modéliser les différents mécanismes en jeu. Le présent chapitre se propose de dresser un tableau de cet arrière plan. Il s'agit de se faire une idée claire de la complexité du système, des données et modèles à mobiliser.

3.1 Caractéristiques de la région d'étude

La région qui sert de contexte à ce travail, les rives du Ngalenka, fait partie d'un territoire irrigué situé au nord du Sénégal, dans la vallée du fleuve (figure 3.1). L'irrigation constitue pour cette région sahélienne, la seule option possible pour assurer l'alimentation et les revenus de ses habitants. Le climat sahélien de cette région et les périodes de sécheresse de ces dernières décennies rendent de plus en plus aléatoire la réalisation de récoltes suffisantes pour une population au taux de croissance démographique soutenu (3,7%[1]).

La zone du Ngalenka amont se situe dans la région communément appelée Fouta-Toro en moyenne vallée du fleuve Sénégal. Elle occupe la partie méridionale de la cuvette de Nianga qui se trouve au sud de la ville de Podor ($16°39'N - 14°58'O$). Au plan administratif, cette zone appartient au département de Podor et fait partie de la région de Saint-Louis (figure 3.2).

La cuvette de Nianga a une forme elliptique (25km est-ouest sur 10km nord-sud) et est ceinte par une digue depuis 1974. Sa partie nord-ouest est occupée par le périmètre rizicole de Nianga (1200 ha) et le sud est bordé par la route du Diéri. La zone Ngalenka amont au nord de cette route, s'étend de Ndiayène à Taredji (figure 3.3).

Avant l'endiguement de la cuvette de Nianga, en 1974, le secteur Ngalenka amont était alimenté en eau par la rivière du même nom, le Ngalenka défluent du fleuve Sénégal, qui traversait le secteur d'est en ouest. L'endiguement de Nianga avec la création du périmètre irrigué de Nianga a coupé la rivière de sa source d'alimentation, le Ngalenka est devenu un bras mort, ce qui a empêché la venue des crues qui permettaient l'exploitation agricole (cultures de décrue) des terres bordant les rives du Ngalenka.

Pour compenser la perte des crues permettant cette exploitation agricole, la SAED[2] a décidé de la remise en eau du marigot Ngalenka entre Diambo et Ndiayène en 1995. Le projet d'aménagement des rives du Ngalenka envisage la création sur 1500ha au total d'une dizaine

1. selon l'annuaire des statistiques du Sénégal, 1993
2. Société Nationale d'Aménagement et d'Exploitation des terres du Delta du Fleuve Sénégal et des vallées du fleuve Sénégal et de la Falémé

de Périmètres Irrigués Villageois (PIV), le Ngalenka servant de canal dans lequel les PIV prendront l'eau.

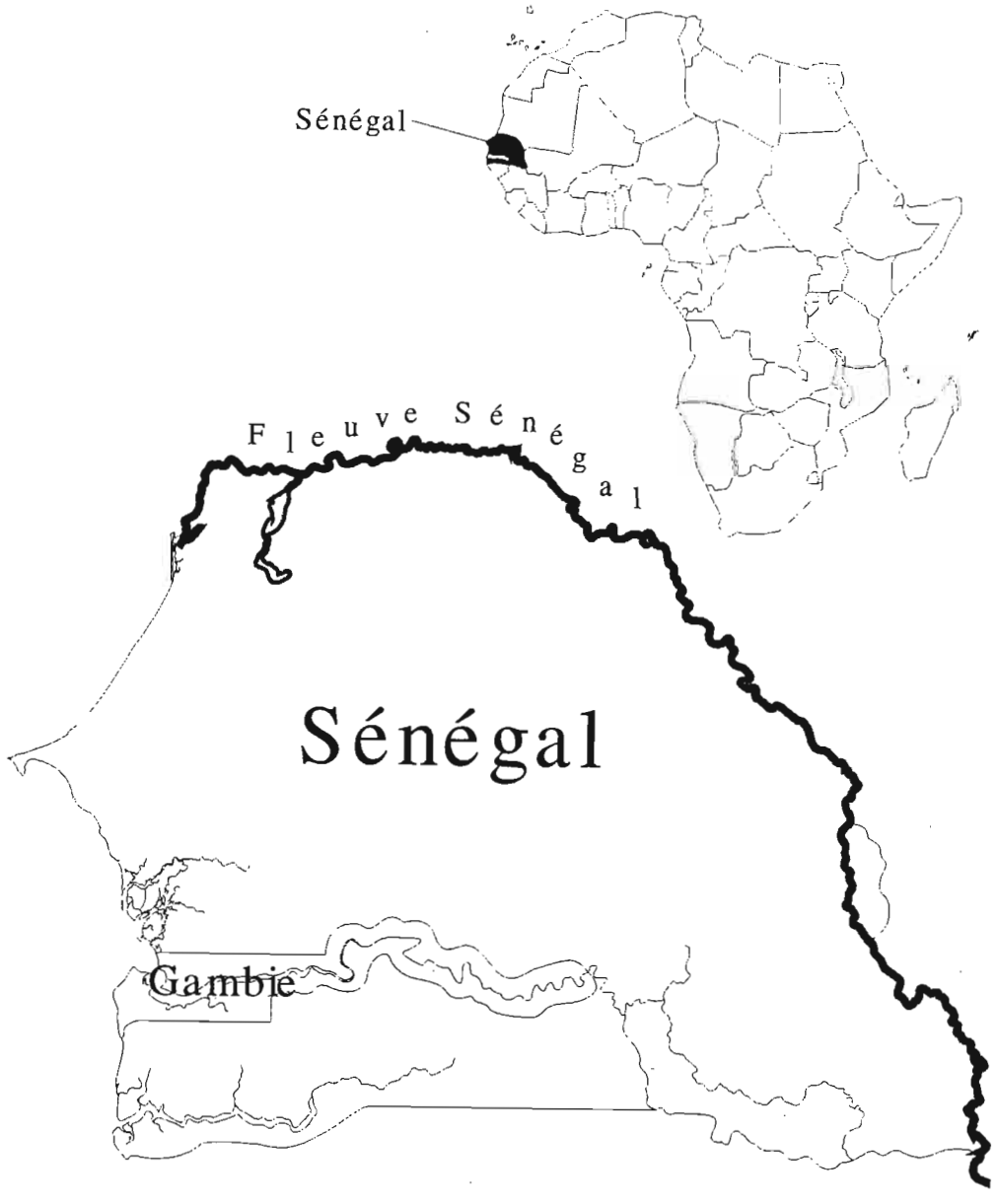


FIG. 3.1 – *Le fleuve Sénégal*

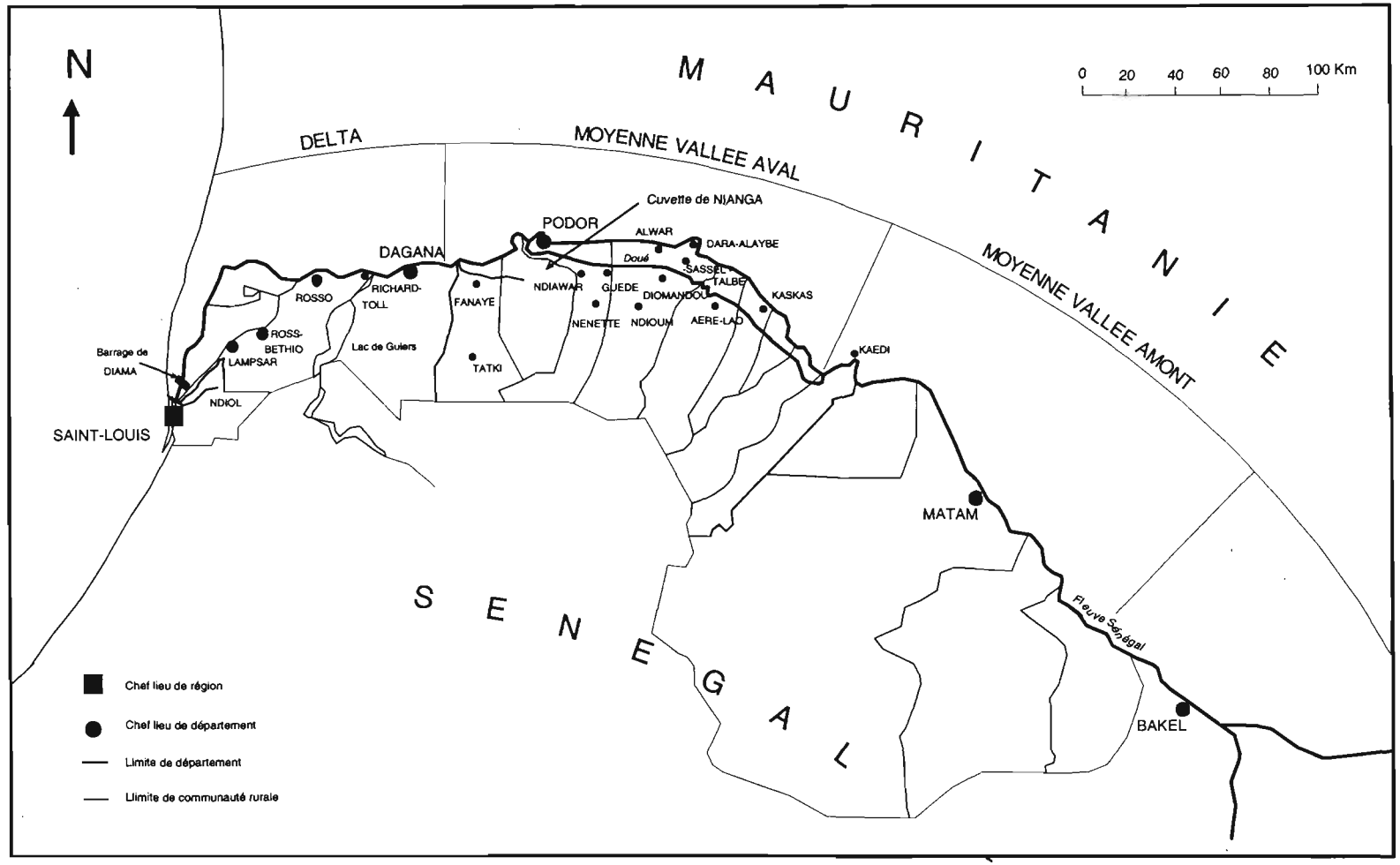


FIG. 3.2 – *La région du fleuve*

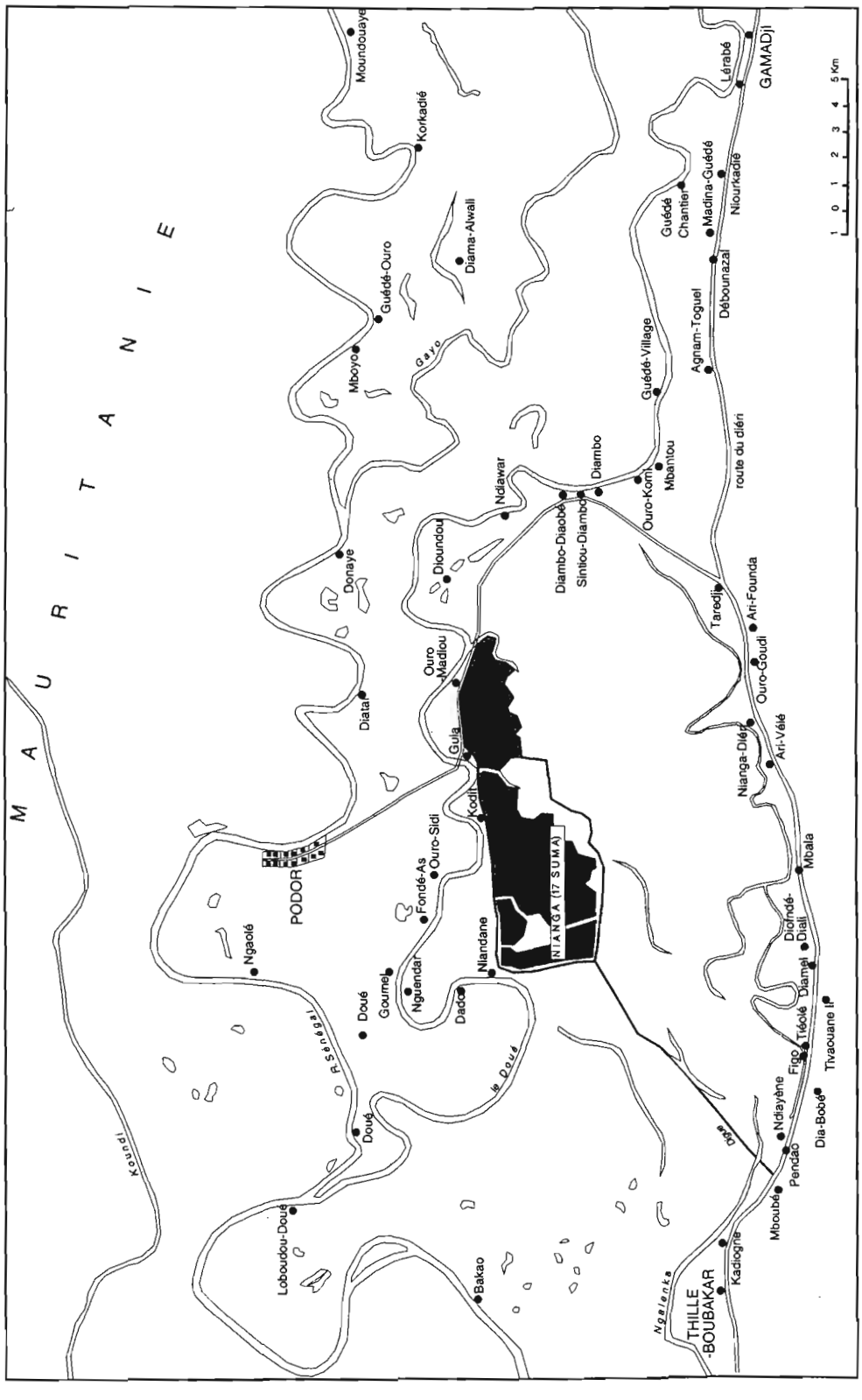


FIG. 3.3 – *La zone de Nianga et du Ngalenka*

3.2 Irrigation et risques de salinisation...

... au niveau mondial L'irrigation permet de minimiser la dépendance des cultures vis à vis des précipitations, cependant en zone aride et semi-aride, elle présente des risques pour l'environnement. Il s'agit principalement de la dégradation saline des sols et des eaux. A l'échelle mondiale, on estime que sur environ 50% des terres irriguées, l'activité agricole connaît des limites, pour cause de salinisation. Aux Etats-Unis, la chute des rendements du fait de la salinisation des sols se manifeste dans environ 30% de l'ensemble des terres irriguées. En Israël, sur le plateau de Paran, il a fallu trouver des variétés de tomates adaptées à des salures élevées (plus de 10 ms/cm).

... au Sénégal Différentes études des précédents sites irrigués (le périmètre irrigué de Nianga) ont montré que dans la vallée du fleuve Sénégal, après moins de trente ans d'irrigation, la salinisation touche 20% des terres exploitées [Zante, 1995]. L'irrigation est en effet réalisée dans une région du Sahel où la longue saison sèche (allant de 10 à 12 mois) s'accompagne d'un processus d'évaporation important (environ 7mm/jour) dont l'effet est accentué par un déficit pluviométrique important. Ce processus d'évaporation favorise la concentration de sels solubles dans les retenues d'eau de surface peu profondes (canaux d'irrigation).

L'histoire géologique de la région contribue également au risque de salinisation. La vallée du fleuve a été envahie aux environs des 5.500 BP[3] par un golfe marin comblé aujourd'hui par des alluvions mais qui a laissé des sédiments de nature chlorurée sodique et magnésienne, [Senghor, 1997b].

La mise en eau des terres par irrigation favorise la mobilisation et la redistribution de ces sels chlorurés sodiques jusque là piégés dans les couches de sol à quelques mètres de profondeur [Boivin, 1995].

3.3 Caractéristiques du phénomène de salinisation

La dégradation saline des sols et des eaux, désignée ici par l'expression générique de salinisation, ou encore évolution de la salinité, est la résultante d'un enchevêtrement de divers processus. En premier lieu, on trouve les mécanismes physico-chimiques contrôlant la concentration et la précipitation des sels dans l'eau, les échanges entre le sol et l'eau, l'évolution des propriétés des sols. Ces mécanismes de base interagissent avec les mécanismes hydriques du transport de l'eau et des solutés à l'intérieur du sol. Interviennent également, les processus contrôlant l'évolution des "conditions aux limites" des mécanismes précédents : processus hydrologiques contrôlant l'évolution du niveau des nappes et des cours d'eau, processus climatiques contrôlant les précipitations et l'évaporation, processus décisionnels contrôlant les

3. Before Present

choix culturaux et la gestion de l'eau dans les parcelles.

Pour décrire complètement les phénomènes de salinisation, il faut donc d'abord rendre compte des processus hydriques de transport de l'eau dans le sol, des processus chimiques de concentration des sels dans le sol (annexe A), des processus géochimiques d'échange entre le sol et sa solution, et enfin des processus mécaniques de dégradation physique des sols. Il faut prendre en compte ensuite les processus agissant à une autre échelle.

3.3.1 Les mécanismes du transport de l'eau et des solutés dans le sol

3.3.1.1 Des modèles microscopiques

Les modèles microscopiques décrivent les lois physiques d'évolution d'un fluide dans un milieu poreux, ici l'eau dans le sol.

La loi du transport de l'eau dans le sol non saturé est décrite par l'équation de Richards. C'est le résultat de la combinaison de la loi de Darcy (qui établit un lien entre le gradient de pression (capillaire, osmotique, gravitaire, etc.) et les flux d'eau via une conductivité hydraulique caractéristique du milieu) et celle de la conservation de la masse de solution. Moyennant certaines hypothèses simplificatrices telles que: le milieux poreux est inerte et indéformable, les phénomènes d'hystérésis[4] existant entre les processus d'humectation et de drainage ne sont pas pris en compte, les effets thermiques sont considérés comme négligeables et l'air présent dans le milieux poreux est à la pression atmosphérique, l'écoulement peut être décrit par l'équation de Richards:

$$\frac{\partial\theta}{\partial t} = \frac{\partial}{\partial z}[K(\theta)(\frac{\partial\psi}{\partial z} + 1)]$$

où θ représente la teneur en eau du sol, $K(\theta)$ la conductivité hydraulique, ψ la pression de l'eau, z est la profondeur du point de sol considéré. L'évaporation figure dans les conditions aux limites.

La loi du transport de soluté dans le sol est également basée sur la résolution d'une équation aux dérivées partielles. Le soluté est caractérisé par sa concentration dans la solution qui va être modifiée par différents processus: convection, dispersion, diffusion, adsorption, décomposition. Différentes équations existent au niveau microscopique, selon ceux des processus qui sont retenus. L'équation de convection-dispersion souvent utilisée s'écrit:

$$\frac{\partial(\theta c)}{\partial t} = \frac{\partial}{\partial z}(\theta.D(\theta,v)\frac{\partial c}{\partial z} - v\theta c)$$

où c est la concentration en soluté, v la vitesse de l'eau au point considéré, $D(\theta,v)$ un coefficient de dispersion hydrodynamique.

Ces différentes équations sont résolues par différents schémas de résolution numériques. Il

4. Phénomène consistant dans l'influence sur l'évolution d'un système des états précédents de ce système. Dans un tel phénomène, les valeurs obtenues pour une grandeur dont on étudie l'évolution en fonction d'un paramètre ne sont pas les mêmes selon le sens de variation du paramètre. Exemple: différentes relations pression/teneur en eau en phase de saturation ou de désaturation.

existe différents modèles numériques réalisés à partir de ces différentes équations par différentes équipes de recherche. Le modèle Leachm réalisé par l'université de Cornwell est l'un des plus connus.

3.3.1.2 Des modèles macroscopiques

Pour les modèles macroscopiques, l'idée est de s'affranchir de l'écriture (et de l'intégration dans l'espace) des termes des lois physiques en considérant une zone de sol macroscopique sur laquelle on effectue des bilans des variables eau et solutés. Ces modèles macroscopiques conduisent à la discrétisation spatiale du profil en différentes couches horizontales. Pour un pas de temps donné, ces bilans se font classiquement par une différence entre les termes sources et les termes puits, ce qui modifie la valeur des variables à l'instant précédent. On obtient ainsi un ensemble de réservoirs reliés par des flux d'eau et de solutés.

Les modèles macroscopiques sont plus simples que les modèles microscopiques et sont intéressants pour la modélisation de zones homogènes suivant un critère donné (drainage latéral). Cependant, le flux entre les différents réservoirs n'etant plus calculé de manière microscopique à l'interface, il ne peut être qu'estimé à partir de variables macroscopiques (diffusion entre couches, terme de percolation). Ces lois empiriques sont souvent des lois paramétrées ce qui conduit à une perte du caractère générique de la modélisation.

Selon l'échantillonage temporel et la discrétisation spatiale considérée, on distingue les modèles à 1 couche (zone sol avec des termes), à 2 couches (zone racinaire et réservoir au dessous); les modèles continus ou discrets.

3.3.2 Processus d'échelle plus large influençant la salinisation

Les modèles microscopiques et macroscopiques décrivent les processus de salinisation dans un profil de sol. Cependant les concentrations de soluté sont également modifiées par l'écoulement de l'eau en surface, dans la zone saturée (nappe) et dans les zones latérales du profil.

En fonction de la nature du sol, différents niveaux de pression peuvent exister entre le profil et son environnement latéral ce qui entraîne des écoulements latéraux et modifie la teneur en soluté du profil considéré.

Les conditions aux limites représentent les hypothèses que l'on pose aux frontières du profil afin de calculer l'écoulement vertical dans le profil. En surface, ces conditions vont changer en fonction de la pluviométrie, des processus d'évaporation et de transpiration des plantes, des phénomènes de crues et de décrue du fleuve. Sous le profil, l'existence éventuelle d'une nappe peut également influencer l'écoulement de l'eau du profil (situation de remontée de nappe par exemple).
La modélisation des processus physiques de salinisation ne peut suffire à totalement décrire le processus de salinisation. La mise en eau des terres dépend du choix cultural réalisé par

le paysan. Les processus de salinisation peuvent être minimisés par une bonne utilisation du draînage.

L'action des agriculteurs a une double importance: d'une part en choisissant leurs cultures et la quantité d'eau qu'ils apportent à chacune d'elles (à hauteur d'une certaine proportion de la demande évaporation maximale), d'autre part parce qu'ils peuvent gérer efficacement ou non leur terre.

Si l'on veut tester l'impact de différents scénarios, il est donc nécessaire de modéliser ces prises de décision qui sont du domaine de la micro-économie.

3.4 Connaissances accumulées sur la région Ngalenka

La vallée du fleuve Sénégal est considérée comme une zone idéale pour la culture irriguée. On y trouve de l'eau en abondance grâce à la présence du fleuve Sénégal, une terre à fort potentiel et une bonne luminosité. Les rendements potentiels de la riziculture dans cette zone dépassent de deux fois ceux des pays asiatiques [Boivin, 1997; Boivin et Poussin, 1997]. Porteuse de nombreuses attentes, elle a été l'objet de nombreuses recherches en pédologie, hydrologie et en agronomie. Dans la section 3.3, les principes des processus dont dépend la salinisation ont été évoqués; cette section 3.4 recense les connaissances accumulées par les travaux conduits sur la région de la moyenne vallée du fleuve Sénégal qui ont rapport avec ces processus et leur déroulement.

3.4.1 Les données cartographiques

Les données cartographiques qui existent concernant la zones du Ngalenka décrivent les propriétés physiques du milieu (pédologie, topographie, hydrographie). Elles sont cependant parfois incomplètes, imprécises ou très anciennes [Patris, 1997].

Il existe des fonds topographiques IGN de la région du fleuve Sénégal (carte Podor et Dagana) permettant de donner les courbes de niveau de la région avec des équidistances de 1 mètre. Ces cartes au 1/50000ème ont été mises à jour pour la dernière fois en 1959, et elles ne couvrent pas la zone de *diéri*[5]. Les aménagements du fleuve ayant eu lieu après l'établissement de ces cartes (construction d'une digue, création du périmètre de Nianga en 1974, aménagement de certains marigots) les tracés de cours d'eau figurant sur ces cartes ne sont plus d'actualité.

Ces fonds topographiques ont servi de base à l'établissement des cartes pédologiques et géomorphologiques établies par la FAO. Par ailleurs une carte pédologique au 1/15000ème de la partie nord-est de la cuvette de Nianga a été établie par l'ORSTOM en 1979 [Patris, 1997].

Il existe également une carte de salinité dressée par conductivité électromagnétique, établie

5. haut pays bordant la vallée

par le laboratoire de pédologie du centre ORSTOM de Dakar-Hann en 1996 [Patris, 1997].

Il n'existe pas de carte hydro-géologique, ni de carte piézométrique de la zone, et il n'y a donc pas d'information précise sur la localisation géographique des nappes.

3.4.2 Les études hydrologiques, agro-climatiques et agronomiques

De nombreuses études ont été menées sur la région du Ngalenka [Jamin, 1995]. Elles suivent deux axes principaux : d'une part les études portent sur la disponibilité de l'eau, la qualité de sols, la rentabilité des différentes cultures et la qualité des sols, d'autre part, les agronomes se sont associés à des économistes pour décrire les systèmes de culture des paysans.

Les études hydrologiques commencées au début du 20ème siècle ont fait l'objet d'une synthèse par l'ORSTOM [Rochette, 1974; Seguis, 1995] qui permet de préciser les disponibilités en eau aux différentes périodes de l'année et les risques d'occurrence de crues importantes ou d'étiages[6] précoces et sévères.

Les recherches en agro-climatologie sont initiées en 1957 sur le sorgho de décrue, puis développées sur le riz et la canne à sucre. Un programme financé par différents bailleurs de fonds (FAO/PNUD/OMVS) permet en 1976 d'établir des tables de besoin en eau des plantes (riz, blé, maïs, sorgho, coton) et d'étudier la fréquences des pluies et des températures.

Le calage des cycles a fait l'objet de nombreux travaux. Il s'agit d'étudier l'adaptation des variétés aux différentes saisons de culture afin de pouvoir réaliser la double culture (culture d'hivernage et culture de saison sèche froide).

Jamin note cependant que la conduite des recherches agronomiques dans la vallée est marquée par la multiciplité des intervenants et par la discontinuité des travaux, menés pour l'essentiel sur des financements extérieurs. Les ressources financières sont souvent liées à des projets de développements ponctuels gérés par la SAED[2]. Cette instabilité ne favorise pas l'accumulation de connaissances car il manque un projet d'ensemble. Une des conclusions de l'article [Jamin, 1995] est que l'éparpillement de ces recherches s'est accompagné de dérives géographiques, qui n'ont pas favorisé les collaborations pluridisciplinaires : si les recherches techniques se sont surtout focalisées sur l'aval de la vallée, où la culture irriguée est plus développée, les recherches socio-économiques se sont plus intéressées aux sociétés traditionnelles de l'amont.

3.4.3 Des modèles[7] *ad hoc*

La démarche classique utilisée dans les études des processus physiques est basée sur une exploration du terrain, la prise de mesures *in situ* qui sont ensuite analysées pour permettre une description du phénomène généralement dans une monographie.

6. niveau moyen le plus bas d'un cours d'eau

7. les modèles auxquels on s'intéresse ici sont les modèles disponibles sous forme de codes informatiques opérationnels, objectivant les connaissances sur les mécanismes décrits plus haut.

Cette méthode, par son caractère descriptif, présente des désavantages par rapport à un objectif d'exploitation ultérieure de ses résultats. En effet, les résultats obtenus sont difficilement extrapolables hors des sites sur lesquels ils ont été trouvés, ce qui oblige à multiplier les études. Le coût d'une telle approche peut alors s'avérer être très élevé.

L'utilisation des modèles offre une voie différente, mieux adaptée à la réutilisation des études effectuées. Une fois calibrés et validés, les modèles permettent de dépasser les limites de l'approche expérimentale, notamment dans le cadre de la prévision de phénomènes survenant dans des conditions différentes de celles pour lesquelles l'expérience était effectuée. Depuis une dizaine d'années, la modélisation est de plus en plus employée par les pédologues, hydrologues et agronomes qui travaillent sur la région du Ngalenka. A son travail traditionnel (prise de mesures *in situ* puis analyse et compréhension des mécanismes en jeu), le thématicien va ajouter la construction d'un modèle résumant sous forme mathématique ou informatique ce qu'il a compris du mécanisme qu'il étudie. Cela va lui permettre de réutiliser cette structure dans des situations variées.

Les hydrologues qui étudient les propriétés mécaniques, physiques et chimiques des eaux marines ou continentales, se sont penchés très tôt sur l'utilisation des modèles. L'équation de Richards qui permet de décrire le mouvement de l'eau en zone non saturée - équation aux dérivées partielles fortement non linéaire - a conduit à la conception de nombreux modèles numériques en fonction du milieu sur lequel ils vont être appliqués. Un exemple de modèle est MHNS_2D, un modèle numérique de transfert d'eau en milieux poreux non saturés [Diaw, 1996]. Dans sa thèse de doctorat, l'auteur propose un modèle numérique de l'équation de Richards basée sur la méthode des éléments finis mixtes hybrides pour simuler la migration des eaux en milieux poreux saturés et non saturés. Cette méthode est plus adaptée à la description de milieux hétérogènes (en terme de conductivité hydraulique) que la modélisation numérique basée sur la théorie des éléments finis classiques. Pour valider son modèle plus adapté à des milieux hétérogènes, ainsi que le montre une comparaison avec un modèle classique (SWMS_2D) en condition de laboratoire, l'auteur l'applique au périmètre de Donaye (conditions de terrain), sur lequel est pratiquée une riziculture inondée en période d'hivernage. Les conclusions de l'étude montrent cependant des écarts importants entre le modèle et le terrain. L'auteur explique ces écarts par la non prise en compte d'un certain nombre de facteurs : influence des échanges entre la nappe et le fleuve, influences des inondations survenues lors de la campagne où les observations ont été effectuées, dynamique hydrique complexe des vertisols, etc. Ainsi, un travail de modélisation respectant toutes les exigences de sa discipline (physique dans ce cas), ne reflète pas la réalité du terrain parce que ne prenant pas en compte des phénomènes modélisés par ailleurs (mouvement de l'eau dans la nappe par exemple).

Les agronomes ont également entrepris de mettre leurs connaissances sous forme de modèles opérationnels. On peut citer comme exemple le modèle RIDEV [Dingkuhn et al., 1995] conçu par des agronomes pour permettre aux agriculteurs d'apprécier les risques liés au choix de

leurs cultures en fonction du moment de l'année où ils décident de réaliser le semis, afin de permettre notamment un système de double culture dans l'année. Sur deux sites tests (Ndiaye dans le delta et Fanaye dans la moyenne vallée), différentes variétés de riz ont été semées à différentes périodes de l'année. Des mesures mensuelles de la biomasse aérienne ont été pratiquées ; une station météorologique automatique, installée dans chaque site a permis de collecter toutes les heures les températures de l'eau, de l'air et du sol, ainsi que la radiation solaire. Tous les jours des observations micro-météorologiques ont été réalisées par la SAED[2]. Les dates de semis, repiquage, épiaison, floraison et maturité physiologique, ainsi que les rendements en grain et en paille ont été consignés. Au bout de deux ans (1990-1992) de collecte de données, un certain nombre de paramètres clés ont été déterminés (température de l'air, de l'eau, date et mode de semis, etc) qui ont permis l'élaboration d'un modèle rendant compte de l'ensemble des processus physiologiques et micro-climatiques expliquant la stérilité des épillets de riz. La première version du modèle résultat permettait à un utilisateur de constater les relations entre développement de la plante et température de l'air et de l'eau, entre stérilité et température à la montaison, de différencier les phénomènes de croissance et de développement. Cependant, ce modèle ne pouvait répondre à lui seul à des problèmes de gestion de calendrier de travail, problèmes qui se posent lorsque l'on est dans un cadre de double culture. Les auteurs ont donc envisagé le couplage de RIDEV avec un modèle conçu par eux-mêmes, simulant la dynamique des parcelles avant récolte, puis ils ont ensuite procédé au couplage de leur modèle avec OTELO, un logiciel de l'INRA [Attonaty et al., 1990] permettant de traiter les problèmes d'organisation du travail dans l'agriculture. OTELO permet de reproduire le mode d'organisation des agriculteurs pour la récolte et la mise en place de la culture suivante, puis d'évaluer l'intérêt de modifications diverses, face à divers scénarios climatiques. Il est interessant de noter que dans la liste des auteurs du simulateur OTELO figure un des auteurs du modèle RIDEV, le couplage ayant consisté à adapter chacun des codes en présence afin de leur permettre de fonctionner ensemble.

Conclusion

Ce chapitre a permis d'illustrer sur un exemple (la zone Ngalenka), la notion de complexité des systèmes naturels. Cette complexité explique le nombre et la diversité des études (ainsi que leur caractère parfois dispersé) qui leur sont consacrées.

La présentation, dans un contexte précis, des mécanismes de la dynamique saline a permis de montrer que l'évaluation du risque salin nécéssitait le prise en compte, l'intégration, de nombreux modèles et ensembles de données. Il est intéressant d'étudier quelle démarche le modélisateur adopte lorsqu'il se trouve confronté à ce type de question. C'est l'objet du chapitre suivant.

Chapitre 4

Coupler des modèles de processus pour étudier la salinité du Ngalenka

RÉSUMÉ :

Dans ce chapitre nous décrivons la problématique de couplage rencontrée lorsque l'on veut modéliser un phénomène tel que la dynamique saline du Ngalenka. La démarche adoptée pour étudier un problème similaire (étude de la dynamique saline de la région du Chishtian au Pakistan [Belouze, 1996]) est analysée dans un second temps. Cette démarche classique possède des avantages et des inconvénients qui sont discutés. Nous exposons ensuite les caractéristiques souhaitables que doit présenter une démarche de couplage.

La salinisation est un phénomène mettant en jeu de nombreux processus qui agissent sur la circulation de l'eau et des sols dissous dans un environnement donné (figure 4.1) ; la salinisation agit sur différents composants de l'environnement et elle évolue différemment suivant différents facteurs écologiques et économiques. L'étude de ce phénomène nécessite donc de prendre en compte la dynamique résultant d'un ensemble de processus en interaction, ce qui revient à traiter une question de couplage. Dans l'exemple que nous présentons dans la première partie

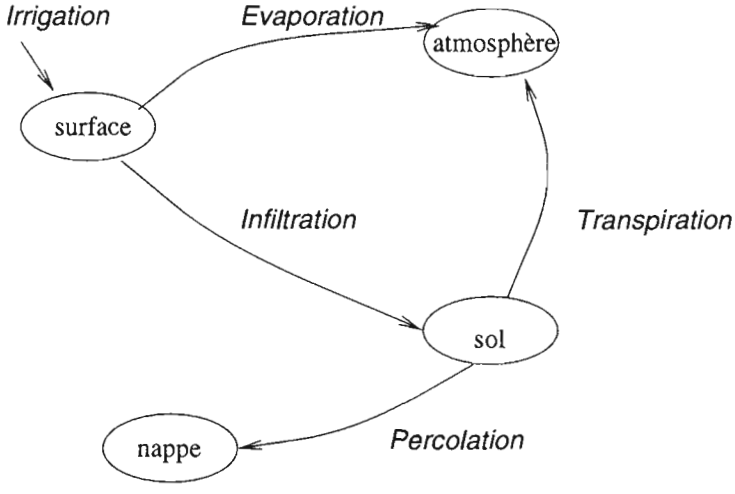


FIG. 4.1 – *Processus de circulation de l'eau dans une zone irriguée*

de ce chapitre, l'étude de la salinisation est traitée par l'application d'une démarche de couplage fort que nous détaillons. Cette approche nécessite un certain nombre de prérequis : une bonne connaissance de différents concepts appartenant à différentes disciplines, afin de pouvoir le cas échéant procéder à des simplifications de paramètres ; des mises en relations de concepts existant sous différentes formes dans des domaines différents. Il faut, de plus, disposer d'un ensemble de connaissances suffisantes pour pouvoir procéder à leur intégration.

Dans le cas de la zone Ngalenka, la disponibilité des connaissances requises n'était pas toujours assurée au départ de ce travail de thèse. De plus, situant ce travail dans le cadre d'une recherche informatique, il nous a semblé approprié et intéressant d'orienter la réflexion vers la détermination de concepts et d'architectures informatiques permettant à un thématicien spécialiste du problème de la salinité de faciliter son travail d'intégration. C'est ainsi que la direction de la thèse s'est portée vers la proposition d'une démarche de couplage de modèles de processus. Nous présentons le cadre de notre proposition dans la deuxième partie de ce chapitre.

4.1 Intégrer des processus pour étudier une question complexe

Une démarche naturelle adoptée pour l'étude d'une question complexe comme l'étude de la salinité consiste pour un thématicien (hydrologue, agronome ou économiste) à prendre en compte l'ensemble des processus impliqués dans le processus de salinisation en effectuant une synthèse, qui peut éventuellement donner lieu à la construction d'un modèle intégré.

Un exemple intéressant à présenter à cause de sa problématique comparable à celui de cette thèse est le travail réalisé dans le cadre d'une collaboration entre deux organismes de recherche l'IIMI[1] et le Cemagref[2]; cette collaboration ayant pour objectif d'effectuer des recherches contribuant à améliorer la performance de l'agriculture irriguée au Pakistan.

Pour permettre l'évaluation de la durabilité et des rendements d'une région irriguée de la zone irriguée de Chishtian dans le sud du Penjab (Pakistan), Belouze [Belouze, 1996] a construit un modèle intégré de système irrigué pour prendre en compte les différents phénomènes qui entrent en jeu dans l'évolution d'un tel système.
Ce travail faisait partie d'un ensemble de travaux tendant à comprendre les processus impliqués dans le fonctionnement du système irrigué à différents niveaux (le réseau hydraulique, le système agraire, les sols et leur évolution). Des études disciplinaires préalables avaient ainsi permis la construction :

- d'un logiciel de simulation hydraulique permettant de modéliser la distribution d'eau de canal et d'étudier l'influence des modes de gestion tactiques [Litrico, 1995; Visser, 1996];
- d'un modèle micro-économique simulant l'anticipation des agriculteurs concernant le choix des cultures, des rendements espérés et de l'utilisation de l'eau souterraine [Rinaudo, 1994];
- d'un modèle de distribution de l'eau dans la zone non saturée des sols permettant de prédire la concentration en sel sous diverses pratiques d'irrigation [Smets, 1996].

Le but de l'étude suivante était alors d'établir des liens entre les diverses disciplines en s'appuyant sur les modèles existants et en proposant une maquette dans laquelle le système irrigué était vu de manière interdisciplinaire.

4.1.1 Caractéristiques de la démarche

Pour atteindre cet objectif, l'auteur est allé à la recherche d'explications sur les différents concepts émanant des différentes disciplines impliquées dans son entreprise d'intégration de connaissances. Après une importante recherche bibliographique, il a suivi une démarche méthodologique qu'il décompose en trois principales étapes (figure 4.2) :

1. Institut International pour le Management de l'Irrigation
2. Institut de recherche français pour l'ingénierie de l'agriculture et de l'environnement

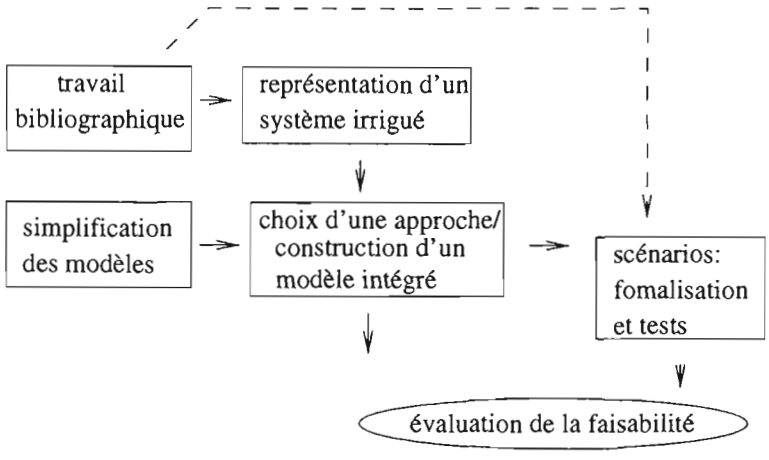


FIG. 4.2 – *déroulement méthodologique d'une intégration de processus. [Belouze, 1996]*

4.1.1.1 La simplification des modèles

Les modèles existants sur lesquels l'intégration devait être basée, n'ont pas toujours été réalisés à l'échelle du Chishtian, la région irriguée étudiée. Certains modèles présentaient des niveaux de détail très fins (ce qui pouvait s'avérer trop exigeant en terme de temps de calcul, d'occupation de la mémoire). Cela semblait inadapté a priori à l'objectif d'opérationalité. Ces raisons expliquent la simplification des modèles de départ.

Le logiciel de simulation hydraulique de transport de l'eau dans les canaux primaires [Litrico, 1995] et secondaires [Visser, 1996] pour la subdivision de Chishtian s'appuie sur le logiciel SIC[3], développé par le Cemagref. Ce logiciel de simulation permet de simuler le transport de l'eau dans un ou plusieurs canaux d'irrigation reliés. Le modèle de base est obtenu par l'écriture des lois de la dynamique des fluides appliquées à un élément de fluide. Il est décrit par deux équations de continuité et de dynamique proposées par Saint-Venant(1871). Ces équations non linéaires aux dérivées partielles sont résolues numériquement dans le logiciel SIC par la méthode de Preissman(1961) qui consiste à discrétiser en temps et en espace les équations de Saint-Venant et à approximer les dérivées partielles par différences finies. S'appuyant sur des travaux antérieurs (Malaterre, 1994), Belouze a choisi de remplacer le modèle hydraulique non linéaire par un modèle linéaire de canal. L'une des raisons citées est la volonté d'éviter le re-développement du logiciel SIC sous l'environnement informatique choisi a priori (Matlab).

Le modèle micro-économique qui permet de déterminer les décisions d'anticipation des agriculteurs [Rinaudo, 1994] a été réalisé sur la base d'une étude approfondie des habitudes des agriculteurs de huit watercourses[4]. Cela a permis de déterminer onze types d'agriculteurs ho-

3. Simulation de Canaux d'Irrigation
4. ensemble de périmètres irrigués présentant une homogénéité du point de vue de sa gestion

mogènes en terme de pratiques agricoles et d'orientation vers une pratique de marché ou au contraire vers une auto-consommation. Cette typologie a été reprise sur l'ensemble du Chishtian en version simplifiée : un seul type d'agriculteur a ainsi été associé à un watercourse donné en fonction de la nature des contraintes existant dans le watercourse en question (offre en eau, etc).

Le modèle hydrodynamique du transport d'eau et de sel dans le sol [Smets, 1996] SWAP a finalement été abandonné pour un modèle macroscopique ("modèle à bilan") qui calcule un bilan en eau et en sel à l'échelle de la saison culturale. La principale raison est que SWAP demande un niveau de précision sur les paramètres d'entrée pas toujours existant (la fréquence d'irrigation doit être fournie à SWAP à l'échelle de l'heure ou de la minute, alors que l'information est disponible sous forme mensuelle). Son utilisation à des fins opérationnelles sur un large espace hétérogène est délicate selon Belouze.

4.1.1.2 La construction d'un modèle intégré

Certaines des études disciplinaires ayant abouti à la construction de modèles créés à l'aide du logiciel Matlab, qui par ailleurs se prêtait bien à la construction de structure de programmes interconnectés, la maquette a été réalisée sous environnement Matlab. Les modèles simplifiés ont été, lorsque cela s'est avéré nécessaire, traduits dans cet environnement.

Les modèles de départ ont été réalisés sur des horizons temporels différents :

- d'une part, les objets hydrauliques sont modélisés sur l'ensemble de la saison culturale (avec un pas de temps de 10 minutes pour les modèles linéaires de canaux).
- d'autre part, les objets agriculteurs et sols comportent des processus modélisés en début d'une année concernant les deux saisons à venir.

Entre les objets hydrauliques (canaux primaires et secondaires) et les autres sous systèmes (agriculteurs et sol) différentes fonctions de sommation ont été réalisées pour mettre en cohérence les différentes informations existant à des échelles de temps différentes. Par exemple les données hydrauliques ont été sommées sur l'année pour la partie sol et sur le mois pour la partie agriculteur, le mois étant la durée du passé à partir de laquelle les agriculteurs anticipent l'avenir.

4.1.1.3 Evaluation de la faisabilité

Pour tester la maquette, différents scénarios de changement pour l'amélioration de la performance du système irrigué ont été définis, puis formalisés dans le contexte du logiciel Matlab. Cette maquette avait en effet pour rôle de permettre une discussion sur la faisabilité d'un modèle intégré pour la problématique générale (améliorer les performances de l'agriculture irriguée au Pakistan).

4.1.2 Remarques

La méthodologie suivie dans ce travail présente l'avantage d'avoir permis l'exploitation de toutes les connaissances de départ. Elle n'a pu être employée que parce que l'auteur de l'étude, thématicien lui même (hydrologue) est allé à la rencontre des autres disciplines pour s'approprier les connaissances nécessaires à la simplification des modèles de départ.

Ce travail n'a pris en compte que les modèles mathématiques construits antérieurement dans le même cadre. Cela a permis de réutiliser l'expérience acquise, mais cela s'est fait à travers une simplification de ces modèles et leur réécriture dans l'environnement informatique Matlab. Cette réécriture aboutit à un modèle intégré manipulant des connaissances uniquement quantitatives.

La maquette créée, avait pour principal objectif d'évaluer la faisabilité et l'intérêt d'utiliser un modèle intégré pour évaluer l'impact de scénarios de changement de gestion d'une zone irriguée, à la suite de ce travail une plate-forme plus complète devait être réalisée. Cependant, étant donné les limites de la maquette initiale (prise en compte de modèles mathématiques sous environnement Matlab), elle doit être reconstruite si l'on veut inclure par la suite des connaissances qualitatives concernant la région [Belouze, 1996].

4.2 Une démarche de couplage des modèles de processus pour étudier la salinité

L'analyse de l'exemple précédent montre que le modèle intégré réalisé n'est pas réutilisable sans modification préalable pour peu que l'on veuille intégrer d'autres connaissances non prises en compte au départ. Cela amène à réfléchir sur les concepts utiles permettant de dépasser ces limites dans le cadre de l'étude d'une question complexe, d'un travail pluridisciplinaire, types de problèmes auxquels se confronte de plus en plus souvent la science environnementale.

4.2.1 Objectifs

Le caractère dispersé et peu complet des informations concernant la région du Ngalenka à laquelle on s'intéresse a déjà été noté dans le précédent chapitre. Par ailleurs, les modèles qui permettent de représenter la dynamique des processus physiques dans la zone Ngalenka, n'ayant pas été déterminés par les mêmes équipes avec des objectifs semblables, ils peuvent ne pas exister sous une forme compatible permettant une adaptation en vue d'un fonctionnement commun. Enfin, ces connaissances n'appartenant pas à la même discipline scientifique, il est souhaitable de considérer le système de la façon la plus neutre possible afin de s'affranchir des points de vue particuliers inhérents à chaque discipline.

Ces observations, ainsi que la réflexion sur la plus-value d'un travail de recherche informatique

sur une problématique environnementale ont conduit à préciser l'objectif du travail vers la détermination de concepts et d'une architecture informatique, facilitant l'ouverture et la flexibilité du modèle intégré résultant. Ainsi plutôt que de réaliser un travail d'intégration pour répondre à la question environnementale sur l'évolution de la salinité au Ngalenka, l'objectif s'est déplacé sur la détermination d'outils permettant au modélisateur de réaliser un travail de couplage, dans le contexte de cette question précise.

4.2.2 La proposition de cette thèse

Les objectifs dégagés conduisent à notre proposition de thèse: nous avons choisi d'orienter ce travail de thèse sur les concepts utiles à une entreprise de couplage de modèles dans un domaine environnemental. Il nous semble également important de proposer au modélisateur un "mode d'emploi", une démarche d'intégration qui lui soit la plus naturelle possible. Ces différentes idées (concepts, architecture, démarche d'intégration) ont été mises en œuvre sur un exemple test ayant pour cadre l'étude de la dynamique d'une région irriguée, par le couplage de deux modèles numériques MOC (modèle de nappe) et LEACHM (modèle d'infiltration) présentés au chapitre 7.

Coupler des modèles revient à définir les façons de gérer les différences, *l'hétérogénéité* des modèles en question. Les modèles que l'on cherche à coupler sont caractérisés par le fait qu'ils s'inscrivent dans le temps (ils représentent une dynamique), dans l'espace (ils caractérisent un système réel décrit d'un certain point de vue) et que leur actions sont différentes les unes des autres. Pour proposer des outils facilitant le couplage de tels modèles, la réflexion doit principalement porter sur ces propriétés clés.

4.2.2.1 L'hétérogénéité temporelle

Si l'on considère comme étant totalement indépendants les modèles de départ que l'on choisit de coupler, alors, on ne peut tenir pour acquis qu'ils aient été réalisés selon la même conception de la gestion du temps de la simulation (les pas de temps des modèles peuvent être fixes ou non, on peut devoir inclure des modèles dont la gestion du temps est réalisée par événements discrets, etc).

4.2.2.2 L'hétérogénéité spatiale

Les modèles ayant été construits par des spécialistes de disciplines aux objectifs scientifiques divers, un même objet du monde peut bénéficier de diverses représentations d'un modèle à l'autre, en fonction de la discipline à laquelle il appartient.

4.2.2.3 L'hétérogénéité comportementale

Les modèles en entrée peuvent être de "sophistication" différentes. Certains peuvent être simplifiés, d'autres très détaillés.

Conclusion

L'étude d'une démarche classique d'intégration de processus a permis de déterminer une problématique informatique de couplage de modèles dans un contexte environnemental. Cette question a été explorée suivant trois axes de recherche concernant l'hétérogénéité. La suite de cette thèse en présente les principaux résultats.

Deuxième partie

Etat de l'art : Simulations, Agents et Couplage

Chapitre 5

Modélisation, Simulation et Agents

RÉSUMÉ :

Ce chapitre présente les notions de modélisation, de simulation et d'agent, que nous employons dans cette thèse. La simulation consiste à faire évoluer dans le temps un modèle. La notion d'agent peut être définie soit du point de vue du modélisateur, comme un outil conceptuel lui permettant de modéliser son monde; soit du point de vue de l'informaticien comme un concept technique. Les notions de simulation et d'agent se rejoignent dans le domaine de la simulation multi-agents. La simulation multi-agents permet l'intégration de connaissances provenant de sources différentes.

5.1 Notion de simulation

5.1.1 Modélisation

La modélisation est la démarche consistant à construire à partir d'un système que l'on veut étudier une abstraction de celui-ci (le modèle), qui ne retient du système que les grandeurs caractéristiques (les variables d'état) jugées pertinentes par le modélisateur. En fonction du type de système à étudier, le modélisateur va choisir entre différents types de modèles.

Un système dont l'évolution est fonction du temps va être représenté par un modèle dynamique. Parmi les modèles dynamiques, en fonction de :

- la nature des variables d'état,
- la façon dont le système évolue dans le temps,

on va distinguer différents types de modèles.

5.1.1.1 Etat, espace d'états

L'état du système à un instant t décrit son comportement d'une façon mesurable, c'est à dire les valeurs que prennent chaque variable caractéristique du système.

L'espace d'états d'un modèle est l'ensemble des valeurs qui peuvent être prises par les variables caractérisant le système modélisé. L'espace d'états est continu lorsque les valeurs sont continues (volume d'eau dans un réservoir en cours de remplissage); il est discret lorsque les valeurs sont discrètes (état rempli, vide ou non vide d'un réservoir d'eau).

5.1.1.2 Temps continu, temps discret

Les variables d'état du système peuvent évoluer dans le temps de façon continue (le changement de valeur peut se faire à tout instant) ou à des moments ponctuels discrétisés dans le temps (dates des événements caractérisant un changement de l'état du système). Selon la perception du temps que le modélisateur va retenir, le système sera considéré comme étant continu ou discret. En croisant ces deux informations (temps et états), on peut distinguer quatres grandes familles de modèles.
En fonction de la nature des variables d'état (aléatoires ou non), on va de plus distinguer les modèles déterministes qui ne contiennent aucune variable d'état aléatoire, des modèles aléatoires ou stochastiques, pour lesquels il est nécessaire d'étudier un grand nombre de simulations et de réalisations aléatoires.

5.1.2 Simulation

La simulation consiste à faire évoluer une abstraction d'un système au cours du temps. L'utilisation de la simulation permet d'aider à comprendre le fonctionnement et le comportement de

ce système et à appréhender certaines de ses caractéristiques dynamiques, ceci dans l'objectif d'évaluer différentes décisions.[Hill, 1993]

La principale tâche de la simulation est de vérifier que le modèle (= abstraction du système) conserve un lien avec l'expérience (= réalité observée). La simulation est une des formes d'études d'un système. La simulation est particulièrement utile dans le cas où le système modélisé est un système dynamique complexe dans lequel se réalisent des actions aléatoires. En météorologie, par exemple, il n'est pas possible de connaître théoriquement le comportement de l'atmosphère à moyen ou long terme car c'est un système dynamique évolutif sensible aux conditions initiales. Le modèle représentant un tel système est en général difficilement formulable à l'aide des seuls outils mathématiques actuels et il est plus facile d'utiliser des techniques de simulation pour approcher le fonctionnement du système étudié.

La simulation ayant un lien direct avec la modélisation, les techniques de simulation sont également classées dans une des deux principales catégories de simulation : simulation continue ou simulation discrète, en fonction de la façon dont on considère le temps. La simulation continue est une technique de simulation utilisant des modèles comportant uniquement des composants analytiques, où le temps évolue de manière continue.
La simulation discrète - à événements discrets - est une technique de simulation à temps et états discrets. Sachant que les modèles mis en œuvre par la simulation peuvent être soit déterministes soit stochastiques on peut définir plusieurs types de techniques de simulation (hybride, mixte, combinée, etc.) selon le degré d'appartenance aux mondes analytique, discret, stochastique et déterministe des modèles mis en jeu [Coquillard et Hill, 1997]. L'intérêt principal présenté par la simulation à évènements discrets est qu'elle permet de modéliser un système pour lequel il est difficile d'obtenir des équations décrivant son fonctionnement de façon analytique ou, lorsque ces équations existent, qu'elles ne trouvent pas de solution du fait de leur complexité. Le principe de la simulation discrète est de permettre une description pas à pas dans le temps du fonctionnement du système sous forme d'un algorithme exécutable sur un ordinateur [Erard et Déguénon, 1996; Fianyo, 2000].

5.2 Notion d'agent

5.2.1 La modélisation individu-centrée

Selon les promoteurs de l'approche systémique [Checkland, 1976; LeMoigne, 1980; Checkland, 1981], la notion de système implique des propriétés telles que : l'émergence, l'interaction, l'interdépendance, la finalité et l'évolution. Ces propriétés sont applicables indépendamment de la dimension et de la nature de l'unité organisée faisant l'objet d'une étude. Les systèmes sont pour les systémiciens des construits théoriques, des hypothèses, des façons parmi d'autres de concevoir les ensembles.

Le modélisateur qui étudie un système, étudie donc *une unité globale organisée d'interrelations entre éléments, actions ou individus* [Morin, 1990]. Il est alors naturel de voir apparaître l'individu dans la définition du système, individu qui peut être suivant le type de système un simple élément ou un élément de décision. Les premiers modèles tenant compte d'interactions au niveau microscopique pour étudier un phénomène macroscopique apparaissent en thermodynamique statistique avec les travaux de Boltzmann [Cercignani, 1998] et de Gibbs [Jolls, 1990] dès la fin du 19ème siècle. Mais leurs théories reliant les propriétés et le comportement des atomes et des molécules aux propriétés et comportement macroscopique des corps qu'ils constituent, mettront du temps à être acceptées.

Les premiers modèles individu-centrés apparaissent en écologie dès la fin des années 40. Chitty [Chitty, 1960] utilise des matrices de Leslie [Leslie, 1945] pour modéliser l'évolution d'une population d'animaux subdivisée en classes d'âge. C'est l'une des premières modélisations de population qui prend en compte la différence de comportement due à l'âge de ses individus.

Cette approche se différencie de l'approche classique qui considère que la découverte d'idées de portée générale en écologie ne peut se réaliser qu'à l'aide des modèles possédant le moins de détails possibles [Smith, 1974]. L'approche traditionnelle de l'écologie considère qu'un modèle explique un phénomène lorsqu'on en a fait une théorie admissible, au terme de laquelle il apparaît que ce phénomène ne pouvait pas ne pas avoir lieu [Coquillard et Hill, 1997].

L'utilisation des modèles individu-centrés va néanmoins commencer à se répandre à partir du début des années 1980, en même temps que l'idée, chez les écologistes, que l'organisme individuel est une unité de base logique pour la modélisation de phénomènes écologiques [Judson, 1994]. Cette idée se propage d'autant mieux que les modèles traditionnels ne suffisent pas pour expliquer les dynamiques de nombreux systèmes observés dans la nature [DeAngelis et Rose, 1992; Babovic, 1996] et qu'elle s'appuie sur deux principes fondamentaux de la biologie [DeAngelis et Gross, 1992] à savoir :

- les individus biologiques sont uniques, ils différent les uns des autres par leur physiologie et leur comportement, en fonction de leur spécificités génétiques et des influences de leur environnement.
- un organisme donné est affecté principalement par les organismes qui se trouvent dans son voisinage spatio-temporel. C'est le principe biologique de la *localité*.

Dans cette approche, on va considérer que l'on a compris un phénomène lorsque, ayant exprimé des hypothèses structurelles et fonctionnelles en un langage symbolique manipulable (informatique), on peut explorer les conséquences de ces hypothèses et quand on retrouve alors (sous le même langage symbolique) le phénomène observé ; les hypothèses émises sont alors acceptées comme explications du phénomène [Coquillard et Hill, 1997].

L'ordinateur devient un outil nécessaire au modélisateur qui utilise sa puissance en continuelle augmentation pour créer des modèles comprenant un nombre d'individus de plus en plus

grand. Il va également s'approprier les nouveaux concepts de programmation qui émergent en informatique comme le concept d'agent auquel on associe des caractéristiques (interaction, autonomie, émergence) qui rejoignent complètement les propriétés de l'individu écologique.

5.2.2 L'agent : un outil conceptuel pour le modélisateur

La définition de l'agent d'un système multi-agents[1] la plus généralement utilisée par les modélisateurs est celle d'une entité logicielle (ou matérielle) d'un système informatique qui exhibe *du point de vue du modélisateur* les propriétés suivantes :

- *autonomie* : les agents ont le contrôle de leurs actions et de leurs états internes;
- *réactivité* : les agents perçoivent leur environnement et réagissent aux changements qui s'y produisent;
- *pro-activité* : les agents n'agissent pas seulement par réaction à leur environnement mais ils sont capables de prendre des initiatives en fonction de buts qui leur sont propres;
- *sociabilité* : les agents interagissent les uns avec les autres par un langage de communication quelconque (qui peut être l'environnement).

Wooldridge considère que cette définition très générale est la moins contentieuse et qu'après tout si l'on réussit à créer des applications intéressantes et utiles ce n'est pas un problème si l'on ne s'entend pas sur des détails terminologiques potentiellement triviaux. Il qualifie cependant cette définition de faible et porteuse de problèmes potentiels : le terme d'agent utilisé sans définition plus précise deviendrait un terme fourre-tout, qui perdrait tout sens [Wooldridge, 1999]. S'adressant à la communauté multi-agents informatique, Wooldridge et Jennings écrivent : *Agent might become a noise term, subject to both abuse and misuse, to the potentiel confusion of the research community* [Wooldridge et Jennings, 1995].

Ferber considère que l'approche consistant à définir l'agent par un ensemble de propriétés à respecter est pragmatiquement intéressante mais se révèle très vague et *un peu limité[e] pour fonder scientifiquement quelque chose* [Ferber, 1998].

1. *un s ou pas* : l'orthographe de multi-agents reste une question ouverte. Si le tiret semble avoir partie gagnée (multiagent ne se rencontre pratiquement plus), il n'en va pas de même pour le *s* final. La consultation de différents sites web montre que Le LIP6 (laboratoire de pointe en recherche multi-agents) a adopté l'invariant singulier de même que l'équipe SMA du CIRAD (à l'origine de la plate-forme CORMAS). Mais l'équipe MAGMA de Grenoble (autre équipe réputée en recherche SMA) a adopté l'invariant pluriel pour tout texte français, tout en conservant l'invariant singulier pour les textes en anglais. De plus la principale conférence francophone JFIADSMA sur le sujet adopte le *s* final. J'ai finalement choisi ce dernier usage, qui me paraît le plus logique.

5.2.3 L'agent : un concept technique

Selon Wooldridge et Jennings [Wooldridge et Jennings, 1995] une définition forte de la notion d'agent doit préciser la définition générale donnée plus haut en explicitant trois points clés :

- une théorie de l'agent définissant conceptuellement ce qu'est un agent et utilisant un formalisme précis pour représenter et raisonner sur les propriétés des agents;
- une architecture de l'agent représentant le modèle informatique de l'agent, implantant les concepts définis par la théorie de l'agent via la construction de structures logicielles et matérielles;
- un langage d'agents, c'est à dire un langage informatique facilitant la programmation des applications multi-agents par l'intégration des différents principes.

De nombreuses théories d'agents ont été (et continuent à être) proposées dès les années 1980 [Bond et Gasser, 1988].
Les premiers travaux sur les agents se sont attachés à formaliser ce que Gasser appelle les aspects macro à savoir tout ce qui concerne la sociabilité des agents : interaction et communication entre agents, décomposition et distribution de tâches, coordination et coopération, résolution de conflits par la négociation. On peut citer le modèle d'acteur de Hewitt [Hewitt, 1977] – l'acteur selon Hewitt est un objet qui gère des états internes et peut répondre à des messages provenant d'objets similaires; les systèmes multi-agents classiques comme MACE [Gasser et al., 1987], DVMT [Lesser et Corkill, 1981] où sont implantés des concepts de coopération et de coordination permettant la résolution de problème par décomposition et distribution de tâches; le protocole *contract net* standardisant la communication entre agents [Smith, 1980]; les travaux de planification distribuée de Rosenchein [Rosenschein, 1982].
Cette tendance vers les aspects cognitifs de la recherche sur les agents s'explique par le fait que les chercheurs provenaient pour la plupart de l'Intelligence Artificielle et considéraient l'agent comme une entité logicielle intelligente i.e. possédant un modèle symbolique du monde explicitement représenté et décidant de ses actions après avoir effectué un raisonnement symbolique sur ce modèle.

A partir des années 1990, l'intérêt s'est porté sur les systèmes composés d'agents agissant pour atteindre un but commun, les agents étant conçus comme des membres d'une équipe (chargés d'intégrer des informations trouvées sur internet [Decker et al., 1997]; de simuler un environnement d'entrainement virtuel [Tambe, 1995]; de jouer au football [Veloso et al., 1997]). Cela a conduit la communauté multi-agents informatique à se pencher sur l'aspect micro de l'agent i.e. sa constitution, son architecture interne [Müller, 1997]. Les aspects d'interaction entre agents, particulièrement importants dans ce cadre, sont au centre de nombreuses recherches [Müller, 1994; Huget et al., 2000], dans le but de mieux comprendre l'évolution au cours du temps de ce concept [Delepoulle et al., 2001] ou dans celui d'en proposer des formalismes [ElFallah-Seghrouchni et al., 2001].

Des théories de l'agent ont émergé qui précisaient ses propriétés : autonomie, réactivité et pro-activité. Certaines de ces théories ont été concrétisées dans la définition d'architectures d'agents et différents critères de classement des agents ont été proposés.
Les agents ont été classés selon leur mobilité : leur capacité à se déplacer dans leur environnement; on distingue les agents statiques des agent mobiles [Appleby et Stewaed, 1994].

Le processus de prise de décision de l'agent a également été un critère de classement : les agents réactifs [Agre et Chapman, 1987; Brooks, 1991; Ferber, 1994] réagissent à des stimuli provenant de leur environnement extérieur par des actions pré-câblées alors que les agents rationnels raisonnent sur un modèle symbolique pour décider de leur action.

On a également classé les agents en fonction des propriétés que l'on veut qu'ils exhibent. Nwana et Ndumu [Nwana et Ndumu, 1996] proposent la typologie de la figure 5.1, considérant qu'un agent devrait idéalement exhiber des propriétés d'autonomie, d'apprentissage et de coopération.

Une autre classification s'est faite sur la base de la fonction de l'agent, les agents d'interface

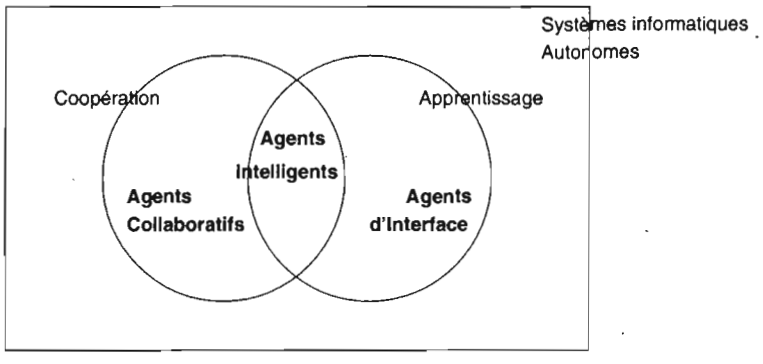


FIG. 5.1 – *Une typologie d'agent d'après [Nwana, 1996]*

[Maes, 1994; Kozierok et Maes, 1993] aussi appelés *personal digital assistant* ou PDA ont pour rôle de faciliter le travail de l'utilisateur humain en lui suggérant par exemple de meilleures façons de réaliser sa tâche. Les agents d'information (ou agents internet) ont pour rôle de gérer, manipuler, rapporter, exploiter l'information pertinente en provenance de diverses sources d'information distribuées.

On peut finalement considérer avec Nwana [Nwana, 1996] que l'agent existe dans un espace de propriétés multi-dimensionnel et qu'une classification exhaustive des types d'agents est difficile à réaliser dans un domaine où les recherches sur l'agent sont encore en cours.

Nous reprenons finalement la classification la plus simple basée sur les processus de décisions de l'agent qui permet de distinguer trois grandes classes d'agents :

- les agents à base logique ou cognitifs encore appelés agents rationnels : ils sont caractérisé par un processus de prise de décision basé sur la déduction logique. Les agents BDI (Beliefs-Desire-Intention) sont les descendants des agents rationnels : la prise de décision dépend de la manipulation de structures de données représentant les croyances, désirs et intentions de l'agent;
- les agents réactifs : le processus de prise de décision est implanté plus directement, à chaque situation dans laquelle se trouve l'agent correspond une action qu'il effectue;
- les agent hybrides : leur architecture est composée de plusieurs couches, chaque couche raisonnant sur l'environnement à différents niveaux d'abstraction et le processus de prise de décision pouvant varier du très symbolique au complètement réactif en fonction des situations.

5.3 Simulation et agent

5.3.1 Les paradigmes de la programmation agent

La programmation basée sur l'agent se distingue de la programmation classique par deux principes généraux :

- un programme n'est pas vu comme un agencement de blocs d'instructions mais comme une société d'agents autonomes qui interagissent;
- le langage de programmation inclut de nouveaux outils permettant d'expliciter les comportements individuels (y compris les buts et les émotions) des agents.

L'idée de décrire un programme en tant qu'éléments logiciels interagissant existe depuis le modèle d'acteur de Hewitt et la programmation orientée objet. On peut par ailleurs noter que le premier langage de programmation contenant des concepts de l'orienté objet (la notion de classe, par exemple) est le langage SIMULA67 [Kirkerud, 1989], langage destiné à la simulation. Mais les propriétés d'autonomie et de proactivité ne se retrouvent pas dans ces types de programmation.

Pour inclure ces propriétés dans le langage de programmation, la principale technique consiste à donner la possibilité d'attribuer aux agents des buts explicites, c'est à dire des représentations concrètes et accessibles de ce qu'ils sont supposés accomplir.

L'explicitation des buts permet de distinguer les objets des agents : les agents ne sont pas simplement des entités logicielles ayant un ensemble de capacités, ce sont des entités ayant un travail à faire. Gasser [Gasser et Briot, 1998] parle de concept d'*action persistante structurée* et considère que c'est le point clé, le principal apport du paradigme agent à la programmation. L'explicitation des buts permet de renforcer la propriété de réactivité des agents : les agents vont réagir aux évènements qui affectent leur buts. L'explicitation des buts permet également

de détecter les conflits pouvant surgir lorsque les actions d'un agent interfèrent avec les buts d'un autre agent.

5.3.1.1 Intégrer les buts : un apport intéressant pour la programmation

Depuis l'apparition de la programmation structurée, on peut considérer un programme comme un ensemble de modules. Chaque module est défini en fonction d'un but recherché par le programmeur. Ce but est évident pour ceux qui connaissent bien, et l'application dans son ensemble, et les relations de ses différentes parties, mais peut ne pas être évidente pour un nouveau venu cherchant à modifier l'application, par exemple. Les informaticiens préfèrent en général le développement de nouvelles applications à la maintenance d'anciennes parce qu'il est difficile de toujours garder à l'esprit la logique utilisée dans le développement des anciennes. De la même façon que le langage de programmation orientée objet permet de savoir à tout instant le type, les propriétés et les actions possibles réalisables par l'objet, la programmation à base d'agent (à travers l'explicitation de buts) permet d'atteindre un certain nombre d'objectifs [Travers, 1996]:

- *vérification*: les buts permettent aux agents de contrôler le succès ou l'échec des actions réalisées (les buts sont satisfaits quand les actions à réaliser l'ont été avec succès). Cela peut faciliter le processus de débuggage puisque les parties de programme qui posent problème équivalent aux agents échouant à atteindre leurs buts;
- *détection de conflits*: les situations de conflits sont courantes dans les systèmes où le contrôle est distribué. La présence de buts explicites permet à un agent de détecter les situations dans lesquelles d'autres agents interfèrent avec ses buts et permet au programmeur de mieux comprendre le problème;
- *lisibilité*: Des buts explicites rendent le programme plus compréhensible par un observateur qui lit le programme ou le regarde s'exécuter. Les buts servent de commentaires expliquant dans un style déclaratif ce que chaque procédure essaie de faire. Mais à la différence de commentaires textuels, les buts sont utilisables une fois le programme compilé.

5.3.1.2 Quelques systèmes de programmation basés sur l'agent

L'agent vu comme extension du processus

SWARM[2] est une plate-forme de simulation multi-agents créée au Santa Fe Institute [Minar et al.]. L'originalité du système provient de la notion d'activité associée à un *swarm* (essaim) permettant de contrôler le déroulement des actions effectuées par les agents. Un swarm est le composant de base du système. L'élaboration d'une application SWARM consiste à spécifier le comportement d'un ensemble d'agents (regroupés dans un *swarm*) sous la forme d'un

2. http://www.swarm.org/

ensemble d'actions à déclencher suivant un ordre déterminé par un *scheduler*. Le groupement d'actions dirigé par le scheduler est appelé activité. Il est possible de créer plusieurs *swarms* qui vont être liés entre eux par des liens hiérarchiques. Dans une application SWARM, un agent n'a pas d'autonomie interne : il possède un ensemble de capacités (les méthodes) qui sont utilisées par l'utilisateur pour spécifier une activité. Cela en fait un outil très flexible pour la spécification de modèles de simulation : il est possible de construire avec SWARM tout type de simulation.

StarLogo[3] est un environnement de modélisation programmable pour étudier les systèmes décentralisés. L'intérêt du système vient de la facilité avec laquelle on peut contruire des applications visuelles composées de très nombreux agents en interaction. Contrairement à SWARM, le comportement de chaque agent est spécifié de façon complètement interne et l'utilisateur ne peut spécifier aucun type de contrôle central.

L'agent vu comme extension de l'objet

Le formalisme AOP (Agent Oriented Programming) [Shoham, 1993] spécifie un paradigme de programmation orienté agent qui se veut l'héritier du paradigme de programmation orienté objet. Les agents sont vus comme des objets spécialisés. Alors que les objets peuvent contenir n'importe quel type d'attribut et communiquer à l'aide de n'importe quel type de message, la structure interne d'un agent ne peut contenir que des attributs de type croyance, capacité ou décision. Le type de message pouvant être traité par l'agent est aussi défini en terme d'activités mentales. Dans le formalisme de Shoham, les types de messages possibles viennent de la théorie des actes de langage. On distingue les messages d'information(inform), de demande(request), de promesse(promise) et de refus(decline). L'état d'un agent est défini comme un état mental. Les concepts du formalisme AOP ont été implantés dans le prototype Agent-0. Le langage PLACA [Thomas, 1994] est une extension du langage Agent-0 qui ajoute la notion de but. Contrairement aux agents Agent-0, les agents PLACA peuvent avoir des buts et planifient des actions en vue de les atteindre. Mais PLACA, résultat d'un travail de thèse, reste un langage expérimental.

CORMAS[4] (Common-pool Resources and Multi-Agent Systems) est un environnement de programmation dédié à la création de systèmes multi-agents. Comme SWARM, il est axé vers la construction de modèles de simulation , avec une spécificité dans le domaine de la gestion des ressources renouvelables. Il offre un cadre de développement de modèles de simulation des modes de coordination entre des individus et des groupes qui exploitent ces ressources en commun. Ce cadre se structure en trois modules. (i) Un premier module permet de définir les entités du système à modéliser, que l'on appelle des agents informatiques, et leurs

3. http://el.www.media.mit.edu/groups/el/Projects/starlogo/
4. http://www.cirad.fr/presentation/programmes/espace/cormas/

interactions. Ces interactions s'expriment par des procédures de communication directe (envois de messages), et/ou par le fait plus indirect de partager le même support spatial. (ii) Le second module concerne le contrôle de la dynamique globale (ordonnancement des différents événements d'un pas de temps du modèle). (iii) Un troisième module permet de définir une observation de la simulation selon des points de vue. Cette fonctionnalité autorise l'intégration, dans le processus de modélisation, des modes de représentation.
CORMAS facilite le travail de construction du modèle en proposant au sein des ces trois modules des éléments prédéfinis. Parmi ces éléments figurent les entités-type, qui sont des classes SmallTalk génériques à partir desquelles, par spécialisation et affinage, l'utilisateur définit des entités particulières pour les besoins de son application.

L'agent vu comme extension de la règle

Le formalisme de programmation téléo-reactive [Nilsson, 1994] a été proposé pour introduire la notion de la programmation continue dans la programmation classique. Un programme téléo-réactif présente la même structure qu'un système de production de règles classique: il est composé d'un ensemble de règles de production. Cependant, les actions possibles ne sont pas des actions discrètes comme *changer* le contenu d'une variable mais des actions continues comme *se déplacer*. Les actions continues vont persister tant que les conditions de déclenchement seront vérifiées.

5.3.2 La simulation multi-agents

La simulation multi-agents reprend le principe de la simulation discrète consistant à décrire le fonctionnement du système sous forme d'algorithmes exécutés par des processus informatiques (les agents). Elle se différencie de la simulation discrète classique par l'ajout de deux autres principes forts qui sont:

- utiliser des modèles centrés sur les agents et leurs interactions (et non sur des relations entre des valeurs mesurées);
- considérer que la dynamique générale du système est issue des interactions entre les agents. [Ferber, 1998]

La simulation multi-agents (les écologistes parlent de simulation individu-centrée) est une méthode d'étude de système basée sur la création dans un ordinateur d'un ensemble d'agents en interaction dans un environnement composant un monde artificiel (= le modèle multi-agents). Les agents du système vont effectuer des actions et des interactions locales ce qui va permettre d'observer des conséquences globales représentant la dynamique du système que l'on simule.

Les simulations multi-agents ont fait leur apparition dans les années 1980 en reprennant les concepts provenant de la programmation orientée-objet[Goldberg et Robson, 1983] et du modèle d'acteur [Hewitt et Atkinson, 1977; Agha, 1986]. Reynolds [Reynolds, 1987] en animation informatique a simulé le comportement d'un essaim d'oiseaux en implantant chaque oiseau

de l'essaim comme un acteur indépendant naviguant dans un environnement dynamique en fonction de perceptions locales. Le déplacement de l'essaim observé à l'écran n'était que le résultat de l'ensemble des interactions entre les oiseaux agissant en fonction de leurs règles propres.

Ces principes ont été repris dans de nombreux travaux appartenant à des domaines différents. En écologie, l'étude du comportement de sociétés d'animaux a donné lieu à la construction de nombreux simulateurs. On peut citer par exemple [Pichon et Mullon, 1991] pour les parasites, [Drogoul, 1993] pour les fourmis , [Sumpter, 1997] pour les abeilles, [Shin et Cury, 2001] pour les poissons. En sciences sociales, les simulations multi-agents ont permis d'étudier le comportement de foules [McPhail, 1997; Farenc et al., 1998; Farenc et al., 2000] ; en recherche économique les théories microéconomiques reposant sur les préférences d'individus expliquant des phénomènes macroéconomiques comme le fonctionnement des marchés sont testés dans les nombreux travaux d'économie comportementale [Kagel et Roth, 1995].

Certains de ces travaux ont permis de poser les bases d'un nouveau domaine de recherche : la vie artificielle [Langton, 1994] dont l'objectif est de reproduire la vie *in sillico*, i.e. d'utiliser les modèles d'organisations rencontrés en biologie, en écologie et en génétique pour construire des systèmes complexes, dotés en particulier de capacités d'adaptation. C'est ainsi qu'en travaillant en collaboration avec des psychologistes, Ph. Preux [Preux et al., 2001] a capturé dans un monde virtuel certaines des règles qui expliquent l'acquisition et le développement comportemental des animaux, ce qui a permis de simuler par l'utilisation d'algorithmes d'apprentissage par renforcement, l'acquisition du mouvement par un bras humain.

D'autres domaines de recherche, comme la théorie des jeux ont pris un nouvel essor avec les possibilités présentées par les méthodes de simulation multi-agents. Comme cela se passe souvent dans les sciences nouvelles, les travaux de simulations multi-agents ont été réalisés dans des langages et avec des formalismes très divers en fonction des diverses expériences des concepteurs [Fianyo et al., 1997]. Cela rendait difficile non seulement la réutilisation de concepts intéressants, mais aussi la maintenabilité de telles applications. Pour contribuer à résoudre ces problèmes, des plates-formes de simulation multi-agents sont apparues. Le plus abouti est SWARM. Il en existe d'autres, comme XRaptor d'une équipe de chercheurs de l'université de Mainz ou Ecoland conçu par John Holland. Ce genre d'outil permet notamment de faire se rencontrer des chercheurs de différentes disciplines utilisant tous le même langage pour manipuler des concepts différents.

Les simulations multi-agents sont en effet un support au travaux interdisciplinaires. Depuis le travail de [Bousquet et al., 1994], on a vu apparaître des simulateurs capitalisant le résultats de travaux scientifiques provenant de différentes disciplines scientifiques [Quensière, 1994; Skinner et al., 1997; Hasler et al., 1999].

La construction d'un simulateurs multi-agents pose néanmoins un certains nombre de problèmes qui restent ouverts :

- *des problèmes techniques* :
  - la gestion d'un nombre important d'agents indépendants pose des problèmes de rapidité des applications, de capacité des machines, etc.
  - il est difficile de vérifier le bon fonctionnement de telles applications, il n'existe pas encore de modèle de tests
- *des problèmes de modélisation* : il n'est pas toujours évident de trouver le niveau pertinent de définition des agents : lorsqu'on s'intéresse à des phénomènes sociaux, doit-on agentifier tous les individus, ou des groupes d'individus homogènes ? Lorsque l'on s'intéresse à des phénomènes se réalisant dans l'espace, comment représenter ceux-ci ? [Treuil et al., 1997; Treuil et al., 2001]
- *des problèmes conceptuels* : comment maîtriser le concept d'émergence sur le plan conceptuel et formel ? Peut-on dégager une théorie de l'interaction et des structures d'organisation ? [Ribeiro, 2000]

Conclusion

La simulation multi-agents permet de modéliser un monde sous forme d'interactions entre agents. Ces agents représentent des entités du monde modélisé, que l'on a doté d'un comportement. Les connaissances de différentes sources sont intégrées dans le comportement des agents. Dans la plupart des exemples, il s'agit d'une intégration forte. Des architectures de couplage faible basées sur les agents ont cependant été proposés. Nous en présentons quelques unes dans le chapitre suivant.

Chapitre 6

Agents et Couplage de logiciels existants

RÉSUMÉ :

Des architectures de couplage faible basées sur le concept d'agent et sur le concept d'objet, ont été proposées. Nous présentons dans ce chapitre quelques unes de ces architectures. Il est intéressant d'analyser ces systèmes afin d'en extraire les idées qui fondent ces approches. Elles peuvent se révéler utiles dans notre contexte.

La démarche du couplage est présente en informatique, où on utilise le terme d'*interopérabilité*. En base de données notamment, l'interopérabilité de systèmes d'information a fait l'objet de nombreux travaux de recherche. L'interopérabilité vise à permettre le partage, le contrôle et l'échange de données entre plusieurs sources d'information hétérogènes. En base de données, les outils d'interopérabilité se classent en deux familles :

- les solutions classiques reposent sur des méthodes d'intégration de schémas [Sheth et Larson, 1990], où chaque site participant au système d'information hétérogène va exporter son schéma dans un modèle commun. Une base de données fortement couplée va utiliser un schéma fédéré global qui est une fusion de tous les schémas des sites locaux. Une base de données faiblement couplée va utiliser un schéma fédéré qui résulte de l'intégration d'un sous-ensemble des schémas locaux.
- les solutions basées sur des architectures de médiation. Dans ce cas, les schémas initiaux ne sont pas touchés, mais on définit un niveau constitué de médiateurs pour assurer la liaison entre les différentes sources de données. Les médiateurs sont chargés de résoudre les conflits sémantiques entre les données des différentes sources. Pour cela, ils s'appuient sur les wrappers qui assurent les fonctions d'encapsulation des bases locales et se chargent du traitement des requêtes locales.

La deuxième approche repose sur le paradigme objet [Liu et Pu, 1997] et un certain nombre de travaux se réclamant de cette approche utilisent le concept d'agent afin de spécifier *wrapper* et médiateurs.

Le système DECA [Benslimane et al., 1998] est un exemple de ce type de solution. DECA est une architecture d'interopérabilité de systèmes d'information hétérogènes bâtis sur trois types d'agents : les agents wrapper ; les agents de coopération (AC) jouent le rôle de médiateurs ; l'agent ontologie définit le vocabulaire commun utilisé par les agents AC pour s'échanger des requêtes sans avoir à travailler sur un schéma global. L'architecture générale du système est représentée sur la figure 6.1.
Dans cette solution, le couplage consiste à définir des correspondances entre les différentes relations (appelées concepts), les attributs et les valeurs par l'utilisation de trois prédicats (EC(C,C') : équivalence des concepts C et C'; EA(C.A, C'.A') : équivalence de deux attributs); EV(C.A, V1, V2) : équivalence de deux valeurs pour un même attribut. Les agents utilisent le langage KQML (Knowledge and Query Manipulation Language) pour s'envoyer informations qui portent donc sur les données.

Les systèmes de gestion de bases de données ont pour caractéristique principale, la propriété de répondre à un ensemble de fonctionnalités communes. Ils manipulent un ensemble de concepts communs. L'objet d'interopérabilité est donc toujours bien déterminé : il s'agit de mettre en relation les différents schémas de données. Dans le couplage de modèles, l'interconnexion doit se faire au niveau de la dynamique appliquée aux données et non au niveau des données elles-mêmes.

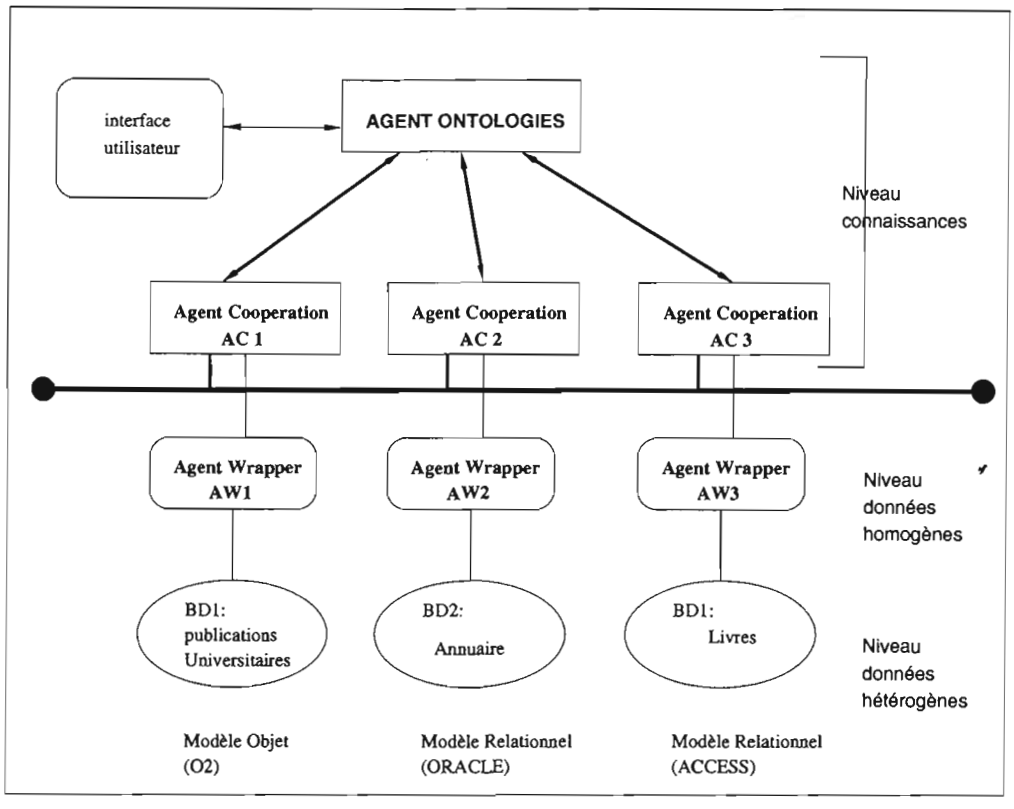


FIG. 6.1 – *Architecture générale de DECA [Benslimane et al., 1998]*

Dans un autre domaine, S. Pinson a proposé un système, le système CREDEX, permettant de synthétiser différents points de vue donnés par différents modèles [Pinson, 1991; Pinson, 1992]. Pour ce faire, le système fait appel à un méta-modèle muni d'une structure de tableau noir, qui pilote l'activation des différents modèles et fait la synthèse des évaluations. Elle l'a appliqué à l'évaluation du risque crédit entreprise.

6.1 Couplage à l'aide d'agents

Les systèmes multi-agents étant nés (entre autre) de la volonté de faire coopérérer les systèmes experts, certains ont naturellement été bâtis dans le but de coupler des logiciels qui existaient déjà. Les paragraphes qui suivent présentent quelques applications multi-agents illustratives du couplage d'applications de différents types.

6.1.1 Couplage de logiciels industriels : le projet ARCHON

ARCHON (ARchitecture for Cooperative Heterogeneous ON-line systems) [Wittig et al., 1994; Jennings et al., 1996] est un projet européen d'application des outils de l'intelligence artificielle

distribuée au domaine industriel. ARCHON propose une méthodologie et une architecture multi-agents pour intégrer des logiciels industriels. Un environnement informatique industriel est en général constitué d'un nombre important de logiciels hétérogènes, créés de façon *ad hoc* pour résoudre un problème spécifique sans prendre en compte l'existence des autres outils logiciels. La méthodologie ARCHON consiste à appliquer une approche "top-down" pour l'analyse du problème d'intégration (déterminer quels sont les composants nécessaires à la résolution du problème) ; on applique simultanément une approche "bottom-up" pour faire l'inventaire de tous les logiciels qui existent. On peut alors intégrer dans la plateforme ARCHON de nouveaux outils logiciels supportant les fonctions non réalisées par les logiciels existants. ARCHON peut être considéré comme un outil de couplage car il offre la possibilité de considérer les logiciels préexistants comme des composants de base qu'un agent ARCHON va encapsuler. L'architecture ARCHON (figure 6.2) est basée sur le concept d'agent. Un agent ARCHON est composé d'un module appelé couche ARCHON (ARCHON Layer - AL) (chargée de dialoguer avec les autres agents) et d'un système intelligent (IS - l'application à coupler) (chargée de gérer les connaissances du domaine). Le système ARCHON a été appliqué dans deux environnements

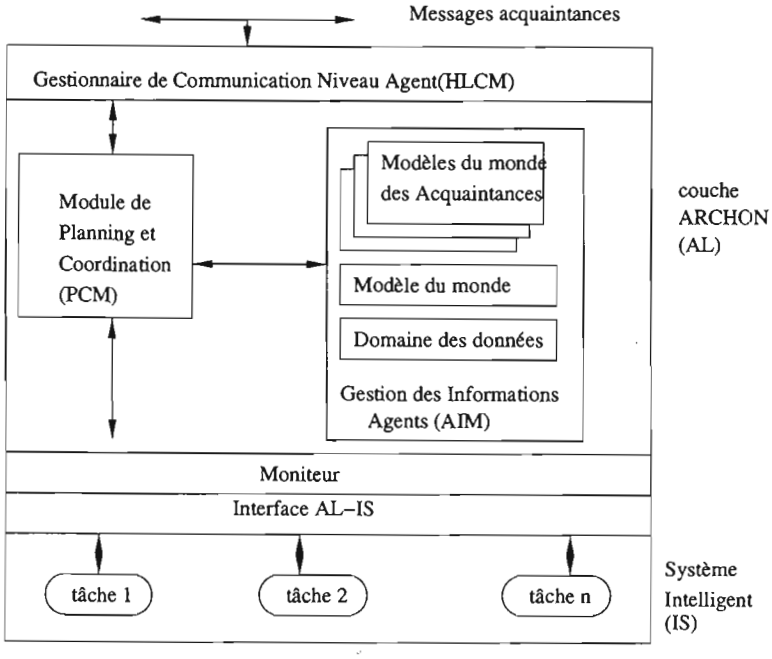


FIG. 6.2 – *Architecture d'un agent ARCHON [Jennings et al., 1996]*

industriels :

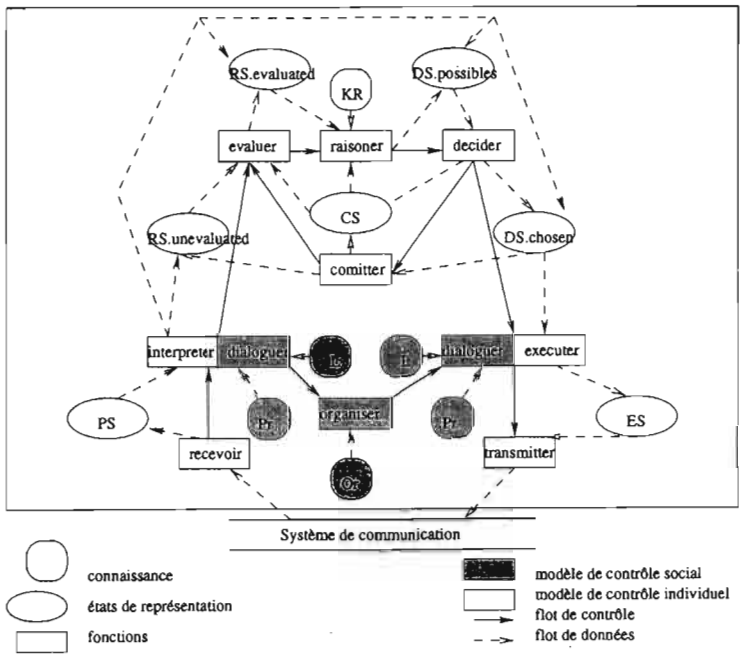
- dans une société industrielle de production d'électricité du nord de l'Espagne Iberdrola, le problème était de construire puis de coupler différents logiciels (d'analyse, de diagnostic) permettant d'aider l'opérateur humain à prendre rapidement les bonnes décisions en cas

de panne provenant dans le réseau de transport de l'électricité contrôlé (apparition d'une alarme dans le système de monitoring).

- ARCHON a également été utilisé au CERN, institut de recherche européen en énergie nucléaire pour contrôler et diagnostiquer les pannes dans l'accélérateur de particules "Proton Synchroton".

Le système ARCHON, construit à partir de l'analyse de problèmes d'expertise provenant du système industriel, présuppose que les applications à coupler sont des systèmes experts (ils raisonnent sur le système physique qu'ils contrôlent et sont capables de réaliser différentes tâches). Les applications ARCHON utilisent de ce fait un nombre restreint d'agents (deux pour l'application de contrôle d'un accélérateur de particules au CERN, sept pour l'application de contrôle d'un système de transport d'électricité à Iberdrola) et de l'avis même des auteurs [Jennings et al., 1996], il n'est pas adapté à un environnement où l'utilisation d'un nombre important d'agents serait nécessaire.

6.1.2 Couplage de fonctions en traitement d'images: le modèle ASIC



L'architecture de contrôle ASIC. PR, RS, DS, CS, ES sont les états de perception, de raisonnement, de décision, d'engagement et d'exécution; KR les connaissances, L le langage d'interaction, Pr les protocoles d'interaction et Or les organisations. Le modèle de contrôle social est en grisé, le modèle de contrôle individuel est en blanc.

FIG. 6.3 – *Architecture de contrôle ASIC [Boissier et Demazeau, 1997]*

Le modèle d'agent ASIC (Architecture for Social and Individual Control) fait partie d'une architecture générique multi-agents définie dans le but de construire des systèmes intégrés de vision (SIV) à la fois ouverts et flexibles [Boissier et Demazeau, 1997]. Les traitements réalisés par les SIV sont nombreux et très différents. Il y a des traitements numériques (extraction des indices 2D d'une scène) qui agissent directement sur l'image avec un comportement déterministe; il y a également des traitements symboliques (construction d'une description symbolique d'une scène). Par ailleurs, les SIV doivent permettre l'intégration de nouveaux traitements, en fonction d'un objectif donné (utilisation d'une fonction donnant de meilleurs résultats par exemple) et doivent permettre aux différents traitements de se contrôler les uns les autres : il doit être possible de vérifier la certitude d'hypothèses émises au niveau d'une description symbolique par un nouveau traitement numérique d'exploration d'une autre partie de l'image par exemple. Ces échanges entre traitements dépendent fortement de la situation courante, il faut donc pouvoir les définir dynamiquement. L'architecture multi-agents ASIC définie pour adresser toutes ces préoccupations est basée sur un modèle d'agent générique de type BDI(Belief-Desire-Intention). Un agent va être instancié sur un traitement particulier (approximation polynomiale, extraction 2D, etc) sur lequel il va exercer un contrôle particulier. Ce contrôle est décomposé en : (i) un contrôle social relatif à la définition de modes de coopération entre les agents, (ii) un contrôle individuel relatif à la définition de modes de fonctionnement du traitement associé à l'agent (figure 6.3). L'architecture ASIC peut être considérée comme une architecture de couplage de fonctions concernant le traitement d'image. Une fonction de traitement d'image peut être couplée avec d'autres fonctions dès lors que l'on a défini pour elle et pour les autres fonctions un langage générique, issu du domaine d'application, capable de définir la structure d'échange commune. Sur chaque fonction de traitement dotée d'un langage du domaine va être instancié un agent ASIC. Les agents ASIC vont utiliser un langage d'interaction propre (le langage commun L de la figure 6.3) pour s'envoyer des messages de contrôle.

Dans les deux exemples de couplage précédents, les applications à coupler ont été créées au moins en partie par les concepteurs de l'outil d'intégration. Le modèle ASIC est un travail réalisé suite à la construction de deux systèmes de vision (Multi-Agent system for Visual Integration -MAVI et système Multi-AGents pour l'Interprétation et Compréhension de scènes -MAGIC) par l'auteur de l'article. Le projet ARCHON a été un projet d'équipe dont les membres ont travaillé à la construction de chaque application qui a par la suite été intégrée.

6.1.3 Couplage de Systèmes d'Aide à la Décision à l'aide d'Agents Logiciels Intelligents

Afin de favoriser la coopération entre des Systèmes d'Aide à la Décision (SAD) dédiés à la gestion de crise, [Jaber et al., 1998] ont proposé une architecture d'Agents Logiciels Intelligents (ALI) chargés de s'échanger un ensemble de services.

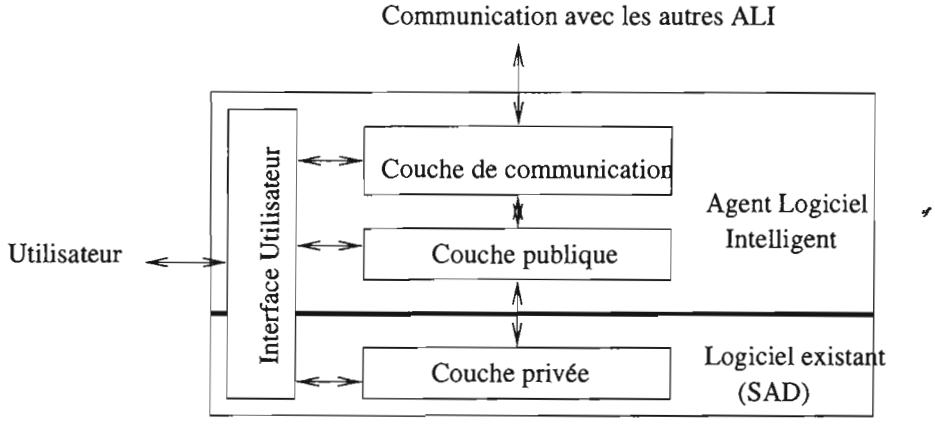


FIG. 6.4 – *Le modèle de l'ALI [Jaber et al., 1998]*

Dans l'application développée, il y a cinq applications que l'on veut intégrer : le système Bosque est dédié à la détection précoce de feux de forêt, le système Florinus assure la localisation des moyens terrestres et aériens et permet de donner le point d'une situation sur le terrain, le système Météo assure la gestion et l'analyse de données météorologiques, le système Wilfried analyse le risque feux de forêt, MCI est une Main Courante Informatisée qui gère l'ensemble de l'information reçue. Chacun de ces SAD est conçu selon un même principe : il propose un ensemble de services à un utilisateur. Ces SAD existent sur différents systèmes (PC et Unix). Un Agent Logiciel Intelligent va être associé à chacun de ces SAD.
Le modèle ALI (Agent Logiciel Intelligent) est représenté sur la figure 6.4. Pour garantir l'ouverture du système multi-agents vers des applications qui ne sont pas fondées selon le modèle d'Agents Logiciels Intelligents, les auteurs ont repris les principes de communication émanant de l'ISO et utilisent le langage ASN.1 (Abstract Syntax Notation One) pour la description des messages envoyés entre agents et le standard BER(Basic Encoding Rules) pour les codages de messages sous forme binaire transmis sur le réseau.

6.1.4 Comparaison et synthèse

Dans tous les exemples de couplage exposés, l'agent utilisé est composé de deux parties :

- une partie spécifique au domaine auquel il est dédié : le Système Intelligent (IS) de ARCHON, le modèle individuel du modèle ASIC et la couche privée du modèle ALI;

- et une partie dédiée à la mise en oeuvre du couplage, comportant des capacités de raisonnement sur les autres. C'est la couche ARCHON (AL) de ARCHON, le modèle social de l'ASIC et l'ALI dans le modèle de [Jaber et al., 1998]. Cette partie couplage définit la communication entre agent et organise cette communication entre agent.

6.1.4.1 La partie spécifique au domaine

Dans ARCHON, chaque IS représente un ensemble de tâches décrites par des noms spécifiques à leur environnement d'exécution. A chaque tâche correspond un nom ARCHON au niveau du moniteur. Ce nom est indépendant de l'environnement d'exécution et l'interface AL-IS est en chargée de faire la traduction entre les deux notions.
Dans ASIC, c'est KR, la base de connaissances qui contient l'ensemble des compétences de l'agent.
Dans le modèle ALI, la description détaillée de chaque SAD est stockée dans un module spécifique appelé "self-model".

La partie mise en oeuvre du couplage fait dans tous les cas appel à une notion de langage commun entre les agents, à la notion de message et de structure de message.

6.1.4.2 Structures de communication entre agents

Dans ARCHON, c'est le module de gestion de l'information de l'agent (AIM) qui fournit les services de coopération entre agent. Pour l'ensemble des applications du système, on va spécifier un AIM spécifique (selon le domaine, l'ensemble des messages gérés par l'AIM ne sera pas le même). La communication entre agents dépend fortement des composants à coupler. Le modèle du monde des autres agents correspond à l'ensemble des messages qui peuvent être compris par les accointances de l'agent considéré.

Dans le modèle ASIC, le langage d'interaction entre agent est très formalisé. Une interaction entre deux agents est définie par (i) l'ensemble des informations utiles pour le routage des messages sur le réseau ; (ii) le langage multi-agents qui définit les informations relatives au contrôle des agents et est indépendant de tout domaine d'application (le message envoyé est-il un but à atteindre, une expression d'hypothèse, etc) (iii) le langage du domaine qui exprime la sémantique du message. Celui-ci dépend fortement du domaine d'application.

le modèle de l'ALI définit une couche de communication chargée de la réception et de l'envoi des messages entre agents. Un message est composé d'un en-tête qui contient des informations générales sur la communication et d'un corps de message qui appartient à un des quatre types suivants : libre, demande de service, réponse à une demande et envoi d'une information. Ces différents types de messages font partie d'un langage de communication à six primitives qui organisent la communication entre les agents.

6.1.4.3 Organisation de la coopération (communication) entre agents

Dans tous les cas, chaque système définit un module chargé de gérer la communication avec les autres agents. Cela consiste notamment à définir dans quel cas la tâche demandée peut être exécutée localement, lorsqu'une tâche ne peut être exécutée localement par l'agent à quel(s) agent(s) la déléguer, de quelle façon (décomposition ou non de la tâche en différentes sous tâches), etc.

Dans ARCHON c'est le module de planification et de coordination (PCM) qui organise la coopération entre les agents. Le PCM est composé d'un ensemble de règles génériques concernant la coordination entre agents et d'un ensemble de règle spécifiques à l'application considérée sur lesquelles il se base pour exploiter les informations contenues dans l'AIM en fonction de la situation courante et en fonction des buts qu'il poursuit.

Dans le modèle ASIC, l'ensemble Pr définit les protocoles d'interactions possibles au niveau de chaque agent. Un protocole d'interaction définit *a priori* l'enchainement des interactions possibles. Chaque fois qu'un agent reçoit un message il consulte sa copie locale de protocoles (Pr) et choisit en fonction de son but le protocole le mieux adapté.

Dans le modèle ALI, le module de communication comprend un algorithme de communication à quatre niveaux, que l'agent applique pour déterminer sa prochaine action.

Le modèle d'agent utilisé appartient dans tous les cas à la famille des agents cognitifs décrits dans la section 5.2.3.
Plus le degré de spécialisation de l'architecture est grand, plus le langage utilisé est riche : le modèle ALI qui est très général n'utilise que six actes de communication, ASIC très spécifique au domaine de la vision utilise des 10-uplets dont chaque terme possède trois ou quatre valeurs possibles, ARCHON utilise un mini-système expert (le PCM) pour gérer la communication entre agents.

6.2 Couplage à l'aide d'objets

L'apparition du concept objet a permis l'émergence de différentes techniques facilitant l'interopérabilité entre systèmes informatiques. Les deux principales architectures sont CORBA, DCOM et DCE qui reposent toutes sur le protocole RPC, lui même reposant sur les *sockets*.

6.2.1 Quelques architectures d'interopérabilité

6.2.1.1 Les principaux outils

la programmation socket. Un socket est un canal de communication à travers lequel des applications peuvent se connecter et communiquer (écriture et lecture de données dans le socket). L'utilisation des sockets est la façon la plus directe de transmettre des données. Elle est très efficace, mais les applications communicantes doivent partager le même langage de programmation afin d'interpréter correctement les données échangées. C'est un outil de bas niveau.

le Remote Procedure Call. RPC offre à l'utilisateur une interface orientée fonctions au-dessus des sockets. L'appel de la procédure distante constitue la requête cliente, le retour de la procédure constitue la réponse serveur. Un appel de procédure obéit à un fonctionnement synchrone : une instruction suivant un appel de procédure ne peut pas s'exécuter tant que la procédure appelée n'est pas terminée (figure 6.5).

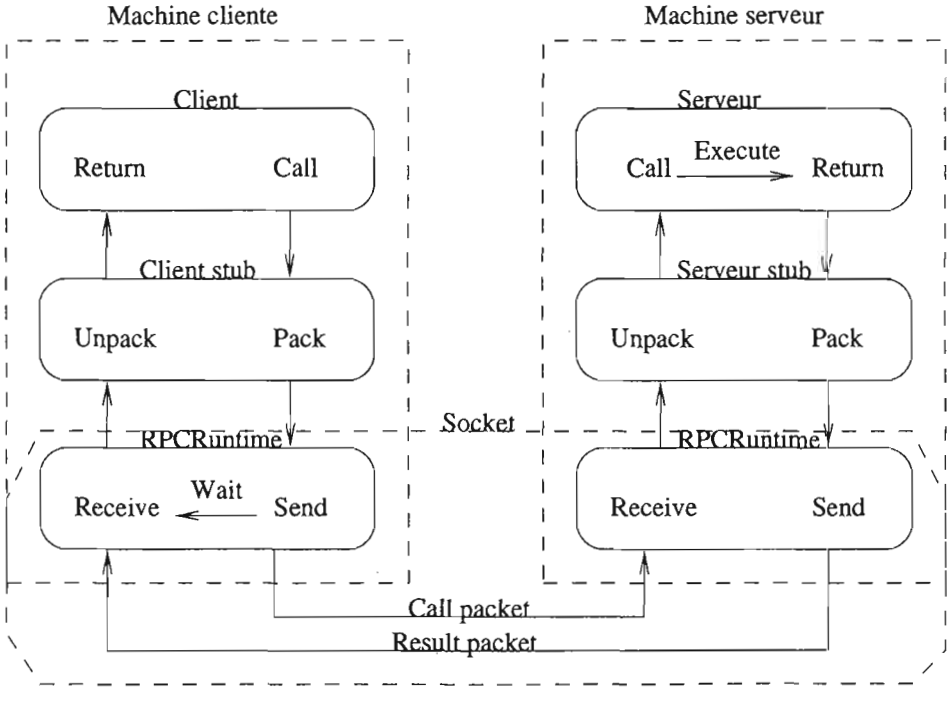


FIG. 6.5 – *structure d'une communication RPC*

6.2.1.2 le Distributed Computing Environment de l'OSF

Le Distributed Computing Environment (DCE) de l'OSF (Open Software Foundation) est un ensemble de standards incluant un standard RPC, qui propose comme CORBA un environnement de programmation distribué. Il est composé de six paquetages : un paquetage de gestion des threads le *threads package*, un paquetage pour la gestion des RPC le *Remote Procedure Call facility*, un service de gestion du temps du système, le *Distributed Time Service*, un paquetage de gestion des noms le *Name services*, un paquetage de gestion de la sécurité le *Security service*, et le *Distributed File Service*, le système distribué de gestion de fichiers.

6.2.1.3 l'architecture CORBA

L'architecture CORBA (Common Object Request Broker Architecture) spécifiée par l'OMG (Object Management Group) permet le développement d'applications réparties.
Un objet CORBA est placé sur un serveur et peut à la fois émettre et recevoir des requêtes/réponses d'autres objets placés sur le même serveur ou sur des serveurs distants. Les interactions entre les objets sont gérées par un bus logiciel, l'ORB (Object Request Broker). Grâce à l'ORB, les mécanismes de communication et d'activation des objets sont totalement transparents pour les clients. C'est l'ORB qui gère l'ensemble des relations client/serveur entre objets.
La technique de mise en oeuvre (figure 6.6) consiste à décrire l'interface de l'objet à l'aide

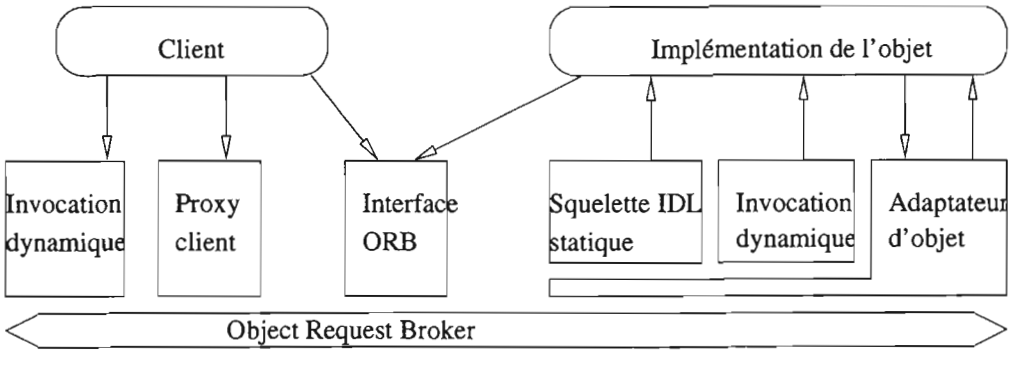


FIG. 6.6 – *Structure des interfaces ORB. D'après [OMG, 1999]*

d'un langage standardisé, l'IDL(Interface Definition Langage), puis à compiler cette interface afin d'obtenir une classe *proxy*[1] dont l'instance sera le représentant local de l'objet distant, et une classe *squelette*[1] dont l'implémentation de l'objet devra hériter. Pour effectuer une

1. proxy : application recevant des requêtes qui ne lui sont pas destinées et qui les transmet aux autres serveurs. Stub est un synonyme de proxy. Un squelette a également la même fonction. On parle aussi de serveur mandaté. Dans la littérature DCOM, les serveurs mandatés coté clients sont appelés proxy, coté serveurs, on parle de stub.

requête, le client peut soit utiliser une invocation statique via le proxy local de l'objet, soit la construire dynamiquement. L'ORB est chargé de localiser l'objet référencé et d'invoquer la méthode à travers le squelette. Le client et l'implémentation de l'objet peuvent accéder directement à l'ORB à travers une interface spécifique. Une seconde interface d'accès à l'ORB, l'adaptateur d'objet, est réservée à l'implémentation de l'objet (enregistrement, désactivation d'objets, etc.)
En plus de la gestion des interactions, l'ORB fournit aussi divers services directement intégrés à l'architecture et possédant une interface IDL. Ces services font partie de l'Object Management Architecture (OMA).

6.2.1.4 le Distributed Component Object Model de Microsoft

DCOM est l'extension distribuée de COM (Component Object Model) qui construit une couche objet RPC (ORPC) au dessus du RPC standard pour communiquer avec les objets distants (figure 6.7). Un serveur COM peut créer une instance d'objet à partir de différentes classes d'objets. Un objet COM peut avoir plusieurs interfaces, une interface représentant une vue différente ou un comportement d'un objet. Un client COM interagit avec un objet COM en obtenant un pointeur sur une des interfaces de l'objet et en invoquant des méthodes de l'objet à partir de ce pointeur, comme si l'objet résidait dans son espace d'adressage. Une interface COM suit une gestion de la mémoire standardisée. Cette standardisation étant réalisée au niveau binaire, cela permet l'intégration de composants binaires qui ont pu être écrits en différents langages de programmation. Le modèle DCOM présente des services équivalent à

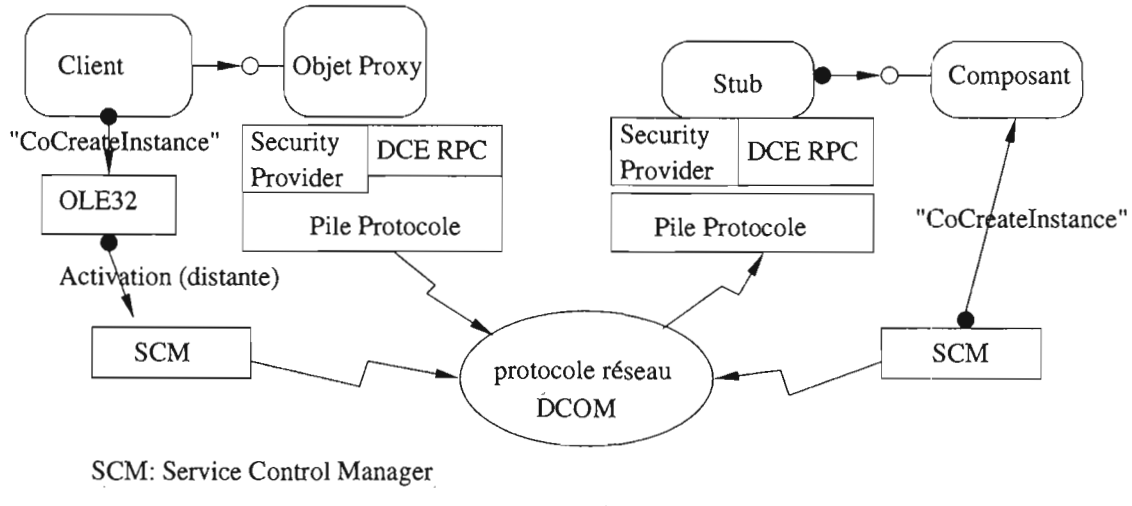


FIG. 6.7 – *Architecture DCOM. [Horstmann et Kirtland, 1997]*

ceux de CORBA et est bien adapté aux environnement Windows. Il existe des ponts CORBA-DCOM permettant aux objets des deux familles de communiquer.

6.2.1.5 Comparaison/Synthèse

L'architecture DCE est principalement utilisée dans la communauté UNIX. Mais ce sont les architectures CORBA et DCOM qui tendent à s'imposer. La différence entre les architectures CORBA et DCOM qui offrent des services similaires, est que DCOM gère des objets possédant plusieurs interfaces, alors que dans CORBA un objet n'a qu'une interface, mais le modèle permet à une interface d'hériter de plusieurs autres interfaces [Chung et al., 1998].

Le monde de la simulation s'est également attaqué au problème du couplage. Dans les exemples rencontrés dans la littérature, c'est l'utilisation des objets CORBA qui est la plus fréquente.

6.2.2 Le projet IMPRESARIO

IMPRESARIO est une plate-forme intégrée de modélisation, de simulation et de visualisation de données pour faciliter le couplage de modèles de différents processus physiques créés indépendamment dans un système unifié [Kernan, 1996].
Les problèmes auquels s'intéresse le laboratoire Sandia touchent à de nombreuses disciplines de la physique. Par exemple, l'analyse d'un crash et de l'incendie qui en découle fait appel aux connaissances en mécanique, au transport thermique, et aux interactions chimiques.
IMPRESARIO a été construit pour résoudre un problème prototype de la thermodynamique. L'objectif était de tester l'intérêt d'un couplage entre codes déjà existants pour résoudre un problème complexe en le décomposant en fonction des applications existantes et non plus en fonction de la charge de calcul sur différents processeurs comme c'est l'usage dans le domaine de la simulation parallèle [Fujimoto, 1993]. Deux objets CORBA ont été définis qui encapsulent chacun une des deux applications suivantes :

- ALEGRA, un programme de simulation de la dynamique des solides utilisé pour calculer la déformation d'un solide suite à un choc, écrit en C++,
- COYOTE, un programme de simulation d'analyse thermique pour calculer le transfert de chaleur dû à la radiation et à la conduction, écrit en FORTRAN.

La réalisation de ces deux applications a montré que le couplage de programmes provenant de différentes spécialités de la physique pour analyser des problèmes physiques était possible. Cela s'avérait d'autant plus intéressant qu'il existe un grand nombre de logiciels bien validés en transfert de chaleur, en physique des ondes de choc, en mécanique quasi-statique, en kinétique chimique et en électromagnétique. Cependant, les travaux de couplage utilisant la technologie CORBA sont pour l'instant arrêtés parce que les superordinateurs qui constituent l'essentiel

du parc informatique des US National Laboratories n'offraient pas de support pour le protocole CORBA[2].

6.2.3 Le projet HLA

L'architecture HLA (High Level Architecture) [Calvin et Weatherly, 1996; Dias et al., 1998; Lutz, 2000] est proposée par le Defense Modeling and Simulation Office (DMSO) du Département de la Défense américain (DoD) pour faciliter la simulation distribuée de systèmes. Chaque système à coupler est appelé un fédéré (*federate*) et l'ensemble du système distribué est désigné par le terme fédération. HLA est composée de trois parties définissant chacune un standard:

- *les règles HLA* décrivent les responsabilités des fédérés et des fédérations. Elles comprennent la description d'un ensemble de principes techniques permettant la mise en oeuvre de l'architecture HLA. Pour les fédérations, les règles définissent le modèle objet de la fédération (FOM), la représentation objet et l'échange de données. Pour les fédérés, les règles définissent un modele objet de la simulation (SOM);
- *la spécification de l'interface d'un fédéré* : dans l'architecture HLA, chaque fédéré interagit avec un RTI (Runtime Infrastructure : une sorte de système d'exploitation distribué miniature) pour établir et gérer une fédération et pour permettre l'échange des informations utiles entre les différents fédérés. L'interface de spécification HLA définit la nature de ces interactions qui sont spécifiées par les services RTI;
- chaque fédéré et chaque fédération doit avoir un modèle objet décrivant les entités représentées dans les simulations et décrivant également les données à échanger entre fédérés. *le modèle OMT (Object Model Template) de HLA* définit la méthode d'enregistrement dans les modèles objets, pour inclure les objets, les attributs, les interactions et les paramètres sans *a priori* sur leurs types respectifs.

Pour Banks [Banks, 2001], malgré ses objectifs louables (faciliter l'interopérabilité, promouvoir la réutilisation des simulations), l'architecture HLA doit être grandement simplifiée et révisée (elle contient trop de concepts peu explicites) afin que l'on puisse l'utiliser pour développer des modèles de façon transparente.

Conclusion

Nous nous proposons de transposer l'application du concept d'agent pour le couplage de logiciels existants dans le domaine de la simulation en construisant une plate-forme d'intégration de logiciels de simulation. La notion d'agent offre la possibilité d'exprimer dans un langage

2. *comm. pers.* juillet 2000, Randall M. Summers, chef du projet ALEGRA, Sandia National Laboratories, DoE.

de haut niveau, donc plus expressif, les règles qui doivent être respectées par chacun des logiciels à coupler. La notion d'agent offre un cadre conceptuel permettant de s'extraire du code pour réaliser un couplage faible (ne pas modifier les applications de départ). Cependant, la coordination entre agents dans les différentes applications présentées traite le temps de façon qualitative: dans le projet ARCHON comme dans le modèle ASIC ou le système d'ALI, l'attention est portée sur l'ordre des différentes actions à effectuer au niveau de chaque agent mais il n'y a pas de contrôle global de l'ensemble des actions effectuées dans ces différents systèmes à un instant donné. Les informations échangées sont d'ordre qualitatif, il n'y a pas de contraintes de temps. Dans le domaine d'intégration de logiciels à des fins de simulation, il devient nécessaire de tenir compte d'un état global du système distribué afin que le monde simulé soit dans un état cohérent.

Troisième partie

le système OSIRIS

Chapitre 7

La démarche de couplage d'OSIRIS : exemple pour l'intégration des dynamiques d'infiltration et de nappe

RÉSUMÉ :

Ce chapitre présente un cas exemple à partir duquel vont être illustrés les différents concepts proposés dans cette thèse. L'exemple consiste à coupler deux modèles : LEACHM (modèle d'infiltration de l'eau dans le sol) et MOC (modèle de dynamique de nappe) sur un système constitué de parcelles de périmètre, de profils de sol et d'une nappe.

7.1 Contexte de mise en œuvre

La dynamique saline dépend essentiellement des mouvements de l'eau contenue dans les zones saturées et non saturées de la région considérée. Ils déterminent la concentration des cations dans le sol. Ces mouvements sont eux mêmes conditionnés par les apports d'eau (dans le cas d'un environnement désertique, ils proviennent essentiellement de l'irrigation) et les sorties d'eau (évaporation et transpiration des plantes, drainage).

Les processus qui gouvernent les mouvements de l'eau à l'intérieur du sol la dynamique saline d'une région irriguée sont le processus infiltration de l'eau dans le sol et le processus de dynamique de la nappe. Ces processus sont formalisés à l'aide d'équations mathématiques résolues numériquement à l'aide de modèles informatiques, les processus d'irrigation et de drainage, quant à eux, dépendent étroitement des actions humaines et sont décrits de façon moins formalisée. Pour simuler la dynamique saline d'une région considérée, nous nous sommes attachés à considérer prioritairement les deux processus infiltration et dynamique de nappe.

Une caractéristique limitative des modèles numériques hydrologiques est le grand nombre de paramètres d'entrée qui doivent être fournis ou correctement estimés afin de pouvoir faire fonctionner le modèle [Rossiter, 1994; Shaffer, 1995; Belouze, 1996; HydrologicStudiesUnit, 2000]. L'Unité d'Etudes Hydrologiques du département de qualité environnementale de l'Etat du Michigan considère que l'application d'un modèle hydrologique sur une zone donnée n'est possible que si les informations minimales hydrologiques et géochimiques suivantes sont disponibles :

- les données caractérisant la géologie du sous-sol,
- les caractéristiques topographiques (y compris les hauteurs des cours d'eau),
- les mesures de la charge du plafond de la nappe,
- la conductivité hydraulique estimée à partir des mesures effectuées sur la nappe,
- la localisation des sources et des points de sortie d'eau ainsi que la mesure de leur flux,
- l'identification des éléments chimiques concernés par la modification du milieu considéré,
- les paramètres géochimiques appropriés (pH, etc).

Il arrive cependant que les données nécessaires au fonctionnement du modèle ne soient pas disponibles. C'est le cas de la région Ngalenka, où comme cela a été souligné au chapitre ??, les données concernant le site sont très complètes à l'échelle du profil de sol sur lequel s'effectuent les analyses pédologiques, alors que les données concernant la nappe sont pratiquement inexistantes, en raison du coût des mécanismes nécessaires à leurs caractérisations. La démarche classique de simplification des modèles utilisée par Belouze (extrapolation des fonctions produites au niveau d'un périmètre pour être appliquées à l'ensemble de la zone étudiée, chapitre ??) ou celle plus flexible d'application du modèle intégré sur chaque maille

de la région étudiée considérée sous forme de grille (figure 2.3) pour le modèle PLM se révèle alors inadéquat, pour appliquer un modèle intégré uniformisé.

L'approche adoptée dans le système OSIRIS consiste à s'inspirer des méthodes proposées pour le couplage de logiciels à l'aide d'agents afin de permettre l'application des modèles à intégrer uniquement sur les entités qui en subissent l'action. Le système étudié est considéré comme un ensemble d'éléments de différentes natures pouvant être en interaction. Ainsi dans le cas d'une étude de la dynamique saline d'une zone d'étude donnée où l'application d'un modèle 3D est impossible pour cause de manque de données, OSIRIS offre la possibilité d'appliquer des modèles 1D sur l'ensemble des profils sur lesquelles les données sont connues, connectés à un modèle 2D de dynamique de la nappe aux caractéristiques estimées sur seulement deux dimensions, donc moins exigeant en données. Le système considéré par OSIRIS est un ensemble de composants subissant l'action de processus différents, figure 7.1. L'intégration des modèles s'effectue donc par l'action qu'ils réalisent au niveau des composants du système auxquels ils sont rattachés.

Dans la suite de ce chapitre, nous présentons l'application du système OSIRIS au couplage des processus d'infiltration et de nappe, pour étudier l'évolution d'indicateurs géochimiques relatifs à la salinité d'un ensemble de périmètres irrigués. Ce couplage s'appuie sur deux modèles de ces processus présentés dans la section suivante.

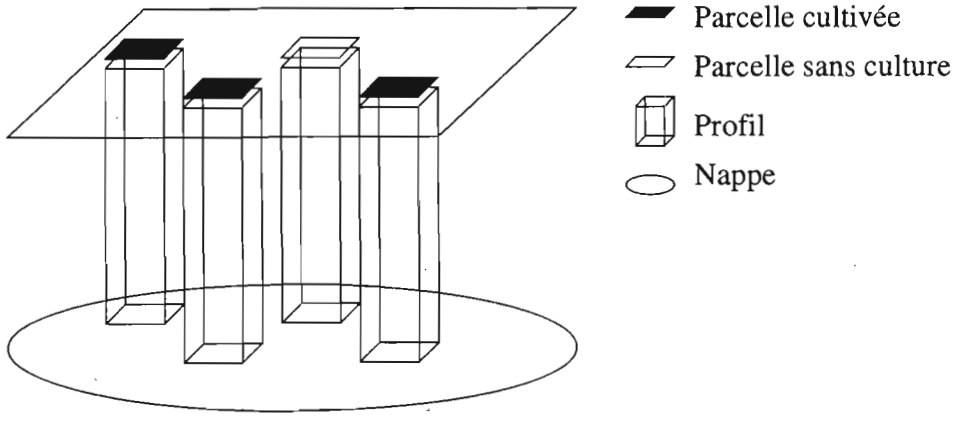


FIG. 7.1 – *Organisation spatiale du modèle OSIRIS appliqué à un problème d'étude de la dynamique saline d'une région irriguée. Les parcelles subissent l'action du processus irrigation qui diffère selon la culture réalisée. Les profils de sols vont subir l'action du processus infiltration conditionné par l'action du processus d'irrigation. Le processus de nappe subit l'influence des différents processus d'infiltration avec lesquels les différents profils de sol le mettent en contact.*

7.2 Présentation des deux modèles numériques

7.2.1 Le modèle d'infiltration LEACHM

LEACHM (Leaching Estimation and Chemistry Model) conçu à l'université de Cornell [Wagenet et Hutson, 1992] fait référence à un ensemble de modèles décrivant l'écoulement de l'eau et des solutés qu'elle transporte dans la zone de sol occupée par les racines.

LEACHM a été développé par John Hutson et Jeff Wagenet et a connu un nombre important d'extensions depuis 1984. LEACHM comprend quatre modèles de simulation et différents utilitaires. Les modèles de simulation utilisent différents shémas de résolution numérique pour simuler le mouvement vertical dans le sol de l'eau et des solutés. Ces shémas diffèrent par leur description de l'équilibre chimique, de la dynamique de transformation et de dégradation des solutés (absorption, volatilisation, etc). LEACHW décrit uniquement l'écoulement de l'eau. Les autres modèles de simulation décrivent en plus la dynamique des solutés :

- LEACHP simule l'écoulement des pesticides,
- LEACHN simule l'écoulement du phosphore et du nitrate,
- LEACHC simule la dynamique saline des sols; c'est cette version que nous avons utilisée.

Ces modèles ont bénéficié de nombreuses améliorations donnant naissance à des codes de noms variés (LEACHNA, LEACHNR, LEACHF, LEACHG [Hutson et al., 1997]).
L'écoulement de l'eau et des flux est décrit par l'équation de Richard et les équations de convection-dispersion résolues numériquement (équations aux dérivées partielles [différences finies, 1D], équations différentielles ordinaires).

En entrée, LEACHM requiert différents paramètres de sol (propriétés physiques du sol, ses caractéristiques de rétention d'eau). Il est également nécessaire de fournir des estimations ou les données concernant les apports en eau (par la surface et par la nappe), la température journalière maximale et minimale.
En sortie, sur un profil de sol considéré, divisé en couches, le modèle calcule les teneurs en eau de chaque couche qui vont être utilisées pour résoudre les équations de convection-dispersion qui décrivent le mouvement des solutés (figure 7.2) Le modèle LEACHM est valide en zone non saturée.

7.2.1.1 Gestion du temps

Le modèle LEACHC fonctionne avec des pas de temps dynamiques, en fonction de la quantité d'eau apportée dans le modèle et des propriétés du sol (conductivité hydraulique) influençant la vitesse d'écoulement de l'eau dans le profil.
L'utilisateur donne en entrée une période de simulation (en jours) durant laquelle va fonctionner le modèle, ainsi que la quantité maximale d'eau qu'il souhaite voir se déplacer (le flux

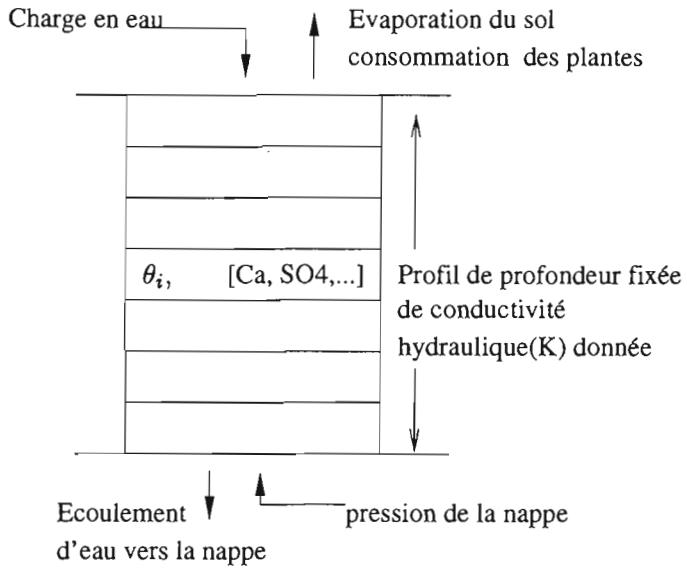


FIG. 7.2 – *Application du modèle LEACHC. Etant donnés une charge en eau de départ, une conductivité hydraulique caractérisant le sol, un calendrier de consommation d'eau suivant la culture présente sur la surface du profil considéré et une pression de départ exercée par la nappe, LEACHC calcule pour chaque couche du profil les teneurs en eau θ_i ainsi que les concentrations des différents éléments chimiques du sol.*

d'eau en mm/mm (hauteur d'eau/hauteur de sol)) pour un pas de temps donné. Le nombre d'itérations de fonctionnement du modèle va dépendre de la quantité d'eau déplacée (le flux d'eau) calculée à chaque pas de temps. Si ce flux s'approche de la limite donnée par l'utilisateur, le pas de temps va être raccourci afin que la prochaine valeur de flux calculée reste dans les limites souhaitées par l'utilisateur.

7.2.1.2 Informations d'entrée sortie

LEACHC fonctionne en mode "batch" à partir d'un fichier d'entrée et génère un fichier de sortie pouvant contenir en fonction des options choisies dans le fichier d'entrée, les détails transitoires des calculs. Les informations manipulées par LEACHC sont décrites dans les paragraphes suivants.

En entrée. Les informations nécessaires à LEACHC contenues dans le fichier d'entrée sont résumées par les tableaux 7.1, 7.2, 7.3 et 7.4.

Types	Nom	Unité	Commentaire
Paramètres de sol	Profondeur profil	mm	Profondeur de la zone étudiée
	Epaisseur d'un segment	mm	A partir de la profondeur du profil et d'épaisseur segment, LEACHM détermine le nombre de couches
	Hauteur de nappe	mm	Cette information fait partie de l'ensemble des informations de conditions aux limites
Propriétés physiques du sol			
(pour chaque couche)	Composition en argile, vase et carbone organique	%	Composition du sol
	densité des particules d'argile, de vase et de carbone organique	kg/dm^3	Ces valeurs sont utilisées pour calculer le coefficient de porosité effective du sol
	θ ou potentiel du sol	m^3/m^3 ou kPa	Teneur en eau initiale ou capacité de rétention d'eau du sol
	racines	%	Pourcentage d'espace occupé par les racines
	Densité apparente	kg/dm^3	
	K	mm/d	conductivité hydraulique
	dispersivité	mm	paramètre de dispersivité du sol

TAB. 7.1 – *Les caractéristiques du sol en entrée pour LEACHC*

Types	Nom	Unité	Commentaire
Cultures	nombre de cultures		
Pour chaque culture	date de germination		
	date d'émergence		
	date de maturité		
	fraction de couverture	%	proportion d'espace recouvert par la culture
	Pan factor		
	Annual N Uptake	kg/ha	

TAB. 7.2 – *Les caractéristiques des cultures en entrée pour LEACHC*

Types	Nom	Unité	Commentaire
Cations et anions (pour chaque couche)	Ca	$mmol/l$	concentration en calcium
	Mg	$mmol/l$	concentration en magnesium
	Na	$mmol/l$	concentration en sodium
	K	$mmol/l$	concentration en potassium
	Cl	$mmol/l$	concentration en chlorure
	SO4	$mmol/l$	concentration en sulfate
Concentrations (Na, Mg, K, Cl, SO4) sous le profil de sol			Ces concentrations font partie des informations de conditions aux limites
Paramètres pour procédures d'équilibre chimique			
	paramètre de diffusion		
	paramètres d'ajustement		
Composition de l'eau apportée (pluie ou irrigation)			
Pour chaque apport	date de l'apport		
	quantité apportée	mm	
	densité du flux de surface	mm/d	
	concentrations en anions et cations	$mmol/l$	Ca, Mg, Na, K, Cl, SO4

TAB. 7.3 – *Propriétés chimiques du profil de sol en entrée pour LEACHC*

Types	Nom	Unité	Commentaire
Informations relatives au climat			
	Evapotranspiration potentielle	mm	par semaine
	Température moyenne	degre C	
	Amplitude moyenne	degre C	L'amplitude et la température sont également des données hebdomadaires

TAB. 7.4 – *Propriétés climatiques du sol*

En sortie. Les informations en sortie données par LEACHC concernent la teneur en eau du profil détaillée pour chaque couche et la teneur en cations pour chaque couche ainsi que les différentes quantités d'eau et de cations échangés avec l'extérieur. Ces informations sont résumées dans le tableau 7.5.

Types	Nom	Unité	Commentaire
Prédiction sur la rétention et la conductivité hydraulique du sol			
Pour chaque couche	paramètre de rétention	kPa	Caractérise la qualité de rétention en eau du sol
	θ	m^3/m^3	la teneur en eau volumique est calculée pour différentes valeurs du paramètre de rétention du sol et pour le paramètre de rétention du sol obtenu par application de la résolution de l'équation de Richards
Totaux cumulés et bilan de masse initiaux et finaux après calcul des pertes(par drainage, par consommation des plantes, par absorption du sol (calcite/gypse) et calcul des apports (par infiltration, pluie, provenant du sol (calcite/gypse))pour :			
	eau	mm	
	corps (SO4, HCO3, CO3, Mg, Na, K, Cl)	$mmol/m^2$	
Propriétés d'échange des ions compte tenu de la présence de culture ; en fonction de la proportion du volume occupé par les racines			
Pour chaque couche	ions dissous	me/l	pour les cations Ca, Mg, Na, K
	ions apparus	me/kg	pour les cations Ca, Mg, Na, K
	ESP	%	Taux de sodium en solution échangeable (Exchangeable Sodium Percentage)
	SAR		fixation du sodium sur les argiles (Sodium Absorption Ratio)

TAB. 7.5 – *Données en sortie de LEACHC*

7.2.1.3 Conditions d'applicabilité du modèle

Lorsque les apports d'eau conduisent à une saturation du profil, les valeurs calculées par LEACHM deviennent incohérentes. En effet, l'équation de Richards utilisée n'est valable que pour les environnements non saturés, lorsque l'on s'intéresse à la teneur en eau.

7.2.1.4 Description technique

LEACHM a été développé avec le langage FORTRAN 77 et est disponible en utilisation libre sous forme d'exécutable pour le système d'exploitation DOS. Nous avons utilisé la version LEACHC du modèle.

7.2.2 Le modèle de nappe MOC

MOC (Method Of Characteristics ground-water flow and transport model) est un modèle numérique de simulation, conçu à l'US Geological Survey, relatif au transport de solutés et à la dynamique de flux dans une nappe [Goode et Konikow, 1989; Konikow et al., 1994]. Le modèle calcule les changements de concentration de solutés causés par les processus d'advection (déplacement suivant la vitesse moyenne de la nappe), de dispersion (diffusion locale due à l'agitation microscopique de la nappe), de mélange (modification de la composition en solutés) ou de dilution (apport en solutés) déclenchés par des apports et des pertes d'eau.

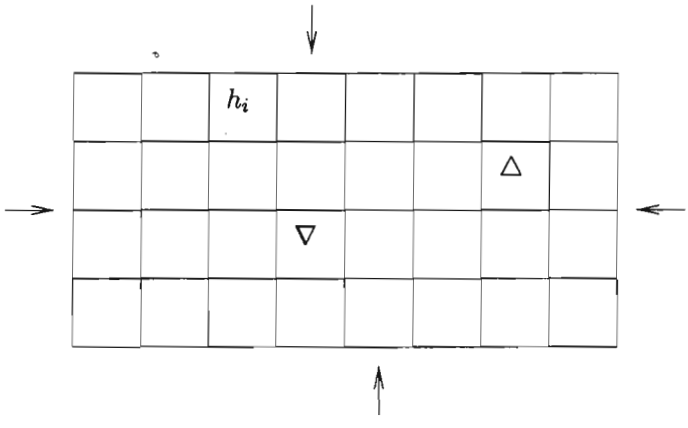


FIG. 7.3 – *Représentation spatiale d'une nappe considérée par le modèle MOC. Le modèle considère la nappe comme une grille composée de mailles uniformes (les nœuds). Pour chaque maille, il calcule la hauteur de nappe h_i ainsi que les concentrations des différents éléments chimiques. Les conditions aux limites des différentes frontières de la nappe doivent être précisées. Les apports en eau sont effectués par la présence de sources(▽) localisées sur une maille. Les pertes d'eau s'effectuent soit par les frontières (conditions au limites) soit par des puits (△) localisés sur des mailles.*

Le modèle couple une équation décrivant la dynamique d'écoulement de la nappe avec une équation de dynamique du transport de solutés. Le programme utilise une méthode itérative pour résoudre une approximation aux différences finies de l'équation décrivant l'écoulement d'eau. L'équation de transport de solutés est résolue selon la méthode des caratéristiques qui donne son nom au modèle. Une procédure de suivi des particules (*particle-tracking*) modélise

le déplacement des solutés dû au processus d'advection. Une procédure aux différences finies calcule en deux étapes les effets du processus de dispersion hydrodynamique et du processus d'apport en eau.

En entrée, MOC requiert les conditions aux limites de la nappe, les propriétés de la nappe à chacun de ses nœuds (concentrations initiales des solutés, hauteur de nappe) et les descripteurs de la grille (nombre de nœuds, localisation des puits et des sources).

En sortie, MOC calcule pour chaque nœud les nouvelles hauteurs de nappe, les concentrations dues aux différents processus de dynamique de nappe, ainsi que les caractéristiques de la dynamique de la nappe (vitesse des processus).

7.2.2.1 Gestion du temps

Le modèle MOC fonctionne avec des pas de temps fixes. L'utilisateur choisit en entrée une période de simulation en années et peut également spécifier le nombre d'étapes transitoires pour lesquelles il souhaite observer les résultats. Le modèle MOC étant un modèle lourd qui effectue de nombreux calculs transitoires pour un pas de temps donné, on se contente en général de demander un seul état final en sortie.

7.2.2.2 Informations d'entrée sortie

MOC fonctionne en mode batch. A partir d'un fichier d'entrée (.dat) MOC génère un fichier de sortie (.out) qui peut, en fonction des options contenues dans le fichier d'entrée, être accompagné de fichiers décrivant certaines parties du résultat final (on peut obtenir de façon séparée les concentrations, les hauteurs de nappe, etc).

Contrairement au modèle LEACHC qui manipule les quantités particulières (concentration en $mmol/l$) d'ions particuliers (K, Na, Ca, Mg), MOC manipule des solutés génériques à l'aide de différents paramètres qui permettent de les caractériser (nombre de particules, caractéristiques de ces particules). Les différentes grandeurs n'ont pas d'unité fixée, elles doivent cependant respecter des dimensions (Dim.) données grandeurs de longueur (L), de masse (M), de temps (T).

En entrée. Les informations nécessaires au modèle MOC sont présentées dans le tableau 7.6

Types	Nom	Dim	Commentaire
Description de la grille représentant la nappe			
	nombre de lignes		
	nombre de colonnes		
	taille d'une ligne	L	
	taille d'une colonne	L	
Pour chaque noeud	hauteur initiale de la nappe	L	
	transmissivité (leakance)	L^2/s	
	conductivité hydraulique	L/s	
Description d'une sous grille pour l'application des procédures de transport de particules			
	nombre de lignes		
	nombre de colonnes		
	numéro premier nœud	(L,L)	
	numéro dernier nœud	(L,L)	
Paramètres hydrologiques et chimiques			
	porosité effective		coefficient positif inférieur à 1
	β		coefficient de dispersivité longitudinale
Paramètres utilisés pour les procédures de calcul des équations d'absorption et de réaction			
	masse volumique apparente	M/L^3	
	coefficient de distribution		
	facteur de retard		
Paramètres d'exécution du modèle			
	nombre de particules par noeuds		

TAB. 7.6 – *Informations en entrée pour le modèle MOC*

En sortie. Les informations données en sortie par le modèle MOC concernent les concentrations des solutés calculés pour chaque noeud ainsi que les caractéristiques hydrologiques calculés de la nappe (Tableau 7.7).

Types	Nom	Dim	Commentaire
pour chaque nœud	hauteur de nappe	L/s	
	vitesse	L	suivant deux axes (X, Y)
	concentration	M/L^3	calculés pour les nœuds de la sous grille
	bilan de masse cumulé en eau	L^3	apport en eau (recharge et injection); retraits dus au pompage et à l'évapotranspiration; quantité d'eau restante ; fuites d'eau
	bilan de masse chimique	M	masse restante; masse perdue (pompage, absorption par corps solides, masse dissoute)
Les calculs transitoires ainsi que les résultats ci-dessus peuvent être obtenus pour un ou plusieurs nœuds spécifiés dans le fichier d'entrée			

TAB. 7.7 – *Informations fournies par le modèle MOC*

7.2.2.3 Conditions d'applicabilité

Les mouvements de nappe sont considérés comme étant significatifs à l'échelle de l'année (sur un temps long). Si l'on souhaite estimer les mouvements de la nappe à une périodicité plus raprochée (suite à d'importants apports en eau provenant des profils par exemple), il est plus judicieux d'utiliser un modèle simplifié, moins gourmand en calcul.

7.2.2.4 Description technique

MOC a été écrit en FORTRAN 77. Le code source est disponible et peut être compilé sur toutes les plates-formes. Cependant sur un environnement PC, il nécessite pour fonctionner la présence d'un coprocesseur arithmétique. La version de MOC que nous avons récupérée est un exécutable compilé pour l'environnement Solaris.

7.3 La démarche de couplage d'OSIRIS pour coupler les processus

Avec le système OSIRIS, le couplage de modèles s'effectue par l'intermédiaire d'un système pivot sur lequel vont agir les différents processus. La première étape d'un couplage utilisant l'outil OSIRIS consiste donc à définir les différents éléments de ce système pivot.
Une fois les entités spécifiées, les modèles numériques peuvent leur être associés. Cette association est réalisée à travers la définition des processus, qui vont agir sur les entités du système

(les entités cibles).

7.3.1 Construction d'un système pivot

La construction d'un système pivot permet d'obtenir un système intégré indépendant des points de vue (paramètres et unités manipulés) présentés par les modèles de départ. L'objectif poursuivi est d'éviter de privilégier une des représentations par rapport aux autres et de permettre à l'utilisateur de représenter le système avec le point de vue qui lui est propre.

Le système pivot de l'architecture OSIRIS est un système non uniformisé: ce système est décrit sous la forme de différentes entités sous l'action de divers processus, décrits par des modèles numériques.

Dans le cadre d'une étude de la salinité d'une région irriguée, il est naturel de représenter les différentes zones de sol concernées par ce phénomène, il faut également inclure dans le système l'ensemble des informations qui influencent les différents processus en jeu. Les entités prises en compte ont été déterminées sur la base d'une étude préalable concernant les effets des opérations hydro-agricoles sur le milieu physique de la région du Ngalenka [Fianyo et al., 1998a]. Le système irrigué est représenté par les entités décrites dans les paragraphes suivants.

7.3.1.1 Les entités spatiales

Les entités spatiales sont les entités principales sur lesquelles agissent les phénomènes de salinisation. Elles peuvent être classées en trois types : les entités de surface, les entites de sol (zone non saturée) et les entités de sous-sol (zone saturée).

La nappe. Il est supposé l'existence d'une seule nappe phréatique continue dans le sous-sol de la région. Elle est caractérisée par une étendue et une fonction associant à chaque point de cette étendue une profondeur (plafond de nappe, également appelé hauteur de nappe) et une épaisseur (permettant de déterminer le plancher de la nappe). Ces propriétés physiques sont complétées par des caractéristiques chimiques (composition ionique pour chaque point).

Le profil de sol. Le profil de sol est caractérisé par un type de sol (en fonction de sa composition en argile, on distingue les sols peu argileux - moins de 17% d'argile, des sols argileux - de 45 à 90% d'argile). Il est décrit par un ensemble de couches caractérisées chacune par une conductivité hydraulique, une teneur en eau et des propriétés chimiques (pH, compositions en ions, taux de sodium echangeable).

La parcelle de périmètre. Un périmètre est composé d'un ensemble de parcelles, sur lesquelles sont cultivées différentes espèces de cultures. Un périmètre (ensemble de parcelles) est associé à un groupe MotoPompe qui assure son approvisionnement en eau. Dans un premier temps, le processus d'approvisionnement en eau peut être modélisé par le choix de la culture associée à la parcelle.

Le cours d'eau. Le marigot Ngalenka est la ressource en eau principale du système. Un cours d'eau est caractérisé par son débit, sa hauteur d'eau, caractéristiques données pour tous les points du parcours du cours d'eau.

Les entités spatiales de surface peuvent être représentés comme le montre la figure 7.4.

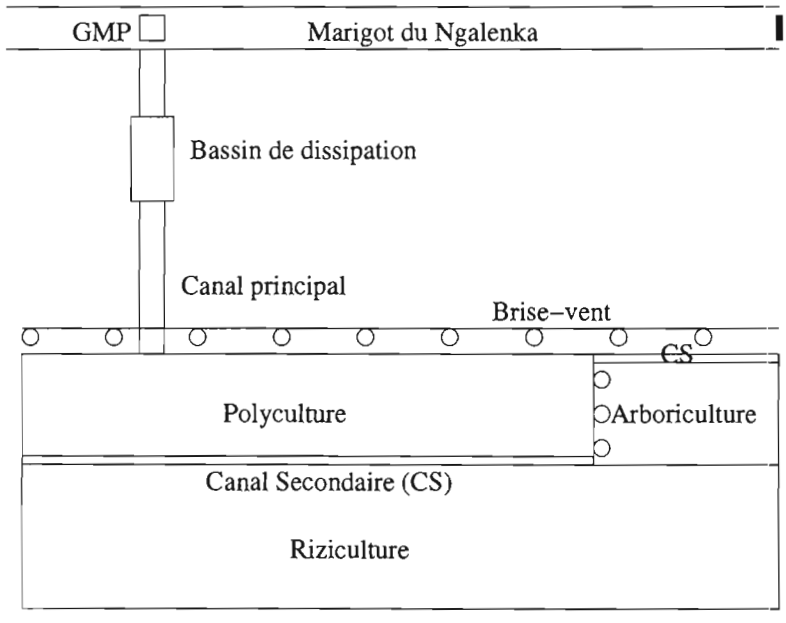


FIG. 7.4 – *Plan type d'un PIV(Périmètre Irrigué Villageois). Un système d'irrigation dans un PIV consiste principalement en : (i) un groupe moto-pompe(GMP) posé sur un bac flottant, (ii) un bassin de dissipation raccordé au GMP par une tuyauterie flexible, (iii) un réseau de canalisations sommaires qui partent du bassin. d'après [Senghor, 1997a].*

7.3.1.2 Les entités de service

Les entités de service capturent toutes les informations relatives au contexte influençant les phénomènes qui se déroulent dans le système. Elles peuvent être classées en deux types : les entités relatives au contexte socio-économique (les activés humaines) et les entités relatives au contexte physique.

Culture. La culture permet de déterminer l'apport en eau d'une parcelle à laquelle elle est associée. Une culture est accompagnée d'un calendrier cultural qui précise pour les différentes périodes de croissance de la plante considérée (durée de germinaison, date d'émergence, date de maturité) la quantité de consommation en eau. Elle dispose également de la propriété qui détermine sa surface occupée (taux de remplissage de la surface).

Type de sol. Le type de sol permet de déterminer la composition (en argile, vase et matière organique) du sol, ses qualités physiques (quelle conductivité hydraulique à saturation par exemple) et diverses autres caractéristiques (évaporation du sol - ETP).

Contexte physique. Cette entité permet de disposer de l'ensemble des informations relatives au contexte climatique (pluies, températures) considéré. Elle est composée d'un ensemble de tables, qui déterminent de façon journalière l'apport en eau du à la pluie (les précipitations) et la température de la région. Ces tables ont été établies à partir d'études menées sur de longues périodes de temps (les premières mesures climatiques son basées sur des données remontant au milieu du 20ème siècle [Lericollais et Sarr, 1995]). La température détermine les pertes en eau dues à l'évaporation du sol, à partir de fonctions (par exemple l'équation de Penman pour l'évapotranspiration). Ces fonctions modélisent les processus à l'aide de paramètres relatifs au sol et aux cultures.

7.3.1.3 Les entités actives

Les entités actives sont relatives aux composants du système disposant en principe d'une autonomie de fonctionnement et dont les actions modifient l'évolution de l'ensemble. Leur description se limite dans cet exemple à présenter les différentes propriétés qui les caractérisent.

le GIE (groupement d'intérêt économique). Le GIE est composé d'un ensemble d'exploitants qui gèrent le périmètre irrigué. Le GIE peut contraindre ses exploitants à effectuer certains choix culturaux sur leurs parcelles. C'est également le GIE qui a la responsabilité de maintenir les installations permettant l'apport en eau par irrigation du système. Selon la manière dont cette gestion s'effectue (mauvais approvisionnement en essence de la moto-pompe du périmètre par exemple) l'apport en eau nécessaire aux différentes parcelles sera ou non possible. Dans cet exemple, nous considérons que la gestion des périmètres par le GIE est idéale, l'apport en eau des parcelles est toujours possible.

L'exploitant. L'exploitant possède une ou plusieurs parcelles qu'il a la responsabilité de gérer. Cette gestion inclut le choix de la culture sur sa parcelle, les apports en eau nécessaires à la culture qu'il a choisie de mettre en place, le draînage de la parcelle (opération de retrait

d'eau) quand cela s'avère nécessaire. Le gestionnaire de la parcelle peut cependant ne pas procéder aux différentes opérations nécessaires (approvisionnement insuffisant de sa parcelle, drainage insuffisant), ce qui influence la salinisation de la région. Dans cet exemple, nous considérons que les actions de l'exploitant sont conformes aux attentes des cultures (l'apport en eau de la parcelle correspond à l'apport en eau nécessaire à la culture choisie).

7.3.2 Spécification des interfaces de processus

Une fois le système représenté à l'aide des différentes entités, l'étape suivante de la démarche de couplage de l'outil OSIRIS consiste à spécifier les interfaces de processus. La spécification d'une interface de processus permet de définir un processus faisant évoluer l'état d'une ou plusieurs entités du système ainsi que ses différentes caractéristiques : quelles sont les entités sur lesquelles il agit ou dont l'état interne l'influence (les entités cibles); quels sont ses différents comportements possibles (les modèles numériques qui permettent de le décrire et les conditions d'applications de ces modèles); quel est son rythme d'action. Dans cet exemple, les processus qui doivent être intégrés sont les processus de dynamique d'écoulement de l'eau dans le sol (le processus d'infiltration) et dans la nappe (le processus de nappe). On va donc spécifier deux interfaces de processus, infiltration et nappe.

7.3.2.1 Définition de l'interface processus infiltration

(a) Les entités cible

L'infiltration représente la dynamique d'écoulement de l'eau dans le sol, l'entité principalement ciblée par ce processus est donc l'entité profil de sol. L'action du processus influence également l'état de la nappe située sous le profil (l'eau qui s'écoule dans le profil peut finir par sortir du profil en direction de la nappe). Enfin, les apports en eau de surface résultent de l'irrigation de la parcelle en surface pour permettre la croissance de la culture qui a été mise en place. Ce processus est donc influencé par l'état de la parcelle sous laquelle le profil de sol sur lequel il agit est situé.
Les entités cible du processus infiltration sont donc le profil de sol, le périmètre et la nappe, avec le profil de sol en entité cible principale.

(b) Les modèles de comportement.

Le modèle LEACHM. Le modèle décrivant l'action du processus d'infiltration est le modèle LEACHM. Ce modèle fonctionne avec les différents paramètres décrits dans les tableaux 7.1, 7.2, 7.3 et 7.4. Ces paramètres doivent être reliés aux attributs des entités cible. Il n'y a pas toujours bijection entre un paramètre et l'attribut d'une entité. Par exemple, le paramètre de composition du sol en argile, vase et carbone organique doit être donné pour chaque couche

du profil de sol alors qu'au niveau de l'entité profil de sol cette donnée existe au niveau de tout le profil (attribut type de sol). Pour permettre le fonctionnement du modèle, le paramètre composition du sol pour chaque couche va être déduit de l'attribut type de sol par application de la valeur globale du profil au niveau de chaque couche.
LEACHM permet de calculer de nouvelles teneur en eau et concentrations de corps ioniques dans le profil de sol et également dans la nappe : une certaine quantité d'eau et de moles sont perdues vers la nappe. A chaque modification de la hauteur de nappe calculée par LEACHM, le processus de nappe va en être informé afin d'en tenir compte.

Paramètre	**Entité.attribut**
teneur en eau[ ]	parcelle.apportEnEau, profilDeSol.teneur en eau[ ]
concentration[ ]	profilDeSol.concentration[ ]
composition en argile[ ]	profilDeSol.typeDeSol.composition en argile
conductivité hydraulique[ ]	profilDeSol.conductivité hydraulique
perteVersNappe	profilDeSol.perteVersNappe

TAB. 7.8 – *Correspondances entre paramètres d'entrée de LEACHM et attributs d'entités du système*

Paramètre	**Entité.attribut**
teneur en eau[ ]	nappe.teneur en eau, profilDeSol.teneur en eau[ ]
concentration[ ]	nappe.concentration, profilDeSol.concentration[ ]
conductivité hydraulique[ ]	profilDeSol.conductivité hydraulique

TAB. 7.9 – *Correspondances entre paramètres de sortie de LEACHM et attributs d'entités cible du système*

Le modèle d'infiltration à bilan. L'action du processus infiltration peut être modélisée de façon plus "grossière" par un modèle à bilan basé sur l'application d'une équation simple modifiant les quantités d'eau et d'ions contenus dans les différentes couches du profil en fonction des paramètres de température et de transpiration du sol. Ce modèle peut être appliqué sur un profil d'état saturé.

Paramètre	Entité.attribut
teneur en eau[ ]	parcelle.apportEnEau, profilDeSol.teneur en eau[ ]
concentration[ ]	profilDeSol.concentration[ ]
composition en argile[ ]	profilDeSol.typeDeSol.composition en argile
conductivité hydraulique[ ]	profilDeSol.conductivité hydraulique
ETP	contextePhysique.ETP

TAB. 7.10 – *Correspondances entre paramètres d'entrée du modèle à bilan d'infiltration et attributs d'entités cible*

Paramètre	Entité.attribut
teneur en eau[ ]	nappe.teneur en eau, profilDeSol.teneur en eau[ ]
concentration[ ]	nappe.concentration, profilDeSol.concentration[ ]
conductivité hydraulique[ ]	profilDeSol.conductivité hydraulique

TAB. 7.11 – *Correspondances entre paramètres de sortie du modèle à bilan d'infiltration et attributs d'entités cible du système*

(c) Les pas de temps (ou rythme) du processus

Le pas de temps du processus va dépendre du caractère significatif de ses actions sur le système. Lorsque le profil est non saturé, l'action du processus est rapide et on peut décider de choisir un pas de temps court (15 jours par exemple). Quand le profil de sol est saturé, l'eau apportée a tendance à rester captive et à ne plus s'écouler, le pas de temps peut être ralenti.

(d) Le choix d'un modèle d'application courant

A un moment donné de la simulation, le processus d'infiltration est représenté soit par le fonctionnement du modèle LEACHM, soit par le fonctionnement du modèle d'infiltration à bilan. Le choix du modèle courant est effectué sur la valeur de la teneur en eau de la première couche (en partant de la surface) du profil. Lorsque celle-ci se rapproche de la teneur en eau à saturation ($K \approx K_s$), alors le modèle LEACHM n'étant plus applicable, le modèle courant doit changer ainsi que le pas de temps du processus.

7.3.2.2 Définition de l'interface processus nappe

(a) Les entités cible

La dynamique de la nappe modélisée par le processus de même nom manipule en priorité

l'entité nappe. Les apports en eau de la nappe proviennent des profils situés à sa frontière supérieure, qui peuvent être considérés comme des sources pour la nappe.
Les entités cible du processus nappe sont donc le profil de sol et la nappe, avec la nappe en entité cible principale. Contrairement au processus d'infiltration qui est associé à *une seule* entité de chaque type, le processus nappe fonctionne avec des données provenant de *plusieurs* entités de type profil de sol pour une nappe.

(b) Les modèles de comportement.

Le modèle MOC. Le modèle principal décrivant l'action du processus de nappe est le modèle MOC. Ce modèle fonctionne avec les différents paramètres décrits dans le tableau 7.6. Ces paramètres vont être reliés aux attributs des entités cible.
MOC calcule de nouvelles teneurs en eau et concentrations de corps ioniques dans la nappe en fonction des apports provenant des profils. Les profils peuvent donc être considérés comme des sources pour la nappe. Le processus nappe prend également connaissance du résultat du calcul de la hauteur de nappe par le processus d'infiltration, le compare avec son propre calcul afin d'estimer la marge d'erreur du calcul de cette valeur.

Paramètre	**Entité.attribut**
hauteur initiale[ ][ ]	nappe.hauteur[ ][ ]
conductivité hydraulique[ ][ ]	nappe.conductivité hydraulique[ ][ ]
teneur en eau[ ][ ]	profilDeSol[ ].perteVersNappe,
	nappe.teneur en eau[ ][ ]
β	nappe.concentration[ ][ ]

TAB. 7.12 – *Correspondances entre paramètres d'entrée de MOC et attributs d'entités du système*

Paramètre	**Entité.attribut**
hauteur de nappe[ ][ ]	nappe.hauteur[ ][ ]
bilan cumulé en eau	nappe.teneur en eau[ ][ ],
	profilDeSol.teneur en eau[ ]
bilan de masse chimique	nappe.concentration[ ][ ],
	profilDeSol.concentration[ ]
vitesse	nappe.conductivité hydraulique[ ][ ]

TAB. 7.13 – *Correspondances entre paramètres de sortie de MOC et attributs d'entités cible du système*

Le modèle de nappe simplifié. L'action du processus nappe peut être modélisée de façon plus simplifiée par un modèle basé sur l'application d'une équation simple de répartition des apports d'eau et d'ions d'un noeud de la nappe à ses noeuds voisin.

Paramètre	Entité.attribut
hauteur de nappe[ ][ ]	nappe.hauteur[ ][ ]
teneur en eau[ ][ ]	profilDeSol[ ].perteVersNappe, nappe.teneur en eau[ ][ ]
concentration[ ][ ]	nappe.concentration[ ][ ], profilDeSol.concentration[ ]
conduc. hydraulique[ ][ ]	nappe.conductivité hydraulique[ ][ ]

TAB. 7.14 – *Correspondance entre paramètres d'entrée du modèle simplifié de nappe et attributs d'entités cible*

Paramètre	Entité.attribut
teneur en eau[ ][ ]	nappe.teneur en eau[ ][ ], profilDeSol[ ].perteVersNappe
concentration[ ][ ]	nappe.concentration[ ][ ],
conductivité hydraulique[ ][ ]	nappe.conductivité hydraulique[ ][ ]

TAB. 7.15 – *Correspondances entre paramètres de sortie du modèle simplifié de nappe et attributs d'entités cible du système*

(c) Les pas de temps (ou rythme) du processus

Le pas de temps du processus va également dépendre la vitesse de modification qu'il effectue sur le système. Si l'on souhaite observer une dynamique de nappe rapide, le pas de temps approprié va être le mois. Le pas de temps moyen de la nappe est l'année.

(d) Le choix d'un modèle d'application courant

Le choix du modèle courant de simulation est basé sur la périodicité à laquelle on souhaite voir évoluer la dynamique. En cas d'importants apports d'eau augmentant significativement la teneur en eau globale de la nappe, il faut adopter un modèle qui donne l'évolution de façon plus rapide.

7.3.3 Exécution du modèle intégré

La troisième étape de la démarche de couplage dans l'outil OSIRIS est l'exécution du modèle intégré. Les précédentes étapes ont permis de spécifier de façon générale les types d'entités et

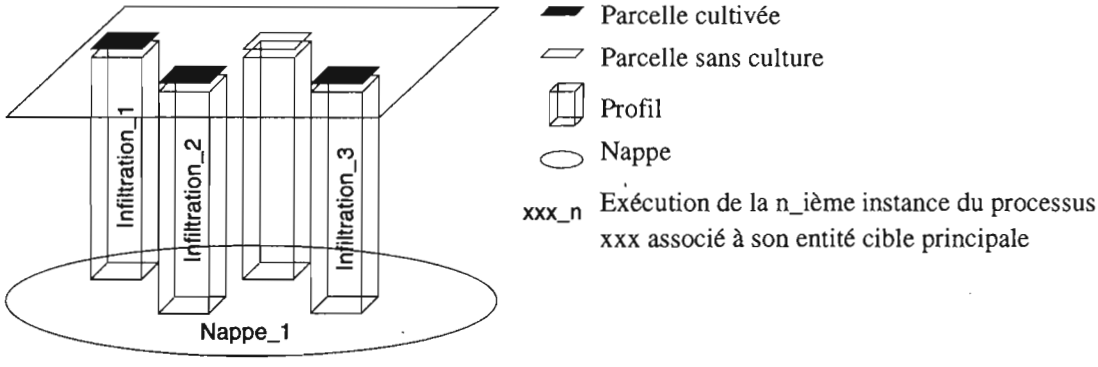


FIG. 7.5 – *L'exécution du modèle intégré est définie comme l'exécution des différents modèles numériques courants des processus associés aux différentes entités du système.*

les types de processus qui représentent leur dynamique. Pour lancer la simulation et mettre en marche les dynamiques, il faut instancier les entités et associer sur chacune d'elles les différents processus. Si l'on considère la figure 7.1 comme une instanciation des différentes entités définies (un système intégré particulier), alors l'exécution du modèle intégré peut être représenté comme le montre la figure 7.5.

7.4 Les outils nécessaires à la démarche de couplage OSIRIS

L'application de la démarche d'intégration présentée dans la section précédente nécessite de surmonter les difficultés introduites par trois types d'hétérogénéité :

L'hétérogénéité temporelle. Il faut pouvoir changer dynamiquement (au cours de la simulation) le pas de temps des différents processus : le processus infiltration peut évoluer à pas de temps jours ou à pas de temps mois. Il faut pouvoir faire coexister différents pas de temps : les processus infiltration et dynamique de nappe n'évoluent pas aux mêmes pas de temps.

L'hétérogénéité des représentations. Les modèles utilisent des représentations (les paramètres et leur unités de mesure) pas toujours utilisables sans transformation préalable. Pour appliquer la méthode OSIRIS, il faut pouvoir traduire les informations vers les modèles et vers les entités dans des systèmes de représentations qu'ils peuvent manipuler.

L'hétérogénéité comportementale. Il faut mettre en place le mécanisme permettant à un processus de changer de modèle d'exécution pendant la simulation.

Les concepts et l'architecture proposés pour résoudre ces difficultés sont présentés dans le chapitre suivant.

Conclusion

Dans ce chapitre, la présentation d'un exemple de couplage concernant deux processus a permis d'introduire des notions (système de représentation pivot, entités, interface de processus, pas de temps, modèle d'application) qui vont être explorées de façon plus formelle dans les chapitres suivants. L'exemple illustre notamment la structure des relations entre processus et entités : l'action d'un processus donné peut se réaliser sur plusieurs entités appartenant à des types différents (c'est le cas du processus infiltration qui agit sur une nappe, un profil et une parcelle), ou sur plusieurs entités appartenant au même type (c'est le cas de la nappe qui agit sur plusieurs profils de sol).

Chapitre 8

Des concepts pour maîtriser l'hétérogénéité en modélisation

RÉSUMÉ :

Les concepts proposés pour le couplage de modèles concernent la gestion du temps de simulation (comment prendre en compte de multiples pas de temps variables dans un système de simulation), la possibilité de faire correspondre différentes représentations concernant la même réalité, le contrôle dynamique du comportement des processus au cours de la simulation. Chacun de ces concepts est présenté de façon générale, l'objectif du chapitre étant de les comparer à d'autres concepts proposés pour résoudre des problèmes similaires.

OSIRIS (Outil de Simulation et d'Intégration des processus physiques pour l'étude d'une Région Irriguée au Sénégal) est un simulateur (il a donc pour rôle d'exécuter un modèle dans le temps) mais son modèle d'exécution est le résultat de l'intégration, du couplage de différents modèles déjà existants, des entités informatiques qui peuvent se situer sur des environnements hétérogènes.

OSIRIS est le support de la réflexion menée sur la définition de concepts informatiques intéressants pour le modélisateur dans une entreprise de couplages de modèles. L'hétérogénéité est une question traitée dans le domaine de la simulation. Un des principaux objets de la recherche en simulation distribuée (PDES : Parallel Discrete Event Simulation [Fujimoto, 1990; Banks, 1998; Cassandras et Lafortune, 1999]) est la détermination d'architectures permettant l'exécution distribuée de programmes de simulation et leur validation formelle [Haddad, 1993; Haddad et al., 2000]. Cependant Fujimoto note dans un article [Fujimoto, 1993] que les concepts définis ne sont pas adoptés par les modélisateurs à cause de la complexité de leur mise en œuvre.

Par rapport à ces travaux, OSIRIS se démarque par la volonté de rester proche du modélisateur (appartenant, dans le cas de la thèse, au monde de l'hydrologie) en proposant des concepts facilement manipulables au travers d'une interface visuelle. un des principaux obstacles auquel se heurte le modélisateur lors de sa démarche d'intégration, comme le montre le travail de Belouze présenté dans la première partie de ce document, est la mise à l'échelle de différents temps de simulation. Une autre difficulté importante pour le modélisateur intégrateur réside dans la mise en correspondance des différentes transcriptions, des différentes représentations d'une même réalité différemment considérée selon les modèles préliminaires. Enfin, comme exposé dans le chapitre 2, il peut exister pour un même problème différents modèles présentant des pas de résolutions différents et qui, suivant le contexte de la simulation, seront ou pas adaptés.

8.1 La gestion de l'hétérogénéité des pas de temps

Différentes notions de temps coexistent dans un système de simulation.

Le *temps physique* ou *temps de la réalité* représente le temps dans lequel le phénomène que l'on observe, le système que l'on simule évolue. Par exemple, dans la simulation du phénomène de croissance d'une plante, le temps physique peut s'étendre sur 6 mois, la croissance de la plante ayant été observée du 30 septembre 1992 au 30 mars 1993;

Le *temps de simulation* ou *temps virtuel* est la représentation du temps de la réalité par le simulateur. Ainsi le temps de simulation du phénomène de croissance d'une plante peut être représenté par un nombre entier qui peut prendre ses valeurs dans l'intervalle [0, 23], chaque

unité du temps de simulation représentant une semaine du temps de la réalité;

Le *temps de "computation"*, le *temps de calcul* de la simulation représente le temps de la réalité durant lequel les instructions du simulateur sont exécutées par le processeur. Le simulateur de croissance d'une plante peut prendre 25 minutes comme temps d'exécution sur un ordinateur donné. Si le lancement de la simulation est effectué à 12h, alors de temps de calcul de la simulation s'étendra de 12h à 12h25.

Dans certains cas, l'avancement du temps de simulation peut s'effectuer au même rythme que le temps de la réalité (le temps de calcul du simulateur est équivalent au temps de la réalité). Dans les simulations d'entraînement, où des humains sont plongés dans des environnement virtuels, cette correspondance est nécessaire pour que l'environnement créé apparaisse réaliste à la personne subissant l'entraînement. Dans ce cas, le simulateur réalisé appartient à la catégorie des applications temps-réel.

Le plus souvent, les simulations suivent un objectif de rapidité d'exécution. Le simulateur est créé à des fins d'analyse, de test ou d'évaluation. Le modélisateur souhaite alors que la simulation donne des résultats le plus rapidement possible.

Dans la suite de ce paragraphe, nous nous intéressons au temps de simulation aussi appelé temps virtuel ou temps logique et aux différents mécanismes qui ont été proposés pour son implémentation. Une définition de cette notion a été proposée par [Lamport, 1978] en tant que moyen d'ordonnancement des événements survenant dans un système distribué.

Tout système de simulation doit respecter la règle de causalité qui stipule que le futur du monde simulé ne peut pas avoir influencé le passé. Cela signifie que lorsqu'un événement A est la cause d'un événement B, l'événement A doit avoir été généré avant l'événement B dans le temps de computation du système. Il existe deux principales approches de gestion du temps virtuel des systèmes de simulation.

8.1.1 Gestion du temps classique : simulation dirigée par l'horloge

Dans une simulation dirigée par l'horloge, le temps virtuel est discrétisé en un certain nombre d'intervalles de taille identique. Ces intervalles de temps sont désignés par le terme de pas de temps. La représentation du temps est symbolisée par une horloge virtuelle de référence qui exprime l'avancement de la simulation et autorise le déclenchement des différents événements (représentés par des actions effectuées) dans le simulateur. Il peut arriver qu'à un pas de temps donné, il n'existe aucune action à effectuer. Le cas où il existe une seule action à effectuer est également possible. Enfin, il peut également y avoir plusieurs actions à effectuer à un pas de temps donné. Dans ce dernier cas, les actions seront toutes associées à la même date virtuelle. Le simulateur doit assurer que l'ordre dans lequel chacune de ces actions est effectuée n'a pas d'incidence sur le futur de la simulation (le système est synchrone). De nombreux simulateurs

utilisent ce type de gestion du temps virtuel [Belouze, 1996; LePage et Ginot, 1997; Shin et Cury, 2001]. C'est le mécanisme le plus simple à implémenter. Cette représentation du temps de simulation est efficace dans le cas où les actions réalisables sont connues a priori. En effet, le concepteur du système doit choisir le pas de temps le plus approprié permettant de prendre en compte toutes les actions significatives du système. La détermination du meilleur pas de temps possible n'est pas toujours évidente comme le notent [LePage et Ginot, 1997].

8.1.2 Gestion du temps en environnement distribué : simulation dirigée par les événements

Dans les systèmes de simulation à événements discrets, la progression du temps virtuel dépend des occurrences d'événements qui, lorsqu'ils sont générés, sont associés à une date virtuelle précise qui permet de les ordonnancer. Dans cette approche, lorsqu'il y a une période d'inactivité, la simulation avance directement jusqu'au prochain événement significatif pour le système. La gestion du temps de simulation est déléguée à un noyau de synchronisation qui assure que lorsque des événements surviennent à la même date, le traitement de ces événements respecte la cohérence du système [Erard et Déguénon, 1996; Cassandras et Lafortune, 1999].

Lorsque le système de simulation à événement discret est distribué, le monde simulé est modélisé sous forme de groupes d'entités communicantes, désignés sous le terme de processus logiques (LP). Chaque LP gère une horloge virtuelle locale qui définit un temps virtuel local. Les différents LP opèrent de façon indépendante et échangent des informations en cas de nécessité [Radhakrishnan, 1997]. Le simulateur est associé à un temps virtuel global qui doit respecter la règle de causalité. Deux approches ont été proposées pour le maintien de cette règle de causalité :

8.1.2.1 L'approche sûre ou pessimiste

Dans cette approche proposée par [Chandy et Misra, 1981], les événements sont traités par les LP uniquement dans le cas où il est garanti qu'aucune erreur de causalité ne peut survenir suite à ce traitement. Pour garantir que seuls les événements sûrs (c'est à dire ceux qui respectent la règle de causalité) vont être réalisés, tout LP sur le point de générer des événements non sûr est bloqué. Cette technique peut aboutir à un état de blocage permanent (deadlock) du système. Pour éviter ces états de blocage, les méthodes sûres permettent d'éviter l'apparition d'un état de blocage dans le système [Chandy et Misra, 1981; Misra, 1986; Fujimoto, 1990]. Les méthodes pessimistes présentent l'avantage d'être facile à implémenter, mais elles ne permettent pas d'exploiter pleinement le parallélisme du système. Le comportement des LP peut être trop pessimiste et augmenter significativement le temps d'exécution de la simulation [Radhakrishnan, 1997].

8.1.2.2 L'approche risquée ou optimiste

Dans l'approche optimiste proposée par [Jefferson, 1985], les erreurs de causalité sont autorisées. Les méthodes risquées permettent de détecter un état de violation de la règle de causalité et proposent des techniques de recouvrement afin de rétablir un état sain du système [Jefferson, 1985; Chandy et Sherman, 1989; Fujimoto, 1990]. Le protocole optimiste de maintien de la causalité le plus utilisé est le mécanisme "time warp".

Le mécanisme time warp. Dans ce mécanisme proposé par [Jefferson, 1985], les sous-systèmes composant le système distribué opèrent de façon asynchrone et s'envoient des messages représentant la survenue d'événement, messages estampillés par un temps local. Lorsque l'événement reçu est associé à une date dépassée, un mécanisme de recouvrement est appliqué pour que tous les événements ayant eu lieu après l'événement à prendre en compte soient annulés. Ce mécanisme consiste en l'envoi d'anti-messages qui sont des copies *négatives* des messages envoyés auparavant. La notion de temps virtuel porte un sens spécial. La date virtuelle est le date du temps logique global du système distribué avant laquelle l'application du mécanisme de recouvrement n'est plus possible. Cela garantit la cohérence du système, en effet une action non réversible comme une opération d'entrée-sortie ne peut être réalisée qu'en se référant au temps global. Ce temps virtuel est calculé par un gestionnaire de coordination temporel et est envoyé à tous les sous-systèmes en fonction d'un algorithme à définir selon un mécanisme de ferroutage (le temps virtuel est communiqué entre les LP par l'utilisation de canaux de communication réservés, figure 8.1). De nombreux auteurs ont proposé des implémentations du mécanisme time warp, [Deelman et Szymanski, 1997],[Sharma et al., 1999],[Schmerler et al., 1998].

Les applications de simulation temps-réel utilisent généralement des mécanismes de gestion du temps dirigée par l'horloge. Cependant pour garantir que les contraintes temps-réel sont respectées, un certain nombre de techniques sont proposées [Kopetz, 1997]:

- les algorithmes de *synchronisation d'horloge* garantissent que les horloges qui génèrent les temps de simulation sur des processeurs pouvant être géographiquement distribués sont correctement synchronisés;
- *l'ordonnancement temps-réel* est utilisé pour garantir que les deadlines des traitements vont être respectés afin d'assurer des interactions correctes entre matériel et/ou participant humain;
- la *compensation de temps* est utilisée pour tenir compte des temps de latence dus aux communications. Lorsqu'une information est envoyée, le destinataire de l'information en question peut extrapoler sa mise à jour en fonction du temps de communication que l'information a pris pour lui parvenir.

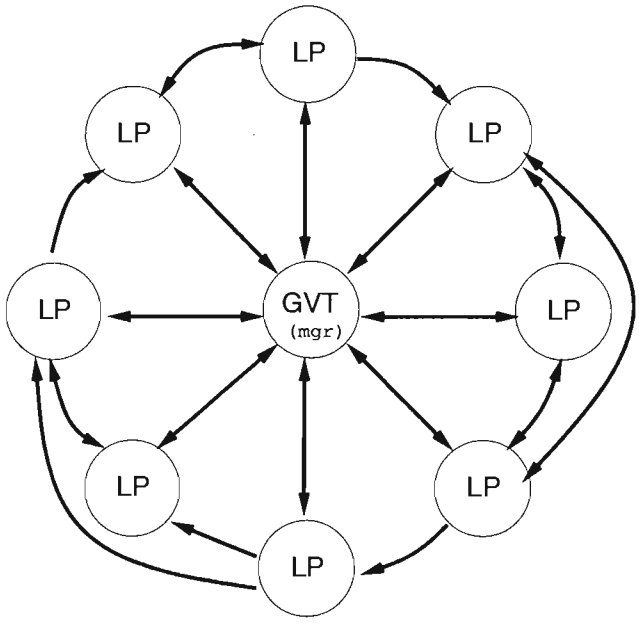


FIG. 8.1 – *Shéma de propagation du temps virtuel global (GVT) dans le protocole time warp*

8.1.3 Gestion du temps dans OSIRIS : prendre en compte des multiples pas de temps variables

Dans notre projet, nous sommes concernés par la coexistence de différents pas de temps, différents rythmes [Fianyo et al., 1998b]. En effet, les processus dynamiques ont été modélisés en fonction de leur rapidité, de la fréquence de leurs actions sur le domaine étudié. Le processus d'évaporation d'une culture est un processus assez lent; et il faut le relier au processus d'irrigation qui est beaucoup plus rapide. Par ailleurs ces processus dynamiques peuvent changer de rythme en fonction de conditions extérieures. L'eau va s'infiltrer plus vite dans le sol dans le cas où la charge d'eau sur le sol augmente rapidement et de façon importante, suite à une opération d'irrigation par exemple. Maîtriser la variabilité et la diversité de rythmes (de pas de temps) est une question indépendante de la façon dont le temps virtuel est implanté. L'objectif à atteindre est, dans ce dernier cas, d'offrir la possibilité de modifier dynamiquement le pas de temps de la simulation (ou de modifier la manière dont les événements sont générés).

La gestion du temps dans le système OSIRIS se range dans la famille des mécanismes pessimistes de gestion du temps de simulation. Seules les actions sûres sont autorisées, les composants dynamiques désirant effectuer des actions correspondant à la génération d'événements futurs du système sont mis en attente jusqu'à ce que les autres composants dynamiques atteignent le temps virtuel considéré correspondant. Le temps de simulation est fondé dans OSIRIS sur la notion de pas de temps. Chaque modèle à coupler agit sur le monde à une

vitesse particulière qui est ramenée à un pas de temps, un rythme particulier. L'utilisation de pas de temps est la façon classique de représenter le temps en simulation orientée à des fins d'analyse ou d'évaluation.

La notion de contrôleur de temps

Le système de simulation OSIRIS est composé de composants dynamiques représentant chacun des modèles à coupler. A chaque composant dynamique est associé un contrôleur de temps qui assure le déroulement du cycle de ce composant.

Un contrôleur de temps est donc un composant dynamique particulier du simulateur (un processus informatique) dont le rôle consiste à autoriser le déroulement du cycle de fonctionnement des composants dynamiques qui lui sont associés. Son comportement consiste à permettre l'avancement du temps de simulation d'une durée d'un pas de temps donné. Il existe autant de contrôleurs de temps que de pas de temps intéressants pour la simulation. La règle de causalité est assurée par deux mécanismes : (i) Tout composant dynamique ne peut avancer dans le temps (c'est à dire effectuer des actions sur le monde) que si le contrôleur de temps auquel il est associé lui en donne l'autorisation. (ii) Chaque contrôleur de temps est relié à un autre contrôleur de temps, son contrôleur de temps voisin. Ce dernier représente le contrôleur de temps dont le pas de temps associé est le plus proche du contrôleur de temps considéré. C'est le contrôleur de temps voisin qui autorise un contrôleur de temps à continuer son exécution. Par ailleurs, un composant dynamique peut changer de contrôleur de temps associé en cours de simulation. Il est possible de définir de nouveaux contrôleurs de temps en les insérant dans la chaîne des controlêurs de temps existants. Cette insertion consiste à mettre à jour les contrôleurs de temps voisins d'au plus deux contrôleurs de temps.

Implantée de cette manière la gestion du temps de simulation permet de prendre en compte des pas de temps non prévus au départ d'une simulation. Elle permet de ralentir ou d'accélerer la vitesse d'exécution des processus lorsque nécessaire.

8.2 La gestion de l'hétérogénéité des représentations

L'hétérogénéité des représentations de la réalité pose problème lorsque l'on s'attaque à une question de couplage de modèles. Chaque modèle représente son système suivant un point de vue particulier, en fonction de l'objectif qu'il cherche à atteindre. On se trouve donc confronté à des représentations diverses qui peuvent se rapporter au même concept mais qui ne sont pas toujours cohérents entre eux.

Par exemple, dans le contexte d'application de ce travail, la représentation du sol est très importante, puisque le sol est l'entité qui enregistre une hausse ou une baisse du taux des sels. Cependant, suivant la discipline scientifique considérée, l'entité sol bénéficie d'une représenta-

tion spécifique.

Pour un modèle hydrologique, le sol est traditionellement représenté sous la forme d'un profil de sol. La spécification d'un profil de sol consiste à définir une profondeur totale subdivisée en un ensemble de couches d'épaisseur fixe ou variable (la somme des épaisseurs étant égale à la profondeur considérée.) La définition du profil de sol consiste à spécifier sous forme de plusieurs tableaux de réels associés au profil (la dimension de chaque tableau dépendant du nombre de couches), les propriétés physiques (conductivité hydraulique) et chimiques (concentration des différents anions) qui le caractérisent, couche par couche.
Pour un modèle agronomique, l'entité sol est représentée sous forme de réservoir : le réservoir sol représentant la partie racinaire est la seule partie du sol qui intéresse l'agronome. La profondeur du sol du point de vue d'un modèle agronomique dépend de la limite jusqu'à laquelle peuvent s'enfoncer les racines de la plante considérée. A cette entité sol vue de façon globale, vont être associés différents réels correspondant à ses caractéristiques chimiques et physiques.
Pour un modèle de nappe, le sol n'est pas considéré du point de vue de sa profondeur. La nappe est en effet représentée sous forme d'un ensemble de mailles, chaque maille couvrant une unité de surface. La limite supérieure de la nappe est le début de la zone non saturée et représente le sol. Ainsi pour chaque maille de la nappe, on dispose d'un ensemble de renseignements concernant le sol, mais ces informations sont disponibles de façon horizontale selon un plan (données en 2D) et non pas en profondeur, de façon verticale (données 1D du profil de sol).

8.2.1 Différentes approches

La réflexion sur la gestion de diverses représentations de données est présente dans de nombreux domaines de l'informatique, notamment en bases de données et en intelligence artificielle. En IA, la représentation de la connaissance est un des sujets de recherche les plus importants. En base de données, de nombreux modèles ont été proposés afin de gérer efficacement les informations (modèle relationnel, modèle objet) ; des formalismes permettant de manipuler de façon cohérente les informations ont été proposés (les différentes formes normales pour le modèle relationnel); des mécanismes de transformation entre données existent (opérations de jointure, extraction, etc) De plus, comme cela a été mentionné en introduction du chapitre 6, différents travaux ont pour objet de proposer de nouveaux mécanismes d'intégration plus complexes entre bases de données. Cependant, les recherches effectuées en base de données sur la traduction entre représentations de données prennent pour primitives, pour objet de base, les schémas de base de données (i.e. la définition dans un formalisme donné d'une collection d'informations structurée) et non de simples variables typées qui constituent dans la grande majorité des applications de simulation la représentation de données.

8.2.2 La représentation de la réalité dans OSIRIS

8.2.2.1 un système de représentation pivot

Pour maîtriser l'existence de représentations potentiellement hétérogènes proposées par différents modèles, il est possible d'en choisir une parmi celles qui existent et de modifier les autres représentations afin de les adapter à celle que l'on considère comme référence. En adoptant cette démarche, on privilégie le point de vue particulier d'un modèle au détriment des autres. Lorsqu'un modélisateur d'une discipline donnée réalise une opération de couplage, il aura tendance à adopter cette démarche. C'est le choix qu'a effectué Belouze dans le travail présenté au chapitre 4.

Dans ce travail, nous avons souhaité pouvoir considérer tous les modèles à coupler selon un point de vue extérieur, qui peut être le point de vue d'un non-spécialiste à qui l'on essaie de transmettre l'ensemble des connaissances apportées par les différents modèles de la façon qui lui est la plus naturelle. Ainsi, la représentation d'un système s'effectue dans OSIRIS par la définition de différents types d'entités (différentes classes d'objets munis de leurs attributs respectifs). Ces entités sont définies par l'utilisateur d'OSIRIS, qui va ensuite associer, accrocher, greffer sur les entités créées, les différents modèles à coupler en établissant une correspondance entre les paramètres manipulés par les modèles et les attributs des entités existant dans la représentation de référence.

8.2.2.2 La notion des fonctions de correspondance entre système pivot et représentations particulières aux modèles

Dans le contexte d'OSIRIS, il peut y avoir bijection entre un paramètre d'un modèle à coupler et un attribut d'une entité du système, il peut y avoir divergence entre un attribut d'une entité du système et différents paramètres appartenant à différents modèles. Suivant la représentation de référence choisie et suivant les différents modèles en présence, les mécanismes de traduction peuvent être différents. Il n'est donc pas possible de proposer un mécanisme unique de traduction entre les différentes représentations. Il est cependant toujours nécessaire d'effectuer une opération de mise en correspondance entre représentations. Aussi nous définissons la notion de fonction de correspondance comme étant un programme permettant la transformation d'un (ou plusieurs) paramètres en un attribut. Il existe différentes fonctions de correspondance prédéfinies dans OSIRIS, mais un utilisateur peut en ajouter d'autres.

8.3 La gestion de processus concurrents agissant sur le même système

Le couplage faible de modèles pose le problème du changement dynamique des paramètres d'exécution des modèles en cours de simulation. Lorsqu'un modèle fonctionne seul, tout chan-

gement du système est provoqué par le modèle en exécution, celui-ci est préparé à le prendre en compte. Lorsque plusieurs modèles agissent sur le même système, les changements effectués par un des modèles sur le système peuvent avoir des répercussions sur d'autres modèles (s'ils modifient des valeurs utilisées comme paramètres par d'autres modèles par exemple) et nécessiter des modifications du comportement de ceux-ci (mise à zéro d'une variable utilisée comme diviseur par un autre modèle, par exemple).

Dans une entreprise de couplage fort de modèle, la cohérence entre les différentes dynamiques est assurée par une modification des différents modèles lorsque cela est nécessaire. Il s'agit en général d'une sérialisation des différentes dynamiques. Dans l'exemple du couplage Boltzmann/Navier-Stokes présenté au paragraphe 2.3.1, l'algorithme appliqué consiste à exécuter à chaque itération de la simulation les instructions du modèle Boltzmann suivies des instructions du modèle Navier-Stokes. Dans l'exemple de l'intégration de modèles pour l'évaluation de la salinisation d'une région présenté au chapitre 4, il y a également sérialisation du fonctionnement des trois modèles à coupler. Prenant appui sur le logiciel Matlab et son outil de simulation Simulink fondé sur la représentation de chaque modèle à connecter comme une boite noire (un subsystem) munie d'interfaces permettant la connexion du modèle considéré avec d'autres modèles (les *inports* et les *outports*), les modèles hydraulique, économique puis agronomique sont exécutés l'un à la suite de l'autre (figure 8.2). Le modèle hydrologique est exécuté à une fréquence journalière, certains paramètres sont cumulés sur le mois afin d'être transmis au modèle économique qui se déroule selon une fréquence mensuelle. Enfin après sommation de certains paramètres du modèle économique sur l'année, le modèle agronomique peut fonctionner.

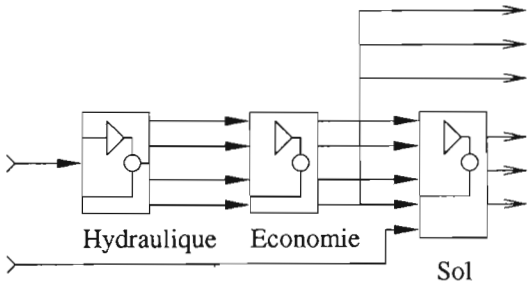


FIG. 8.2 – *Le modèle intégré d'étude de la salinisation [Belouze, 1996]*

Dans le cas d'un couplage faible, c'est l'interface qui assure la cohérence entre les différentes actions du système. Dans l'architecture HLA (High Level Architecture) présentée au chapitre 6, la définition de l'interface associée à un fédéré (modèle à coupler) comprend la spécification des contraintes existant sur les différents attributs manipulés ; les interactions entre le fédéré

et les autres membres de la fédération ne peuvent se faire que par l'intermédiaire de l'interface qui est implantée en tant qu'un ensemble de services appartenant au Runtime Infrastructure (RTI).

Dans OSIRIS, c'est l'agent de contrôle qui permet le maintien de la cohérence entre les différents processus indépendants du système.

8.3.1 L'agent de contrôle

Dans un outil générique d'intégration de modèles, les contraintes associées aux différents modèles doivent pouvoir être définies et modifiées par l'utilisateur du système. Dans le système OSIRIS, l'agent de contrôle est le concept qui permet d'assurer cette fonctionalité.

Le mécanisme agent de contrôle est inspiré de l'étude des systèmes ARCHON, ASIC et ALI présentés au chapitre 6. Dans ces différents systèmes, un agent est associé à un composant à coupler ; il stocke sous une forme particulière (collection des tâches de l'Intelligent Service (IS) chez ARCHON, Knowledge Rules (KR) chez l'ASIC, self-model dans le modèle ALI), l'ensemble des connaissances relatives au composant lui permettant de modifier dynamiquement son comportement au cours de l'exécution du système couplé.

L'agent de contrôle du système OSIRIS est également associé à un composant logiciel à coupler (l'interface processus) et permet un stockage des connaissances relatives au composant lui permettant de modifier son comportement en cours de simulation. Par exemple, un processus de nappe peut être modélisé par différents modèles numériques, chacun des modèles étant adapté à un contexte donné de la simulation. C'est l'agent de contrôle associé au processus nappe qui va choisir le modèle dynamique courant en fonction de règles spécifiées par l'utilisateur.
Par rapport aux différents systèmes présentés, OSIRIS permet de spécifier les différentes connaissances relatives à l'utilisation des modèles au niveau de l'utilisateur du système (avant le lancement de l'exécution) et non comme dans les systèmes classiques au niveau de l'implémentation du système. Cette flexibilité est nécessaire dans le contexte de ce travail. En effet, un même modèle peut être utilisé pour représenter différents phénomènes. Le modèle Leachm, par exemple, peut servir à calculer l'écoulement de l'eau dans la zone non saturée du sol, mais il peut également être utilisé pour déterminer une hauteur d'eau de la nappe.

8.3.2 Les connaissances nécessaires au contrôle dynamique

Plusieurs types de connaissance sont nécessaires à la maîtrise des dynamiques concurrentes de différents processus.

Les connaissances de contrôle du modèle sont relatives aux paramètres d'exécution du modèle représentant le processus. Selon le contexte de la simulation, il peut s'avérer nécessaire de modifier la périodicité d'exécution d'un modèle. Par exemple, pour un processus d'infiltration, la

vitesse d'infiltration de l'eau dans le sol dépend de la charge d'eau en surface. Si la quantité d'eau en surface augmente sous l'action d'un processus pluie, alors il y aura une augmentation plus rapide de la teneur en eau dans le sol. Pour que le système tienne compte de cette accélération, il faut augmenter la fréquence d'exécution du processus d'infiltration. Par ailleurs, également suivant le contexte de la simulation, un processus donné peut être représenté de façon plus adéquate par un modèle donné plutôt que par un autre. Il est très fréquent que pour un même phénomène, le modélisateur dispose de modèles microscopiques permettant un niveau de résolution très fin et de modèles macroscopiques, les modèles à bilan par exemple, plus simples et donnant des résultats agrégés. Dans l'exemple précédent de l'infiltration, lorsque la charge en eau en surface est pratiquement nulle (pas d'apport en eau dû à des pluies ou à l'irrigation), il est inutile d'utiliser un modèle microscopique, puisque l'action d'infiltration est pratiquement négligeable. De même, lorsque le profil, suite à des apports importants d'eau, devient saturé (sa teneur en eau est proche de celle d'une nappe), le mécanisme d'infiltration n'est plus le même et l'application d'un modèle microscopique peut donner des résultats incohérents. Dans ces différents cas, il devient plus judicieux d'utiliser un modèle simplifié. L'utilisation d'un modèle microscopique n'est par ailleurs pas toujours possible, les modèles microscopiques nécessitent souvent la connaissance d'un nombre important de paramètres qui ne sont pas toujours disponibles.

Un autre type de connaissance nécessaire à la maîtrise des différentes dynamiques est relatif aux données du système. Il est nécessaire d'assurer que les modèles ne mettent pas à jour les entités du système avec des valeurs devenues incohérentes parce que le contexte de la simulation a changé pendant le calcul des résultats. Il est également nécessaire de résoudre le problème de l'accès concurrent aux différents attributs des entités du système par les différents modèles fonctionnant en parallèle. Lorsque l'on couple un modèle d'infiltration et un modèle de nappe, ces deux modèles calculent une hauteur de nappe. Pour le modèle d'infiltration, cela correspond à la limite inférieure de la zone non saturée, pour le modèle de nappe c'est le plafond. La valeur finalement retenue dans le système peut-être donnée par le modèle d'infiltration, par le modèle de nappe, elle peut être calculée en fonction de ces deux résultats. Le choix de la valeur prise en compte pour la mise à jour est un problème de modélisation qui doit pouvoir être effectué par l'utilisateur.

8.3.2.1 Les règles de contrôle

Les types de connaissances nécessaires au contrôle portent sur des valeurs de variables (attributs d'entité, paramètres de modèle) et doivent être entrés dans le système par l'utilisateur. La manière la plus naturelle de représenter tous ces types de connaissances est d'utiliser le formalisme classique "si condition-vérifiée alors action-réalisée".
La règle de contrôle est le formalisme permettant à l'utilisateur du système OSIRIS de spécifier des connaissances relatives au fonctionnement parallèle de différents modèles. Pour chaque

processus évoluant dans le système, l'utilisateur a la possibilité de définir des règles de contrôle qui manipulent les différents attributs des entités du système, ainsi que les différents attributs propres au processus (contrôleur de temps, modèle associé). Ces règles de contrôle vont être mises en œuvre par l'agent de contrôle associé au processus considéré.

8.4 Une architecture pour maîtriser le couplage de modèles fondée sur un modèle d'agent

Sur la base des concepts présentés dans les paragraphes précédents, l'outil OSIRIS s'organise dans une architecture en 3 couches [Fianyo et Boivin, 2000; Fianyo et al., 2001]; les agents de contrôle dirigent le comportement du système (figure 8.3).

De l'intérêt du contrôleur de temps. L'intérêt principal du contrôleur de temps est qu'il permet d'expliciter le temps virtuel dans lequel se déroulent les différents modèles. L'explicitation du temps virtuel permet sa manipulation et c'est ainsi qu'un agent de contrôle peut adapter le rythme des différents processus en fonction du cours de la simulation et des règles que l'utilisateur a spécifiées.

De la justification de la notion d'interface processus. L'agent de contrôle est associé à un modèle explicitant la dynamique d'un phénomène physique. Cependant, un phénomène peut être représenté par différents modèles, de résolutions différentes, chacun d'eux étant adapté à des circonstances particulières de la simulation. Ceci justifie l'introduction de la notion de processus encapsulant différents modèles. Le concept de processus dans l'architecture OSIRIS permet de manipuler une même interface pour un phénomène considéré, ce phénomène pouvant être représenté par différents modèles. Un agent de contrôle est donc associé à un modèle par l'intermédiaire d'une interface processus.

De l'utilité de la notion d'accointance. Les connaissances permettant le contrôle des différentes dynamiques sont spécifiées dans un langage symbolique (parce que plus proche du langage naturel de l'utilisateur). Ces informations sont définies par l'utilisateur au niveau de l'interface processus et vont être stockées dans une base de règles. L'agent de contrôle associé à une interface processus donnée scanne cette base de règles en cas de nécessité. Le déclenchement de la vérification des règles est obtenu grâce à l'introduction de la notion d'accointance entre agents de contrôle. En effet, les changements qui interviennent dans le système sont le résultat de l'exécution des modèles. Chaque modèle est contrôlé par un agent de contrôle (par l'intermédiaire d'une interface processus). Une accointance d'un agent de contrôle est défini comme tout agent de contrôle dont l'interface processus associée partage au moins un attribut avec l'agent de contrôle considéré. Ainsi lorsqu'un processus modifie

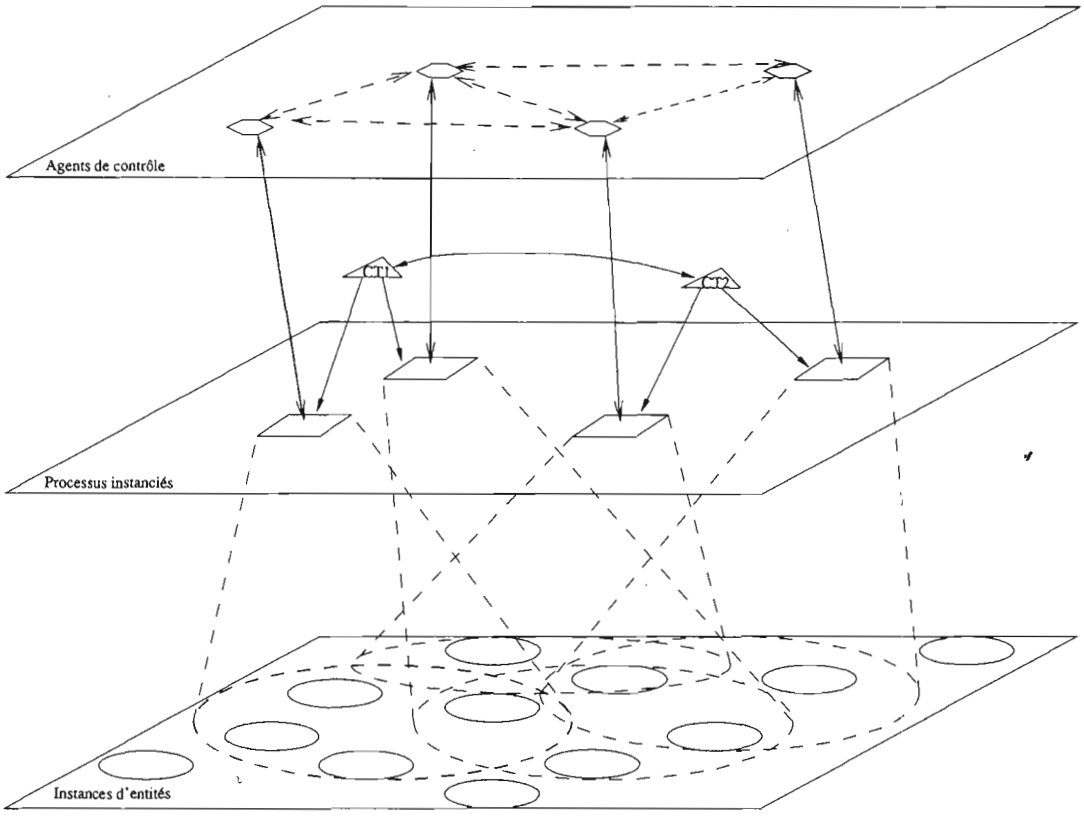


FIG. 8.3 – *Organisation conceptuelle d'OSIRIS. Le système à simuler est représenté sous la forme d'un ensemble d'entités, qui composent la première couche. La dynamique du système est représentée par un ensemble de processus qui modifient les états des entités, ils appartiennent à la deuxième couche du système. Ces processus sont contrôlés temporellement par des contrôleurs de temps (CT1 et CT2). Le contrôle de la dynamique du système (quel pas de temps, quel modèle d'exécution pour un processus donné) est effectué par les agents de contrôle aui appartiennent à la troisième couche.*

l'attribut d'une entité quelquonque, il notifie l'ensemble de ses accointances qui peuvent alors vérifier leur base de règle et éventuellement procéder à des réajustements. Il y a donc dans OSIRIS deux types de temps virtuels : les processus et leur modèles évoluent dans le temps défini par les contrôleurs de temps. Les agents de contrôle évoluent de façon indépendante en fonction des différents messages qu'ils recoivent de leurs accointances.

Conclusion

Dans ce chapitre, nous avons essayé de situer les concepts proposés dans OSIRIS par rapport aux différents domaines de recherche en informatique. La définition de temps virtuel est un

sujet de recherche ouvert en simulation [Radhakrishnan, 1997]. La représentation des informations fait l'objet de nombreux travaux en Intelligence Artificielle et en Base de Données, la proposition d'architectures permettant le contrôle dynamique de comportements est l'objet de recherche domaine des Systèmes Multi-Agents.

Chapitre 9

Architecture du système OSIRIS

RÉSUMÉ :

Les deux chapitres précédents ont introduit les principaux concepts du système OSIRIS. Dans ce chapitre, nous présentons de façon détaillée et formalisée l'architecture de l'outil d'intégration selon différents éclairages. La perspective statique permet la définition des différents composants du système (entités, processus, agent de contrôle et contrôleurs de temps) ainsi que les différentes relations qui existent entre eux. L'architecture de l'agent de contrôle est détaillée dans une section particulière (9.1.2). La perspective dynamique présente le mode de fonctionnement des différents composants (quelles actions pour chaque composant, quelles interactions). Enfin, la perspective utilisateur définit le mode d'utilisation de l'outil OSIRIS.

9.1 Perspective statique du système

La perspective statique du système OSIRIS peut être résumée par un shéma conceptuel de données en suivant le formalisme UML (Unified Modeling Langage). Le diagramme classes/associations de la figure 9.1 illustre les relations d'association qui existent entre les différents composants d'OSIRIS. Dans la suite du paragraphe, ces composants et leurs relations sont présentés de façon détaillée.

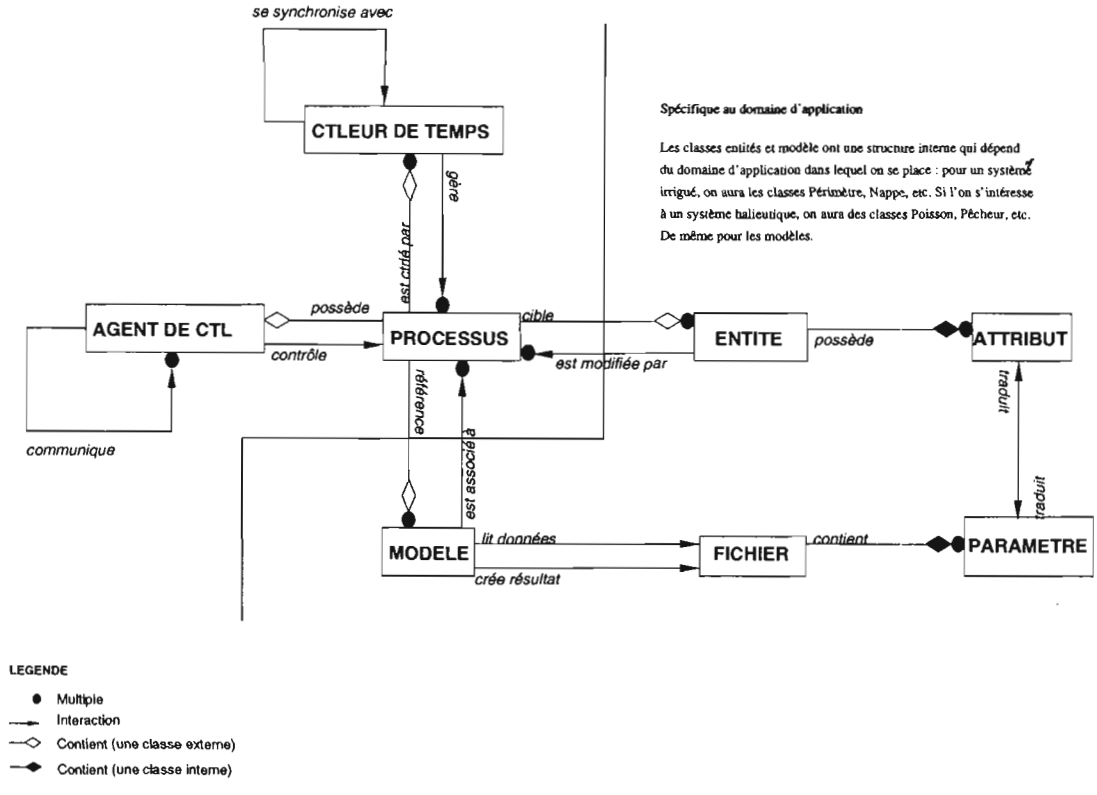


FIG. 9.1 – *Diagramme classe/association résumé du système OSIRIS*

9.1.1 Les différents composants

9.1.1.1 Les entités

Un système à simuler est décrit dans OSIRIS sous forme d'entités. De façon générale une entité peut être définie comme étant le réceptacle d'informations utiles pour l'application. Le découpage en entités capture le point de vue de l'utilisateur sur le système à simuler, en organisant d'une certaine manière les informations. La définition d'une entité calendrier cultural par exemple, permet de déterminer la consommation en eau d'une culture au cours d'une saison qui modulera l'apport net en eau du sol lui-même.

Entité spatiale. Une entité spatiale présente la propriété supplémentaire d'exister de façon objective dans le système à simuler et de disposer de coordonnées géographiques. Une entité spatiale est géographiquement située. Comme exemple, on peut citer le périmètre, la nappe, le profil de sol. Une entité spatiale est pertinente pour le système si elle permet de représenter l'effet des processus que l'on étudie. Les entités spatiales bénéficient d'une représentation graphique dans le système.

9.1.1.2 Les processus

La notion de processus dans le système OSIRIS réfère à la définition donnée en écologie. Un processus est défini comme la description dans l'espace et dans le temps de l'action d'un phénomène donné et des mécanismes de cette action [Blondel, 1995]. Les processus sont étudiés et donnent naissance à des modèles en fonction des objectifs recherchés par les modélisateurs. Dans OSIRIS, le processus est le concept servant d'interface permettant de manipuler ces modèles.

9.1.1.3 Les modèles

Les modèles auxquels on s'intéresse dans OSIRIS sont des modèles informatiques. Ces modèles sont des codes représentant le comportement d'un processus donné selon le point de vue de leur concepteur. Pour un processus donné, il est possible de trouver plusieurs modèles le représentant, ces modèles fonctionnant à des résolutions d'espace et de temps différents. Ainsi un processus infiltration peut être décrit par un modèle microscopique comme Leachm ou par un modèle macroscopique à bilan. Dans notre outil de couplage de modèles, les codes informatiques (programmes) qui sont pris en compte sont ceux qui à partir d'un ensemble de paramètres en entrée construisent un ensemble de paramètres de sortie contenant les résultats calculés par le modèle (programme). Un modèle est un exécutable qui peut fonctionner sous différents environnements systèmes. Il peut donc évoluer sur une machine différente de celle où évolue le système OSIRIS. OSIRIS va manipuler le modèle par le biais d'une interface : la partie cliente du modèle présente sur la plate-forme OSIRIS va échanger différentes informations avec la partie serveur du modèle éventuellement située sur une machine distante.

9.1.1.4 Les contrôleurs de temps

Les contrôleurs de temps sont les mécanismes de représentation explicite du temps de la simulation. Dans OSIRIS, il n'y a pas de définition de temps global de la simulation. Le temps simulé est représenté par un ensemble de contrôleurs de temps associés chacun à une unité de temps donnée. Les contrôleurs de temps sont liés deux à deux par un entier qui exprime la correspondance entre les unités de temps.

9.1.1.5 Les agents de contrôle

Un agent de contrôle est un agent logiciel collaboratif qui contrôle dynamiquement lors d'une simulation, l'exécution du processus auquel il est associé en fonction de règles de contrôle qui sont définies par l'utilisateur.

Les règles de contrôle. Une règle de contrôle suit le formalisme suivant :

`<condition> <action>`

La partie condition est de la forme `<variable> <operateur> <valeur>` où :

- `variable` est un attribut d'une entité du système ;
- `operateur` est un opérateur d'égalité (inférieur, supérieur, égal).

La partie action suit le schéma `<variable>` ← `<valeur>`. `<variable>` peut être un attribut d'une entité et peut également désigner un paramètre de contrôle du processus associé à l'agent de contrôle (contrôleur de temps courant, modèle courant). Un exemple de règle peut être : `nappe_0.concentrationCA > 0.3` ⇒ `modèleCourant` ← `emae.NAPPEB` (Si la concentration en cation Ca^{2+} de l'instance numéro 0 de l'entité de type nappe est supérieure à 0.3 alors le modèle courant (du processus pour lequel on spécifie la règle) devient le modèle NAPPEB qui se trouve sur la machine emae.

9.1.1.6 Les fonctions de correspondance

La fonction de correspondance n'est pas un composant du système. Elle explicite le fait qu'il est possible de traduire à l'aide d'une fonction (de correspondance) des paramètres manipulés par un modèle numérique en des attributs d'entité. Dans le diagramme classe/association fig. 9.1, la notion de fonction de correspondance apparaît dans la relation *traduit* entre ATTRIBUT et PARAMETRE.

9.1.2 L'agent de contrôle

L'architecture interne d'un agent de contrôle est présentée sur la figure 9.2. La composition interne de l'agent ainsi que les échanges d'information entre les différents composants sont détaillés dans les paragraphes ci-dessous.

9.1.2.1 Composition interne de l'agent de contrôle

Un agent de contrôle est composé de deux parties : une partie communication gère l'arrivée et la sortie des messages à destination des autres agents; une partie contrôle applique les règles qui sont associées au processus qu'il contrôle.

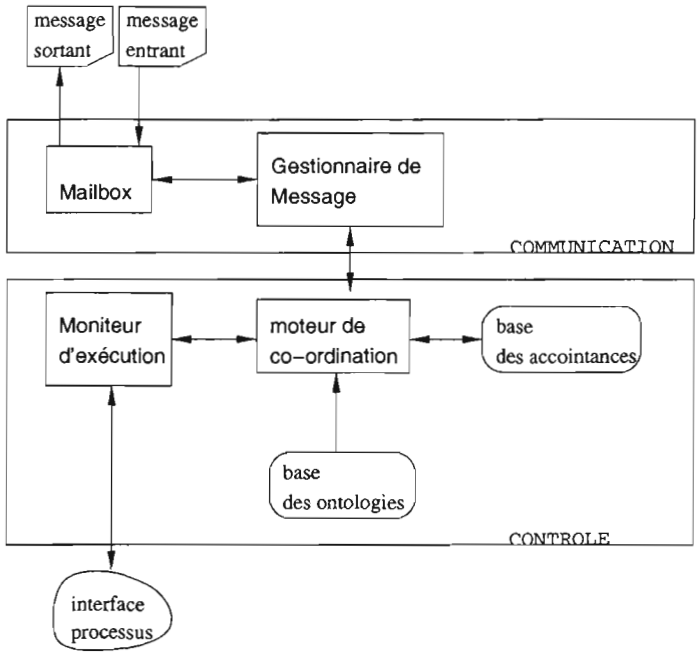


FIG. 9.2 – *Architecture d'un agent de contrôle*

La partie communication est composée d'une mailbox associée à un gestionnaire de message. Le gestionnaire de message active l'agent en cas de réception d'un message.

La partie contrôle est composée du moteur de co-ordination. C'est lui qui met en œuvre la coordination entre agents par l'application du protocole de communication. Dans notre cas, le protocole consiste en l'envoi d'un message standard à toutes les accointances de l'agent.

Une base d'accointances spécifie les agents qui vont recevoir des messages envoyés à l'initiative du moteur de co-ordination.

Une base d'ontologies contient la définition du monde sur lequel opère l'agent. Dans notre cas, elle contient les références (adresses) de l'ensemble des entités du système. C'est par la consultation de la base d'ontologie que l'agent peut appliquer les règles (changement du contenu d'un attribut d'une entité, par exemple) qui on été spécifiées. Un exemple d'ontologie est présenté en annexe (figure C.5).

9.1.2.2 Les flux d'information et de contrôle

Lorsqu'un message est reçu par la mailbox d'un agent, celui-ci avertit le gestionnaire de message qui active le moteur de co-ordination.

Le moteur de co-ordination lance alors la vérification des règles de contrôle qui ont été spécifiées au moment de la définition du processus.

Cette vérification est effectuée par l'activation du moniteur d'exécution qui exécute la fonction de vérification définie au niveau du processus. Si certaines règles sont applicables, les actions correspondantes sont réalisées. Elles sont de deux sortes : il s'agit soit de la mise à jour d'un attribut d'une entité, soit il s'agit du changement d'un paramètre de contrôle du processus.

Lorsque l'action consiste à mettre à jour un attribut d'entité, l'attribut est mis a jour, le moteur de co-ordination lance l'envoi d'un message ses accointances pour les informer d'un changement qui peut les concerner. Cela va déclencher à leur niveau la vérification des règles.

Lorsque l'action consiste à mettre à jour une propriété d'exécution du processus (modèle courant ou contrôleur de temps), les accointances ne sont pas averties : cela est inutile puisqu'elles seront informées d'un changement qui les concernent par la modification des entités qu'elles manipulent.

A la fin de son cycle, le processus demande à son agent de contrôle une vérification des règles. Cette demande est réalisée grâce à l'utilisation de l'interface `ZeusExternal` qui permet à une classe Java extérieure à Zeus d'accéder aux classes publique de l'agent Zeus.

9.1.3 Relations entre composants

9.1.3.1 Processus et Entités

L'action des processus consiste à modifier périodiquement l'état des entités du système. Les entités qui sont accédées soit en lecture, soit en écriture, soit en modification par un processus sont les *entités cibles* du processus en question. Parmi toutes ces entités cibles, il existe une entité sur laquelle l'action du processus repose principalement (il modifie principalement les attributs de cette entité); c'est l'*entité cible principale* du processus.

9.1.3.2 Relations entre Processus et Modèles

Un processus modifie le système par l'application des instructions d'un modèle. Pour un processus donné, il peut y avoir plusieurs modèles possibles (plusieurs fonctions de calcul d'un

résultat donné), un modèle spécifique étant adapté à un contexte de simulation spécifique. Par ailleurs, un modèle peut spécifier l'action de différents processus.

9.1.3.3 Relations entre Entités et Modèles

Les entités cibles d'un processus sont manipulées par le ou les modèles associés à ce processus. Les modèles des processus utilisant des fichiers de paramètres d'entrée et de sortie, il faut définir un mécanisme de traduction entre les attributs des entités et les paramètres utilisés par les modèles. C'est le rôle des fonctions de correspondance.

9.1.3.4 Relations entre Processus et Contrôleur de temps

Tout processus du système est rattaché à un contrôleur de temps. Le contrôleur de temps contribue à synchroniser les processus, en autorisant l'exécution des processus qui lui sont rattachés.

9.1.3.5 Relations entre Processus et Agent de Contrôle

Tout processus est associé à un agent de contrôle, celui-ci dirige dynamiquement les actions du processus en fonction de la base de règle spécifiée par l'utilisateur. L'agent de contrôle effectue deux types de contrôle sur son processus associé :

- il contrôle les paramètres d'exécution du processus : contrôleur de temps et modèles;
- il contrôle les valeurs des attributs des entités cibles du processus.

Ces types de contrôle ne peuvent se faire sans coordination entre agents de contrôle pour éviter tout état d'incohérence du système. C'est à dire, par exemplequ'il faut éviter que deux processus manipulant les mêmes entités ne mettent à jour le même attribut, en même temps.

9.1.3.6 Relations entre Agents de Contrôle

Les agents de contrôle dont les processus associés partagent des entités sont associés par des relations de collaboration. C'est ainsi que se définit la notion d'accointance. L'information concernant une modification effectuée par un processus sur un attribut est ainsi transmise aux accointances de l'agent.

9.2 Perspective dynamique sur le système

La dynamique d'OSIRIS est mise en œuvre par les composants processus, modèle, agents de contrôle et contrôleur de temps. Le modèle de fonctionnement de ces différents composants ainsi que leur fonctions respectives sont résumés dans le tableau 9.1.

9.2.1 Relations entre composants dynamiques

Les relations entre ces différents composants sont exprimées dans le diagramme classe / association détaillé de la figure 9.3. A un instant donné, un processus est associé à un modèle qui est son modèle *courant*. Un processus est également associé à un contrôleur de temps *courant* choisi parmi l'ensemble de ses contrôleurs de temps potentiels. Le choix de ces composants courants est dynamiquement réalisé par l'agent de contrôle auquel est associé le processus.

9.2.1.1 Synchronisation entre Contrôleurs de Temps

La synchronisation entre deux contrôleurs de temps (qui sont alors désignés par le terme de contrôleurs de temps voisins) est représentée à l'aide d'un formalisme résau de Petri sur la figure 9.4.

9.2.1.2 Communication entre agents de contrôle

La communication entre un agent de contrôle et ses accointances se fait par envoi de message. Un agent de contrôle est composé de deux modules : le module Message est en attente de l'arrivée d'un éventuel message provenant d'une accointance. L'arrivée d'un message entraîne l'activation du module Contrôle qui met en œuvre les règles définies.

COMPOSANT	FONCTION
Processus	Image du processus physique dans le système. Interface entre les modèles à intégrer et les entités du monde simulé. Permet la définition de règles de contrôle à respecter par les modèles lors de la simulation.
Modèle	Image du modèle numérique (programme) à exécuter. C'est par le modèle que le code numérique associé peut être manipulé. Un modèle est composé de deux parties : – une *partie serveur* située sur la machine sur laquelle est physiquement présente le code à exécuter. Le serveur est en attente de client potentiel; – une *partie client* permettant la génération des paramètres en entrée et la récupération des résultats (les paramètres en sortie) à partir des attributs des entités référencées par le composant Processus.
Contrôleur de temps	Synchronise temporellement les Processus entre eux.
Agent de contrôle	Met en oeuvre l'ensemble des règles de contrôle définies pour un composant Processus.

TAB. 9.1 – *Fonctions des différents composants dynamiques du système*

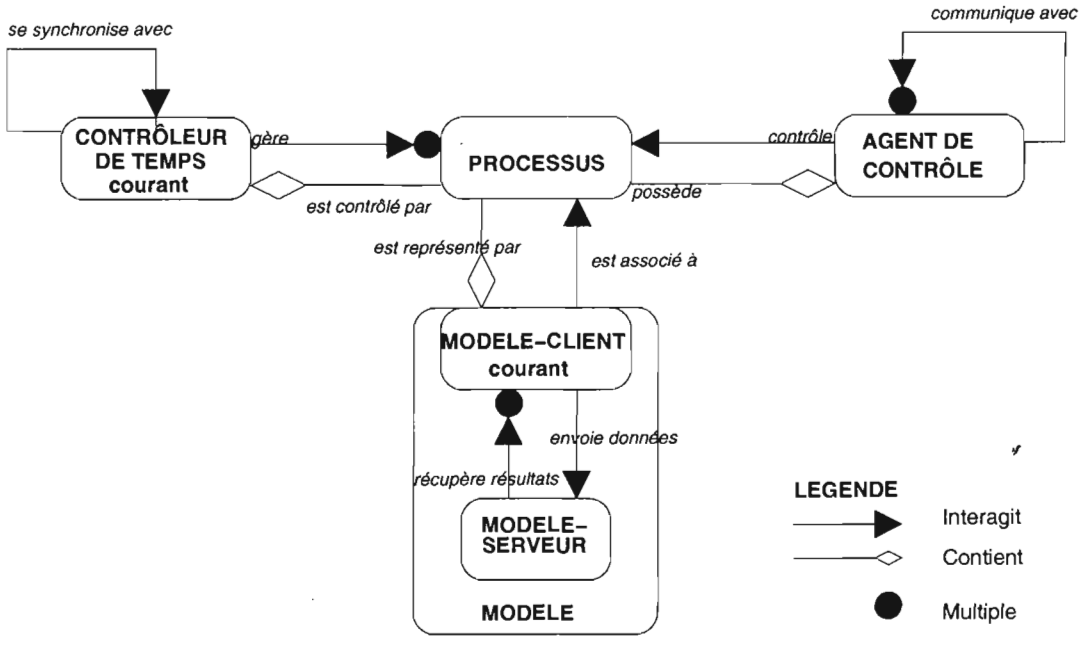


FIG. 9.3 – *Diagramme classe/association des composants constituant la dynamique du système*

9.2.2 Les cycles de fonctionnement des composants dynamiques

Les composants constituant la dynamique du système OSIRIS évoluent parallèlement au cours de la simulation. Ils sont synchronisés par des attentes sur événement émanant d'un composant donné. L'enchaînement général des actions du système peut alors être représenté par des diagrammes de collaboration UML. Les diagrammes de collaboration qui suivent, accompagnés chacun d'un tableau de description des interactions, décrivent, pour chacun des composants, l'enchaînement des actions réalisées.

9.2.2.1 Le cycle du Processus

Durant un cycle, le composant processus attend l'autorisation de son contrôleur de temps associé pour s'exécuter. Il lance ensuite l'exécution du modèle qui lui est associé et permet à l'agent de contrôle de s'assurer que toutes les règles de contrôle le concernant sont bien respectés. L'enchaînement de ces actions est représenté et décrit par la figure et le tableau 9.2.

Le modèle serveur. Le modèle-serveur suit l'architecture classique d'un serveur concurrent en mode connecté. Cela lui permet de gérer plusieurs requêtes de différents clients simultanément. Un thread de traitement est créé pour le traitement de chaque client.

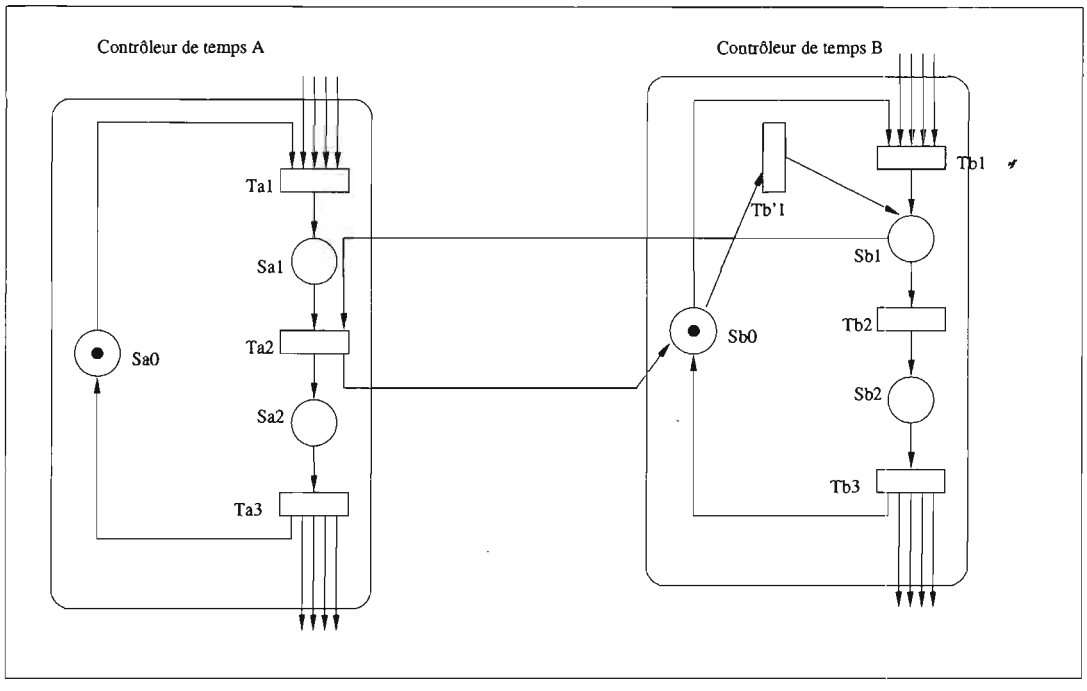
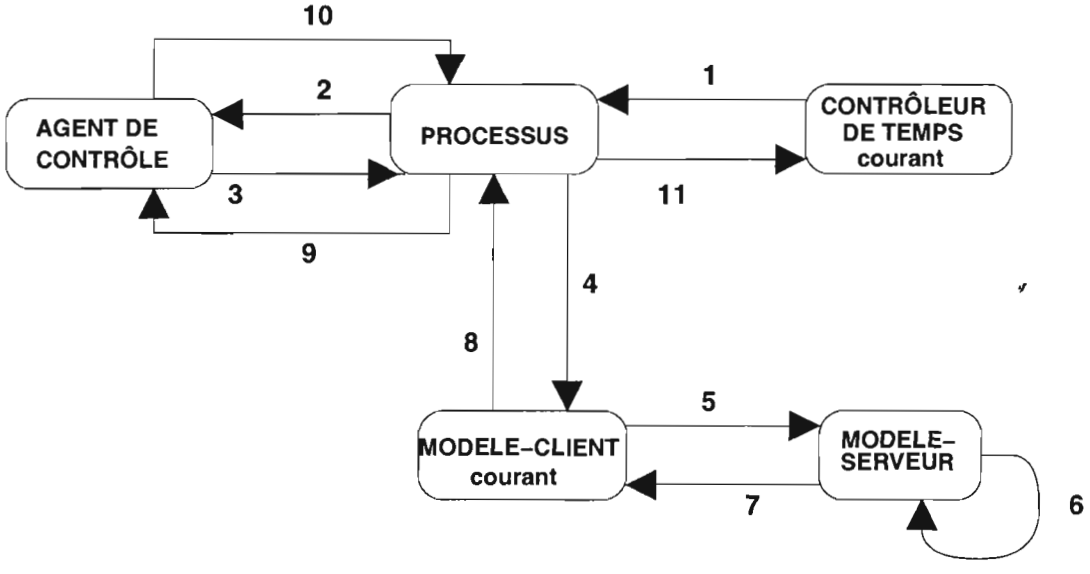


FIG. 9.4 – *Liaison des contrôleurs de temps. Une unité de temps pour le contrôleur de temps A correspond à N unités de temps pour le contrôleur de temps B, N étant le nombre entier reliant les deux temps virtuels. La transition Tb1 va être franchie N fois par le contrôleur de temps B, ce qui donne la possibilité aux processus associés de s'exécuter N fois. La transition Tb'1 va ensuite être franchie, ce qui va permettre au contrôleur de temps A de franchir la transition Ta2. Le franchissement de cette transition représente l'avancement dans le temps du contrôleur de temps A, ainsi que de ses processus associés. En définitive, pour une unité de temps A, le contrôleur de temps B aura avancé de N unités*

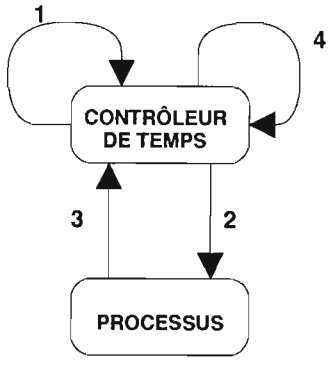


	Description
1	Autorisation par le contrôleur de temps courant de l'exécution d'un cycle par le processus
2	Demande d'autorisation d'exécution
3	Autorisation d'exécution par l'agent de contrôle après vérification des différentes règles relatives au processus
4	Création via le modèle-client courant du fichier de données nécessaire à l'exécution du modèle client
5	Envoi du fichier de données au modèle serveur et attente de la réponse
6	Création d'un thread permettant l'exécution du code numérique associé
7	Envoi des résultats produit par le code numérique
8	Récupération des valeurs via le modèle-client dans le fichier résultat et mise à jour des entités
9	Demande de contrôle des valeurs mises à jour
10	Signalement aux accointances des modifications effectuées par le processus. Vérification des règles de contrôle
11	Signalement de la fin du cycle

TAB. 9.2 – *Description des interactions lors du cycle d'un processus*

9.2.2.2 Le cycle du Contrôleur de Temps

Le contrôleur de temps autorise l'exécution des processus qui lui sont associés puis avertit son contrôleur de temps voisin qu'il peut lui aussi autoriser l'exécution de ses processus associés. L'enchaînement des différentes actions et interactions sont décrites dans la figure et le tableau 9.3.

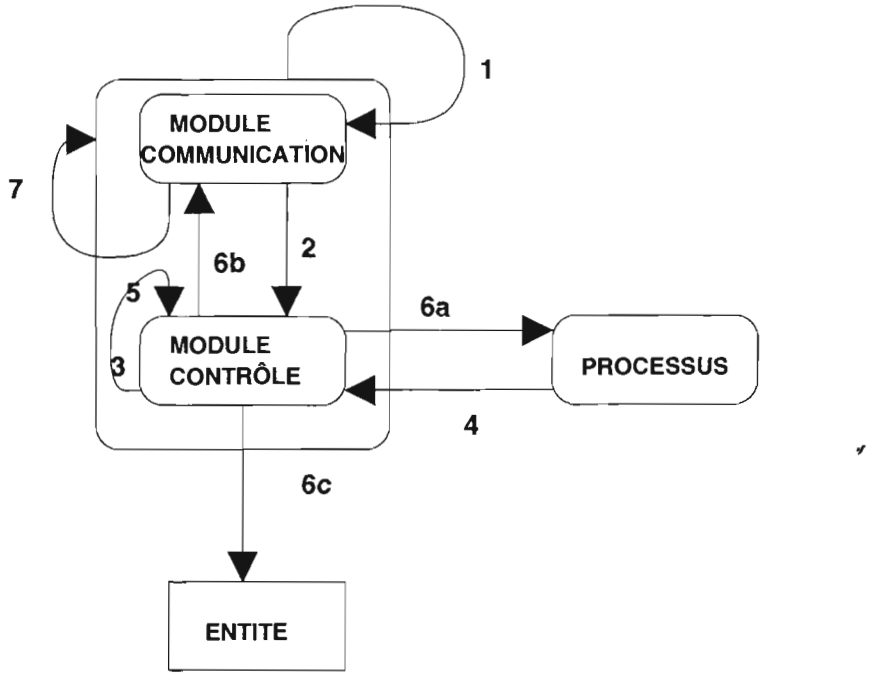


	Description
1	Demande d'autorisation d'avancement dans le temps au contrôleur de temps voisin correspondant à l'unité de temps immédiatement inférieure
2	Autorisation d'exécution envoyée aux processus associés
3	Réception de la fin d'exécution de tous les processus associés. Libération des processus changeant de contrôleur de temps. Accueil des processus demandant ce contrôleur de temps comme contrôleur de temps courant
4	Autorisation d'avancement dans le temps du contrôleur de temps voisin correspondant à l'unité de temps imédiatement supérieure

TAB. 9.3 – *Description des interactions lors du cycle d'un contrôleur de temps*

9.2.2.3 Le cycle de l'Agent de Contrôle

Contrairement aux autres composants dynamiques, l'agent de contrôle évolue de façon asynchrone en fonction de messages qu'il reçoit de ses accointances. De plus, il reçoit à la fin de chaque cycle du processus auquel il est associé une demande de vérification des règles de contrôle qui ont été spécifiées à son niveau. L'enchaînement des actions d'un agent de contrôle est décrit dans la figure et le tableau 9.4 Cette figure constitue une reprise sous une forme plus synthétique des informations contenue dans la figure 9.2 sur laquelle on a représenté la dynamique.



	Description
1	Réception d'un message
2	Déclenchement du processus de vérification des contrôles
4	Demande de vérification des règles de contrôle
3, 5	Mise en œuvre des actions de contrôle
6a	Mise à jour des variables de contrôle du processus si besoin
6b	Mise à jour des variables des entités si besoin
6c	Envoi d'un signal d'une action de changement du monde effectuée
7	Envoi aux accointances d'un message de modification du monde effectué

TAB. 9.4 – *Les actions et interactions du cycle d'un agent de contrôle*

9.3 Détails d'implémentation

Le système OSIRIS a été construit avec le langage de programmation Java 1.3. L'architecture de communication entre les parties cliente et serveur de modèle se base sur l'utilisation des sockets. L'architecture de l'agent de contrôle est empruntée au modèle proposé par la plate-forme Zeus. Ces différents choix sont discutés dans les paragraphes qui suivent.

9.3.1 Le langage d'implémentation : le choix de Java 1.3

OSIRIS est un système de simulation distribué programmé avec le langage Java. Ce choix n'est pas totalement naturel du point de vue de la simulation distribuée (PDES), où les objectifs recherchés sont l'amélioration des performances des applications de simulation par la définition de modèles et outils permettant la distibution de ces applications.

Java est en effet un langage à faible performance : la compilation d'une classe Java produit du byte code qui doit être transformé par un interpréteur (la machine virtuelle java) en fonction de l'ordinateur sur lequel va être exécuté le programme. Ceci explique que Java est 7 à 8 fois plus lent que C++ pour les opérations arithmétiques, 9 à 10 fois plus lent que C++ pour les opérations d'entrées/sorties (lecture de données dans un fichier) [Galyon, 1998]. La lenteur de Java est cependant, en partie compensée dans notre contexte par le fait que les différents calculs sont effectués par des exécutables compilés avec divers langages. OSIRIS est avant tout un outil d'interface.

L'avantage déterminant de Java dans notre contexte est sa portabilité sur la plupart des systèmes d'exploitation existants. Les classes créées pouvant être exécutées sans modification sur les principaux systèmes d'exploitation existant, le problème de l'hétérogénéité des plate-formes est résolu par le choix de ce langage.

Java offre des possibilités de réflexivité : il est possible de manipuler des classes qui ont été définies par ailleurs en utilisant les *méta*-fonctions du package object.reflect. C'est cette fonctionnalité qui permet d'autoriser l'utilisateur à définir ses propres classes (décrire son application) sur lequel va s'appliquer la gestion d'OSIRIS. Cela permet également d'introduire des modèles numériques sans reconstruction du système OSIRIS, pour peu que ces modèles observent les caractéristiques spécifiées (il suffit de placer leur description dans un répertoire donné et que cette description suive un formalisme donné).

L'interface utilisateur est importante dans le système OSIRIS, puisque c'est par elle qu'un utilisateur va pouvoir construire son système de simulation. Java propose un ensemble important de classes permettant de créer des interfaces utilisateur graphiques (GUI) conviviales.

Java est un langage orienté-objet, ce qui permet un style de programmation proche de la manière dont ont été spécifiés les problèmes. Cela est d'autant plus interessant qu'il s'agit de traiter de problèmes de parallélisme. Par ailleurs, concernant le parallélisme, Java offre avec la classe Thread un certain nombre d'outils facilitant sa mise en œuvre. Ce sont principalement ces raisons qui expliquent le choix du langage Java pour la construction du système.

9.3.2 L'implémentation du composant Modèle : un modèle client – serveur basé sur les sockets

Le composant Modèle peut évoluer sur une machine distante. L'utilisation d'un modèle client – serveur est donc naturelle. L'exécutable distant est associé à un programme serveur chargé de traîter les requêtes provenant d'éventuels clients. Un serveur peut avoir à traiter simultanément plusieurs requêtes (dans le cas par exemple où un même modèle est utilisé pour représenter plusieurs processus physiques). C'est la raison pour laquelle le serveur crée un thread pour traiter chaque arrivée de requête. La communication entre client et thread serveur s'effectue en mode connecté par l'interface des sockets. Le mécanisme de bas niveau des sockets a finalement

été choisi par rapport au mécanisme CORBA disponible avec Java 3 (package rmiiop) pour des raisons d'amélioration des performances de l'application. Ce choix diminue cependant la généricité du système OSIRIS : l'ajout d'un modèle nécessite l'introduction d'une classe serveur à contruire par l'utilisateur.

9.3.3 L'implémentation du composant Agent de Contrôle : le choix de la plate-forme Zeus

Zeus[1] [Nwana et al., 1999; Collis et Ndumu, 1999] est une plate-forme de construction d'agents proposée par le groupe de recherche sur les agents intelligents des laboratoires BT Labs(British Telecommunication), permettant de générer des agents collaboratifs à l'aide d'un ensemble d'outils présents dans un ensemble de librairies. Certains des outils offerts par Zeus présentent un intérêt particulier pour le système OSIRIS. La plate-forme Zeus offre notamment :

- un mécanisme d'échange de messages capable de transmettre des performatives FIPA-ACL;
- une librairie de stratégies de co-ordination pré-établie, qui inclut notamment le contract-net protocole;
- différents types de relations organisationnelles dans une société d'agents, i.e. on peut définir des supérieurs et des inférieurs (les supérieurs ordonnent, les inférieurs exécutent), ou des pairs (les agents ne bénéficient d'aucune priorité);
- un ensemble de structures permettant de spécifier les différentes règles que doit faire respecter l'agent : il est possible de spécifier des règles sous forme de primitives, ou sous forme de règles de production;
- un formalisme permettant de décrire le monde sur lequel doit agir un agent;
- un système d'interface permettant l'exécution distribuée des différents agents d'une application.

Nous avons choisi d'utiliser des outils existants plutôt que de bâtir notre propre architecture d'agent [Wooldridge et Jennings, 1999], puisque les possiblités présentées par la plate-forme Zeus répondent à nos besoins. De plus Zeus est implantée et génère des classes en Java 1.3, langage dans lequel OSIRIS a été implanté. La compatibilité est donc totale. Enfin, Zeus est une plate-forme à utilisation libre de droit (comme tout logiciel Open Source) dont les sources sont disponibles et modifiables. L'architecture de l'agent de contrôle d'OSIRIS (architecture simplifiée de celle de l'agent générique ZEUS présentée en annexe B) a donc été implantée en modifiant les codes sources de l'agent générique ZEUS.

1. www.labs.bt.com/projects/agents/zeus/

9.4 Utilisation du système

L'utilisation du système OSIRIS consiste à lancer l'exécution de différents modèles (éventuellement situés sur des machines distantes) appliqués à un ensemble d'entités constituant le système à simuler. Pour aboutir à cette exécution distribuée du système de simulation, un certain nombre d'étapes préalables doivent être accomplies. Il s'agit de définir l'ensemble des entités constituant le système tel que le perçoit l'utilisateur. Une fois le système spécifié, l'utilisateur peut définir les différents processus dont il souhaite voir les actions en les associant aux entités définies. Il faut ensuite spécifier les règles de contrôle à respecter par les différents modèles et processus. Il est alors possible de lancer la simulation. Ces différentes actions sont réalisées à l'aide d'interfaces présentées en annexe C.

9.4.1 Définir les classes et instances d'entités

Définir un système consiste à créer et à spécifier l'ensemble des types d'entités spatiales, actives, ordinaires, qui vont représenter le système considéré. Si l'on s'intéresse à un système irrigué, on va définir le monde sous forme de classes d'entités périmètre, nappe, fleuve, profil, exploitant, etc. Si par contre on s'intéresse à un système de pêche, on décrira ce monde avec les classes d'entités pêcheur, mareyeur, poisson, etc.
Instancier le système consiste à créer et à spécifier des instances de classes d'entités du système.

Dans la version actuelle de l'outil OSIRIS, la description du monde est figée sur le système irrigué. On se borne donc à l'instancier.

9.4.2 Définir les classes et instances de processus

Définir un processus consiste à créer et à spécifier une classe de processus possédant les caractéristiques suivantes :

- la classe de processus est identifiée par un nom unique;
- cette classe de processus est associée à un certain nombre de classes d'entités, qui sont les entités cibles du processus,
- la classe de processus possède une entité cible principale,
- elle est associée à un ou plusieurs modèles utilisables,
- elle est associée à un ou plusieurs contrôleurs de temps utilisables,
- des contrôles sur les attributs des entités cibles peuvent lui être associés
- des contrôles sur les paramètres d'exécution du processus en question peuvent lui être associés.

Instancier un processus consiste à créer et à spécifier une instance de processus. Une instance de processus associe une classe de processus avec des instances d'entités cibles. Pour chaque processus instancié créé il y a:

- création de l'agent de contrôle associé au moment du lancement de la simulation,
- création du fichier de données au moment de l'exécution sur une machine spécifiée d'un modèle donné.

9.4.3 Définir les règles de contrôle

Les règles de contrôle permettent le contrôle dynamique des processus instanciés au cours de la simulation. Les règles de contrôle sont spécifiées au niveau de la définition d'une classe de processus. Une règle de contrôle concerne soit un attribut d'une entité cible du processus, soit un paramètre d'exécution du processus. Les paramètres d'exécution du processus considérés sont les contrôleurs de temps et les modèles possibles.

9.4.4 Lancer la simulation

Une fois les instances de processus créées, la simulation peut être lancée. Le lancement de la simulation consiste à lancer l'exécution du mécanisme des contrôleurs de temps qui va mettre en marche tout le système. Pour chaque processus intancié du système, un agent de contrôle va être créé et activé. L'activité des agents de contrôle se déroule parallèllement à l'activité des processus instanciés du système.

Conclusion

L'architecture d'OSIRIS est très fortement distribuée. Les différents composants dynamiques évoluent de façon indépendante, et se synchronisent uniquement grâce aux différents mécanismes présentés ; les modèles serveurs sont sur des machines distantes.

Chapitre 10

Exemple d'exécution d'une simulation

RÉSUMÉ :

Ce chapitre donne quelques résultats produits par l'exécution du système OSIRIS. Ces résultats proviennent de la simulation d'un système initialisé avec des données fictives, l'objectif poursuivi étant de vérifier le bon fonctionnement des concepts (contrôleurs de temps et agents de contrôle) présentés dans cette thèse. Les résultats obtenus sont donc présentés à titre illustratif, sans discussion de leur validité.

Dans ce chapitre, nous présentons les résultats d'une simulation du système intégré construit en suivant la démarche de couplage présentée au chapitre 7. Cette exécution est donc présentée à titre illustratif. Ses résultats ne peuvent être comparés avec des données provenant d'une situation réelle.

Le système intégré est composé de 4 entités profils de sol sur lesquels sont situés des parcelles (qui peuvent ou non être cultivées), une nappe composée de 12 nœuds, située sous les profils. Les profils de 12 couches chacun, sont initialisés avec des teneurs en eau et des concentrations en ions égales (les données de chaque couche du profil sont uniformes). Les 12 noeuds de la nappe (4 lignes et 3 colonnes) sont également initialisées avec des données uniformes. 3 parcelles sur 4 portent des cultures.

Les exécutables correspondant aux modèles LEACHM, MOC, INFILB (modèle d'infiltration simplifié) et NAPPEB (modèle de nappe simplifié) sont installés sur les machines UNIX Sun Solaris 7(efate.bondy.ird.fr), Solaris 8 (emae.bondy.ird.fr), Windows95 (galapagos.bondy.ird.fr). Le modèle OSIRIS est lancé sur la machine emae.

Les différentes valeurs des attributs des différentes entités ne correspondent pas à une réalité observée. Elles ont été choisies pour permettre un changement de contexte, l'objectif poursuivi par la simulation étant d'observer le comportement des différents processus afin de vérifier le fonctionnement correct des contrôleurs de temps (pas de blocage dû à des attentes infinies) et des agents de contrôle (changement des propriétés d'exécution suite à l'application des règles des règles définies).

Ces résultats sont donc présentés à titre illustratif, mais ne peuvent être comparés avec des données prevenant d'une situation réelle.

10.1 Les données du système

	nappe_0
teneur en eau [0-3][0-2]	0.8
concentration Na [0-3][0-2](mmol/l)	13.9
concentration K [0-3][0-2]	0
concentration Mg [0-3][0-2]	0.288
concentration Ca [0-3][0-2]	0.1
hauteur de nappe [0-3][0-2]	75

TAB. 10.1 – **Données relatives à la nappe**

	profil_0	profil_1	profil_2	profil_3
surface(m^2)	20	20	55	30
hauteur d'une couche	200	200	200	200
teneur en eau [0-11]	0.2	0.3	0.2	0.4
concentration Na [0-11]	13.9	14.1	13.5	12.0
concentration K [0-11]	0	0	0	0
concentration Mg [0-11]	0.3	0.4	0.3	0.3
concentration Ca [0-11]	0.15	0.1	0.2	0.1
pH [0-11]	7.5	7.6	7.5	7.6
perteVersnappe	0	0	0	0
type sol	holllade	hollalde	hollalde	hollalde

	parcelle_0	parcelle_1	parcelle_2	parcelle_3
culture	tomate	oignon	riz	-
date debut (en jours)	0	10	30	-

TAB. 10.2 – **Données relatives aux 4 parcelles et à leurs profils associés**. *Les parcelles sont cultivées en boucle à partir de leur date de début. Par exemple, pour la parcelle_2, le riz est planté au 30ème jour et va poursuivre son cycle de 90 jours (tableau 10.3). Au 91ème jour, on suppose qu'il y a redémarrage de la culture du riz si la simulation n'est pas terminée. Cette supposition, peu réaliste, permet la poursuite de l'alimentation en eau de la parcelle et la modification des comportements des processus infiltration et nappe.*

date(jours), quantite(mm)	riz	tomate	oignon
date germinaison, quantité d'eau nécessaire	(0,300)	(0,150)	(0,150)
date émergence, quantité d'eau nécessaire	(60,300)	(40,150)	(30,100)
date maturité, quantité d'eau nécessaire	(90,0)	(70,0)	(50,0)

TAB. 10.3 – **Données relatives aux cultures**. *Ces données déterminent l'apport d'eau d'une parcelle (quelque soit sa superficie) pour une période donnée. Par exemple, une parcelle associée à la culture* oignon *reçoit 150 mm d'eau pendant 30 jours, puis 100 mm pendant 20 jours.*

	hollaldé	sableux	vertique
composition argile(%)	45	17	60
composition matière organique(%)	15	10	10
composition vase(%)	10	10	10
ETP	1.1	1.2	1.2
teneur en eau à saturation ($cm.cm^{-3}$)	0.7	0.8	0.6

TAB. 10.4 – **Données relatives aux types de sol**

	pluie (mm)	température (^oC)	amplitude (^oC)
0	0	18	7
89	3	30	10
179	0	35	15
269	0	37	10
355	0	25	10

TAB. 10.5 – **Données journalières concernant le contexte physique**. *On suppose qu'il ne pleut pratiquement pas, la période la plus froide de l'année dure 3 mois (les températures journalières varient de 18 à 25 degrés celsius). Le reste du temps le climat est très chaud.*

10.2 Les processus du système

Le système évolue sous l'action de deux processus :

La nappe change de hauteur et de concentration en ion sous l'action du modèle MOC lorsque sa concentration moyenne en Ca^{2+} est inférieure à 0.3. Dans le cas contraire (Ca^2+ supérieure à 0.3) la dynamique de la nappe est représentée par un modèle simplifié NAPPEB.

Les quatre profils évoluent en fonction des apports d'eau dont la quantité est déterminée par la culture associée à la parcelle. Ils sont soumis à l'action d'un processus d'infiltration modélisé par le code LEACHM lorsque la teneur en eau moyenne du profil est inférieure à une valeur donnée. Dans le cas où la teneur moyenne est supérieure à la valeur seuil, la dynamique d'infiltration du profil est représentée par le modèle simplifié INFILB.

10.2.1 La définition des processus

L'utilisation d'OSIRIS permet de définir les processus nappe (tableau 10.7) et infiltration (tableau 10.6) dotés des propriétés suivantes:

Les entités cibles du processus et l'entité principale permettent de déterminer l'ensemble des attributs qui vont être accéder (en consultation et en modification) par le processus.

Les contrôleurs de temps et les modèles associés représentent les propriétés d'exécution du processus considéré; ils déterminent les dynamiques possibles du processus considérés.

Parmi l'ensemble des possibles, les propriétés courantes d'un processus sont déterminées par les règles de contrôle.

L'intérêt des fonctions de correspondance est illustré pour le modèle MOC : les concentrations en ions sont calculées globalement par MOC alors qu'elles existent par espèces dans le système. La concentration en ion est donc sommée pour le modèle MOC et distribuée proportionnement à la répartition des espèces d'ions existants dans le système pour la mise à jour des attributs concentrationNa,Ca,K,Mg tableau 10.7.

Cible	une Parcelle, un Profil, une Culture, un Type de Sol, un Contexte Physique, une Nappe
A. consultés	Culture.date, Culture.quantiteEau, TypeSol.teneurSaturation, Profil.teneurEau, Profil.concentration{Na,Ca,Mg,K}
A. modifiés	Profil.teneurEau, Profil.concentrationEau, Profil.concentration{Na,Ca,Mg,K}, Profil.perteVersNappe{teneurEau, Na,Ca,Mg,K}
Contrôleurs temps	15 jours; 5 mois
Modèles	leachm; infilb
Contrôles condition ⇒ action	Profil.teneurEau >= TypeSol.teneurSaturation ⇒ modeleCourant ← infilb tempsCourant ← 5 mois Profil.teneurEau <TypeSol.teneurSaturation ⇒ modeleCourant ← leachm tempsCourant ← 15 jours

TAB. 10.6 – **Définition d'un processus infiltration.** *les listes A. consultés et A. modifiés ne sont pas exhaustives. Seuls les attributs principaux ont été mentionnés.*

Cibles	une Nappe, plusieurs Profils
A. consultés	Nappe.teneurEau, Profil.teneurEau, Profil.perteVersNappe{teneurEau,Ca,Na,Mg,K}, Nappe.concentration{Ca,Na,Mg,K}, Profil.concentration{Ca,Na,Mg,K}
A. modifiés	Nappe.teneurEau, Profil.perteVersNappe{teneurEau,Ca,Na,Mg,K}, Nappe.concentration{Ca,Na,Mg,K}
Contrôleurs temps	6 mois, 2 ans
Modèles	moc, nappeB
Contrôles condition ⇒ action	Nappe.concentrationCa >= 0.3 ⇒ modèleCourant ← nappeB tempsCourant ← 2 ans Nappe.concentrationCa <0.3 ⇒ modèleCourant ← moc tempsCourant ← 6 mois
Correspondances MOC attribut ↔ paramètre	SOMME(Nappe.concentration{Ca,Na,Mg,K})→ concentrationION Nappe.concentrationCa ← MULTI(concentrationION, DIVISE(Nappe.concentrationCa,SOMME(Nappe.concentration{Ca,Na,Mg,K}))

TAB. 10.7 – **Définition du processus nappe.**

10.2.2 L'instanciation des processus

L'instanciation des processus permet d'associer des entités particulières à une instance de processus. C'est à un processus instancié qu'est associé un agent de contrôle. C'est également lors de l'instanciation du processus qu'il est possible de déterminer la machine serveur qui sur laquelle va être exécuté le modèle numérique.

infiltration_0	profil_0, nappe_0	galapagos.LEACHM, efate.INFILB
infiltration_1	profil_1, nappe_0	galapagos.LEACHM, emae.INFILB
infiltration_2	profil_2, nappe_0	galapagos.LEACHM, efate.INFILB
infiltration_3	profil_3, nappe_0	galapagos.LEACHM, efate.INFILB
nappe_4	profil_0, profil_1, profil_2, profil_3, nappe_0	galapagos.LEACHM, efate.INFILB

TAB. 10.8 – **Instanciation des processus**

10.3 La simulation

10.3.1 Le contrôle du temps de la simulation

Le contrôle du temps de la simulation est assuré par le paramétrage des contrôleurs de temps. Les contrôleurs de temps existent indépendamment des processus qu'ils contrôlent. Ils sont liés deux à deux par des relations de voisinage. Ces relations expriment les rapports de pas de temps entre contrôleurs.

Lorsque les processus ont été instanciés et sont prêts à être simulés, les contrôleurs de temps auxquels ne sont rattachés aucun processus ne necessitent pas de fonctionner pour la simulation considérée.

Dans l'exemple présenté, les seuls contrôleurs de temps actifs sont les contrôleurs de temps mois, année et jour. Le voisin théorique du jour est le contrôleur de temps semaine, son voisin effectif sera le mois pour cette simulation. La correspondance par défaut entre des voisins effectifs est calculée en tenant compte des relations entre voisin théorique (1 mois correspond pratiquement à 28 jours car théoriquement 1 mois correspond à 4 semaines et 1 semaine correspond à 7 jours.) Les correspondances pratiques sont modifiables par l'utilisateur au lancement de la simulation.

	Théorie		Effectif	
	Voisin	Corr.	Voisin	Corr.
seconde	-	1	-	-
minute	seconde	60	-	-
heure	minute	60	-	-
jour	heure	24	-	1
semaine	jour	7	-	-
mois	semaine	4	jour	28
annee	mois	12	mois	12

TAB. 10.9 – **Les contrôleurs de temps**. *La correspondancce pratique est mise à jour au démarrage de la simulation en fonction des caractéristiques des processus présents. Les processus n'utilisant que les contrôleurs de temps année, mois et jours, les autres contrôleurs de temps sont inactifs durant la simulation.*

10.3.2 Le déroulement de la simulation

L'écran de contrôle de la simulation (figure 10.1) affiche pour chaque processus instancié associé à sa cible principale et différents temps de simulation (date de démarrage du processus, date d'arrêt, date actuelle) en fonction du contrôleur de temps courant du processus considéré. Pour les processus actifs, il est possible d'activer une fenêtre de contrôle permettant de suivre le déroulement de l'exécution du modèle numérique sur la machine distante (figure 10.2).

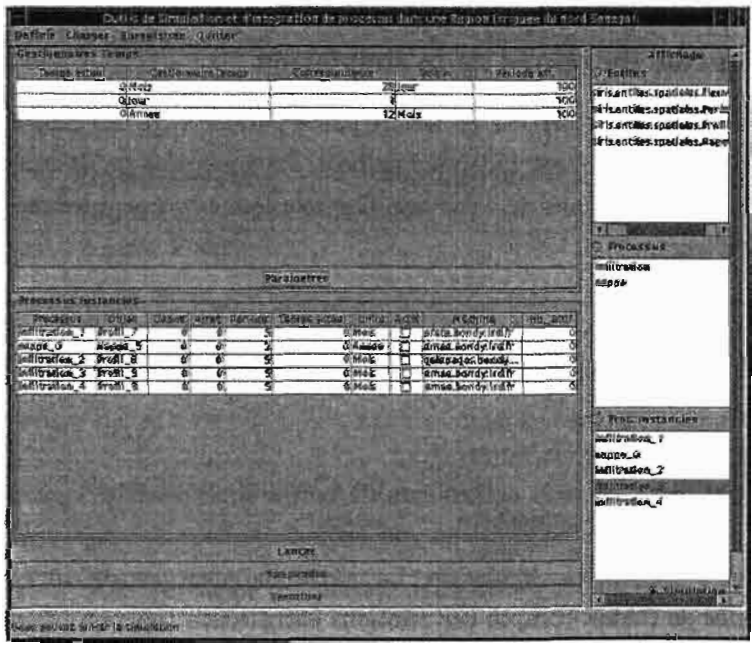


FIG. 10.1 – *Ecran de contrôle de l'outil OSIRIS*

FIG. 10.2 – *Ecran de contrôle distant de l'outil OSIRIS*

10.4 Résultats de la simulation

Le système ci-dessous simulé pendant une période de 5 ans (= 60 mois = 120 quinzaines de jours) donne les résultats ci-dessous. La durée de la simulation (sans activation des fenêtres de contrôle permettant de suivre le déroulement de la simulation) sur une machine distante est de 15 mins environs.

10.4.1 Etat final de la nappe et d'un profil

	teneurEau	concentrationNA	concentrationK	concentrationMg	concentrationCa
	0.1830	0.5	15.3	0.1	0.5
0	0.3495	0.6	11.3	0.1	0.4
1	0.2577	0.8	9.7	0.0	0.1
2	0.2315	0.5	11.6	0.0	0.2
3	0.2311	0.8	13.5	0.0	0.1
4	0.2455	0.8	13.5	0.0	0.1
5	0.2413	0.7	13.5	0.0	0.1
6	0.2473	0.8	13.5	0.0	0.1
7	0.2550	0.7	12.9	0.1	0.2
8	0.2548	0.6	13.5	0.0	0.2
9	0.2635	0.6	13.5	0.0	0.3
10	0.1354	1.5	10.4	0.1	0.4
11	0.1329	1.8	10.4	0.0	0.3

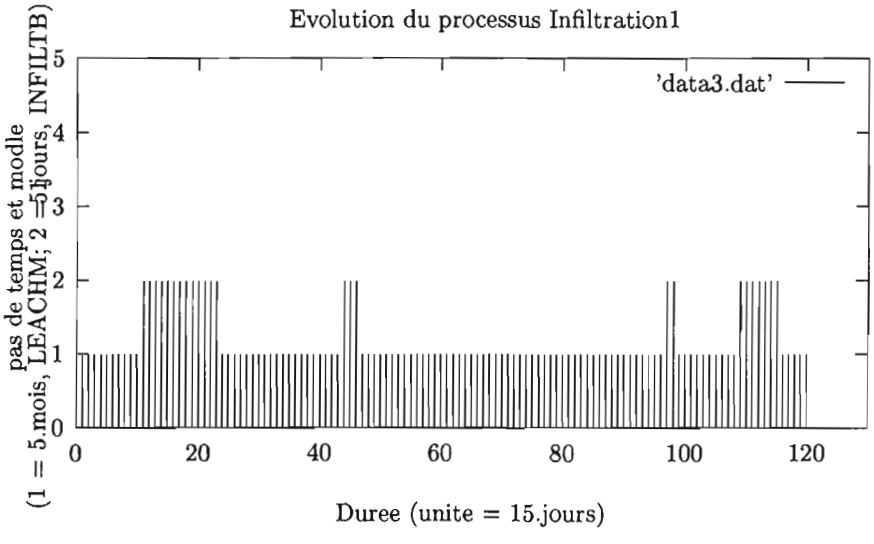
TAB. 10.10 – **Etat final du profil_0**.

	hauteur	concentrationNA	concentrationK	concentrationMg	concentrationCa
0,0	92	2.5	5.5	3.1	5.2
0,1	79	2.1	5.8	3.2	4.2
0,2	80	2.0	5.7	3.3	6.2
1,0	84	2.7	6.5	3.3	3.2
1,1	80	2.3	7.5	3.5	5.2
1,2	78	2.2	5.5	3.2	5.5
2,0	82	2.5	6.2	3.4	5.1
2,1	92	2.3	7.7	3.5	6.2
2,2	85	1.4	5.5	3.3	5.5
3,0	87	2.3	6.6	3.3	3.6
3,1	75	2.2	4.4	3.6	6.6
3,2	80	2.7	5.4	3.4	4.4

TAB. 10.11 – **Etat final de la nappe**.

10.4.2 Exemple d'évolution d'un processus

Le graphique ci-dessous représente l'évolution du processus infiltration_1 au cours de la simulation. Le processus infiltration_1 évolue entre deux états : dans l'état 1, il est représenté par le modèle LEACHM et possède un pas de temps de 5 mois. Dans le l'état 2, sa dynamique est dirigée par le modèle INFILB et son pas de temps est de 15 jours.



Conclusion

Cet exemple de simulation, donné à titre illustratif ne sera pas davantage commenté ici. Dans l'état actuel de l'outil OSIRIS, les simulations effectuées permettent de vérifier le fonctionnement interne de l'architecture (pas d'interblocages entre processus, bon fontionnement des différents composants). L'architecture permet donc de coupler des processus. Une vérification des données produite par le couplage (validation générale de la démarche de couplage, validation spécifique - par rapport à la région et le phénomène étudiés - du choix des entités et des processus, des modèles et des règles de contrôle) doit comparer les résultats fournis par le système initialisé avec des données réalistes aux sorties de modèles intégrés portant sur la même problématique. Cela fait partie de nos futurs travaux.

Quatrième partie

Discussion

Chapitre 11

Discussion

RÉSUMÉ :

Ce chapitre discute l'intérêt de l'outil OSIRIS (quels apports et quelles limites). Nous comparons OSIRIS à d'autres outils de couplage (PLM et modèle Belouze). Une des limites est le manque de validation de l'outil OSIRIS par rapport à des données de terrain. Les problèmes de la validation dans l'utilisation de l'outil OSIRIS sont discutés. L'un des principaux apports d'OSIRIS est la flexibilité qu'il apporte dans une démarche de couplage de modèles environnementaux. Le travail effectué s'est concentré sur l'aspect dynamique du couplage (synchronisation des pas de temps, contrôle des comportements). Concernant l'aspect statique, la réflexion a débouché sur le choix d'un système de représentation pivot. Les problèmes liés aux fonctions de correspondance entre système pivot et représentations particulières aux modèles, bien qu'abordés, méritent d'être approfondis : il faut proposer des cadres plus formels pour la mise en correspondance entre attributs et paramètres, dans le cadre d'une utilisation sur des données réelles, il faut réfléchir à la possibilité de l'utilisation conjointe d'un système d'information géographique pour une meilleure visibilité du système par essence spatialisé.

11.1 Comparaison d'OSIRIS à PLM et au modèle du Chishtian

Afin de mieux déterminer les apports et les limites de l'architecture OSIRIS, il est intéressant de tenter une comparaison par rapport à d'autres outils de couplage. Le modèle de Belouze et le modèle PLM déjà présentés au début de ce document, peuvent servir de base de comparaison. Ce sont en effet deux architectures de couplage de processus environnementaux. Le modèle PLM est comme OSIRIS présenté en tant qu'architecture générique de couplage, le modèle de Belouze s'attaque au même type de problème qu'OSIRIS dans son premier exemple d'application (étude de la salinité).

Les différences entre ces différents modèles peuvent être représentés comme le montre le tableau suivant :

	PLM	**modèle intégré Belouze**	**OSIRIS**
Type	environnement de couplage pour dynamiques physiques et socioéconomiques	modèle intégré d'une région irriguée	architecture de couplage de processus
Caractéristiques	environnement de couplage graphique (SME), langage de modélisation (MML) pour construire l'unité de modèle, utilisation d'un SIG	paramétrage possible par l'utilisateur	interface utilisateur pour définir les caractéristiques des processus à appliquer sur le système à simuler
Validation	validé et calibré pour la région Paxutent	faisabilité démontrée (vraissemblance des résultats du modèle en sortie par rapport aux données du terrain)	vérification du fonctionnement interne
Flexibilité (ajout de modèle)	nécessite la reconstruction de l'unité de modèle	nécessite la reconstruction du modèle intégré	ajout sans modification du système
Généricité (application possible à d'autres contextes)	oui - appliqué à d'autres région inondables	non	oui - à réaliser effectivement

TAB. 11.1 – *Tableau comparatif entre les modèles PLM, Belouze et OSIRIS*

Cette comparaison doit cependant tenir compte de la différence des contextes dans lesquels ces outils ont été proposés. PLM est le résultat d'un ensemble de travaux réalisés par une importante équipe de recherche. Les idées à la base de la construction du modèle PLM et

des différents outils qui l'accompagnent (langage de modélisation MML, environnement de simulation graphique SME) ont été présentés dès 1990 [Costanza et al., 1990]. Le modèle de Belouze, réalisé lors d'un stage DEA de six mois a néanmoins pu bénéficier des résultats de travaux antérieurs réalisés ainsi que le modèle dans le cadre d'un projet d'amélioration de l'agriculture irriguée au Pakistan mené par l'Institut International pour le Management de l'Irrigation (IIMI) et le CEMAGREF.

11.2 Des limites...

Au niveau technique

OSIRIS est une application fortement distribuée. Cependant, nous supposons que le réseau est sûr, sans panne. Dans le cas contraire, aucun mécanisme de recouvrement de panne n'ayant été implanté, le système se bloque dans un état d'attente infinie. C'est une limite certaine au bon fonctionnement de l'outil proposé.
Une deuxième limite est relative aux outils proposés à l'utilisateur pour poser ses règles. Les possibilités d'expression de règles dans le système OSIRIS sont pour l'instant très fortement limitées.

Le problème de la validation

Une démarche de modélisation implique la prise en compte de la question de la validité du modèle proposé [Cordier, 1999]. De façon classique, la validation d'un modèle s'effectue par confrontation entre les résultats fournis par le modèle et les données observées dans la réalité, ce qui dans ce travail n'a pas été réalisé. Dans notre problématique, une telle validation nécessite la prise en compte de trois types de validation :

le premier type réside dans la vérification de la validité des modèles numériques à intégrer ;

le deuxième type de validation se situe dans le contrôle du résultat obtenu par le mécanisme d'intégration. Dans notre cas, un contrôle dynamique est effectué en fonction d'un ensemble de règles posées par l'utilisateur. Il s'agit donc de vérifier la validité de ces règles;

le troisième type réside dans le respect des règles posées au cours de la simulation.

Seuls les deuxième et troisième types de validation concernent spécifiquement le système OSIRIS. On suppose que les modèles à intégrer ont déjà été validés auparavant. Le respect des règles au cours de la simulation peut être assez facilement vérifié par une démarche classique de debugage et de mise au point du système. Le deuxième type de validation pose un problème

conceptuel important : il faut être capable de définir et d'apprécier globalement ce qu'est une bonne simulation de processus intégrés (par exemple, conservation de la masse).

Enfin la validation peut s'appliquer au schéma conceptuel et à la démarche d'OSIRIS et à son caractère générique. L'attitude des modélisateurs, lorsqu'on leur présente l'outil, est de ce point de vue ouverte et positive. Ce " quatrième type " de validation passe par l'application effective de la démarche à d'autres contextes.

OSIRIS en tant que résultat d'une recherche informatique

OSIRIS est le fruit d'une recherche informatique réalisée à partir d'une problématique environnementale. La question informatique a émergé par l'exploration d'une question environnementale avec une perspective informaticienne. Ce cheminement, qui se reflète dans la rédaction de ce mémoire, et qui place cette thèse dans le domaine d'une informatique appliquée, situe l'apport de notre travail aux frontières de différents domaines allant de la modélisation à différents champs informatiques plutôt qu'à un domaine informatique précis. C'est une des caractéristiques des recherches en multi-agents d'être par essence multi-disciplinaires. Dans ce genre de sujet, le fil conducteur à suivre est rarement évident au départ. Celà explique que malgré les différents formalismes utilisés pour décrire le système proposé, il n'y a pas de sémantique (de description en un seul langage du système, d'algorithme) associé à notre travail. C'est (d'un point de vue strictement informatique)une limite de ce travail, mais cette démarche (partir d'un cadre non purement informatique) favorise la mise au jour de questions informatiques originales, intéressantes à traiter. Au delà de ses limites, OSIRIS est porteur d'un ensemble de contributions que nous présentons dans le paragraphe suivant.

11.3 ... aux apports de l'outil OSIRIS

Au niveau technique

OSIRIS permet d'appliquer à un système donné, le fonctionnement d'un ou de plusieurs modèles pouvant être situés sur des machines distantes. Il n'y a pas de modification interne des modèles de départ, tout le travail d'intégration est réalisé au niveau de la plate-forme de couplage OSIRIS.

OSIRIS est un outil d'intégration flexible. Les modèles à coupler étant intégrés par les interfaces processus dont les propriétés sont spécifiées par l'utilisateur, celui-ci a toute latitude pour les associer aux entités qu'il juge pertinentes. De plus, l'utilisateur dispose au moment du lancement du choix des instances d'entités qui vont effectivement se voir appliquées les

processus définis.

OSIRIS est un outil ergonomique. L'interface utilisateur est composée de menus, boites de dialogue, les zones d'entrée de données sont associées, lorsque cela est possible, à des listes créées en fonction des actions déjà réalisées (les attributs ne peuvent être choisis que s'ils appartiennent aux entités préalablement sélectionnées). Les entités spatiales sont représentées par des icônes qui permettent d'obtenir par simple clic de souris les attributs et leurs valeurs courantes. Ces propriétés d'interface utilisateur, classiques dans les logiciels actuels ne sont pas toujours disponibles pour la manipulation de modèles de simulation. Le modèle MOC par exemple, requiert l'utilisation judicieuse, dans la spécification du fichier d'entrée, des caractères de tabulation, espace et ligne vide, ces caractères ayant une signification pour le modèle (une ligne vide dans la description de la grille signifie la présence d'une ligne de données dont valeurs sont spécifiées ailleurs, l'introduction malencontreuse d'une ligne vide peut aboutir au plantage du modèle : pas de cohérence entre la description de la grille et son contenu). Les fichiers étant générés par OSIRIS, ce genre de désagrément échappe à l'utilisateur.

OSIRIS réutilise des outils existants (plate-forme Zeus). Cette réutilisation qui a permis un gain de temps pour la conception de l'architecture, permettra également de bénéficier des améliorations futures du produit. Cela démontre l'intérêt des travaux de conception de plate-formes opérationelles dans le domaine des systèmes multi-agents.

Au niveau modélisation

OSIRIS permet une prise en compte explicite du temps de la simulation. La prise en compte du temps en simulation se fait traditionnellement de façon implicite par l'adoption d'un pas de temps constant, ce qui contraint le modélisateur dans sa démarche. Les phénomènes complexes sur lesquels il se penche ne peuvent pas toujours être modélisés à pas de temps uniformes (imprévisibilité du devenir). Une prise en compte du temps explicite permet au modélisateur de mieux s'adapter au processus qu'il modélise. Et de rester conscient des simplifications qu'il effectue.

OSIRIS permet d'effectuer le couplage au niveau du modélisateur (utilisateur) et non au niveau de l'informaticien : le couplage est réalisé par l'utilisateur via le renseignement d'écrans contrairement à la démarche classique de couplage qui se réalise par une phase de programmation.

OSIRIS oblige le modélisateur à décrire le monde à simuler d'une manière définie et formalisée en terme d'entités, de processus, de règles de contrôle à appliquer. Cette caractéristique aboutit à un système de représentation pivot sur lequel vont se greffer les modèles à intégrer.

Elle est bénéfique, car elle force à l'explicitation des choix de modélisation et facilite donc le dialogue interdisciplinaire.

OSIRIS permet une prise en compte explicite et souple du temps de simulation. Dans OSIRIS, plusieurs pas de temps peuvent être définis et contrôler l'exécution des processus. Un processus peut changer de pas de temps en cours de simulation.

Au niveau capitalisation de connaissances
OSIRIS permet de récupérer des métaconnaissances sur les modèles. Il s'agit de toutes les connaissances implicites que l'utilisateur exprime sous forme de règles de contrôle lors de la spécification des processus. La récupération de connaissances implicites est une préoccupation constante en gestion des connaissances *Knowledge Management* et l'approche utilisée dans OSIRIS peut être rapprochée de celle utilisée pour le système SYRCLAD [Guin-Duclosson, 1999; Nova et Guin-Duclosson, 2000]. Ce système propose un cadre général permettant d'expliciter de manière déclarative des connaissances de reformulation et de résolution sur un problème puis de manipuler ces connaissances pour résoudre le problème, moyennant la définition préalable par l'utilisateur expert d'un ensemble de connaissances (règles, méthodes, etc.).

11.4 Les développements souhaitables

L'essentiel de notre réflexion s'est concentré sur l'aspect dynamique du couplage (synchronisation des pas de temps, contrôle des comportements). Les concepts d'agent de contrôle et de contrôleurs de temps ont été formalisés et implantés. Ce n'est pas le cas du concept de fonction de correspondance: dans la version actuelle du système, une fonction de correspondance est construite de façon *ad-hoc*, il n'y a pas de réelle formalisation de cette notion. Un aspect important du problème de correspondance entre types de données n'a pas été abordé: il s'agit de la gestion des unités (dans notre cas exemple, le modèle MOC utilisant des unités génériques, le problème a été évité).

OSIRIS a vocation à fonctionner sur des données réelles, pas seulement sur des cas d'école. Il faut pouvoir instancier les entités d'OSIRIS avec le contenu d'une base de donnée déjà renseignée par ailleurs. La plupart des données manipulées par OSIRIS sont géoréférencées. Elles bénéficient dans OSIRIS d'une représentation graphique symbolique (objets icôniques) dont le pouvoir expressif est sommaire. Dans ce contexte, une réflexion sur l'utilisation conjointe d'un système d'information géographique (SIG) a été engagée, grâce à l'encadrement d'un stage DESS [Poulet, 2000]. Ce stage a permis d'entamer la réflexion sur l'utilisation du SIG

comme interface d'instanciation des entités et des processus. L'emploi du SIG est également utile comme interface pour l'affichage des résultats de la simulation.

Chapitre 12

Conclusion

La question du couplage faible, également abordée sous le terme d'interopérabilité est un problème d'actualité, dans un environnement informatique caractérisé par l'avènement des réseaux. Si de nombreux travaux de recherche s'attaquent à ce problème pour les domaines industriels, la modélisation écologique manque encore d'outils adaptés à son contexte (systèmes caractérisés par l'interaction de multiples processus, que l'on veut modéliser et simuler en intégrant des connaissances provenant de plusieurs disciplines). Dans cette thèse, nous avons proposé une architecture multi-agents de couplage adaptée à ce type d'environnement. Nous avons appliqué cette architecture sur un exemple d'étude de la dynamique saline d'une région irriguée.

Dans le premier chapitre de cette thèse, nous avons introduit le problème du couplage à partir d'exemples pris dans différents domaines (industriels et environnementaux). Nous avons dégagé les principaux problèmes que posent les différentes démarches de couplage.

Après avoir montré dans un second chapitre en quoi l'étude de la salinité d'une région irriguée est une question de couplage, nous avons présenté dans une partie état de l'art des travaux de recherche qui utilisent la notion d'agent afin de connecter des applications indépendantes.

Nous avons ensuite présenté le système OSIRIS sous différents angles. D'un point de vue applicatif, nous avons illustré le fonctionnement du système pour la simulation d'une région irriguée afin d'étudier sa dynamique saline. Les chapitres suivant ont permis d'expliciter les concepts proposés, l'architecture interne et les détails d'implémentation.

Notre réflexion sur le couplage s'est effectuée autour de trois points clés :
(1) comment permettre le lien entre les différentes informations manipulées par les modèles; pour cela nous introduisons l'idée de système pivot et le concept de fonction de correspondance. Ces notions grâce auxquelles l'application d'OSIRIS à d'autres contextes de simulation

devient possible, assure le caractère générique du modèle.

(2) comment gérer la synchronisation temporelle entre les modèles à coupler; pour cela nous proposons la notion de controleur de temps. Cette notion permet de rendre dynamique le temps de la simulation.

(3) comment maîtriser le fonctionnement des modèles en cours de simulation; pour cela nous introduisons le concept d'agent de contrôle. Ce concept permet l'adaptation dynamique des modèles en fonction du contexte de la simulation.

Notre travail se veut également une contribution au domaine de la gestion des connaissances : il révèle et permet de récupérer des connaissances implicites (des métaconnaissances) détenues par le modélisateur sur les codes qu'il veut intégrer.

Au niveau technique, nous avons choisi de réutiliser des outils existants (utilisation de la plate-forme ZEUS) pour bénéficier de l'amélioration future de ce produit. Cette démarche valide et justifie les travaux précédents en systèmes multi-agents, elle a permis par ailleurs un gain de temps pour la conception de l'outil.

Les travaux environnementaux auxquels nous nous interessons dans cette thèse, opèrent essentiellement sur des données spatialisées. Il serait intéressant d'orienter la réflexion vers l'utilisation de Systèmes d'Informations Géographiques en relation avec l'outil que nous avons construit. Cette réflexion a débuté par l'encadrement d'un stage de DESS (Poulet, 2000), mais il doit se poursuivre. Ce travail permettra notamment d'intégrer des représentations cartographiques dans les interfaces d'OSIRIS et de présenter de façon plus visuelle les résultats obtenus par le déroulement de la simulation.

Références bibliographiques

Agha, G. (1986). *Actors: A Model of Concurrent Computing in Distributed Systems.* The MIT Press, Cambridge, Massachusetts.

Agre, P. et Chapman, D. (1987). PENGI : an implementation of a theory of activity. Dans *proceedings of th 6th national conference on artificial intelligence*, pages 268–272, San Mateo, CA. Morgan Kaufmann.

Allègre, C.-J. et Michard, G. (1973). *Introduction à la géochimie.* Presses Universitaires de France.

Appleby, S. et Stewaed, S. (1994). Mobile software agents for control in telecommunications networks. *BT Technological journal*, 12(2):104–113.

Attonaty, J. M., Chatelin, M. H., Poussin, J. C. et Soler, L. G. (1990). Un simulateur à base de connaissance pour raisonner équipement et organisation du travail en agriculture. Dans Bourgine, P. et B Walliser, P., éditeurs, *Economics and Artificial Intelligence*, pages 291–297.

Babovic, V. (1996). *Emergence, Evolution, Intelligence; Hydroinformatics.* Balkema Publishers, Rotterdam.

Banks, J., éditeur (1998). *Handbook of simulation : Principles, Methodology, Advances, Applications and Practice.* Wiley and Sons.

Banks, J. (2001). The Future of Simulation. Dans *actes de la troisième conférence francophone de MOdélisation et SIMulation*, volume 3.

Belouze, P. (1996). Un modèle intégré d'un système irrigué par la prise en compte de phénomènes hydrauliques, économiques et hydro-pédologiques : application sur le périmètre irrigué de Chishtian, Penjab sud, Pakistan. mémoire de DEA, Université de Montpellier, Montpellier, France.

Benslimane, D., Yetongnon, K., Chraibi, S. et Abdelwahed, E. H. (1998). DECA : une architecture multi-agents pour l'interopérabilité de bases de données hétérogènes. Dans Tchuente, M., éditeur, *Acte du 4ème colloque africain sur la recherche en informatique*, pages 269–281.

Blondel, J. (1995). *Approche écologique et évolutive.* Collection écologie num. 27, éditions Masson.

Boissier, O. et Demazeau, Y. (1997). Une architecture multi-agents pour des systèmes de visions ouverts et décentralisés. *Techniques et Sciences Informatiques*, 16(8):1039–1062.

Boivin, P. (1995). Dégradation des terres irriguées dans la moyenne vallée du fleuve sénégal , mécanismes état et caractérisation. Dans *Atelier ADRAO*. ADRAO.

Boivin, P. (1997). Soil degradation in irrigation schemes in the senegal river middle valley : Mechanisms, characterization methods and actual situation. Dans K.M. Miézan, M.C.S. Wopereis, M. D. J. D. et Randolph, T., éditeurs, *Irrigated rice in the Sahel : Prospects for Sustainable development.*, pages 37–49. West Africa Rice Development Association, Bouaké.

Boivin, P., Dia, I., Lericollais, A., Poussin, J., Santoir, C. et Seck, S., éditeurs (1995). *Nianga, laboratoire de l'agriculture irriguée en moyenne vallée du Sénégal.*

Boivin, P. et Poussin, J.-C. (1997). Les paysans du Fleuve et la culture du riz. *Echanges*, (9). Edité par l'ORSTOM-Dakar.

Bond, A. H. et Gasser, L., éditeurs (1988). *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, San Mateo, CA.

Boudjlida, N. (1996). Environnements Logiciels Intégrés et Base de Données. Dans *Forum Internationel Informatique et Applications - FIIA '96.*

Bourgat, J.-F., LeTallec, P., Mallinger, F., Perthame, B. et Qiu, Y. (1994). Couplage Boltzmann Navier-Stokes. technical report RR2281, INRIA, Unité de recherche INRIA Rocquencourt.

Bousquet, F., Cambier, C., Mullon, C., Morand, P. et Quensière, J. (1994). Simulating fishermen's society. Dans Gilbert, N. et J.Doran, éditeurs, *Simlating societies, the computer simulation of social phenomena.* UCL Press.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.

Calvin, J. O. et Weatherly, R. (1996). An introduction to the high level architecture (hla) runtime infrastructure (rti). Dans *Proceedings of the 14th DIS Workshop*, volume 14, pages 103–114.

Cassandras, C. G. et Lafortune, S. (1999). *Introduction to Discrete Event Systems.* Kluwer Academic Publishers.

Cercignani, C. (1998). *Ludwig Boltzmann, the man who trusted atoms.* Oxford University Press.

Chandy, K. M. et Misra, J. (1981). Asynchronous distributed simulation via a sequence of parallel computations. *Communication ACM*, 24(11):198–205.

Chandy, K. M. et Sherman, R. (1989). Space-time and simulation. Dans *Distributed Simulation.* Society for Computer Simulation.

Checkland, P. (1976). Science and the system paradigm. *International of general systems*, 3:127–134.

Checkland, P. (1981). *Systems thinking, systems practice.* John Wiley and son, London.

Chitty, D. (1960). Population processes in the vole and their relevance to general theory. *Canadian journal of zoology*, 38:99–113.

Chung, P.-Y. E., Huang, Y., Yajnik, S., Liang, D., Shih, J. C., Wang, C.-Y. et Wang, Y.-M. (1998). Dcom and corba side by side, step by step, and layer by layer. *C++ Report*, 10(1):18–29.

Collis, J. et Ndumu, D. (1999). Zeus technical manual. manuel technique, Intelligent Systems Research Group BT Labs.

Coquillard, P. et Hill, D. R. (1997). *Modélisation et simulation d'écosystèmes. Des modèles déterministes aux simulations à événements discrets.* Masson.

Cordier, M.-O., éditeur (1999). *actes de la journée autour de la validation de modèles traitant de processus complexes.*

Costanza, R., Sklar, F. et White, M. (1990). Modeling coastal landscape dynamics. In [Voinov et al., 1999], pages 91–107.

DeAngelis, D. L. et Gross, L. J. (1992). Preface. Dans DeAngelis, D. L. et Gross, L. J., éditeurs, *Individual-based models and approaches in ecology - populations, communities and ecosystems*, New York. Chapman and Hall.

DeAngelis, D. L. et Rose, K. (1992). Which individual-based approch is most appropriate for a given problem? Dans DeAngelis, D. L. et Gross, L. J., éditeurs, *Individual-based models and approaches in ecology - populations, communities and ecosystems*, New York. Chapman and Hall.

Decker, K., Pannu, A., Sycara, K. et Williamson, M. (1997). Designing behaviors for information agents. Dans *Proceedings of the 1st International Conference on Autonomous Agents.*

Deelman, E. et Szymanski, B. K. (1997). Continuously monitored global virtual time. Dans *Proceedings of the International Conference on Parallel and Distributed Processing, Techniques and Applications*, Las Vegas, NV.

Delepoulle, S., Preux, P. et Darcheville, J.-C. (2001). Dynamique de l'interaction. Dans *Modèles Formels de l'Interaction.*

Dias, D., Fox, G., Furmanski, W., Mehra, V., Natarajan, B., Ozdemir, H., Pallickara, S. et Ozdemir, Z. (1998). Exploring JSDA, CORBA and HLA based Mutech's for Scalable Televirtual (TVR) Environments. Dans *proceedings of the Workshop on Object Orientation and VRML.*

Diaw, B. (1996). *Modélisation numérique des trasnferts d'eau en milieux poreux non saturés : le modèle MHNS_2D.* thèse de doctorat, Université de Strasbourg.

Dingkuhn, M., LeGal, P. et Poussin, J.-C. (1995). RIDEV : un modèle de développement du riz pour le choix des variétés et des calendriers. pages 205–222. Orstom éditions.

Drogoul, A. (1993). *De la simulation multi-agent à la résolution collective de problèmes.* Phd thesis, Université Paris VI.

ElFallah-Seghrouchni, A., Haddad, S. et Mazouzi, H. (2001). A formal study of interactions in multi-agent systems. *International Journal of Computers and their Applications*, 8(1):23–32.

Erard, P.-J. et Déguénon, P. (1996). *Simulation par événements discrets - concepts et réalisations en Simula, Ada et Smalltalk.* Presses Polytechniques et Universitaires Romandes.

Farenc, N., Musse, S. R., Schweiss, E., Kallmann, M., Aune, O., Boulic, R. et Thalmann, D. (1998). One step towards virtual human management for urban environment simulation. Dans *ECAI 98 Workshop on Intelligent Human Interface*, Brighton, England.

Farenc, N., Musse, S. R., Schweiss, E., Kallmann, M., Aune, O., Boulic, R. et Thalmann, D. (2000). A paradigm for controlling virtual humans in urban environment simulations. *Applied Artificial Intelligence Journal*, 14(1):69–91. Special Issue on Intelligent Virtual Environments.

Ferber, J. (1994). Simulating with reactive agents. Dans Hillebrand, E. et Stender, J., éditeurs, *Many agent simulation and Artificial Life*, pages 8–28. IOS Press.

Ferber, J. (1998). Introduction à la modélisation et aux systèmes multi-agents. Dans *Compte-rendu séminaire "Modélisation et systèmes multi-agents : présentation et perspectives au Cemagref".*

Fianyo, E. (2000). Simulation des systèmes techniques. Dans *tutoriel "Simulation multi-agents pour l'étude des systèmes naturels : théories, plates-formes et exemples d'application"*, Journées de formation avancée du 5ème CARI. Antananarive,Madagascar. `http://www.bondy.ird.fr/lia/transparentsCARI/finaleCARI.htm`.

Fianyo, E. et Boivin, P. (2000). Coupling Dynamical Physical Processes with Agents. a conference on Group Decision and Negociation. Glasgow, Scotland, UK.

Fianyo, E., Senghor, J.-P., Perrier, E., Boivin, P. et Treuil, J.-P. (1998a). Une démarche informatique pour simuler le fonctionement hydro-salin d'un système irrigué au nord sénégal. Dans Tchuente, M., éditeur, *actes du 4ème Colloque Africain sur la Recherche en Informatique.*

Fianyo, E., Treuil, J.-P. et Demazeau, Y. (1997). Simulation de l'exploitation des perimetres irrigues :etude sur la coordination reactive par apprentissage dans l'utilisation d'une ressource. Dans *actes des 5èmes Journées Francophones Intelligence Artificielle Distribuée Systèmes Multi-Agents (JFIADSMA).*

Fianyo, E., Treuil, J.-P., Demazeau, Y. et Pinson, S. (2001). OSIRIS : une architecture multi-agents de couplage de processus physiques pour simuler le fonctionnement hydrosalin d'une région irriguée. Dans Dolgui, A. et Vernadat, F., éditeurs, *actes de la 3ème conférence francophone de MOdélisation et SIMulation*, volume 2.

Fianyo, E., Treuil, J.-P., Perrier, E. et Demazeau, Y. (1998b). Multi-agent architecture integrating heterogeneous models of dynamical processes: The representation of time. Dans Sichman, Conte et Gilbert, éditeurs, *proceedings of the workshop on Multi-Agent Systems and Agent-Based Simulation*, volume 1534.

Fitz, H. C., DeBellevue, E., Costanza, R., Boumans, R., Maxwell, T., Wainger, L. et Sklar, F. (1996). Development of a general ecosystem model for a range of scales and ecosystems. *Ecological Modelling*, 88(1):263–295.

Freire-Junior, J. C. (1997). *Ingénierie des systèmes d'information: une approche de multi-modélisation et de méta-modélisation.* thèse de doctorat, Université Joseph Fournier - Grenoble I.

Fujimoto, R. M. (1990). Parallel discrete event simulation. *CACM*, 33(10):30–53.

Fujimoto, R. M. (1993). Parallel discrete event simulation: will the field survive? *ORSA Journal on computing*, 5(3):245–248.

Gabrielle, B., Gosse, G. et Nicolardot, B. (1997). Différentes échelles de temps et de complexité dans la modélisation des bilans environnementaux des cultures.

Galyon, E. (1998). C++ vs Java Performance. rapport technique, Colorado State University.

Gasser, L., Braganza, C. et Herman, N. (1987). *MACE: A flexible testbed for Distributed AI Research.* Pitman Publishing/Morgan Kauffmann Publishers, San Mateo, CA.

Gasser, L. et Briot, J.-P. (1998). (Jean-Pierre Briot interviews Les Gasser on) Agents and concurrent objects. *IEEE Concurrency*, 6(4):74–81. Special series on Actors and Agents, edited by Dennis Kafura and Jean-Pierre Briot.

Goldberg, A. et Robson, D. (1983). *SMALLTALK-80, the language and the implementation.* Addison-Wesley publishing company, Reading, Massachusetts.

Goode, D. et Konikow, L. (1989). Modification of a method-of-characteristics solute-transport model to incorporate decay and equilibrium-controlled sorption or ion exchange. Investigations Report 89-4030, U.S. Geological Survey Water-Resources. 65 pages.

Guin-Duclosson, N. (1999). SYRCLAD: une architecture de résolveurs de problèmes permettantd'expliciter des connaissances de classification, reformulation et résolution. *Revue d'Intelligence Artificielle. éditions Hermès*, 13(2).

Haddad, S. (1993). Parallélisme et généricité. Dans *congrès biennal de l'AFCET.*

Haddad, S., Ilie, J.-M. et Ajami, K. (2000). A model checking method for partially symmetric systems. Dans *proceedings of the 13th international conference on Formal Description Techniques for Distributed Systems and Communications Protocols (FORTE'91).*

Hasler, B., Wier, M. et Andersen, J. (1999). Integrated modeling of changes in agricultural policy, environmental and economic effects. Dans Nordisk-Jordbrugsforskning, éditeur, *Nordiske Jordbrugsforskeres Forening XXI Kongres*, volume 81, pages 450–458.

Hewitt, C. (1977). Viewing control structure as patterns of passing messages. *Artificial Intelligence*, 8(3):323–364.

Hewitt, C. et Atkinson, R. (1977). Parallelism and synchronization in actor systems. Dans *fourth ACM symposium on principles of programming languages POPL 1977*, pages 267–280, Los Angeles, California.

Hill, D. R. (1993). *Analyse orientée-objet et modélisation par simulation.* Addison Wesley.

Horstmann, M. et Kirtland, M. (1997). DCOM architecture. *MSDN Library.* `http://msdn.microsoft.com/library`.

Huget, M.-P., Koning, J.-L. et Bergia, L. (2000). Une plate-forme pour l'ingénierie des protocoles et son application au projet de télé-enseignement baghera. Dans Sylvie Pesty, C. S.-F., éditeur, *Systèmes multi-agents : méthodologie, technologie et expérience, 8emes Journées Francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents, JFIAD-SMA'00.*

Hutson, J. L., Wagenet, R. J. et Niederhofer, M. E. (1997). Leaching estimation and chemistry model: a process based model of water and solute movement, transformations, plant uptake and chemical reactions in the unsaturated zone. Versions LEACHF and LEACHG (for simulating nitrogen and phosphorus transformations, cycling and transport). Research report R97-1, Cornell University, Department of Soil, Crop and Atmospheric Sciences, Ithaca, New York. 138pages.

HydrologicStudiesUnit (2000). Criteria for Groundwater Modeling Reports. rapport technique, Land and Water Management Division, Department of Environmental Quality.

Irvine, D., Kristjanson, J., Robertson, S., Uso, J., Brebbia, C. et Power, H. (1998). PECOS: 150 person-years experience in teaching, learning and conducting interdisciplinary. Dans Uso, J. et Brebbia, C., éditeurs, *Advances in ecological sciences*, volume 1. Computational Mechanics Publications.

Jaber, A., Guarnieri, F. et Wybo, J. L. (1998). Un système d'agents logiciels intelligents pour favoriser la coopération entre des systèmes d'aide à la décision dédiés à la gestion de crise. Dans *actes des 6èmes JFIADSMA 98.* Hermes.

Jamin, J. (1995). Evolution des recherches agronomiques dans la vallée du fleuve sénégal. Dans [Boivin et al., 1995], pages 139–151.

Jefferson, D. R. (1985). Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3):404–425.

Jennings, N. R., Corera, J., Laresgoiti, I., Mamdani, E. H., Perriolat, F., Skarek, P. et Varga, L. Z. (1996). Using ARCHON to develop real-word DAI applications for electricity transportation management and particle accelerator control. *IEEE Expert*, 11(6).

Jolls, K. R. (1990). Gibbs and the art of thermodynamics. Dans Mostow, G. D. et Caldi, D. G., éditeurs, *Proceedings of the Gibbs Symposium*, pages 293–321, Providence, R.I. American Mathematical Society.

Judson, O. (1994). The rise of individual-based modelling in ecology. *Trends in Ecology and Evolution*, 9(1):9–14.

Kagel, J. H. et Roth, A. E., éditeurs (1995). *The Handbook of Experimental Economics.* Princeton University Press. Paperback edition,Fall 1997.

Kernan, D. (1996). Coupling finite element codes using corba-based environments. rapport technique, Department Of Energy - USA / Sandia Laboratories.

Khettry, D. et Sun, H. (2000). A Windows NT virtual collaboratory for technical computing. *Advances in Engineering software*, 31(8-9):717–722.

Kirkerud, B. (1989). *Object Oriented programming with SIMULA.* Addison-Wesley, Reading Massachusetts, New-York, Amsterdam.

Klein, U. (2000). Simulation-based distributed systems : serving multiple purposes through composition of components. *Safety Science*, 35(1-3):29–39.

Konikow, L., Granato, G. et Hornberger, G. (1994). User's guide to revised method-of-characteristics solute-transport model(moc–version 3.1). Investigations Report 94-4115, U.S. Geological Survey Water-Resources. 63 pages.

Kopetz, H. (1997). *Real-Time Systems - Design Principles for Distributed Embedded Applications*, volume 395 de *The Kluwer International Series in Engineering and Computer Science.* Kluwer Academic Publishers, Boston. ISBN 0-7923-9894-7.

Kozierok, R. et Maes, P. (1993). A learning interface agent for scheduling meetings. Dans *proceedings of ACM- SIGCHI International Workshop on Intelligent User Interfaces*, pages 81–93.

Lamport, L. (1978). Time, clocks and the ordering of events in a distributer system. *Communications of ACM*, 21:558–564.

Langton, C., éditeur (1994). *Artificial life III.*

LeFur, J. (1998). Adaptabilité potentielle de modèles de pêche artisanale tropicale (complexe) aux exploitations halieutiques méditerranéennes. *Gaps in Mediterranean Fishery Science, CIESM, Workshop series*, (5):31–39.

LeFur, J. et Bommel, P. (1998). Couplage d'un modèle multi-agents de la dynamique d'une ressource marine avec un modèle multi-agents de l'exploitation halieutique artisanale sénégalaise. Séminaire smas, INRA Montpellier.

LeMoigne, J.-L. (1980). *la théorie du système général, Théorie de la modélisation.* Collection Systèmes-Décisions, Presses Universitaires de France.

LePage, C. et Ginot, V. (1997). Vers un simulateur générique des peuplements piscicoles. Dans *Actes des 5èmes JFIADSMA.* Hermes.

Lericollais, A. et Sarr, A. (1995). Histoire de périmètres. Dans [Boivin et al., 1995], pages 5–36.

Leslie, P. (1945). On the use of matrices in certain population mathematics. *Biometrika*, 33:183–212.

Lesser, V. R. et Corkill, D. D. (1981). Functionally accurate, cooperative distributed systems. *IEEE Transactions on systems, Man and Cybernetics*, C11(1):81–96.

Liao, H. et Tim, U. (1997). an interacting modeling environement for non-point source pollution control. *Journal of the American Water Resources Association*, 33(3):591–604.

Litrico, X. (1995). Alternative scenarios for improved operations at the main canal level : a study of Fordwah Branch, Chishtian subdivision, using a mathematical flow simulation model. rapport de DEA, USTL-ENGREF.

Liu, L. et Pu, C. (1997). An adaptative object-oriented approach to integration and access of heterogeneous information sources. *Distributed and Parallel Databases*, 5:167–205.

Lutz, R. R. (2000). Migrating the HLA Object Model Template to an IEEE standard. *Johns Hopkins Apl. Technical Digest*, 21(3):337–347.

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40.

Marlet, S. (1996). *Alcalinisation des sols dans la vallée du fleuve Niger(Niger) - Modélisation des processus physico-chimiques et évolution des sols sous irrigation.* diplôme de doctorat en sciences agronomiques, Ecole Nationale Supérieure Agronomique de Montpellier.

Maxwell, T. et Costanza, R. (1995). Distributed Modular Spatial Ecosystem Modelling. *International Journal of Computer Simulation*, 5(3):247–262. special issue on Advanced Simulation Methodologies.

Maxwell, T. et Costanza, R. (1997). A language for modular spatio-temporal simulation. *Ecological Modelling*, 103(2/3):105–114.

Maxwell, T. et Villa, F. (1998). Collaborative Modular Modeling of Environmental Systems. Dans *proceedings of the 1998 Conference on Simulation Methods and Applications - CSMA98.*

McPhail, C. (1997). Stereotypes of the crowd and collective behavior revisited. Dans Michael Katovich, D. M. et Saxton, S., éditeurs, *Constructing complexity : symbolic interaction and social forms, a festchrift for Car J. Couch.* JAI Press, Greenwich, CN.

Minar, N., Burkhart, R., Langton, C. et Askenazi, M. The swarm simulation system: A toolkit for building multi-agent simulations. Santa Fe Institute working paper 96-06-042.

Misra, J. (1986). Distributed discrete event simulation. *Computing surveys*, 18(1):39–65.

Morin, E. (1990). *introduction à la pensée complexe.* E.S.F éditeur, Paris.

Müller, J.-P. (1994). Modelling interacting agents in dynamic environments. Dans *Proceedings of the 11th European Conference on Artificial Intelligence - ECAI'94.*

Müller, J.-P., éditeur (1997). *The Design of Intelligent Agents*, volume 1177 de *LNAI.* Springer-Verlag, Berlin, Germany.

Nilsson, N. J. (1994). Teleo-reactive programs for agent control. *Journal of ARtificial Intelligence Research*, 1:139–158.

Nova, N. et Guin-Duclosson, N. (2000). Liens entre l'apprentissage à partir d'exemples et le raisonnement à partir de cas - apports pour les environnements informatiques pour l'apprentissage humain. rapport interne RR2000-2, LISI.

Nwana, H. et Ndumu, D. (1996). An introduction to agent technology. Rapport technique, BT Laboratories.

Nwana, H. S. (1996). Software agents: an overview. *Knowledge engineering review, Cambridge University Press*, 11(3):205–244.

Nwana, H. S., Ndumu, D. T., Lee, L. et Collis, J. (1999). ZEUS: A tool-kit for building distributed multi-agent systems. *Applied Artifical Intelligence Journal*, 13(1):187–208.

OMG (1999). *The Common Object Request Broker: Architecture and Specification - CORBA/IIOP 2.3.1 Specification.* OMG.

Patris, S. (1997). Constitution d'un SIG sur la zone ngalenka amont pour une étude environnementale et socio-économique. DESS système d'information géographique, Université de Caen.

Picavet, M. (1997). *La complexité dans la modélisation du système d'information de l'entreprise : proposition de solutions concepts, outils et démarche.* Habilitation à diriger les recherches en sciences mathématiques, Université des Sciences et Technologies de Lille - Laboratoire d'Informatique Fondamentale de Lille.

Pichon, G. et Mullon, C. (1991). Le logiciel CINEFIL : simulation des relations hôtes - parasites. Dans *Sur les distributions rencontrées en parasitologies. SEMINFOR 5.*

Pinson, S. (1991). Surface Knowledge and Decision Models in a Credit Risk Assessment System. Dans *Workshop AI and Business AAAI'92.*

Pinson, S. (1992). Integrating diverse multiattribute information processing models in an evaluation expert system. Dans *proceedings of the Information Processing and Management of Uncertainty in Knowledge Based Systems IUPMU'92.*

Pohl, K., Weidenhaupt, K., Domges, R., Haumer, P., Jarke, M. et Klamma, R. (1999). PRIME - Toward process-integrated modeling environments. *ACM Transactions on Software Engineering and methodology*, 8(4):343–410.

Poulet, N. (2000). Couplage entre un SIG et un simulateur : réalisation de la partie SIG de ce couplage avec le logiciel Arcinfo. rapport de stage, IRD.

Pouliot, J. (1999). *Définition d'un cadre géosémantique pour le couplage des modèles prévisionnels de comportement et des SIG; application pour les écosystèmes forestiers.* thèse de doctorat, Ecole Polytechnique Fédérale de Lausanne, Lausanne.

Preux, P., Delepoulle, S. et Darcheville, J.-C. (2001). Selection of behaviors by their consequences in the human baby, software agents, and robots. Dans *Computational Biology, Genome Information Systems and Technology.*

Quensière, J., éditeur (1994). *La pêche dans le Delta Central du Niger. Approche pluridisciplinaire d'un système de production halieutique*, volume 1. ORSTOM-Karthala-IER,.

Radhakrishnan, R. (1997). Formal specification and discrete event simulation algorithm. rapport technique, dept. of ECECS - University of Cincinnati.

Reynolds, C. W. (1987). Flocks, herds and schools : a distributed behavioral model. Dans Stone, M. C., éditeur, *SIGGRAPH'87*, pages 25–34.

Ribeiro, A. (2000). *Un Modele d'Interaction Dynamique pour les Systemes Multi-Agents.* Thèse de doctorat, Université Joseph Fournier, Grenoble.

Rinaudo, J. D. (1994). Development of a tool to assess the impact of water markets on agricultural production in pakistan. rapport de DEA, Université de Montpellier I.

Rochette, C. (1974). Le bassin du fleuve Sénégal. Monographies hydrologiques ORSTOM. édition ORSTOM.

Rosenschein, J. S. (1982). Synchronization of multi-agent plans. Dans *Proceedings AAAI-82*, pages 115–119.

Rossiter, D. (1994). Land Evaluation Lecture Notes. manuel de cours, Cornell University, College of Agriculture and Life Sciences, Department of Soil, Crop and Atmospheric Sciences.

Rouchier, J., Requier-Desjardins, M. et Dugue, P. (1998). L'interdisciplinarité pour la modélisation dans la recherche-développement : une application aux relations élevage agriculture en zone soudano-sahélienne au cameroun. Dans *Actes de l'atelier : les flux de biomasse et la gestion de la fertilité à l'échelle des terroirs, Montpellier, France, 5-6 mai 1998.*

Russell, C. (1996). Integrating ecology and economics via regional modeling. *Ecological Applications*, 6(4):1025–1030.

Sarjoughian, H., Zeigler, B. et S.Park (2000). Collaborative distributed network system : a lightweight middleware supporting collaborative DEVS modeling. *Future Generation Computer Systems*, 17(2):89–105.

Schmerler, S., Tanurhan, Y. et Müller-Glaser, K. D. (1998). Advanced optimistic approaches in logic simulation. Dans *Proceedings of the 1998 Design Automation and Test in Europe (DATE '98)*. IEEE Computer Society.

Seguis, L. (1995). Hydrologie d'une cuvette du lit majeur du Sénégal : exemple de la cuvette de Nianga. Dans [Boivin et al., 1995], pages 49–66.

Senghor, J.-P. (1997a). Fonctionnement hydrosalin et dynamique socio-économique des systèmes irrigués sahéliens. projet de recherche, Université de Montréal.

Senghor, J.-P. (1997b). Un univers multi-agents pour évaluer le fonctionnement hydro-salin et la cinétique socio-économique du secteur ngalenka. Rencontre LIA du 03 au 15. 11.1997.

Shaffer, M. J. (1995). Fate and Transport of Nitrogen - What Models Can and Cannot Do. working papar 11, USDA, Agricultural Research Service, Great Plains Systems Research Unit, Fort Collins, Colorado.

Sharma, G. D., Radhakrishnan, R., Rajasekaran, U. K. V., Abu-Ghazaleh, N. et Wilsey, P. A. (1999). Time warp simulation on clumps. Dans *13th Workshop on Parallel and Distributed Simulation (PADS'99)*, pages 174–181.

Sheth, A. P. et Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236.

Shin, Y.-J. et Cury, P. (2001). Exploring fish community stability through trophic interactions using a spatialized individual-based model. *Aquatic Living Resources*. in press.

Shoham, Y. (1993). Agent Oriented Programming. *Artificial Intelligence*, 60(1):51–92.

Shrestha, P. (1996). An integrated model suite for sediment and pollutant transport in shallow lakes. *Advances in Engineering Software*, 27(3):201–212.

Skinner, J., Lewis, K., Bardon, K., Tucker, P., Catt, P. et Chambers, B. (1997). An overview of the environmental impact of agriculture in the U.K. *Journal of Environmental Management*, 50(2):111–128.

Sky, R. et Buchal, R. (1999). Modeling and implementing concurrent engineering in a virtual collaborative environment. *Concurrent Engineering Research and Applications*, 7(4):279–289.

Smets, S. (1996). Modeling the effects of irrigation management on soil salinity and crop transpiration at the field level. M.sc. thesis, Wageningen Agricultural University, Wageningen.

Smith, J. M. (1974). *Models in ecology*. Cambridge University Press.

Smith, R. G. (1980). The contract net protocol: high-level communication and control in distributed problem solver. *IEEE Transactions on computers*, C29(12):1104–1113.

Strosser, P. (1997). *Analyzing alternative policy instruments for the irrigation sector - an assessment of the potential for water market development in the Chishtian sub-division.* Phd thesis, Wageningen Agricultural University, the Netherlands. 243p, 39 figures, 40 tables, 3 appendices.

Sumpter, D. (1997). Investigating the organisation of insect societies using multi-agent models. talk at the Winter Meeting of the IUSSI.

Tambe, M. (1995). Recursive agent and agent group tracking in a real-time, dynamic environment. Dans *Proceedings of the 1st International Conference on Multi-Agent Systems.*

Thomas, S. R. (1994). The PLACA Agent Programming Language. Dans Wooldridge, M. et Jennings, N., éditeurs, *ECAI Workshop on Agent Theories, Architectures, and Languages*, volume 890. LNCS.

Travers, M. (1996). *Programming with Agents: New metaphors for thinking about computation.* Ph.d in media arts and sciences, MIT Media Laboratory, Cambridge, MA. Nominated for ACM Doctoral Dissertation Award.

Treuil, J.-P., Mullon, C., Perrier, E. et Piron, M. (2001). *Modèles en analyse spatiale*, chapitre Simulation multi-agents de dynamiques spatialisées. Hermès.

Treuil, J.-P., Perrier, E. et Cambier, C. (1997). Directions pour une approche multi-agents de la simulation de processus physiques spatialisés. Dans *actes des 5èmes JFIADSMA*, pages 221–228. Hermes.

Valentin, J. L. (1987). Ecological modelling, an objective tool for an integrated study of ecosystems. *Neritica*, 2:43–60.

Vedeld, P. (1994). The environment and interdisciplinarity : ecological and neoclassical economical approches to the use of natural resources. *Ecological Economics*, 10(1):1–13.

Veloso, M., Stone, P., Han, K. et Achim, S. (1997). CMUnited: a team of robotic soccer agents collaborating in an adversarial environment. Dans *Proceedings of the 1st International Workshop on RoboCup*.

Visser, S. (1996). Canal water distribution at the secondary level in the Punjab, Pakistan. M.sc. thesis, University of Technology, Delft.

Voinov, A., Costanza, R., Wainger, L., Boumans, R., Villa, F., Maxwell, T. et Voinov, H. (1999). Patuxent Landscape Model : integrated ecological economic modeling of a watershed. *Environmental Modelling and Software*, 14:473–491.

Wagenet, R. J. et Hutson, J. L. (1992). LEACHM: Leaching Estimation and Chemistry Model - A process based model of water and solute movement, transformations, plant uptake, and chemical reactions in the unsaturated zone, version 3. rapport de recherche, SCAS Research Series number 92-3, Cornell University, Department of Soil, Crop and Atmospheric Sciences, Ithaca, New York. 127pages.

Wittig, T., Jennings, N. R. et Mamdani, E. H. (1994). ARCHON - a framework for intelligent cooperation. *IEE-BCS Journal of Intelligent System Engineering*, 3(3):168–179. special issue on Real-time Intelligent Systems in ESPRIT.

Wooldridge, M. (1999). chapter : Intelligent agents. Dans Weiss, G., éditeur, *Multiagent systems*. The MIT Press.

Wooldridge, M. et Jennings, N. (1995). Intelligent agents : theory and practice. *Knowledge engineering review, Cambridge University Press*, 10(2).

Wooldridge, M. J. et Jennings, N. R. (1999). Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, 3(3).

Zante, P. (1995). Etude du comportement physique de sols argileux soumis à l'irrigation dans la moyenne vallée du fleuve Sénégal : distribution d'agrégats et courbe de retrait. Dans ORSTOM, éditeur, *Structure et fertilité des sols tropicaux : deuxième réunion du groupe thématique, 12-13 septembre 1994*.

Liste des tableaux

Table des figures

Annexes

Annexe A

Mécanismes chimiques de la salinisation

Les eaux d'irrigation contiennent de nombreux sels chimiques dissous (chrorure de sodium, sulfate de calcium, etc.) En milieu liquide (rivières, canaux) les concentrations de ces sels sont insuffisantes pour conduire à leur précipitation. Mais une fois que ces eaux sont répandues sur les champs, les pertes par évaporation qui ne concernent que l'eau solvant amènent à une augmentation de la concentration de ces sels dans la phase liquide du sol et cela peut aboutir (si les exportations de sels hors du milieu considéré sont insuffisantes) à leur précipitation. Il en est de même pour l'eau du sol qui peut atteindre la surface directement par remontée capillaire et déposer les sels dans les couches superficielles du sol.

Un équilibre de précipitation s'écrit $aA + bB \xleftrightarrow{K_s} A_aB_b$, K_s étant le produit de solubilité du précipité. A la saturation[1] on a l'égalité $K_s = [A]^a.[B]^b$ où $[X]$ indique la concentration du soluté X dans la solution.

Cette égalité a pour conséquence une loi dite loi de T (figure A.1): comme K_s est constant, quand une solution se concentre du fait de l'évaporation, les concentrations de A et B ne peuvent pas augmenter toutes les deux une fois que la saturation est atteinte.
L'eau du fleuve Sénégal est de composition constante au cours de l'année. Elle est très peu chargée ($58\mu S/cm$ de conductivité électrique), de pH neutre (6.9 in situ) et plus riche en calcium qu'en sodium. Elle est cependant légèrement excédentaire en carbonates (une base faible). Lors d'une phase de concentration, le premier minéral qui va précipiter est la calcite (carbonate de calcium) suivant la réaction :

$$2HCO_3^- + Ca^{2+} \xleftrightarrow{K_s} CaCO_3 + CO_2 + H_2O$$

Soit le coefficient d'alcalinité résiduelle calcique ARC $= [HCO_3^-] - 2[CA^{2+}]$

- si ARC est inférieur à 0, $[HCO_3^-]$ diminue et $[CA^{2+}]$ augmente pendant la concentration.

1. Etat d'une solution en présence de la quantité maximale de soluté qu'elle peut dissoudre à une température donnée

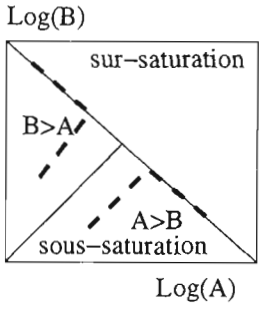


FIG. A.1 – *Loi de T: diagramme de saturation par rapport à un minéral AB. [Marlet, 1996]*

- si ARC est supérieur à 0, $[HCO_3^-]$ augmente et $[CA^{2+}]$ diminue pendant la concentration,

Le sol est un complexe argilo-humique que l'on peut concevoir grossièrement comme un ensemble de lamelles argileuses entourées de cations [majoritairement calcium(Ca^{2+}), sodium (Na^+), potassium (K^+) et magnésium (Mg^{2+})][2]. Ces cations assurent d'une part la cohésion électrostatique à la partie solide du sol, et d'autre part sont en contact avec la solution du sol et constitue le complexe d'échange cationique.

Lorsque ARC est inférieur à 0, l'augmentation de Ca^{2+} ne modifie pas la structure du sol où Ca^{2+} est déjà majoritaire. L'accumulation des calcium a donc avant tout des conséquences de toxicité pour les plantes. On obtient ainsi des sols dits salins.

Lorsque ARC est supérieur à 0, l'eau est excédentaire en anions carbonates(HCO_3^-). Lorsqu'une concentration de l'eau conduit à une précipation de la calcite, les HCO_3^- deviennent un peu plus excédentaires. En cas de nouvelle précipitation, celle-ci va s'effectuer pour une teneur en calcium plus faible et une teneur en anions HCO_3^- plus élevée. Le phénomène s'auto-amplifie et aboutit à une eau fortement déséquilibrée, riche en carbonates et donc alcaline et très pauvre en calcium donc sodisante. Cette accumulation de bases faibles va provoquer une hausse du pH du sol, le rendant impropre à la culture. C'est le phénomène d'alcalinisation.

Par ailleurs, la diminution des cations Ca^{2+} a d'autres conséquences, cette fois sur la structure du sol. En effet, les cations sodium (Na^+) tendent à remplacer le calcium sur le complexe d'échange. Du fait d'une structure électrostatique différente, la stabilité du sol peut être modifiée : des phénomènes de gonflement des argiles peuvent se produire en cas de brusques apports d'eau, puis un entraînement des sodium par dilution qui peut conduire à un tassement des lamelles d'argile (du fait de l'absence de répulsion électrostatique). Les sols peuvent alors se

2. La croûte terrestre de 6 à 70 km d'épaisseur est composée de 46% d'oxygène, 28% de silicium, 5% de fer, 3,6% de calcium, 2,8% de sodium, 2,6% de potassium, 2% de magnésium et 2% d'autres éléments [Allègre et Michard, 1973]

dégrader physiquement (en particulier diminution de la perméabilité hydraulique), et alors dits sodiques, caractéristiques de processus de sodisation.

L'eau du fleuve Sénégal étant exédentaire en carbonates, les processus alcalinisation et sodisation disposent de conditions favorables sur les périmètres irrigués.

Annexe B

L'architecture d'un agent ZEUS

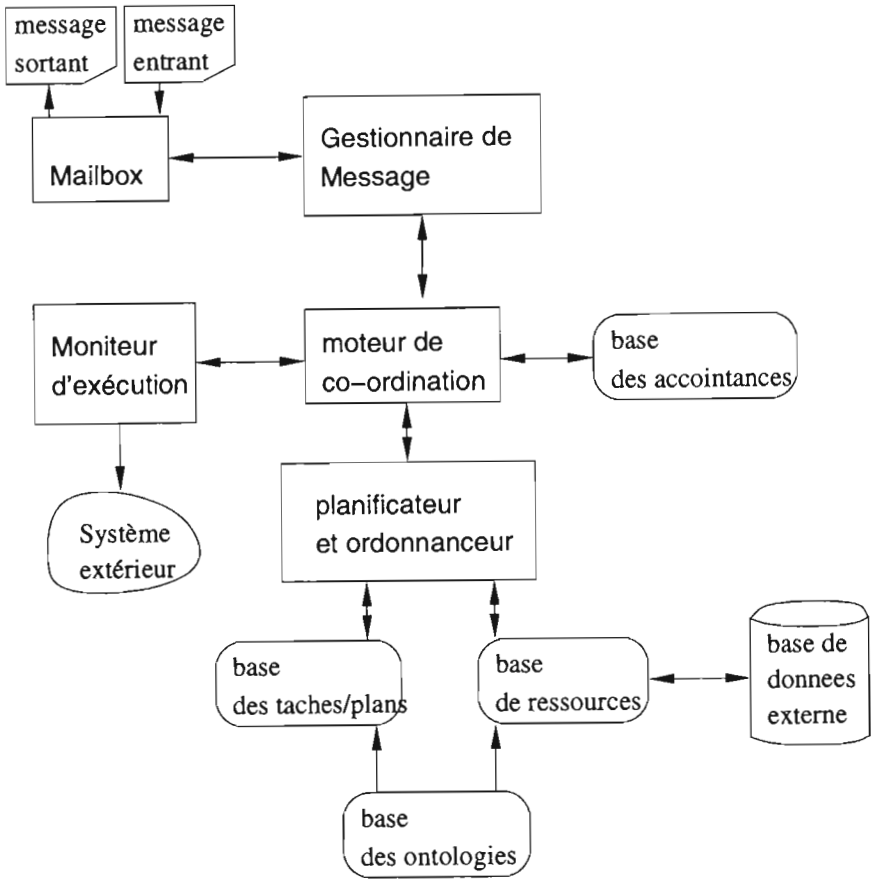


FIG. B.1 – *Architecture d'un agent ZEUS [Nwana et al., 1999]*

B.1 Les composants

Comme le montre la figure B.1, un agent générique ZEUS inclut les composant suivants :
une Mailbox permet de gérer la communication entre l'agent et les autres agents du système.

un Gestionnaire de message traite les messages en entrée provenant de la Mailbox, et les envoie composants appropriés de l'agent.

un moteur de co-ordination prend les décisions relativees aux buts actuels de l'agent (quels sont les buts à poursuivre, quand les abandonner, etc). C'est ce composant qui met en œuvre la coordination entre agents par l'application de protocoles et de stratégies de coordination (contract net protocol, par exemple).

une base des accointances décrit les relations (pairs, supérieur) de l'agent avec les autres agents de la société, ses croyances concernant les capacités des autres agents. Le moteur de co-ordination utilise les informations contenues dans cette base lorsqu'il veut collaborer avec d'autres agents.

un planificateur-ordonnanceur planifie les actions de l'agent en fonction des décisions prises par le moteur de co-ordination. C'est à son niveau que sont spécifiés les ressources (également appelés faits) et les services utilisables par l'agent.

une base de ressource contient la liste des ressources qui appartiennent à l'agent ou lui sont accessibles. La base de ressources sert d'interface aux systèmes extérieurs, elle permet l'utilisation de données provenant de bases de données extérieures.

une base d'ontologies contient la définition logique de tout type de ressource existant dans le système. Cela comprend : les attributs de la ressource, les domaines de valeurs pour chaque attribut, les contraintes sur les valeurs d'attribut, tout type de relation entre la ressource et les autres ressources du système.

une base de plan/tâche contient la description logique des opérateurs de planification (ou des tâches) connus de l'agent.

un moniteur d'exécution gère l'horloge interne de l'agent, lance arrête et contrôle les tâches qui ont été ordonnancées pour exécution ou terminaison par le planificateur-ordonnanceur. Il informe le planificateur-ordonnanceur des conditions de terminaison des tâches qu'il contrôle (succès ou échec). Le moniteur d'exécution a un accès direct au système extérieur.

B.2 Les flux d'information et de contrôle

Lorsqu'un message est reçu par la Mailbox d'un agent, celle-ci le transmet au Gestionnaire de message pour traitement. A la réception du message, le Gestionnaire de message l'interprète comme une demande d'accomplissement d'un but. Il transmet le message au moteur de co-ordination pour déterminer si ce but doit être poursuivi ou non. Dans le cas où le but est accepté, le moteur de co-ordination détermine un plan d'actions permettant de le réaliser. Pour cela, il réveille le planificateur-ordonnanceur afin que celui-ci construise un plan permettant de réaliser le but. Le planificateur-ordonnanceur crée un plan pour le but, en utilisant la description des actions qui se trouve dans sa base de plan, il réserve les ressources qui sont requises par le plan et qui sont disponibles dans sa base de ressources. Lorsque le planificateur-ordonnanceur détermine qu'une ressource requise n'est pas disponible à son niveau, il en avise le moteur de co-ordination pour que celui-ci recherche une assistance exterieure pour produire cette ressource.

Le moteur de co-ordination essaie d'obtenir de l'assistance pour obtenir la ressource manquante. Pour cela, il cherche dans sa base d'accointances, le nom d'agents dont il suppose qu'ils possèdent la capacité de produire la ressource. Dans le cas où il ne connaît aucun agent pouvant résoudre son problème, il envoie un message (via la Mailbox) à un agent facilitateur afin d'obtenir la liste de tous les agents actifs avec leurs capacités. A la réception d'une réponse, la Mailbox transmet le message de réponse au moteur de co-ordination. Celui-ci met à jour sa base d'accointances et envoie des messages aux agents leur proposant une demande de service pour obtenir la ressource manquante.

Une fois que tous les agents contractés ont envoyé leurs offres de service, ou si la date limite de réponse a été atteinte, le moteur de co-ordination transmet toutes les offres au planificateur-ordonnanceur, celui-ci sélectionne le meilleur contractant pour fournir la ressource requise. La valeur d'une proposition dépend de son coût et de son degré de compatibilité avec l'ensemble du plan original. Lorsque le plan est complet, le planificateur-ordonnanceur retourne au moteur de co-ordination la liste des agents contractants retenus à qui il faut confirmer la demande et la liste des agents contractants rejetés, à qui il faut également envoyer des messages.

Cependant, avant d'envoyer le message scellant le contrat et les messages de rejet, le moteur de co-ordination envoie d'abord à l'agent à l'origine de la réalisation du but courant, un message d'information sur sa capacité à atteindre le but fixé et sur le prix que cela va coûter. Le moteur se met ensuite en attente d'une réponse. Si celle-ci est positive, le moteur de co-ordination envoie alors les différents messages aux agents contractants. Si la réponse est négative, des messages d'annulation sont alors envoyés à tous les contractants et le planificateur-ordonnanceur est in-

vité à annuler son plan.

Une fois le plan ordonnancé prêt à l'exécution, le moniteur d'exécution exécute les actions spécifiées en invoquant un programme externe déclaré dans chaque description. Si le plan est exécuté avec succès, les résultats finaux sont envoyés par le moteur de co-ordination et la Mailbox à l'agent qui avait demandé la réalisation du but.

Annexe C

Quelques écrans d'interface de l'architecture OSIRIS

FIG. C.1 – *Interface de visualisation des instances d'une classe d'entité spatiale*

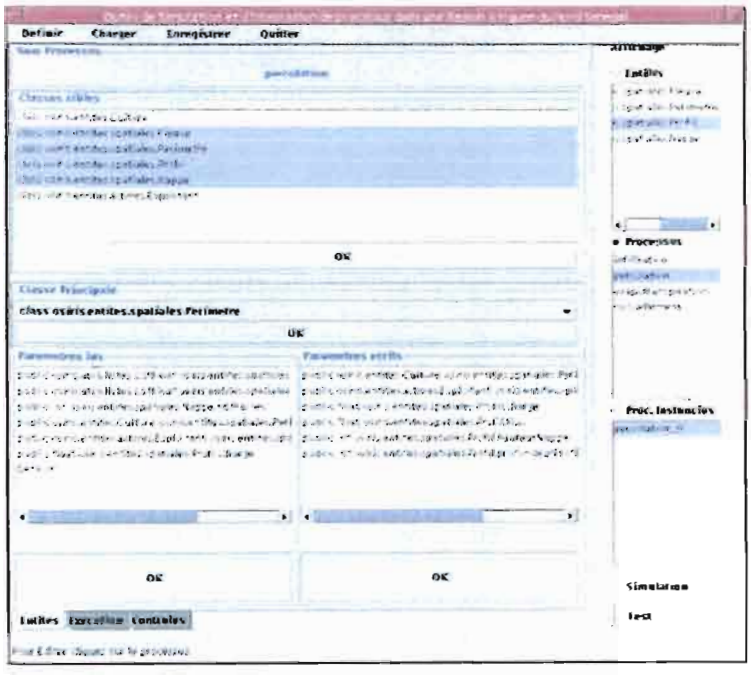


FIG. C.2 – *Interface de visualisation d'une classe de processus*

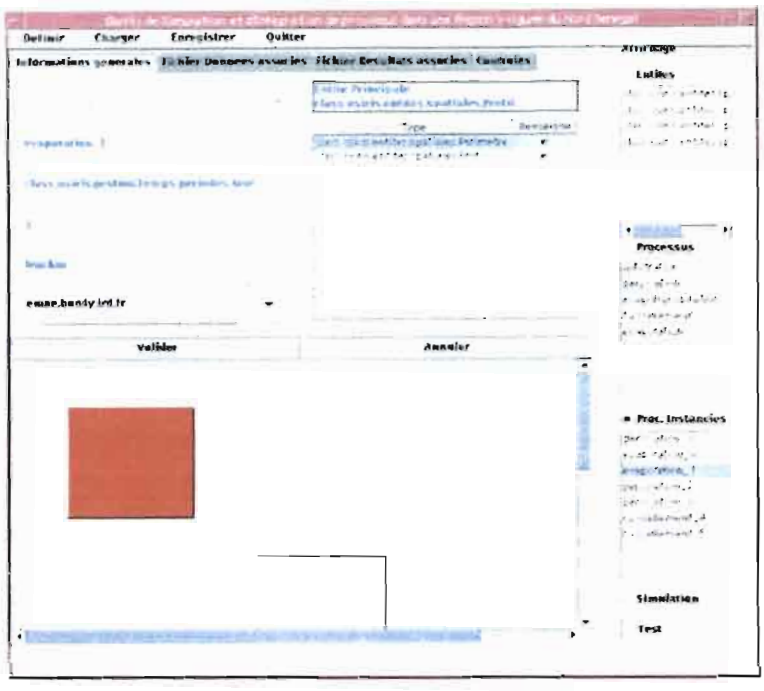


FIG. C.3 – *Interface de visualisation d'un processus instancié*

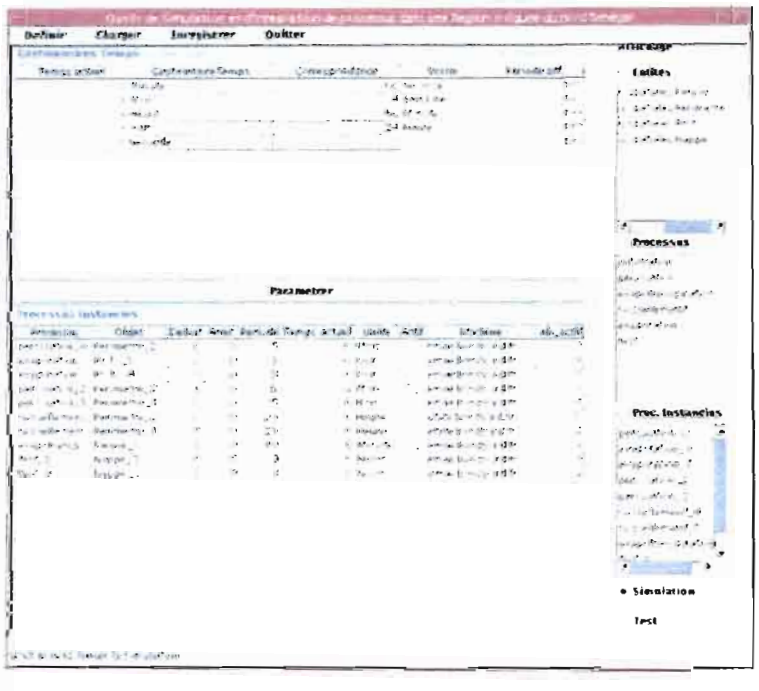


FIG. C.4 – *Interface de visualisation du paneau de contrôle de la simulation*

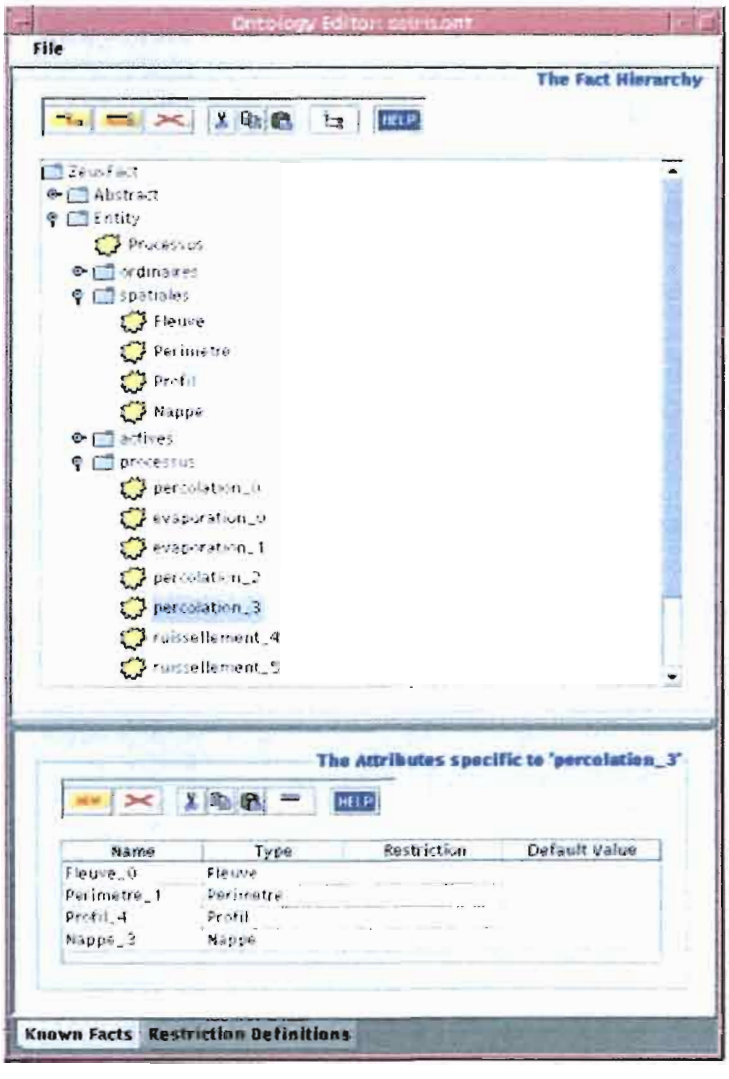


FIG. C.5 – *Interface de visualisation des ontologies correspondant au système*

OSIRIS fils aîné de Geb (dieu de la terre) et Nout (déesse du ciel) succéda à son père sur le trône et régna avec Isis sa soeur devenue son épouse. Il est le père d'Horus. Souverain juste et bon, OSIRIS enseigna à son peuple l'agriculture et les techniques d'irrigation. Il fût assassiné par son frère Seth, jaloux et rêvant de lui ravir son trône, qui dispersa les morceaux de son cadavre. Toutefois, Isis reconstitua son corps, le fit momifier par Anubis et lui redonna la vie. Ressuscité et désormais à l'abri de la mort, le dieu-roi Osiris préfèra cependant se retirer et laisser le trône terrestre à son fils Horus. Il règne depuis sur l'au-delà, le monde souterrain où il accueille les âmes des justes et règne sur les morts.