Approche multi-agent pour la multi-modélisation et le couplage de simulations. Application à l'étude des influences entre le fonctionnement des réseaux ambiants et le comportement de leurs utilisateurs.

Julien Siebert

Nancy-Université
Université Henri Poincaré

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Approche multi-agent pour la multi-modélisation et le couplage de simulations.

Application à l'étude des influences entre le fonctionnement des réseaux ambiants et le comportement de leurs utilisateurs.

THÈSE

présentée et soutenue publiquement le 9 septembre 2011

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1

(spécialité informatique)

par

Julien Siebert

Composition du jury

Président :	Dominique Méry	Professeur, Université Nancy 1
Rapporteurs :	Francine Krief	Professeur, Université Bordeaux 1
	René Mandiau	Professeur, Université de Valenciennes et du Hainaut-Cambrésis
Examinateurs :	Marcelo Dias de Amorim	Chargé de recherche, CNRS
	Philippe Mathieu	Professeur, Université Lille 1
	Laurent Ciarletta	Maître de Conférences, Institut National Polytechnique de Lorraine
Directeur de thèse :	Vincent Chevrier	Maître de Conférences, HDR, Université Nancy 1

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria

Mis en page avec la classe thloria.

Remerciements

Je tiens à remercier les personnes suivantes pour leur collaboration à ce travail de thèse : Tom Leclerc et Tomás Navarrete Gutiérrez pour les travaux expérimentaux autour des réseaux mobiles ad hoc et pair-à-pair ; Joris Rehm, pour la formalisation en Event-B ; Virginie Galtier, pour l'implantation au travers du middleware JMS et François Klein pour la base du logiciel MASDYNE, et les conseils sur les preuves mathématiques.

Ensuite, je tiens à remercier les membres des équipes Maia et Madynes pour leur accueil, leurs conseils, les moments et les discussions que nous avons partagé qu'ils soient scientifiques ou non (autour d'une bière ou autour d'un thé).

Je tiens à remercier également la Région Lorraine, le projet ANR SARAH et l'Université Henri Poincaré (Nancy 1) pour leurs soutiens financiers respectifs.

Plus personnellement, je tiens à remercier Patricia pour son soutien durant l'écriture et la relecture du manuscrit.

Enfin, merci à Vincent et Laurent pour ces 4 ans de recherche.

À mes amis et ma famille
(ceux qui m'ont supporté durant ce travail de thèse),
et à tous ceux qui m'ont donné le goût des sciences
(les responsables).

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Contexte de la thèse : Évaluation et contrôle des technologies des réseaux ambiants

Le travail de recherche présenté dans ce manuscrit a pour contexte la modélisation et la simulation appliquées à l'évaluation des réseaux ambiants. Plus particulièrement, nous nous sommes intéressés aux réseaux pair-à-pair de partage de fichiers et aux réseaux mobiles ad hoc. Les réseaux ambiants, tels que nous les entendons, présentent des caractéristiques nouvelles par rapport aux réseaux informatiques plus classiques comme l'Internet (filaire) et les réseaux de téléphonie sans fil (3G, Edge). Ces caractéristiques sont les suivantes :

- **Décentralisation**. Il n'existe pas de nœud central (tel un serveur) qui coordonne le fonctionnement du réseau ou des services disponibles (architecture pair-à-pair).
- **Ouverture**. Chaque nœud peut se connecter et se déconnecter à tout moment sans que cela nuise au fonctionnement du réseau ni ne perturbe la qualité des services disponibles.
- **Grande dynamique** de la structure et du fonctionnement. La topologie, qu'elle soit physique ou virtuelle, évolue rapidement dans le temps (de l'ordre de la minute).
- **Contrôle local** des services proposés par les usagers. Les usagers ont la possibilité de proposer des services (partage de fichiers, de bande passante) en mettant à disposition du réseau une partie des capacités de leurs propres machines. Ils sont les acteurs effectifs du réseau.

La mise en place et l'évaluation de tels réseaux de grandes tailles est relativement difficile. Ces réseaux sont souvent de grande taille (les réseaux pair-à-pair comptent aujourd'hui plus d'un million d'utilisateurs), le coût de l'architecture d'expérimentation (que ce soient des sondes réseaux ou des sondages sur les comportements des usagers) peut être rapidement prohibitif et il est rarement possible de reproduire une expérience sur un réseau réel car les conditions expérimentales évoluent constamment. Ainsi, la démarche de modélisation et de simulation a pris dans ce domaine une place de plus en plus importante.

La modélisation et la simulation des réseaux ambiants nécessite de prendre en compte à la fois les aspects propres au réseau (algorithmes, protocoles, services *etc.*) mais aussi des aspects connexes comme l'environnement dans lequel prend place le réseau et les comportements des usagers. En effet, dans ce type de systèmes, les performances dépendent, d'une part, des aspects techniques liés aux machines connectées, à la capacité des liens entre elles et aux méthodes logicielles et protocolaires utilisées pour

rendre cet ensemble d'ordinateurs interconnectés communiquant. D'autre part, les usagers de ces réseaux, de par leurs comportements peuvent influencer les performances. Prenons l'exemple des réseaux mobiles ad hoc dans lesquels chaque nœud du réseau (ordinateur portable, *smartphone*, *etc.*) est mobile. La topologie physique (les liens radio entre chaque appareil) évolue donc avec le mouvement des utilisateurs : les **utilisateurs influencent le réseau**. Le mouvement des usagers va dépendre à la fois de leurs comportements (rendez vous, envie d'une promenade *etc.*) mais aussi des contraintes environnementales (obstacles à éviter) et des autres usagers (suivre le groupe, éviter les autres personnes) : les **utilisateurs s'influencent mutuellement** et l'**environnement influence les utilisateurs**. On observe également que les comportements des usagers vont changer en fonction de la qualité de fonctionnement du réseau (déconnexion lorsque le service est indisponible, mouvement vers un point d'accès lorsque la connexion est mauvaise *etc.*) : le **réseau influence les comportements des usagers**. Enfin, la qualité des liaisons radio va changer en fonction de l'environnement physique dans lequel se trouve l'usager (dans un bâtiment, sur un place, *etc.*) : l'**environnement influence le réseau**.

Cet exemple illustre les types de phénomènes d'inter-influences qui existent entre, à la fois, les comportements des usagers, le réseau (l'ensemble de machines connectées et communiquantes) et l'environnement (l'environnement physique des usagers et du réseau). Ce genre de phénomènes complexifie la tâche de modélisation et de simulation. Il est nécessaire de prendre en compte des modèles qui proviennent de domaines de recherche différents (sciences humaines pour les comportements, réseaux informatiques pour les protocoles, physique pour l'environnement). Ces modèles représentent chacun des aspects différents du même système et utilisent des échelles différentes. Il faut alors faire interagir l'ensemble de ces modèles pour obtenir une modélisation satisfaisante.

1.2 Enjeux de la thèse et contribution

Modéliser les phénomènes d'inter-influences n'est pas chose aisée dans la mesure où il est nécessaire de représenter à la fois les usagers, le réseau et l'environnement. Ces trois niveaux d'abstraction font appel à des domaines scientifiques différents et n'évoluent pas aux mêmes échelles. Ainsi, pour chacun on peut trouver des modèles et des logiciels de simulation déjà existants. L'enjeu est alors de pouvoir réutiliser ces modèles et leurs logiciels, de les faire interagir de façon à modéliser le système en son ensemble. Cette question n'est pas propre au seul domaine des réseaux ambiants. Elle apparaît aussi dans les autres domaines scientifiques qui font face aux mêmes types de phénomènes : les systèmes complexes. C'est d'ailleurs un enjeu majeur du domaine des systèmes complexes que de pouvoir représenter un phénomène en utilisant plusieurs points de vues provenant de domaines différents [1] [Bourgine et al., 2008].

Dans ce travail de thèse, nous explorons les solutions apportées par le cadre paradigmatique des systèmes multi-agents allié à la démarche de multi-modélisation et de couplage de simulateurs. Ces deux approches conceptuelles mariées aux apports de l'ingénierie logicielle orientée agents nous permettent d'envisager à la fois la modélisation et la simulation des réseaux ambiants de manière simple et homogène : sous forme d'une "société de modèles et de simulateurs en interaction" [2].

Ce travail de thèse étant mené dans deux domaines scientifiques (systèmes multi-agents et management des réseaux informatiques) et deux équipes de recherche différentes, nous avons opté pour un travail en largeur qui contribue à la fois aux recherches menées dans la communauté des systèmes multi-agents et dans la communauté des réseaux ambiants. Notre contribution comporte deux volets différents mais liés par le fil directeur qu'est l'étude des phénomènes d'inter-influences dans les réseaux ambiants.

D'une part, nous proposons une solution pour mettre en place cette société de modèles et de simulateurs. Pour ce faire nous proposons de créer un système multi-agent en charge des toutes les opérations liées à la multi-modélisation et à la simulation du multi-modèle. Ce travail se situe dans le courant de pensée de la communauté de la simulation multi-agent. Notre contribution dans ce domaine se situe dans le fait de proposer un méta-modèle multi-agent (AA4MM) homogène qui répond simultanément aux défis liés à la multi-modélisation des systèmes complexes, à la réutilisation de simulateurs existants et

1. Source [Bourgine et al., 2008] : How can we simultaneously study multiple levels of organization as it is often required in problems in biology or social sciences ?

2. Le terme est emprunté à la thèse de Stéphane Bonneaud [Bonneaud, 2008]

au couplage de ces derniers, c'est-à-dire à l'ordonnancement de cet ensemble de simulateurs hétérogènes. Sur ce dernier point, il est important de noter que ce méta-modèle propose de coordonner l'ensemble des simulateurs d'une manière décentralisée sans ordonnanceur global. C'est à notre connaissance une première dans ce domaine.

D'autre part, dans le domaine des réseaux ambiants, nous nous sommes intéressés à des cas d'étude à la fois dans le domaine des réseaux pair-à-pair et dans le domaine des réseaux mobiles ad hoc. Notre contribution dans ce domaine est avant tout exploratoire. Cette question étant relativement originale et complexe, il est difficile de donner aujourd'hui une réponse précise quant à l'impact de ces types de phénomènes d'inter-influences sur les performances globales des réseaux ambiants. Les expériences menées sur les différents cas d'étude permettent une première quantification de ces phénomènes. La démarche scientifique adoptée et les outils proposés permettent, quant à eux, d'envisager ce problème d'une manière rigoureuse et ouvre de nouvelles perspectives.

1.3 Plan du manuscrit

Nous proposons une lecture du manuscrit qui peut être découpée en plusieurs parties.
Tout d'abord, les chapitres 2 à 4 présentent les notions basiques importantes à la bonne compréhension du travail effectué.
Le chapitre 2 cible la démarche scientifique de modélisation et de simulation et les notions comme un modèle, un paradigme, un simulateur, *etc.* y sont présentées.
Le chapitre 3 se focalise sur une démarche particulière, celle de la simulation multi-agent. Nous y détaillons ce qu'est un agent, un environnement et quel est l'intérêt de cette approche de multi-modélisation.
Enfin le chapitre 4 dresse un panorama des outils de modélisation et de simulation qui sont présents dans les réseaux ambiants. Nous insistons dans ce chapitre sur le fait qu'un seul modèle ne peut à lui seul décrire convenablement et facilement les phénomènes d'inter-influences existants entre les usagers et le fonctionnement des réseaux ambiants.

Ensuite, les chapitres 5 à 7 proposent un état de l'art de la multi-modélisation sous l'angle des systèmes multi-agents et positionnent ce travail de thèse.
Le chapitre 5 présente les problèmes liés à la multi-modélisation et les contraintes liées à notre cas d'étude.
Le chapitre 6 dresse un état de l'art de la modélisation multi-agent face aux problèmes de multi-modélisation et de couplage de simulateurs. Enfin, le chapitre 7 positionne la problématique traitée dans cette thèse.

La contribution de ce travail de thèse s'échelonne du chapitre 8 au chapitre 11.
Le méta-modèle AA4MM est présenté au chapitre 8.
L'algorithme de simulation distribué que nous avons proposé pour répondre spécifiquement aux exigences de AA4MM est détaillé au chapitre 9.
Ensuite, une preuve de concept est donnée au chapitre 10 à partir d'un exemple de multi-modèle "jouet".
Enfin, le chapitre 11 présente l'application de ce méta-modèle au domaine des réseaux ambiants.

Chapitre 2

Modélisation et simulation

Sommaire

2.1 Introduction

La démarche de modélisation et de simulation est une approche scientifique qui a pour but de produire de la connaissance sur des phénomènes naturels ou artificiels. Cette démarche est utilisée pour comprendre, prédire voire contrôler des objets ou des phénomènes qu'ils soient déjà existants ou encore au stade de la conception. Le principe est de créer une simplification (un modèle) du phénomène ou de l'objet étudié, par exemple une maquette réduite, un prototype ou un modèle numérique. En étudiant le comportement du modèle dans différentes situations il est possible de répondre aux questions que l'on se pose sur le phénomène initial. Une telle démarche trouve son intérêt lorsqu'il s'agit d'étudier des phénomènes ou des objets qui sont difficilement manipulables, observables (de par leur taille, leur dangerosité ou tout simplement à cause du coût financier des expériences) ou qui ne sont pas encore conçus.

Proposer un modèle et en tirer des résultats exploitables n'est pas une activité anodine. Pour que les résultats obtenus soient utilisables, il est nécessaire de respecter une certaine démarche. Depuis plusieurs dizaines d'années, les recherches dans ce domaine ont fait de l'activité de modélisation et de simulation une science à part entière en proposant théories, outils et vocabulaire spécifiques.

Ce chapitre présente les notions de modélisation et de simulation qui seront utiles à la compréhension de ce travail de thèse. La section 2.2 présente les notions conceptuelles associées à un modèle : qu'est-ce qu'un modèle ? quel est son rôle ? de quoi est il composé ? *etc.* La section 2.3 quant à elle, se focalise plus sur les aspects pratiques de la modélisation, à savoir l'implantation, l'exécution et la validation d'un modèle.

2.2 Modélisation

2.2.1 Qu'est-ce qu'un modèle

L'activité de modélisation consiste à créer une représentation simplifiée (appelée modèle) d'un phénomène pour pouvoir l'étudier. Marvin L. Minsky [Minsky, 1965] définit un modèle comme suit : "pour un observateur B, un objet A^* est un modèle d'un objet (ou d'un phénomène) A si B peut utiliser A^* pour répondre à des questions qui l'intéressent sur A" [3]. La modélisation est une activité essentiellement ternaire qui met en relation le phénomène étudié, le modèle et l'observateur. C'est ce dernier qui va décider des questions à poser et du cadre de pensée (appelé paradigme) qui vont définir le type de modèle à utiliser (voir schéma 2.1).

Dans leur ouvrage sur la théorie de la modélisation et de la simulation [Zeigler et al., 2000], les auteurs introduisent la notion de cadre expérimental. Le cadre expérimental peut être vu comme l'ensemble des questions que l'on se pose sur un phénomène et l'ensemble des conditions sous lesquelles est observé / expérimenté ce phénomène. Le cadre expérimental va guider le concepteur du modèle en définissant à la fois les phénomènes et les objets à représenter et les observations et mesures à effectuer sur le modèle pour répondre à ses questions.

Dans la suite du document, on parlera de système étudié *comme étant* le phénomène que l'on considère associé à un cadre expérimental particulier.

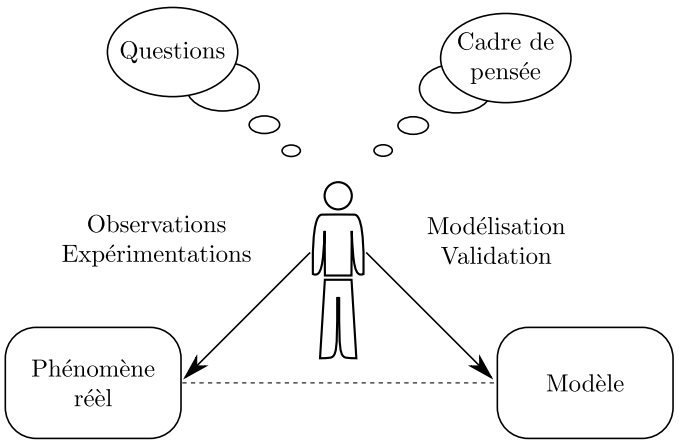


FIGURE 2.1 – Modélisation : représentation schématique. Un modèle est toujours associé à un phénomène à étudier et à un observateur (aux questions qu'il se pose sur le phénomène et à son cadre de pensée).

2.2.2 Niveau et domaine d'abstraction

Un modèle est donc associé à un observateur (ou modélisateur) et à un cadre expérimental. Selon les questions et le point de vue adopté par un modélisateur (ou si deux personnes établissent chacune un modèle), plusieurs modèles peuvent être créés pour un même phénomène.

Dans ce document, nous définissons la notion de domaine d'abstraction *d'un modèle comme* l'ensemble des objets ou phénomènes représentés au sein de ce modèle.

La notion de niveau d'abstraction *sert à désigner* l'échelle spatio-temporelle qu'utilise le modèle *pour décrire un objet ou un phénomène.*

3. La version originale : To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A. The model relation is inherently ternary. Any attempt to suppress the role of the intentions of the investigator B leads to circular definitions or to ambiguities about "essential features" and the like.

Par exemple, on peut décrire un phénomène biologique au niveau d'une seule cellule, d'un ensemble de cellules (un tissu), d'un ensemble de tissus (un organe), *etc.* En fonction de l'étude menée, il est nécessaire de se placer dans un niveau particulier, par exemple au niveau d'une seule cellule si on souhaite étudier les mécanismes qui s'y déroulent. Un niveau d'abstraction peut être qualifié de haut ou de bas. Cette dénomination n'est pas absolue et est liée au point de vue qu'adopte le modélisateur par rapport à son système étudié. Par exemple les mécanismes présents dans une cellule représentent un niveau d'abstraction bas par rapport aux phénomènes se déroulant dans les tissus (niveau plus haut) ou par rapport à ceux se déroulant dans les organes (niveau encore plus haut). Généralement, lorsque l'on modélise un phénomène qui fait intervenir des mécanismes à plusieurs échelles spatio-temporelles, il est fait la distinction entre le niveau microscopique (le plus petit niveau), le niveau macroscopique (le plus haut niveau) et les niveaux mésoscopiques (les niveaux d'abstraction intermédiaires). Un modèle à un haut niveau d'abstraction n'est pas plus ou moins précis qu'un modèle décrit à un plus faible niveau d'abstraction. Il décrit simplement un même phénomène à une échelle spatio-temporelle différente.

2.2.3 Validation d'un modèle et erreur

Ensuite, on dira qu'"un modèle A^* est un bon modèle du phénomène A (un modèle valide), du point de vue de l'observateur B si les réponses fournies par A^* correspondent à celles que B aurait obtenues en travaillant sur A" [4] [Minsky, 1965].

Il est donc nécessaire, pour valider un modèle, de comparer les résultats fournis par ce modèle et quantifier l'erreur faite par rapport au phénomène correspondant. Pour ce faire, il faut collecter des données sur le système étudié par l'observation ou l'expérimentation. De plus, un modèle ne sera d'ailleurs pleinement valide que si ces résultats ont été confirmés par d'autres personnes (les pairs de la communauté par exemple).

2.3 Mise en place d'un modèle et simulation

2.3.1 Activité de modélisation

L'activité de modélisation et de simulation dans son ensemble consiste donc tout d'abord à concevoir un modèle. Ensuite il est nécessaire de simuler ce dernier pour obtenir des résultats. Enfin, ces résultats doivent être vérifiés afin de savoir s'ils peuvent s'appliquer au système étudié. Si tel est le cas, alors le modèle est valide et représente le comportement du système étudié sous certaines conditions énoncées par le cadre expérimental. Il est alors possible d'utiliser le modèle pour comprendre, prédire ou mettre au point des méthodes qui permettent de contrôler le système réel étudié en faisant de nouvelles expériences en simulation.

Si le modèle n'est pas valide alors, il est nécessaire de le modifier et de recommencer ce processus jusqu'à l'obtention d'un modèle valide. Le schéma emprunté à [Quesnel, 2006] et présenté par la figure 2.2 résume ce processus.

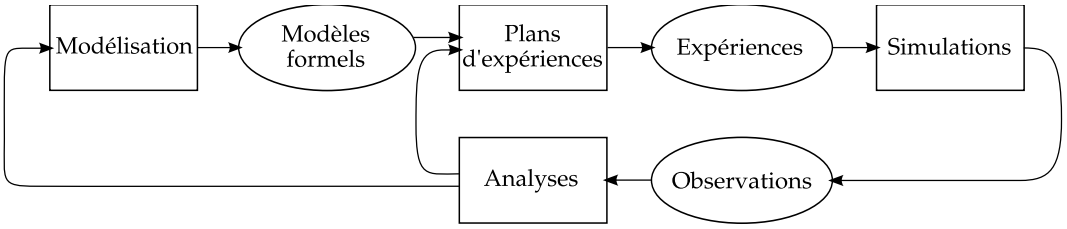


FIGURE 2.2 – Processus de modélisation et de simulation, source [Quesnel, 2006]

À chaque étape du processus de modélisation et de simulation, il est nécessaire de répondre à un certain nombre de questions afin de rendre intelligible l'étude en elle-même et afin de rendre reproductibles les

4. If A is the world, questions for A are experiments. A* is a good model of A, in B's view, to the extent that A*'s answers agree with those of A's, on the whole, with respect to the questions important to B.

résultats de simulation. Ainsi l'activité de modélisation nécessite (selon [Ramat, 2006]) :

- de définir les questions que l'on se pose sur le phénomène réel ;
- de définir le système étudié : les éléments du phénomène réel à prendre en compte, leurs relations, le tout en fonction de la question posée ;
- d'établir un plan expérimental ;
- d'adopter un paradigme et un formalisme (ou plusieurs) pour concrétiser le modèle ;
- de simuler le modèle en fonction du plan expérimental ;
- d'analyser les résultats de simulation obtenus.

Les trois premiers points sont de l'ordre du travail à mener en préalable à la conception d'un modèle. Dans les sections suivantes, nous présentons les différentes étapes constituant la concrétisation et l'utilisation d'un modèle.

2.3.2 Paradigme

Le paradigme est le point de vue qui va diriger la conception du modèle. C'est en quelque sorte la vision scientifique à laquelle adhère le modélisateur et qui va lui permettre d'effectuer sa tâche. L'historien des sciences Thomas S. Kuhn [Kuhn, 1996] a défini le terme paradigme comme étant "certains des exemples acceptés de pratiques scientifiques - ce qui inclut lois, théories, applications et instrumentations - qui fournissent des modèles desquels proviennent des traditions particulières et cohérentes de recherche scientifiques." [5]

Il existe plusieurs paradigmes. On peut citer notamment le paradigme systémique dans lequel tout objet ou phénomène étudié est vu comme une entité appelée système : entité qui possède une frontière, une dynamique interne et un ensemble de flux entrants et sortants (voir [Zeigler et al., 2000]). Ou encore, le paradigme des systèmes multi-agents, dans lequel le monde peut être décrit comme un ensemble d'entités autonomes (appelées les agents) situées au sein d'un ou plusieurs environnements et qui interagissent entre elles, au travers et avec leur environnement (voir section 3.2 et [Ferber, 1997]).

2.3.3 Formalisme

Pour donner corps au modèle, il est nécessaire de le décrire d'une manière formelle. Le langage formel particulier qui permet cette description est appelé formalisme. Pour un paradigme donné, il peut exister plusieurs formalismes. Généralement, ceux-ci sont classifiés selon plusieurs critères : la représentation et la gestion du temps, la représentation de l'espace et des objets modélisés et le rapport au hasard [Miller et al., 2004; Quesnel, 2006]. Une classification des formalismes, provenant de [Ramat, 2006], est présentée par la figure 2.3.

Pour illustrer les différences qui existent entre les formalismes, les figures 2.4a et 2.4b donnent des exemples à la fois d'un modèle décrit par des équations différentielles (figure 2.4a) et d'un modèle décrit par un automate à état finis (dans notre exemple un réseau de pétri, voir figure 2.4b).

2.4 Simulation

Le modèle formalisé permet de répondre aux questions que l'on se pose sur le système étudié, soit de manière analytique (quand cela est possible), soit par la simulation. D'une manière générale, la résolution analytique d'un modèle permet d'obtenir un résultat en un seul calcul alors que la simulation d'un modèle consiste à faire évoluer le temps dans le modèle et ainsi calculer, pour chaque valeur du temps de simulation, un résultat partiel. Un peu plus formellement, la simulation est le processus qui fait changer le modèle d'état en fonction de son état précédent, des paramètres d'entrée du modèle et d'une variable appelée temps de simulation. On peut représenter le processus de simulation graphiquement. Le temps de simulation est représenté par l'axe des abscisses et l'ensemble des états du modèles par l'axe des ordonnées

5. Source [Kuhn, 1996] : [...] I shall henceforth refer to as 'paradigms', a term that relates closely to 'normal science'. By choosing it, I mean to suggest that some accepted examples of actual scientific practice - examples which include law, theory, application, and instrumentation together - provide models from which spring particular coherent traditions of scientific research.

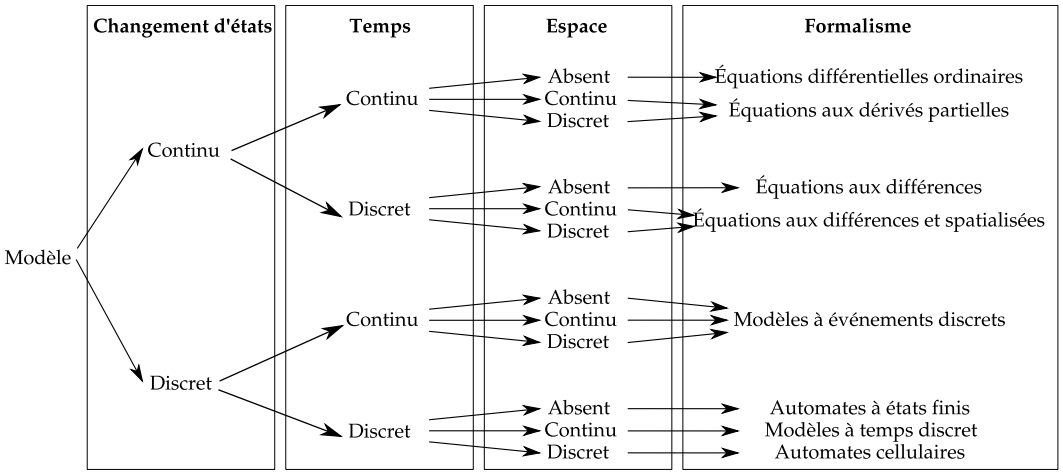
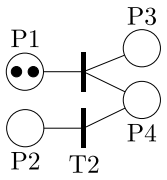


FIGURE 2.3 – Classification des différents formalismes, source [Ramat, 2006]

$$\begin{aligned} \frac{dx}{dt} &= x(\alpha - \beta y) \\ \frac{dy}{dt} &= -y(\gamma - \delta x) \end{aligned}$$

(a) Équation différentielle (équation de Lokta-Volterra). Dans ce formalisme, un phénomène est représenté via les variations ($\frac{d}{dt}$) de certaines valeurs (ici x et y) en fonction du temps t et de certains paramètres (α, β, γ et δ). Dans ce cas, l'ensemble des variables (temps y compris) est représenté de manière continue.



(b) Réseaux de pétri. Dans ce formalisme, un phénomène est représenté par un ensemble d'états (les nœuds du réseau) et un ensemble de transitions (les liens du réseau). Dans ce cas, le temps et l'ensemble des états sont représentés de manière discrète.

FIGURE 2.4 – Différents formalismes

(voir figure 2.5). On voit apparaître sur ce schéma les changements d'états du modèle à certaines valeurs du temps de simulation. Ce sont ces valeurs qui sont importantes et celles-ci sont déterminées par la politique d'exécution du modèle.

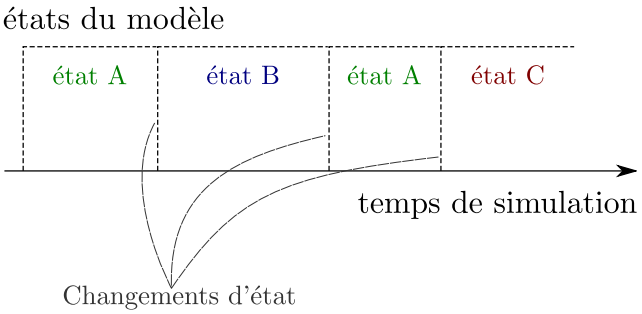
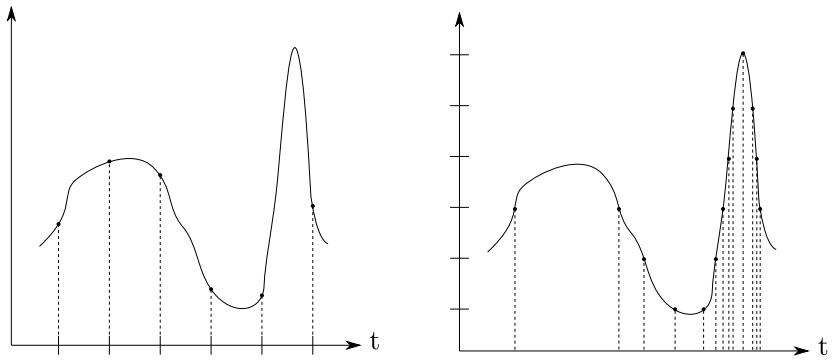


FIGURE 2.5 – Simulation

2.4.1 Différentes politiques d'exécution

Il existe plusieurs politiques de simulation selon la représentation du temps de simulation et de l'état du modèle (discret / continu) (voir les différentes spécifications apportées par [Zeigler et al., 2000] ou encore la description apportée dans [Michel, 2004] au chapitre 2). On peut citer les deux principales qui sont : la simulation cyclique et la simulation événementielle (voir figure 2.6). Dans la simulation cyclique, le temps est représenté de manière discrète et le modèle change d'état pour chaque valeur du temps de simulation (simulation par pas de temps). À l'opposé, dans la simulation événementielle, le temps est représenté de manière continue mais la dynamique du phénomène modélisé est discrétisée : le modèle change d'état de manière discrète. Ainsi, le modèle est exécuté pour chaque changement d'état du modèle (appelé événement de simulation). Par exemple, dans un modèle de réseau, un événement de simulation peut correspondre à la réception d'un message.



(a) Simulation cyclique. Le temps de simulation est discrétisé en "pas" de temps et la dynamique du phénomène modélisé est représentée de manière continue.

(b) Simulation événementielle. Le temps de simulation est continue et la dynamique du phénomène modélisé est représentée de manière discrète.

FIGURE 2.6 – Différentes politiques de simulation

Simuler un modèle consiste donc simplement à faire évoluer le modèle au cours du temps afin d'observer son comportement. La politique d'exécution est dépendante du formalisme utilisé et elle est choisie par le modélisateur en fonction de ses besoins.

Pour avoir un aperçu complet de la dynamique du modèle, il est généralement requis de faire plusieurs simulations, soit pour tester les différentes valeurs des paramètres du modèle soit parce que ce dernier

fait intervenir le hasard [6]. Cet ensemble de simulations est défini au niveau du plan expérimental. Le plan expérimental définit l'ensemble des paramètres à faire évoluer durant les expériences, le nombre de simulations à mener pour chaque expérience, *etc.*.

2.4.2 Logiciel de simulation et simulateur

Exécuter un modèle nécessite de l'implanter informatiquement via un programme.

Le terme logiciel de simulation *désigne l'outil informatique qui permet de simuler numériquement un modèle.*

Le logiciel de simulation contient donc, au minimum, une version opérationnelle du modèle et un mécanisme de simulation. La plupart du temps sont ajoutés des outils de définition de plan d'expériences ainsi que des outils de visualisation et d'analyse des données afin d'offrir au modélisateur les services utiles au processus de modélisation et de simulation.

Par abus de langage, on utilise souvent le terme simulateur en lieu et place de logiciel de simulation. Par la suite nous utiliserons également le terme simulateur, plus pratique à employer. Cependant, par la suite il nous sera nécessaire de faire la distinction entre la partie mécanisme de simulation et la partie purement logicielle. À ce moment nous utiliserons explicitement le terme logiciel de simulation.

En terme d'architecture logicielle, on distingue généralement deux types de simulateurs selon que le programme s'exécute sur une seule machine ou de manière distribuée sur plusieurs. Les simulateurs distribués sont utilisés pour simuler des modèles qui demandent beaucoup de ressources computationnelles (mémoire, calcul, *etc.*). Dans ce dernier cas, le mécanisme de simulation est plus spécifique car distribué.

2.4.3 Simulation distribuée et contrainte de causalité

La simulation distribuée consiste à faire exécuter un modèle par un simulateur qui est distribué sur plusieurs machines. Le but est de gagner en performance soit pour diminuer le temps de simulation, soit pour augmenter la taille du modèle. Dans ce contexte un modèle donné est décomposé en sous-parties (voir figure 2.7).

Un modèle m est distribué en plusieurs parties *notées μ_i si elles-même sont aussi des modèles et leur union donne le modèle m.*

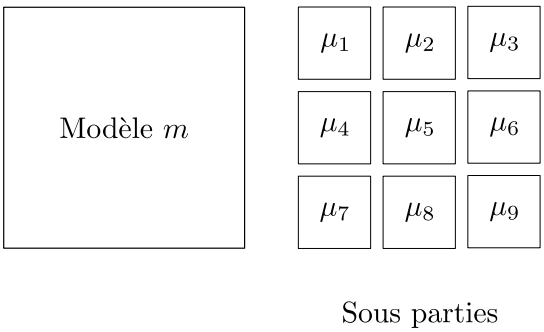


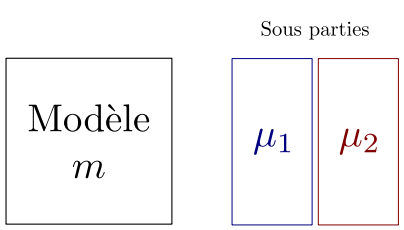
FIGURE 2.7 – Illustration d'un modèle m décomposé en sous parties μ_i, avec $i \in [1, 9]$.

Le problème majeur de la simulation distribuée est de vérifier que chaque sous partie du modèle est exécutée dans le "bon" ordre. Le "bon" ordre d'exécution est l'ordre dans lequel la simulation centralisée et la simulation distribuée donnent le même résultat [Fujimoto, 2001]. Pour cela, il faut que la simulation distribuée respecte le principe de causalité. Le principe de causalité signifie simplement que si un phénomène particulier (appelé cause) produit un autre phénomène (appelé effet) alors l'effet ne peut pas précéder la cause [Sebastien, 2009].

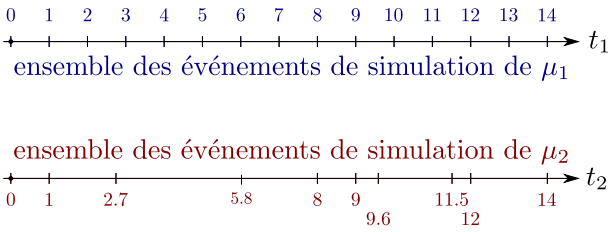
6. En répétant les simulations on obtient une meilleure précision sur les résultats.

> Contrainte de causalité, *d'après [Fujimoto, 2001] : Respecter la contrainte de causalité, c'est faire en sorte que les résultats fournis par la simulation distribuée d'un modèle m soient les mêmes que ceux fournis par l'exécution "classique" (non distribuée) de ce même modèle m.*

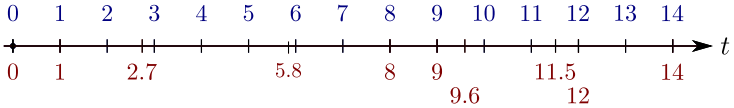
Par exemple, prenons deux événements de simulation (notés e_1 et e_2) qui arrivent dans deux sous parties distinctes du modèle (notées μ_1 et μ_2) à des instants de simulation différents (notés t_1 et t_2). Si $t_1 < t_2$ alors l'événement e_1 doit arriver – dans la simulation – avant e_2. C'est-à-dire que l'exécution des sous parties du modèle μ_1 et μ_2 doit faire en sorte que l'événement e_1 arrive avent e_2 (voir figure 2.8)



(a) Le modèle m est décomposé en deux sous parties μ_1 et μ_2.



(b) Événements de simulation de chaque sous partie du modèle.



(c) Ordre des événements de simulation du modèle quand la contrainte de causalité est respectée.

FIGURE 2.8 – Illustration du respect de la contrainte de causalité en simulation distribuée

Il existe principalement deux manières de gérer le temps dans les simulations distribuées. Dans la première méthode, la contrainte de causalité est toujours respectée : les modèles μ_i s'attendent les uns les autres avant d'être exécutés et de faire avancer le temps de simulation. C'est ce qu'on appelle les méthodes conservatives de simulation distribuée.

Dans la deuxième méthode, les modèles μ_i sont exécutés et leurs temps de simulation t_i sont augmentés jusqu'à ce que la contrainte de causalité soit enfreinte : une partie μ_i du modèle m reçoit une donnée dont elle aurait du tenir compte à un temps de simulation inférieur à t_i. Dans ce cas, il est nécessaire de revenir en arrière dans la simulation et de recommencer l'exécution des μ_i en tenant compte de cette nouvelle donnée (mécanisme de *rollback*). C'est ce qu'on appelle les méthodes optimistes de simulation distribuée. Le mode optimiste a été historiquement développé pour améliorer les performances globales de la simulation. Cependant, dans notre travail de thèse, nous n'abordons que les méthodes de simulation distribuée conservatives. Celles-ci sont présentées ci-après.

2.4.4 Exemples de méthodes conservatives de simulation distribuée

Les simulateurs distribués peuvent se décliner selon deux architectures : centralisée ou décentralisée. Dans la première, le modèle m est distribué en plusieurs sous parties μ_i mais la gestion de la simulation est centralisée dans une entité unique appelée ordonnanceur (voir figure 2.9). Le rôle de cette entitée est de répondre aux questions comme : quel μ_i est exécuté à quel moment ? comment le temps de simulation augmente-t-il ? *etc.*

L'avantage d'une telle approche est d'être conceptuellement relativement simple à mettre en place. En effet, l'ordonnanceur et l'algorithme de gestion du temps n'ont pas besoin d'être décentralisés. Cependant, cela peut limiter les performances de la simulation, l'ordonnanceur devenant un goulot d'étranglement si le nombre de calculs à effectuer devient trop important (pour simuler un grand réseau par exemple).

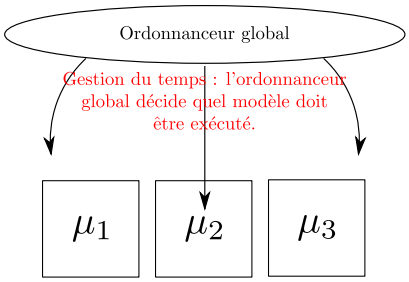


FIGURE 2.9 – Simulation distribuée utilisant un ordonnanceur global

Dans la deuxième architecture, chaque partie μ_i est exécutée par son propre ordonnanceur et ceux-ci se synchronisent via un algorithme décentralisé. L'algorithme Chandy/Misra/Bryant (CMB) du nom de ses co-inventeurs (voir [Chandy and Misra, 1979; Adam et al., 1981]) est certainement l'algorithme classique dans le domaine de la simulation événementielle distribuée. Nous en exposons ici les grands principes (que nous réutiliserons par la suite au chapitre 8). Pour plus de détails sur l'algorithme en lui même, nous invitons le lecteur intéressé à se référer à l'ouvrage [Zeigler et al., 2000] pages 264-270.

L'algorithme CMB et ses déclinaisons s'appliquent aux simulations distribuées orientées événements discrets (*parallel discrete event simulation* : PDES). Dans cette approche, chaque sous partie μ_i est exécutée selon une politique d'exécution événementielle. Les modèles μ_i s'échangent des événements de simulation. Ils possèdent chacun une file dans laquelle ils peuvent stocker les événements externes et ils peuvent exécuter des événements de simulation qui leurs sont externes : provenant d'autres parties distribuées μ_j du modèle m (voir figure 2.10).

Cet algorithme repose sur le concept de processus logique proposé par Leslie Lamport [Lamport, 1978]. Chaque modèle μ_i est associé à un simulateur, noté s_i. Chaque simulateur est vu comme un processus logique qui possède sa propre horloge. L'algorithme CMB permet d'éviter les attentes infinies (*deadlocks*) et de respecter la contrainte de causalité.

Pour cela, les modèles μ_i échangent soit les événements de simulation, soit des événements vides appelés *null messages*. Chaque événement échangé, qu'il soit vide ou non, possède une étiquette temporelle (*timestamp*). Chaque simulateur possède alors une file d'événements ordonnés selon leurs *timestamps* et choisit l'événement de simulation à exécuter (qu'il soit interne ou externe au simulateur).

Pour respecter la contrainte de causalité, l'algorithme se repose sur la notion de *lookahead*. Le *lookahead* peut être vu comme l'intervalle de temps de simulation pendant lequel le simulateur garantit qu'aucune donnée ne sera produite. Si à un instant t_i donné, un simulateur s_i n'a pas d'événement à exécuter, celui-ci envoie un *null message* aux autres simulateurs avec le *timestamp* suivant : t_i+*lookahead*. De cette manière tous les autre simulateurs savent qu'aucun événement en provenance de s_i n'arrivera avant le temps de simulation t_i+*lookahead*. Ainsi, chaque simulateur sait s'il peut exécuter ses événements de simulation ou non.

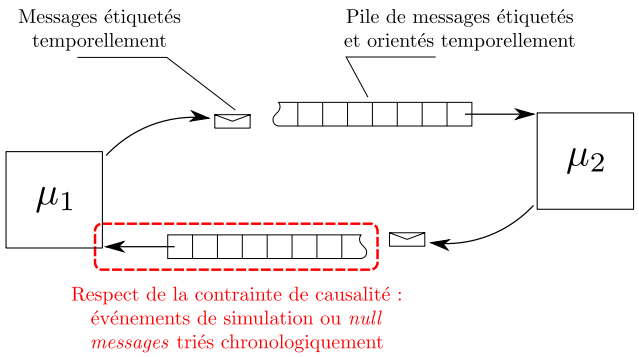


FIGURE 2.10 – Simulation distribuée à événements discrets

2.5 Synthèse

L'approche de modélisation et simulation a pour but d'accroître la connaissance que possède un observateur sur un phénomène donné. L'intérêt de cette démarche réside dans le fait de simplifier l'objet d'étude en créant un modèle. Ce modèle est beaucoup plus simple à manipuler, à observer que le phénomène réel modélisé. Les conditions expérimentales sont toujours maîtrisées et une même expérience peut être rejouée autant de fois que nécessaire. L'enjeu de cette méthode consiste à créer un modèle suffisamment représentatif du phénomène étudié pour que les réponses apportées par le modèle puissent s'appliquer au phénomène réel étudié.

La méthodologie associée à la modélisation et à la simulation est relativement bien structurée. Cependant, il est important de noter qu'elle est liée au cadre de pensée du modélisateur et aux questions posées sur le phénomène réel étudié. Ainsi, un même phénomène peut être représenté de manières différentes et plusieurs modèles et simulateurs hétérogènes peuvent co-exister. Un même phénomène peut être représenté par plusieurs modèles décrits dans des formalismes différents et simulés grâce à des politiques d'exécutions différentes. De même le paradigme de modélisation va changer en passant de l'étude de systèmes non-complexes (qui présentent des mécanismes de causalité linéaires) aux systèmes complexes. Dans le chapitre suivant (chapitre 3), nous présentons l'approche de modélisation et de simulation multi-agent qui est une démarche de modélisation propre aux systèmes complexes.

Chapitre 3

Modélisation et simulation multi-agent

Sommaire

3.1 Introduction

3.1.1 Modéliser les phénomènes complexes

Le domaine des systèmes complexes a adopté la démarche de modélisation et de simulation multi-agent pour saisir la complexité des phénomènes étudiés. "La définition d'un système complexe et celle d'un système multi-agent sont très proches. Mais au delà de leur seule ressemblance, **systèmes multi-agents et systèmes complexes désignent en fait la même chose**" [7] [Fromm, 2004].

Le paradigme multi-agent propose une description d'un système composé d'un ensemble d'entités autonomes, appelées les agents, qui interagissent entre elles, avec, et au travers d'un environnement [Ferber, 1997]. Un système multi-agent se compose donc d'agents et d'un ou plusieurs environnements (voir la définition 3.2.1.1). La dynamique globale du système émerge des interactions au niveau local entre les différents agents [Van Dyke Parunak et al., 1998].

7. Source [Fromm, 2004] chapitre 1, section 2 : The definition of a Complex Adaptive System (CAS) is closely related to the definition of a Multi-Agent System (MAS), and it defines a CAS simply as a form of complex MAS with adaptive agents. [...] CAS and MAS do not only sound similar, they describe the same systems.

3.1.2 Intérêt du paradigme multi-agent

Le paradigme multi-agent apporte un changement radical dans la façon de représenter un système. Tout d'abord l'approche de modélisation multi-agent permet d'introduire différents niveaux d'abstraction dans le modèle. Le niveau d'abstraction microscopique (les agents) et le niveau macroscopique (le système multi-agent en son entier). Ensuite, le paradigme multi-agent permet de décrire localement (pour chaque individu) les causes et les mécanismes du comportement. Il est ainsi possible de directement introduire des comportements comme "en cas d'obstacle tourner à droite", "partager si le nombre de voisins qui partagent est supérieur à 10", *etc.* C'est-à-dire qu'en lieu et place d'une équation qui décrit la dynamique d'un phénomène donné, l'approche multi-agent décompose ce phénomène en plusieurs parties (les agents) et la dynamique globale du système résulte des interactions entre toutes ces parties et leur environnement. On retrouve ici la notion d'émergence [Fromm, 2004]. Il n'est donc pas possible de résoudre un système multi-agent de manière purement mathématique comme le serait un système d'équation. La simulation est l'outil primordial des approches de modélisation multi-agents [Van Dyke Parunak et al., 1998].

L'approche multi-agent répond à l'incapacité des méthodes de modélisation basées sur des équations à représenter les individus hétérogènes, leur environnement et leurs interactions. Les domaines des sciences humaines (économie, sociologie) et biologiques (écologie), qui font face à des phénomènes collectifs pour lesquels une représentation globale (moyenne) est improductive, utilisent le paradigme multi-agent dans leurs études [Amblard and Phan, 2006; Grimm, 1999]. En biologie, la modélisation multi-agent est notamment utilisée pour décrire les comportements d'araignées sociales [Bourjot et al., 2002], de rats [Thomas et al., 2002]. Dans les sciences humaines, cette approche est utilisée pour représenter la mobilité des piétons et /ou des conducteurs dans les villes [Helbing et al., 2001; Doniec et al., 2008], les dynamiques de peuplement de villes [Gil-Guijano et al., 2010] ou encore les évacuations d'urgence en cas de crise [Murakami et al., 2002].

3.2 Systèmes multi-agents (SMA)

3.2.1 Définitions

Définition 3.2.1.1 *On appelle système multi-agent (ou SMA), un système composé des éléments suivants :*

1. *Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.*
2. *Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
3. *Un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq 0$), lesquels représentent les entités actives du système.*
4. *Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.*
5. *Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.*
6. *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.*

Définition d'un système multi-agent, source Jacques Ferber [Ferber, 1997].

3.2.2 Les agents

Les agents sont donc les entité actives du système multi-agent. On entend par là que ce sont des entités autonomes (qui poursuivent leurs propres buts) qui peuvent prendre des décisions (voir définition 3.2.2.1).

Définition 3.2.2.1 *On appelle agent une entité physique ou virtuelle*

a. qui est capable d'agir dans un environnement,

b. qui peut communiquer directement avec d'autres agents,

c. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),

d. qui possède des ressources propres,

e. qui est capable de percevoir (mais de manière limitée) son environnement,

f. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),

g. qui possède des compétences et offre des services,

h. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Définition d'un agent, source Jacques Ferber [Ferber, 1997].

Un agent est donc une entité qui répond à toutes les caractéristiques précédentes. Généralement, il est fait la distinction entre les agents dits réactifs qui possèdent peu de capacités cognitives (peu ou pas de mémoire, de but, de représentation interne), et les agents dits cognitifs qui, eux, possèdent des capacités de réflexion beaucoup plus importantes et souvent une architecture plus développée.

3.2.3 L'environnement

L'environnement est l'espace (généralement muni d'une métrique) dans lequel sont plongés les agents. Il possède sa propre dynamique et ses propres états mais, contrairement aux agents, l'environnement ne possède pas processus de décision. D'une manière générale, si les agents ne peuvent percevoir qu'une partie limitée de l'environnement, ce dernier, quant à lui, a connaissance de tous les agents présents en son sein [Ferber, 1997].

L'environnement est un élément important dans le système multi-agent. En effet, c'est grâce à lui que les agents peuvent co-exister et interagir. Ce dernier est en effet considéré comme le médium des interactions [Weyns et al., 2007]. Ceci étant, l'environnement doit pouvoir être perçu par les agents et ces derniers doivent pouvoir agir dessus et interagir au travers.

3.2.4 La dynamique du système multi-agent : comportements, interactions et organisation

Un système multi-agent est avant tout un système dynamique. Pour faire évoluer un système multi-agent, il faut décrire plusieurs choses : le comportement des agents, la dynamique de l'environnement, les règles qui vont régir leurs interactions et la présence ou non (et le respect) d'une certaine organisation.

Les agents possèdent un comportement. Ce dernier est généralement décrit par une boucle perception-délibération-action [Ferber, 1997] (voir figure 3.1). À un instant donné, l'agent perçoit localement son environnement. Ensuite, il prend une ou plusieurs décisions en fonction de ses perceptions et de ses états internes (représentation du monde, mémoire...). Enfin, l'agent choisit d'effectuer une ou plusieurs actions.

Plusieurs agents peuvent intenter des actions simultanément ou au même endroit. Cela peut amener des conflits dans le système. Par exemple deux agents peuvent simultanément essayer de passer au même endroit (voir figure 3.2). Dans un système réel, cela se traduirait probablement par une collision. Dans un système artificiel comme un programme informatique, on peut par exemple éviter ce genre de situation en empêchant un des deux agents d'avancer.

On voit ici, qu'il est nécessaire de décrire, en plus des agents et de l'environnement, l'ensemble des règles qui régissent les interactions entre les agents et ce qui doit se passer en cas de conflit. L'environnement du système multi-agent est le médium des interactions. C'est lui qui permet de spécifier l'ensemble des règles d'interaction [Weyns et al., 2007; Ricci et al., 2007a; Michel, 2007].

Il est également possible de spécifier une organisation (un ensemble de rôles, de groupes, de contrats...) qui va permettre aux agents d'interagir selon certaines contraintes ou de spécifier aux agents comment ils

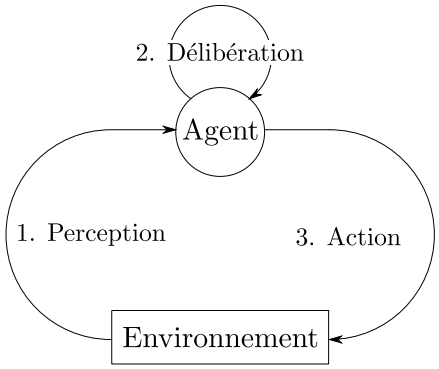


FIGURE 3.1 – Comportement d'un agent : boucle perception-raisonnement-action

peuvent coopérer. Les travaux menés au sein de la communauté ont tendance à inclure cette organisation au sein d'un environnement spécifique appelé environnement social [Stratulat et al., 2009; Kitio et al., 2008].

3.3 Méta-modèle multi-agent

3.3.1 Introduction

En terme de modélisation, le paradigme multi-agent va nécessiter un certain niveau de spécification. Dans la description précédente, nous ne présupposons rien quant à la dynamique interne d'un agent ou d'un environnement. Un environnement peut très bien évoluer de manière continue tandis que les agents ne le perçoivent que de manière discrète. De même, le paradigme multi-agent en lui même ne définit pas ce que doivent représenter un agent ou un environnement particulier. Un agent peut très bien représenter un seul individu (comme un utilisateur des réseaux) ou un groupe d'individus (un ensemble de nœuds d'un réseau), voire être lui-même un système multi-agent.

En matière de modélisation, les agents et les environnements sont des entités qui possèdent des propriétés particulières. Le paradigme multi-agent oblige à décrire, à un niveau abstrait plus élevé que les modèles, quelles sont les entités qui composent un système multi-agent. Ces entités sont au minima les agents et l'environnement mais on peut aussi considérer des objets présents dans l'environnement et utilisables par les agents, la projection d'un agent dans son environnement : son corps avec ses capteurs et effecteurs voire ajouter à la description du système multi-agent des groupes et des rôles. Utiliser un tel point de vue permet de s'abstraire d'un modèle multi-agent particulier et de définir un ensemble abstrait de concepts et de règles de fonctionnement qui peuvent se généraliser à tous les modèles multi-agents. Nous parlons alors d'un **méta-modèle** multi-agent.

> *Un* méta-modèle multi-agent *définit l'ensemble des concepts qui composent un système multi-agent, leurs relations et leurs rôles.*

3.3.2 Intérêt

La notion de méta-modèle est importante dans le domaine de la modélisation multi-agent. Concevoir un modèle multi-agent pose des défis supplémentaires par rapport à la démarche "classique" de modélisation et de simulation. Le fait de faire représenter des agents qui interagissent peut entraîner des conflits. Chaque agent doit percevoir son environnement et doit agir sur ce dernier. Par exemple, si deux agents se déplacent et parviennent au même endroit simultanément, il est nécessaire de définir si les deux agents peuvent co-exister au même endroit ou si l'un ou l'autre peut prendre la place. De plus, au niveau de la gestion du temps de simulation, il est nécessaire de définir dans quel ordre vont être exécutés les agents et leur environnement. Le manque de spécification peut remettre en cause les résultats de simulation obtenus par le modèle multi-agent [Michel, 2004; Meurisse, 2004]. Le rôle du méta-modèle est alors

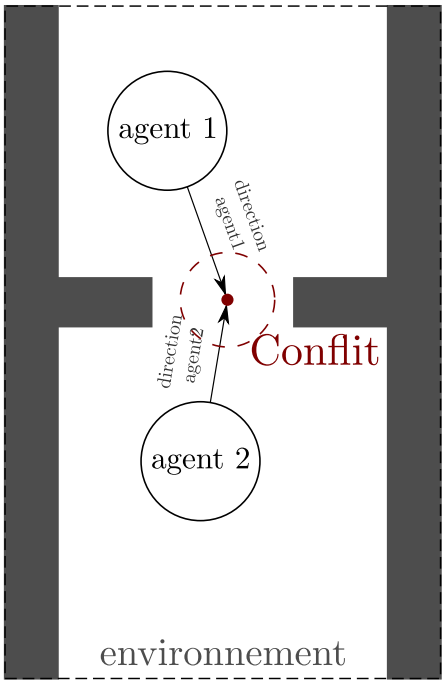


FIGURE 3.2 – Exemple de conflit dans un système multi-agent.

de permettre au modélisateur de spécifier tous ces choix de modélisation qui sont propres aux modèles de systèmes multi-agents.

3.3.3 Exemples de méta-modèles multi-agent

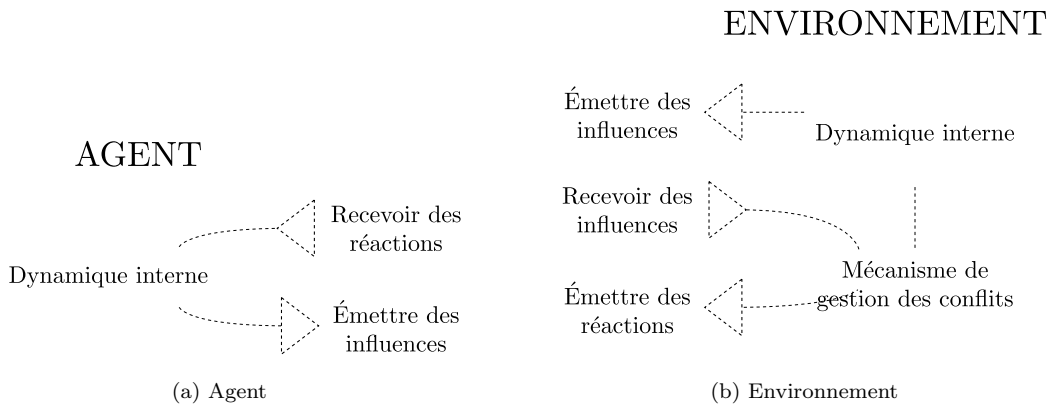
Dans cette section nous présentons différents méta-modèles multi-agents. Ceux-ci sont présentés car la démarche de méta-modélisation est à la base de notre contribution mais aussi en guise d'illustration de ce qu'est un méta-modèle multi-agent.

Méta-modèle influence réaction

Dans le domaine de la modélisation et de la simulation multi-agent, nous pouvons citer le méta-modèle influence-réaction initialement présenté dans [Ferber and Müller, 1996] puis adapté pour la simulation (IRM4S : *Influence Reaction Model for Simulation*) [Michel, 2007] et la simulation avec plusieurs environnements [Morvan et al., 2010]. Le méta-modèle IRM4S a pour but de répondre à la question : "quoi faire lorsqu'apparaissent des conflits dans le modèle multi-agent ?"

IRM4S spécifie, entre autre, que les agents et le (ou les) environnement(s) n'agissent et n'interagissent pas directement mais émettent des influences. Par exemple, un agent A qui prend la décision de bouger vers la droite émet une influence qu'on pourrait traduire par "je souhaite bouger vers la droite". Ensuite, c'est le rôle de l'environnement de vérifier si les influences peuvent effectivement avoir lieu (par exemple, l'agent A bouge effectivement vers la droite) ou, s'il y a des conflits (par exemple, les agents A et B souhaitent tous deux bouger au même endroit), de gérer les conflits (par exemple, l'agent A cède sa place à l'agent B). Une fois l'ensemble des influences résolues, le (ou les) environnement(s) émet(tent) des réactions (par exemple, l'agent A ne peut pas bouger).

Pour résumer, le méta-modèle IRM4S agit à deux niveaux. En terme de simulation ce méta-modèle contraint l'ensemble des actions nécessaire à l'exécution du modèle multi-agent. Il ne s'agit pas seulement de faire évoluer les dynamiques des agents et de l'environnement, il faut aussi gérer les flux d'influences, les conflits et les réactions. Ensuite, en terme de modélisation, ce méta-modèle force à spécifier, en plus de la dynamique des agents et de l'environnement, les choix à faire en cas de conflits (voir figure 3.3)..



(a) Agent (b) Environnement

FIGURE 3.3 – Méta-modèle influence-réaction. Architecture des différentes entités : agents et environnements

Séparation corps-esprit des agents et décomposition de l'environnement en artefacts

Une autre question qui se pose lorsque l'on conçoit un modèle multi-agent est la représentation et le lien entre les agents et le ou les environnements. Un agent doit pouvoir interagir avec son environnement, en percevoir une partie et utiliser les objets présents dans l'environnement. De même, un agent peut tout à fait être présent dans plusieurs environnement à la fois. Nous sommes donc confrontés à des questions comme : "comment un agent peut-il agir sur l'environnement ? que signifie agir et percevoir ? comment modélise-t-on les capteurs et les effecteurs ?".

Le méta-modèle proposé dans [Payet et al., 2006b] et [Stratulat et al., 2009] repose, entre autre, sur le principe de séparation entre le corps et l'esprit de l'agent. L'idée est de séparer les mécanismes décisionnels de l'agent (sa dynamique interne) : son *esprit*, de ses mécanismes d'action (d'influence) et de perception de l'environnement : son *corps* (voir figure 3.4). Les corps des agents sont en fait des instances particulières des agents présentes au sein des environnements. Cela permet à un agent d'être situé dans plusieurs environnements qu'ils soient physiques [Payet et al., 2006b] ou sociaux [Stratulat et al., 2009].

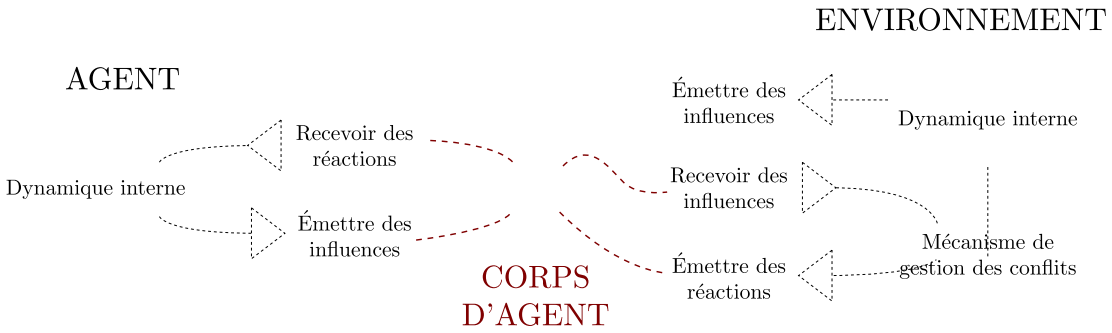


FIGURE 3.4 – Méta-modèle proposé par [Payet et al., 2006b] et [Stratulat et al., 2009] : séparation entre corps et esprits des agents

Dans le même esprit, le méta-modèle Agents et Artefacts (A&A) [Omicini et al., 2008] propose de décomposer l'environnement en un ensemble d'entités passives appelées artefacts. Les artefacts sont des objets visibles et utilisables par les agents (voir figure 3.5 et figure 3.6). Le corps d'un agent et un artefact sont tous deux des entités spécifiques qui permettent de faire le lien entre les agents et l'environnement. Le corps d'un agent permet à ce dernier de percevoir, d'agir sur son environnement et d'être perçu par les autres agents. Un artefact représente un objet de l'environnement qu'un agent peut percevoir et sur lequel il peut agir.

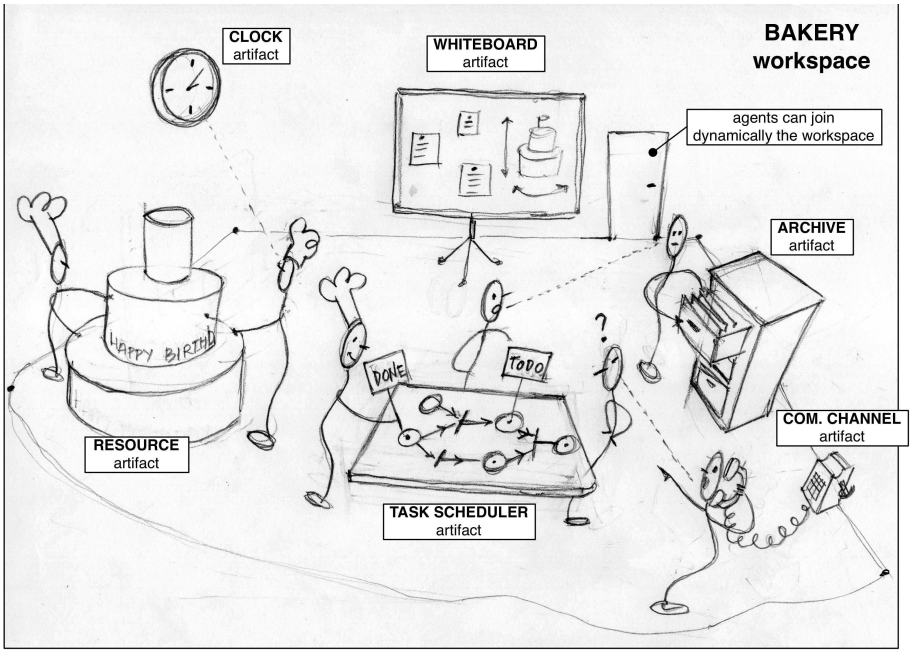


FIGURE 3.5 – Représentation métaphorique d'un système multi-agent selon le méta-modèle Agents et Artefacts (A&A). Les agents, ici des cuisiniers, coordonnent leurs activités grâce à un ensemble d'artefacts présents dans leur environnement. Source [Piunti and Ricci, 2008]

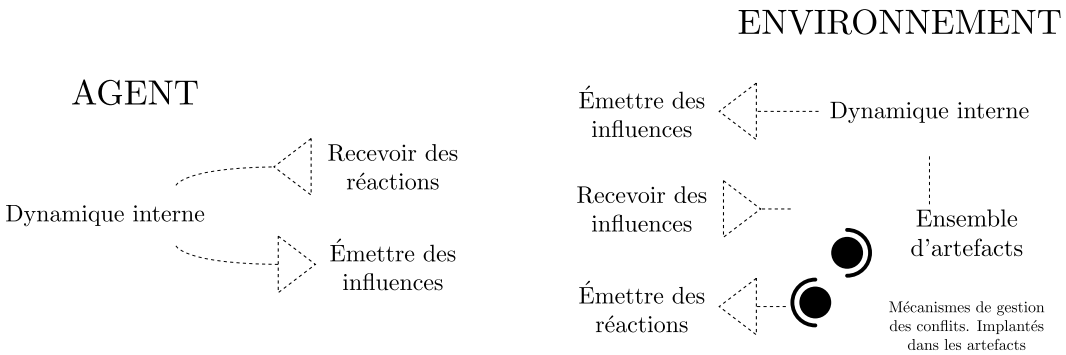


FIGURE 3.6 – Méta-modèle Agents et Artefacts. L'environnement est composé d'un ensemble d'artefacts.

La modélisation de l'environnement en un ensemble d'artefact, la séparation entre la dynamique décisionnelle de l'agent, ses capteurs et ses effecteurs et la dynamique de l'agent permettent toutes deux de forcer le modélisateur à spécifier l'ensemble des interactions possibles entre les agents et l'environnement. La modélisation multi-agent est alors mieux spécifiée. Ensuite, de tels méta-modèles permettent une plus grande modularité dans la conception du modèle. Grâce à eux, il est possible de changer plus facilement un environnement par un autre ou un agent par un autre. Le modèle en lui-même est plus facilement maintenable, vérifiable et réutilisable.

3.4 Synthèse

Dans ce chapitre nous avons présenté l'approche de modélisation et de simulation multi-agent. C'est une démarche de modélisation particulière qui a pour but de représenter les systèmes complexes sous la forme d'un ensemble d'entités autonomes, hétérogènes et dynamiques (les agents). Les agent sont situés dans un environnement et interagissent à la fois entre eux et avec leur environnement. Cette approche est utilisée dans des domaines comme les sciences humaines et la biologie où il est nécessaire de représenter des ensembles d'individus hétérogènes en interaction. Pour notre part, cette approche est intéressante dans la mesure où elle nous permettra de représenter les usagers des réseaux ambiants, leurs comportements et leurs environnements.

Nous avons également présenté la notion de méta-modèle multi-agent. Cette notion est importante dans la mesure où la modélisation multi-agent apporte de nouvelles questions par rapport à la modélisation plus "classique". La modélisation multi-agent étant une démarche particulière de modélisation, celle-ci doit répondre aux mêmes contraintes que celles présentées dans le chapitre précédent (voir chapitre 2). La conception d'un modèle multi-agent se fait comme tout modèle face à une question et un cas d'étude particuliers. Cependant, la modélisation multi-agent apporte avec elle un lot de questions supplémentaires qui ne sont pas directement liées à un cas d'étude (sémantique du modèle) mais qui sont propres à l'approche multi-agent en elle même. Le fait de faire interagir des agents dans un environnement introduit la possibilité d'actions simultanées et de conflits entre agents. De même un agent doit percevoir et agir sur son environnement et il est nécessaire de bien spécifier chacun de ces aspects. Cette spécification se fait grâce au méta-modèle multi-agent. Cela permet d'obtenir le modèle le plus satisfaisant possible par rapport au cas d'étude. Cela permet également à d'autres scientifiques de reproduire les résultats de simulation et de rendre la modélisation multi-agent plus générique.

Au terme de ce chapitre, nous pouvons donc dire que l'approche multi-agent apporte des solutions sur deux tableaux. Tout d'abord, au niveau sémantique, la modélisation multi-agent permet d'aborder des problèmes complexes dans lesquels les modélisations plus "classiques" ont montré leurs limites. Ensuite, au niveau conceptuel, le méta-modèle multi-agent permet de structurer et de forcer la spécification des apsects propres à la modélisation multi-agent.

Dans le chapitre suivant (chapitre 4), nous nous intéressons aux aspects sémantiques de notre cas d'étude. Nous passons en revue les différents modèles et simulateurs existants qui permettent de représenter les réseaux ambiants et leurs usagers. Nous montrerons que, dans le contexte de notre cas d'étude, ces modèles et simulateurs ne sont pas suffisants si ils sont pris séparément. Nous nous intéresserons alors au couplage de modèles et de simulateurs (voir chapitre 5), domaine qui fait apparaître selon nous des problématiques scientifiques fortes au niveau conceptuel.

Chapitre 4

Modélisation et simulation des réseaux ambiants

Sommaire

4.1 Introduction

Dans le domaine des réseaux informatiques ambiants, l'approche de modélisation et de simulation a pris une place importante pour la création et l'évaluation de nouveaux protocoles et services. En effet, il est souvent difficile de mener des expériences sur des réseaux de grandes tailles (les réseaux pair-à-pair comptent aujourd'hui plus d'un million d'utilisateurs) et sur des réseaux hétérogènes (l'Internet n'est pas un seul réseau mais l'interconnection de myriades de réseaux). Le coût de l'architecture d'expérimentation (que ce soient des sondes réseaux ou des sondages sur les comportements des usagers) peut être rapidement prohibitif. Ensuite, se posent des questions légales sur l'utilisation des données recueillies (le plus souvent privées). Mais plus important, il est rarement possible de reproduire une expérience sur un réseau réel car les conditions expérimentales évoluent constamment.

Les technologies des réseaux ambiants ont connu ces dernières années un développement rapide. Le développement grand public d'Internet, la miniaturisation des appareils (machines, capteurs, effecteurs) et la généralisation des communications sans fil ont fait apparaître les possibilités techniques et les besoins d'utilisation d'une informatique à la fois autonome, dynamique et ubiquitaire. Ces technologies sont dites

dynamiques car la topologie (les liens entre les différents nœuds du réseau) évolue rapidement dans le temps (avec un ordre de grandeur entre la minute et l'heure) soit de part la mobilité des nœuds du réseau ou à cause de l'évolution d'une topologie dite virtuelle relative à une application donnée. Les appareils composant l'informatique et les réseaux ambiants possèdent des capacités réduites (énergie, puissance de calculs). Ces appareils peuvent être mobiles ou non et en terme de sécurité, il n'est pas possible de contrôler leur comportement. On voit donc apparaître des services distribués, qui doivent être robustes aux connexions et déconnexions ou aux pannes, être persistants dans la durée, être accessibles en tout point du réseau, *etc.* [Krief, 2008]. C'est en ce sens qu'on peut les qualifier d'omniprésents.

En dehors de ces aspects plutôt techniques, l'informatique ambiante doit, de plus en plus, intégrer l'humain dans la boucle. D'une part, les applications ambiantes ont pour but d'apporter un service de plus en plus personnalisé et doivent pouvoir être utilisées par tout un chacun (quels que soient le niveau d'étude, la classe sociale, l'âge, *etc.*). D'autre part, les usagers, de par leur comportement, influencent le fonctionnement des réseaux soit de par une utilisation erronée des services proposés comme c'est par exemple le cas dans les réseaux pair-à-pair avec le problème du manque de partage [Adar and Huberman, 2000] ou en proposant de nouveaux services comme par exemple en téléphonie avec la possibilité pour les usagers de concevoir de nouvelles applications. D'une manière générale, pour le concepteur du réseau ambiant, il est difficile de prédire quelle sera la dynamique de ce réseau en ne tenant compte que des aspects techniques. La conception d'une application ou de protocoles est dirigée par un besoin et le concepteur a souvent en tête un scénario d'usage bien précis. Dans le cadre des réseaux ambiants, nous sommes confrontés à des scénarios d'usages et des comportements d'usagers qui peuvent varier du tout au tout. Bien qu'il soit toujours possible représenter le fonctionnement du réseau dans le cas ou les usagers font preuve d'un comportement moyen, il devient nettement plus difficile de prédire le fonctionnement du réseau dans le cas de comportements plus atypiques (attaques, utilisation détournée, *etc.*) ou dans le cas de comportements hétérogènes et dynamiques. Il devient donc nécessaire de prendre en compte les comportements des usagers, les interactions entre le réseau et les comportements pour pouvoir prédire le fonctionnement des réseaux ambiants.

Les sections suivantes décrivent les modèles et les simulateurs issus de la communauté des réseaux informatiques qui permettent de décrire les technologies des réseaux ambiants. Ces outils s'appuient sur une longue tradition de modélisation et de simulation du domaine des réseaux informatiques (section 4.2.3). Les aspects réseaux de ces technologies (la topologie, le trafic, les protocoles, les services *etc.*) composent le cœur de ces outils et les modèles et les simulateurs dans ce domaine sont relativement bien éprouvés et utilisés. À l'opposé, le domaine des réseaux ambiants (section 4.3), dans notre cas les réseaux pair-à-pair et les réseaux mobiles ad hoc, a connu un essor important ces dernières années. En conséquence, le nombre de modèles et de simulateurs disponibles a pris de l'ampleur. Ces derniers ciblent les aspects réseaux mais offrent aussi parfois (mais pas toujours !) la possibilité d'y intégrer un modèle de comportements des usagers ou un modèle de l'environnement. En ce qui concerne les comportements des usagers, la plupart des modèles proposés - quand ils existent - sont difficilement adaptables à la représentation de comportements à la fois hétérogènes et dynamiques et à la représentation des interactions entre les usagers et leur environnement. Ainsi, pour étudier les phénomènes d'inter-influences au sein des technologies des réseaux ambiants, les modèles des réseaux ne sont pas suffisants.

4.2 Réseaux informatiques

4.2.1 Bref historique

Les réseaux informatiques sont apparus au cours des années 1970 (Arpanet aux États-Unis, Cyclades, en France) [Schafer]. Aujourd'hui, les réseaux informatiques se retrouvent embarqués dans les véhicules (automobiles, avions), dans les chaînes de montage industrielles ou encore, à l'échelle planétaire avec Internet, une interconnexion mondiale de réseaux informatiques ouverte et décentralisée. Nous utilisons ce dernier comme exemple principal pour discuter du fonctionnement des réseaux informatiques.

Un réseau informatique a pour rôle de permettre l'échange de données (fichiers textes, images, vidéos, *etc.*) entre plusieurs ordinateurs. Du point de vue des utilisateurs du réseau, ce dernier se limite à des applications ou des services communicants (messagerie instantanée, e-mail, transfert de fichiers, *etc.*). Du

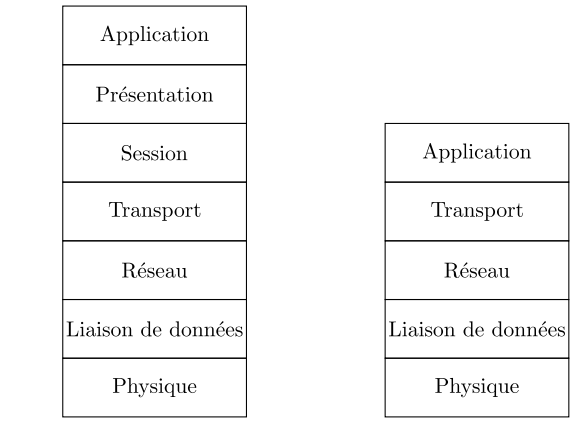
point de vue physique, chaque ordinateur est connecté aux autres par un ensemble de liens tels que les câbles de cuivre, la fibre optique, les liaisons radio, *etc.* C'est par ce réseau physique que transitent, sous forme de signaux électriques ou électromagnétiques, l'ensemble des données échangées. Cependant, entre ces deux extrêmes (usagers et réseaux physiques) se situe un ensemble d'outils à la fois matériels, logiciels et protocolaires qui ont pour but de rendre le réseau fonctionnel.

4.2.2 Principe de fonctionnement des réseaux informatiques composant Internet

Connecter physiquement un ensemble d'ordinateurs ne suffit pas à créer un réseau informatique fonctionnel. En effet, il est nécessaire, pour que ceux-ci puissent communiquer, de relever un certain nombre de défis : comment passe-t-on d'un message numérique à un signal physique ? qui est le destinataire du message ? quel chemin doit parcourir ce dernier ? que se passe-t-il si celui-ci n'arrive pas à destination ? *etc.* Pour répondre à toutes ces questions, un ensemble de solutions a été développé. Par exemple, les machines connectées au réseau ont besoin d'une interface physique pour être branchées et pouvoir envoyer des signaux sur le réseau physique ; chaque machine possède des adresses (adresse MAC, adresse IP) ; les algorithmes de routage établissent des routes sans boucle d'un point A à un point B du réseau ; le protocole de communication TCP s'assure que les messages sont bien arrivés à destination, *etc.*

Une architecture en couche

Ces solutions sont structurées et mises en œuvre dans Internet grâce à une architecture spécifique en couche [Tanenbaum and Van Steen, 2001]. C'est cette méthode qui est adoptée à la fois par le modèle TCP/IP (standard *de facto* et base d'Internet) et par le standard OSI (*Open Systems Interconnection*) proposé par l'ISO (Organisation internationale de normalisation). Cette architecture standardisée permet à chaque machine présente sur le réseau de communiquer avec toutes les autres quelles que soient les différences matérielles ou logicielles.



(a) Modèle de référence OSI : sept couches

(b) Pile protocolaire Internet (modèle TCP/IP) : cinq couches

FIGURE 4.1 – Architecture de communication en couches : modèle OSI et modèle TCP/IP (source [Kurose and Ross, 2008])

Le principe de base (pour TCP/IP ou OSI) est de diviser l'ensemble des défis liés au réseau en plusieurs niveaux ou couches. Chaque couche contient des outils et des méthodes (matériels, logiciels, protocoles) pour résoudre un problème particulier. Les couches sont ordonnées en pile et chacune se repose sur les services offerts par la couche directement inférieure (voir figures 4.1a et 4.1b) [Kurose and Ross, 2008].

L'architecture OSI contient sept couches alors que l'architecture Internet (TCP/IP) n'en contient que cinq. Les différences entre ces deux modèles sont relativement minimes et leur explication n'est pas du ressort de ce manuscrit. Cependant, pour chaque modèle, une distinction est faite entre les trois

premières couches (les couches média : physique, liaison de données et réseau) et la (ou les) dernière(s) (les couches hôte). Les **couches média**, aussi appelées couches basses, ciblent en priorité les problèmes liés à la transmission des données sur les réseaux : la conversion du signal physique en données numériques compréhensibles par l'ordinateur (les bits : *binary digits*), la gestion des erreurs de transmission, l'adressage des machines, le routage des messages au travers du réseau *etc.* Les **couches hôte**, aussi appelés couches hautes, ciblent, quant à elles, les problèmes liés à la communication de bout en bout : la fiabilité de la communication (le message est-il arrivé entièrement au destinataire ?), la sécurité (authentification, cryptographie) *etc.*

Cette représentation est théorique et, en pratique, la distinction n'est pas si fermée. En effet, il est aussi possible de trouver des mécanismes de sécurité dans les couches média (comme WEP, WPA ou IPSec). Cependant, ces légères divergences avec le modèle TCP/IP et le standard OSI ne sont pas du ressort de ce manuscrit et pour une compréhension du fonctionnement des réseaux informatiques IP, une description de ces deux modèles nous semble suffisante.

Description des couches des modèles OSI et TCP/IP (source [Kurose and Ross, 2008])

La couche **application** est l'endroit où résident les applications communicantes et leurs protocoles associés. Un utilisateur du réseau n'a généralement affaire qu'aux logiciels de ce niveau. Par exemple, un client mail et le protocole SMTP (*Simple Mail Transfer Protocol*) sont des outils faisant partie de la couche application. Les données circulant dans cette couche sont appelées **messages**.

Les couches **présentation** et **session** n'apparaissent que dans le standard OSI. En réalité, celles-ci sont incluses dans la couche application du modèle TCP/IP et leurs fonctionnalités sont activées ou non en fonction de l'application communicante. Le rôle de la couche **présentation** est de fournir un service à la couche application afin de lui permettre d'interpréter la signification des messages échangés. Le rôle de la couche **session** est de délimiter et de synchroniser les échanges de messages.

La couche **transport** gère les communications de bout en bout. Pour Internet, les protocoles principaux de ce niveau sont : TCP(*Transmission Control Protocol*) et UDP (*User Datagram Protocol*). TCP permet, d'une manière générale, de fiabiliser la communication en garantissant que les messages arrivent à destination, en contrôlant les flux d'émission et en évitant les congestions. UDP ne propose aucun de ces mécanismes. À ce niveau, les messages sont découpés en **segments** et des informations supplémentaires, appelées **en-têtes**, leurs sont associées (somme de contrôle, numéro du port source, du port de destination, *etc.*). On parle alors d'**encapsulation** des messages en segments.

La couche **réseau** a pour but de construire une route (un ensemble de nœuds adjacents) entre l'émetteur et le destinataire du message. Dans le cas d'Internet, le protocole IP (*Internet Protocol*) assure cette fonction. Cette route n'est pas connue à l'avance et est établie de proche en proche à partir des adresses IP des machines. À ce niveau, les segments sont encapsulés en **paquets**.

La couche **liaison de données** a pour but de transférer les données entre deux nœuds adjacents d'une route. Le protocole Ethernet officie à ce niveau. Ici, les paquets sont encapsulés en **trames**. Lorsque deux trames sont envoyées en même temps par deux émetteurs sur le même lien, une collision peut arriver. Les protocoles de liaison de données comme Ethernet ont aussi pour rôle de spécifier comment les appareils détectent et se remettent de telles collisions.

La couche **physique** est la couche la plus basse dans le sens où elle interagit directement avec le réseau physique. Celle-ci est chargée de la conversion entre les signaux physiques (électriques ou électromagnétiques) et les bits (*binary digits*). Elle est en pratique toujours réalisée par un circuit électronique spécifique (carte Ethernet, carte Wi-Fi).

Distinction entre les différents nœuds du réseau

Au niveau des nœuds (les ordinateurs) composant le réseau, une distinction est faite entre les **nœuds terminaux** (au "bord" du réseau) : des PCs ou des serveurs derrière lesquels se cachent en général un ou plusieurs utilisateurs ; et les **nœuds internes** ("à l'intérieur" du réseau) : en général des machines uniquement utilisées pour leur rôle spécifique dans le réseau comme des switchs ou des routeurs (couches 2 et 3).

La différence provient du fait que les nœuds terminaux implantent toutes les couches de l'architecture Internet alors que les nœuds internes au réseau n'implantent que les trois (ou deux) premières en fonction de leur rôle.

Illustration du fonctionnement du modèle en couche : envoi d'un message

Prenons l'exemple de l'envoi d'un message depuis l'ordinateur A vers l'ordinateur B pour illustrer comment fonctionne cette architecture en couche (voir figure 4.2). Tout d'abord, le message est créé par la couche application de A. Ensuite, il descend couche par couche et, pour chacune des couches traversées, il subit des transformations : le message est encapsulé en segments par la couche transport, puis en paquets par la couche réseau et enfin en trames par la couche liaison de données. Arrivé à la couche la plus basse, la couche physique, l'ensemble des trames est envoyé sur le réseau et passe de machine en machine jusque l'ordinateur B. Ce dernier récupère les trames et leur fait remonter les couches de la plus basse jusqu'à la plus haute. Chacune des couches de B défait les modifications effectuées plus tôt par les couches de A correspondantes : les trames sont recomposées en paquets puis en segments et enfin en messages. C'est ainsi que la couche application de B peut récupérer les messages provenant de A.

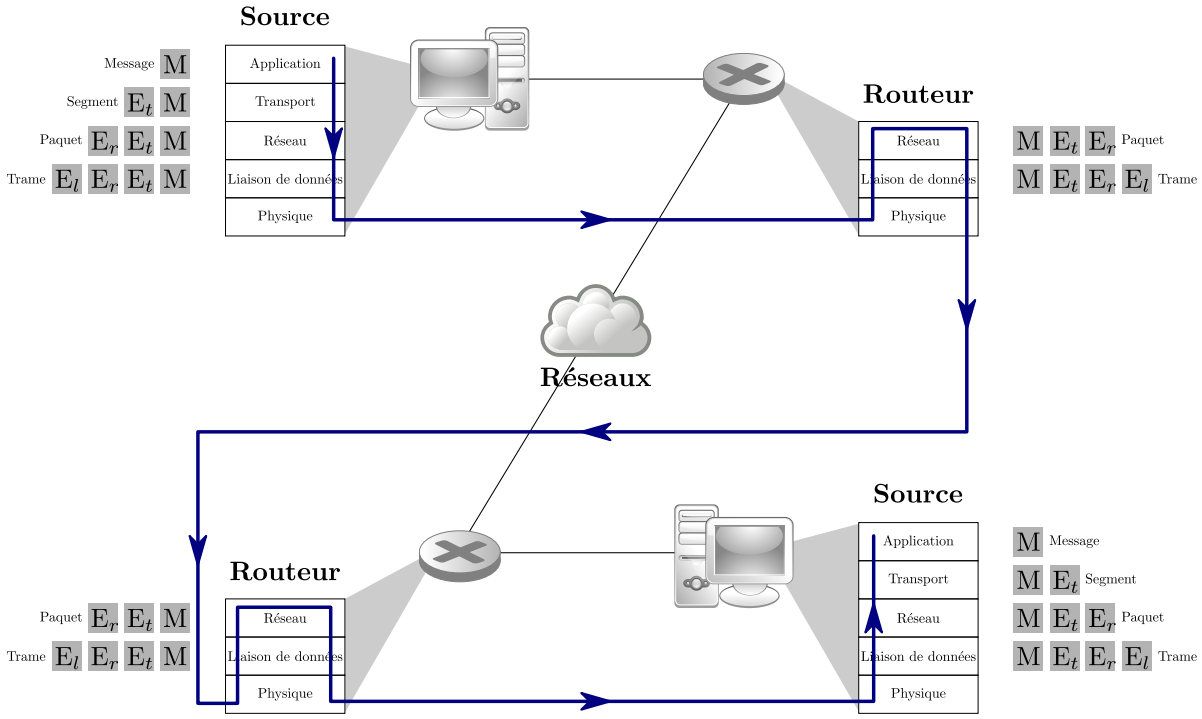


FIGURE 4.2 – Envoi d'un message depuis A vers B (trajet bleu, source [Kurose and Ross, 2008]). Le message M est encapsulé en segments (ajout des en-têtes E_t), puis en paquets (ajout des en-têtes E_r) et enfin en trames (ajout des en-têtes E_l). Les trames sont envoyées au travers du réseau. L'ensemble de nœuds internes aux réseaux travaille soit avec des trames (dans le cas d'un switch) ou avec des paquets (dans le cas d'un routeur). Les trames sont récupérées par la destination, désencapsulées en paquets puis en segments et enfin en message.

Un réseau informatique n'est pas simplement un ensemble de machines connectées entre elles. Pour pouvoir être opérationnel et permettre à plusieurs machines et à leurs usagers de communiquer, il est nécessaire de résoudre un certain nombre de défis et l'étude et la conception d'un réseau informatique est une tâche difficile. Dans le cas des réseaux filaires tels qu'Internet il est nécessaire d'aborder les problématiques dans les différents niveaux : corrections des erreurs de transmission, adressage, routage, sécurité *etc.* Les travaux menés depuis le début des années 1970 qui sont à la fois théoriques (description des protocoles sous formes de machines à états, preuves formelles, *etc.*) et pratiques (les implantations réalisées par différents organismes doivent être interopérables) ont permis d'apporter de nombreuses solutions à l'interconnexion de machines et de réseaux pour composer l'Internet : une toile mondiale utilisée par tout un chacun.

Dans le cas des technologies des réseaux ambiants, s'ajoutent, en plus des défis nouveaux liés à la place de plus en plus importante que tient l'humain dans le système. Les sections suivantes introduisent ces dernières et focalisent sur les réseaux pair-à-pair et les réseaux mobiles ad hoc, deux technologies sur lesquels nous nous sommes plus focalisés.

4.2.3 Modélisation des réseaux informatiques

D'une manière générale, la modélisation du réseau suit l'architecture en couche des modèles standards comme OSI ou TCP/IP [Tanenbaum and Van Steen, 2001; Pujolle, 2008]. Il existe des outils pour modéliser chacune des couches : générateurs de topologie (Brite [Medina et al., 2001], Inet [Winick and Jamin]), générateurs de trafic [Cao et al., 2001], modèles de protocoles des couches deux à quatre (IEEE 802.3, IEEE 802.11, IP, TCP, UDP, *etc.* voir les plates-formes NS2 [8], , son successeur NS3 [9] et GTNetS [10]), modèles de réseaux overlay (simulateurs PlanetSim [Garcia et al., 2005], OverSim [Baumgart et al., 2009], PeerSim [Jelasity et al.]), *etc.* (voir la revue effectuée dans [Casanova et al., 2008]).

De la même manière, les métriques utilisées pour qualifier les performances des réseaux suivent les différents niveaux. Le travail réalisé dans [ITU, 2006] définit les performances et les métriques associées (*grade of service*, *network performances* et *quality of service*) à chacune des différentes couches. Le schéma présenté en figure 4.3 illustre ces différentes métriques.

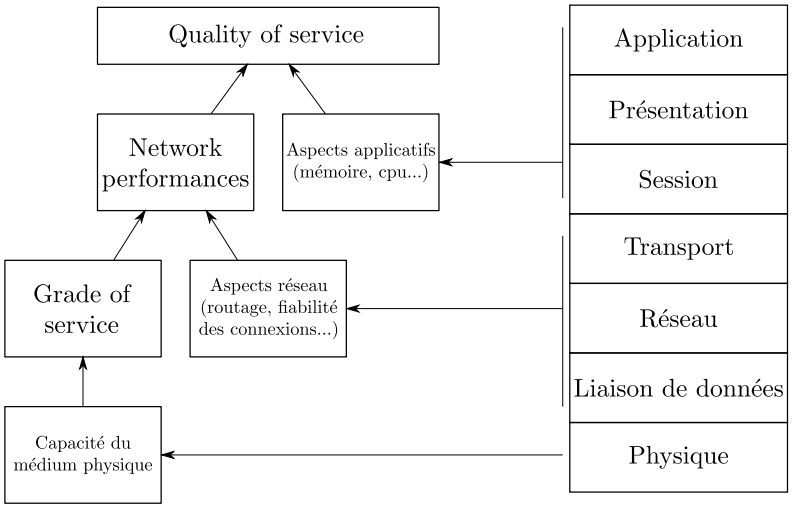


FIGURE 4.3 – Performance globale des réseaux informatiques, sources [ITU, 2006] et [Kurose and Ross, 2008]. Celle-ci peut être décomposée en plusieurs parties : les performances et capacités des liens physiques (appelées *grade of service*), les performances des algorithmes de routage, de nommage, de correction d'erreurs... (appelées *network performance*), les performances et capacités de l'application utilisée (*quality of service*).

8. `http://www.isi.edu/nsnam/ns/`
9. `http://www.nsnam.org/`
10. `http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/`

L'utilisation de chacun de ces outils (modèles, métriques, simulateurs, *etc.*) dépend principalement du phénomène étudié et du degré de précision voulu. Cela dépend également du niveau d'abstraction dans lequel le modélisateur souhaite se placer. Par exemple, les modèles utilisés ne seront pas les mêmes si on cherche à étudier l'efficacité théorique d'un algorithme de routage des requêtes dans un réseau pair-à-pair (une représentation du réseau overlay en graphe peut être considérée comme suffisante) ou ses performances pratiques une fois implanté (un générateur de trafic et une représentation de la topologie deviennent alors nécessaires).

4.3 Modélisation et simulation des réseaux ambiants

Nous souhaitons, dans nos modélisations, tenir compte à la fois des aspects réseaux et des comportements des usagers. Dans cette section, nous passons en revue les outils de modélisation et de simulation qui permettent effectivement de représenter ces deux aspects. Nous nous sommes intéressés à la fois aux outils provenant du domaine des réseaux informatiques et aux outils provenant des sciences humaines qui ciblent explicitement les comportements des usagers des réseaux ambiants.

4.3.1 Des modèles et simulateurs nombreux et changeants

Le premier constat que nous tirons de la littérature est qu'il existe aujourd'hui un grand nombre d'outils de modélisation et de simulation qui peuvent potentiellement nous intéresser. Cependant, seul un petit nombre de simulateurs qui sont maintenus, documentés et utilisés comme outils standards dans la communauté. Les outils comme NS2 [ns2], GTNetS [Riley, 2003] et Omnet++ [Pongor, 1993] existent depuis plusieurs années et sont toujours utilisés et maintenus. Ces outils possèdent un grand nombre de modèles éprouvés qui vont de la couche physique jusqu'à la couche application. Comme le souligne [Casanova et al., 2008], ces outils ont été développés pour modéliser les couches réseaux et sont moins adaptés à la simulation des applications.

En conséquence, d'autres simulateurs ont vu le jour. Certains, comme le simulateur de réseaux pair-à-pair OverSim [Baumgart et al., 2009], se basent sur les outils existants (en l'occurrence Omnet++). D'autres ont été développés spécifiquement pour modéliser les couches hautes du réseau et les applications PlanetSim [Garcia et al., 2005], PeerSim [Jelasity et al.], GloMoSim [Zeng et al., 1998], QualNet [11], Jane [Gorgen et al., 2007].

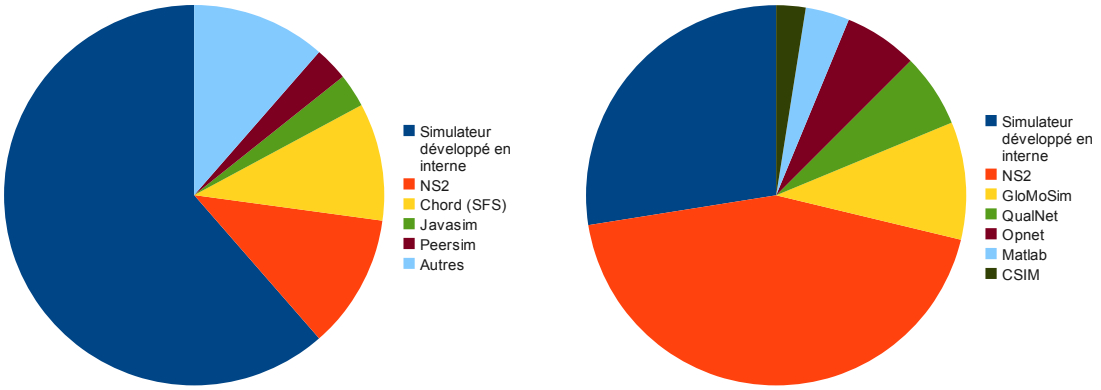
Dans le domaine des réseaux pair-à-pair et des réseaux mobiles ad hoc on observe qu'un grand nombre d'outils de simulation sont développés pour les besoins spécifiques de chaque expérience. Par exemple, dans [Naicken et al., 2006a] les auteurs rapportent que sur 146 articles scientifiques traitant de la simulation des réseaux pair-à-pair, près de la moitié (71) ne spécifient pas l'outil utilisé. Dans les articles restants (75), plus de la moitié des travaux utilisent leurs propres simulateurs (43) et dans le dernier quart, NS2 reste la plate-forme la plus utilisée. Dans le domaine des réseaux mobiles ad hoc, les auteurs de [Kurkowski et al., 2005a] montrent que NS2 est là aussi l'outil le plus utilisé (43,8%) et que plus d'un quart des articles revus utilisent leurs propres simulateurs (27,3%). La figure 4.4 résume les résultats de ces deux revues.

Ainsi, dans la communauté des réseaux ambiants, le nombre d'outils de modélisation et de simulation est relativement élevé. Un petit nombre possède une longue tradition et sont bien ancrés dans le domaine. Cependant, beaucoup d'outils sont développés pour les besoins d'une expérience et ne sont pas réutilisés. Bien que ces derniers puissent être valides et produire des résultats proches des phénomènes réels, il est souvent difficile de reproduire leurs résultats donc de les valider ou de les invalider, de leurs ajouter de nouveaux modèles ou de modifier ceux existants.

4.3.2 Un réel besoin de différents niveaux d'abstraction

Le deuxième constat que nous tirons de la littérature est qu'il existe aujourd'hui un réel besoin d'une modélisation qui tienne compte de plusieurs niveaux d'abstraction. En effet, en fonction des problématiques que l'on souhaite aborder dans ce domaine, il peut être nécessaire d'étudier le fonctionnement soit de la couche physique, soit de la couche IP ou seulement de la couche application, voire

11. `http://www.scalable-networks.com/products/qualnet/`



(a) Simulateurs utilisés dans le domaine des réseaux pair-à-pair [Naicken et al., 2006a].

(b) Simulateurs utilisés dans le domaine des réseaux mobiles ad hoc [Kurkowski et al., 2005a]

FIGURE 4.4 – Revue des simulateurs utilisés.

de tenir compte de plusieurs couches en même temps. De ce fait, les plates-formes de simulation les plus génériques telles que NS2, NS3, Omnet++ ou SimGrid [Casanova et al., 2008] qui proposent des approches d'intégration de modèles sous forme de différents composants (*plugins*) afin de représenter les différentes couches du réseau en un même outil.

Les outils de simulation présents dans la communauté sont développés pour représenter le fonctionnement du réseau. Les outils comme NS2, Omnet++, GTNetS représentent d'une manière précise les couches basses du réseau. Les plates-formes de simulation comme OverSim, PlanetSim, PeerSim, Jane [Gorgen et al., 2007], *etc.* se focalisent davantage sur les couches hautes du réseau (protocoles et services de la couche application). Avec l'avènement des réseaux ambiants, sont apparues les problématiques d'influence du comportement des usagers. Aujourd'hui, les phénomènes de connexions et déconnexions intempestives des usagers dans les réseaux pair-à-pair (le *churn*) et la mobilité des utilisateurs dans les réseaux mobiles ad hoc sont certainement les plus connus dans la communauté. De ce fait, les plates-formes de modélisation et de simulation qui ciblent les réseaux pair-à-pair ou les réseaux mobiles ad hoc incluent en général soit des modèles de *churn* [Naicken et al., 2006a], soit des modèles de mobilité. Par exemple, dans le domaine des réseaux mobiles ad hoc, les auteurs de [Kurkowski et al., 2005a] montrent que 91,2% des papiers revus citent le modèle de mobilité utilisé.

4.3.3 Des influences en sens unique

La plupart des modèles de comportements des usagers qui existent dans la communauté des réseaux ambiants ont pour but de fournir un ensemble de traces au modèle du réseau. Par exemple, les modèles de *churn* dans le domaine des réseaux pair-à-pair indiquent combien d'usagers se connectent et se déconnectent du réseau au cours du temps. De même dans les réseaux mobiles ad hoc les modèles de mobilité indiquent la position des usagers au cours du temps. On dit que ces modèles d'usagers paramétrent les modèles de réseaux (voir schéma 4.5).

Cette approche ne permet pas de représenter, comme nous le souhaitons, des comportements d'utilisateurs qui évoluent en fonction des performances du réseau. En effet, il n'y a pas de retour d'information depuis le modèle du réseau jusqu'au modèle du comportement des usagers. Nous atteignons ici les limites des plates-formes de modélisation et de simulation des réseaux dont le point de vue est centré sur le fonctionnement du réseau. Dans notre cas d'étude, il nous est nécessaire de représenter les causes des comportements des usagers. Or, ce travail n'est pas restreint au seul domaine des réseaux ambiants. Il nécessite le point de vue d'autres domaines scientifiques qui permettent de représenter les comportements humains.

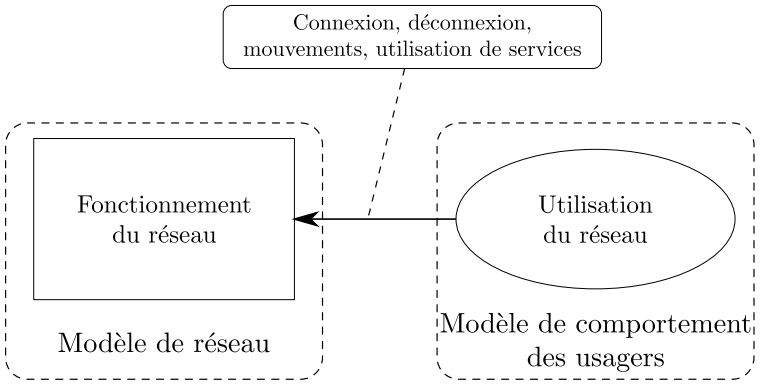


FIGURE 4.5 – Paramétrage d'un modèle de réseau par un modèle de comportements d'utilisateurs.

4.3.4 Des modèles de comportements souvent limités

Il est important de noter que le point de vue "orienté réseau" a une influence sur le type de modèles de comportements des usagers utilisés. En voyant les modèles de comportements des usagers comme des paramètres d'entrée du modèle réseau, il n'est pas nécessaire que ceux-ci représentent une quelconque réaction des usagers au fonctionnement du réseau. Effectivement, dans la littérature, les modèles classiquement utilisés pour représenter les comportements des usagers des réseaux ambiants sont pour la plupart inadaptés à la représentation l'hétérogénéité des comportements.

Par exemple, dans les réseaux mobiles ad hoc, la majorité des travaux (64,5% selon [Kurkowski et al., 2005a]) utilisent le modèle de mobilité *random waypoint* dans lequel chaque usager bouge de manière aléatoire. D'autres modèles de mobilité tentent de prendre en compte des comportements plus proches de la réalité comme les mouvements de groupes ou l'évitement d'obstacle, voir les revues [Camp et al., 2002; Lin et al., 2004; Bai and Helmy, 2004]. De même, dans le domaine des réseaux pair-à-pair, la plupart des modèles de comportements des usagers utilisés ne permettent pas de représenter l'hétérogénéité des comportements. Par exemple, parmi les modèles qui représentent la volonté de partager des utilisateurs ou leur conscience de la pollution des fichiers échangés, les auteurs de [Feldman et al., 2004] proposent une mise en équation du phénomène du manque de partage. Dans ce modèle, le nombre d'usagers connectés au réseau et leurs comportements sont figés dans le temps ce qui, en pratique, n'est pas le cas. D'autres travaux s'intéressent au partage et à la pollution des données [Dumitriu et al., 2005; Lee et al., 2006; Kumar et al., 2006]. Leurs modèles se basent sur une approche itérative qui calcule le nombre d'utilisateurs qui partagent et le taux de pollution dans le temps. Cependant, chaque utilisateur possède le même comportement et, comme dans le modèle précédent, le nombre d'usagers et leurs comportements restent figés dans le temps.

De telles caractéristiques comme l'hétérogénéité, l'évolution des comportements et la réaction des usagers au fonctionnement du réseau sont très difficiles à obtenir avec des modèles mathématiques classiques décrits par des équations ou par des graphes. Seule une approche individu-centrée, notamment avec le paradigme multi-agent, permet d'obtenir de telles propriétés [Van Dyke Parunak et al., 1998].

4.4 Modélisation multi-agent appliquée aux usagers des réseaux ambiants

Dans cette section nous souhaitons montrer les avantages de la modélisation multi-agent pour représenter les usagers des réseaux ambiants. Bien que, dans le domaine des réseaux informatiques, l'approche multi-agent soit utilisée pour proposer de nouveaux algorithmes de routage, de répartition de charge (*load-balancing*) *etc.* (voir [Gaïti and Merghem-Boulahia, 2006]), nous nous focalisons ici uniquement sur la modélisation du comportement des usagers.

En guise d'exemple, nous avons choisi la mobilité des usagers dans les réseaux mobiles ad hoc. Nous aurions également pu présenter les travaux portant sur la question de la réputation et de la coopération

des usagers des réseaux pair-à-pair qui utilisent des modèles de confiance et de réputation basés sur les systèmes multi-agents (voir [Hales, 2004; Morge and Mathieu, 2007; Grizard et al., 2006]). Cependant, nous pensons que l'exemple de la mobilité des usagers reflète à lui seul l'intérêt de l'approche multi-agent pour la modélisation des usagers des réseaux ambiants.

4.4.1 Modèles classiques de mobilité

Nous nous appuyons ici sur les revues effectuées dans la littérature pour décrire l'ensemble des différents modèles de mobilité utilisés dans les simulations des réseaux mobiles ad hoc [Camp et al., 2002; Bai and Helmy, 2004]. Les différentes catégories de modèles de mobilité peuvent être classés selon les caractéristiques représentées dans chaque modèle (voir figure 4.6).

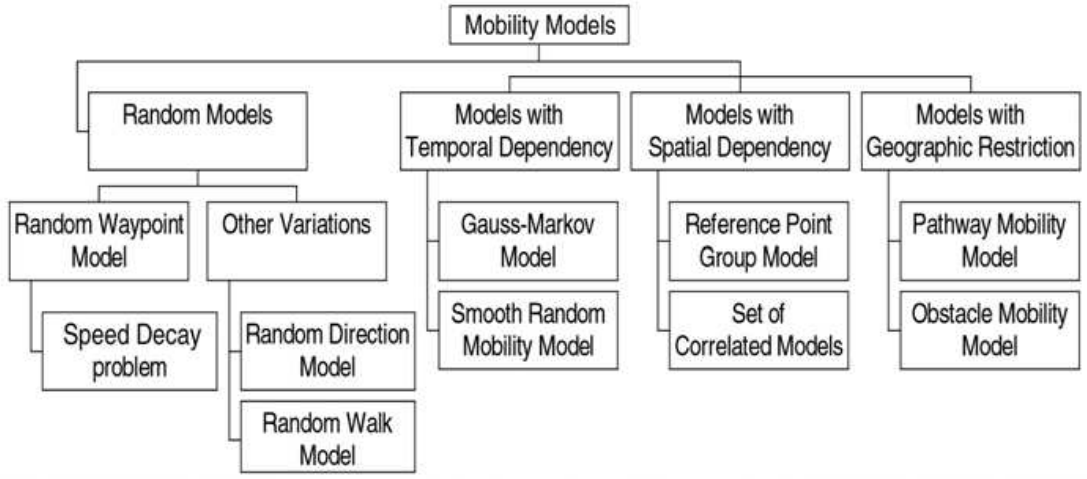


FIGURE 4.6 – Les différents types de modèles de mobilité utilisés dans la communauté des réseaux mobiles ad hoc. D'après [Bai and Helmy, 2004]

Tout d'abord, les modèles de mobilité aléatoires (*random models*) représentent les usagers comme des entités qui se meuvent de manière aléatoire. Cette catégorie contient, par exemple, le modèle très répandu *random waypoint* dans lequel les utilisateurs bougent dans une direction pendant un temps donné puis changent aléatoirement leurs directions et leurs vitesses.

Les modèles avec dépendance temporelle (*temporal dependancy*) représentent le fait que, dans la plupart des cas, les utilisateurs ne changent pas brutalement de direction et de vitesse et que ces dernières caractéristiques sont souvent corrélées dans le temps. Par exemple, les modèles de mobilité de type *Gauss-Markov* et *Smooth random* prennent en compte la vitesse et la direction du passé (au temps $t-1$) pour calculer la vitesse et la direction à l'instant présent (au temps t).

Les modèles avec dépendance spatiale (*spatial dependancy*) considèrent, quant à eux, que le mouvement d'un utilisateur dépend aussi des mouvements des autres usagers. Cela permet de représenter, par exemple, des mouvements en groupe ou d'éviter les collisions entre les différents usagers. Le modèle de mobilité type *Reference point*, représente un groupe d'usagers et un guide. La vitesse et la direction des membres du groupe dépend, dans ce modèle, de la vitesse et de la direction du guide auxquelles s'ajoutent une déviation propre à chaque utilisateur.

Enfin, les modèles avec restriction géographique (*Geographic restriction*) considèrent que les mouvements des usagers des réseaux mobiles ad hoc sont contraints par leur environnement (rues, obstacles, *etc.*). Une solution relativement simple pour intégrer les informations propres à l'environnement est de restreindre les mouvements des usagers sur des chemins prédéfinis. Ainsi le modèle *path mobility* définit l'ensemble des chemins que peut parcourir un usager comme étant un graphe (les nœuds étant les destinations et les liens les chemins) puis chaque utilisateur choisit aléatoirement une destination et se déplace suivant le chemin le plus court.

4.4.2 Modélisation multi-agent appliquée aux réseaux mobiles ad hoc

La plupart des modèles de mobilité permettent de représenter chacun une caractéristique particulière de la mobilité des utilisateurs des réseaux mobiles ad hoc. L'approche multi-agent permet de prendre en compte plusieurs caractéristiques simultanément (restriction géographique + dépendance temporelle + dépendance géographique) voire d'en intégrer de nouvelles (buts à atteindre, environnements dynamiques). Le paradigme multi-agent permet de représenter directement les causes de la mobilité comme la présence d'obstacles, de points de rendez-vous ou tout simplement les autres usagers et d'en décrire un ou des comportement(s) résultant(s) (mouvements de groupes, évitement *etc.*). De plus, les travaux dans le domaine de la modélisation urbaine fournissent des modèles multi-agents relativement réalistes issus d'observation des mouvements de foule, de piétons, de flux de trafic *etc.* [Helbing et al., 2001; Teknomo et al., 2000]. Dans un contexte où les réseaux mobiles ad hoc ne sont pas encore déployés à grande échelle, obtenir des traces des mouvements d'utilisateurs est difficile. De plus, les données issues du réseau sont sujet aux lois sur la protection des données personnelles. En plus d'être difficile à obtenir, ces données doivent être anonymisées pour pouvoir être utilisées. S'appuyer sur des modèles multi-agents de mobilité tels que ceux présents dans les sciences sociales (modélisations urbaines) apporte l'avantage de pouvoir créer facilement des ensembles de trajectoires réalistes d'usagers (des traces). Ces traces sont ensuite utilisées comme paramètre pour évaluer les protocoles et les services des réseaux mobiles ad hoc.

L'approche de modélisation multi-agent a été utilisée pour modéliser la mobilité des usagers des réseaux mobiles ad hoc. Par exemple, le simulateur multi-agent MMTS (*Microscopic Multi-Agent Traffic Simulator*) [Sommer, 2007] développé par l'ETH Zurich est utilisé pour générer les trajectoires des usagers des réseaux mobiles ad hoc et mesh. Bien que le terme multi-agent n'apparaisse pas textuellement, l'algorithme d'évitement des obstacles de l'outil mobiReal [Maeda et al., 2005] reprend les principes multi-agents. De même, les auteurs de [Legendre et al., 2006; Borrel et al., 2009] utilisent aussi la modélisation multi-agent (appelée ici *behavioral modeling*) pour définir des modèles de mobilité plus réalistes et plus évolués que les modèles classiquement utilisés.

4.4.3 Description d'un modèle de mobilité multi-agent de type piéton

Le but de cette section est double. Il s'agit, d'une part, de présenter un modèle de mobilité multi-agent qui est aujourd'hui une référence dans le monde de la modélisation urbaine et, d'autres part, de montrer que ce modèle répond bien aux différentes caractéristiques énoncées par [Bai and Helmy, 2004].

Dans les modèles microscopiques de mobilité et notamment dans les modèles multi-agents, les usagers (aussi appelés nœuds mobiles) sont représentés en tant que tel par des entités spécifiques (des agents) qui possèdent une certaine dynamique. Dans le cas des modèles multi-agents, est en plus considéré un environnement. La plupart du temps, le comportement des usagers est décrit sous forme de règles relativement simples comme par exemple "éviter un obstacle", "suivre un autre utilisateur" ou "rebondir sur les murs". Ces règles sont ensuite composées pour former un comportement plus complexe. Il est à noter que la composition des règles de comportements n'est pas anodine et doit se faire suivant un ordre précis pour ne pas introduire d'erreur dans le modèle. Par exemple, l'usager modélisé doit en premier lieu éviter les obstacles avant de suivre ses différents buts si l'on souhaite que ce dernier ne passe pas au travers des obstacles [Brooks, 1986].

Dirk Helbing propose un modèle microscopique de mobilité des piétons. L'ensemble des détails est décrit dans [Helbing, 1991; Helbing et al., 2001]. Nous ne décrivons ici que les principes de ce modèle. Celui-ci se base sur quelques règles relativement simples :

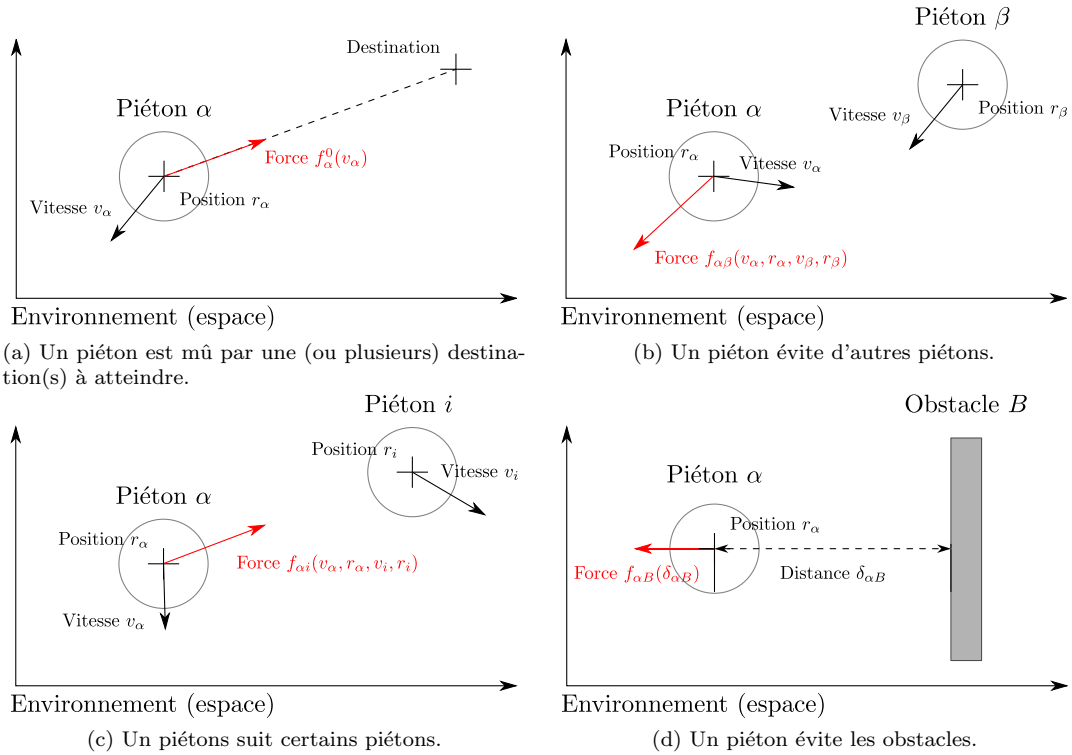
- Un piéton (noté α) se déplace généralement vers une ou un ensemble de destinations à une certaine vitesse (v_0 : la vitesse préférée plus ou moins une certaine marge).
- Un piéton possède un angle de vision et un champ de vision duquel il peut percevoir son environnement (les obstacles et les autres piétons).
- Un piéton évite les obstacles de l'environnement (murs, lampadaires, *etc.*).
- Un piéton peut éviter les autres piétons et/ou suivre certains piétons (déplacement en groupe).

Dans ce modèle, à chaque règle qui intervient sur le mouvement (évitement des obstacles, atteindre les destinations, *etc.*) correspond une force (voir figure 4.7). La composition de ces différentes règles fournit le mouvement d'un piéton et se calcule suivant l'équation 4.1 qui stipule que l'accélération d'un piéton est égale à la somme des forces qui s'appliquent à lui (notées $f_\alpha(t)$) plus une certaine marge (notée $\xi_\alpha(t)$). La somme des forces reprend l'ensemble des règles énoncées précédemment : la volonté d'atteindre un but (notée f^0_α), l'évitement des obstacles (notée $f_{\alpha B}$), l'évitement des autres piétons (notée $f_{\alpha\beta}$) et le suivi de certains piétons (noté $f_{\alpha i}$) (voir l'équation 4.2). Chaque force dépend elle même de différents paramètres comme, par exemple, la position du piéton (notée r_α), sa vitesse (notée v_α), la distance du piéton à un obstacle (notée $\delta_{\alpha B}$). La figure 4.7 illustre les différentes règles composant le modèle de comportements d'un piéton selon [Helbing et al., 2001].

$$\frac{\mathrm{d}v_\alpha}{\mathrm{d}t} = f_\alpha(t) + \xi_\alpha(t) \tag{4.1}$$

$$f_\alpha(t) = f^0_\alpha(v_\alpha) + f_{\alpha B}(\delta_{\alpha B}) + \sum_{\beta(\neq\alpha)} f_{\alpha\beta}(v_\alpha, v_\beta, r_\alpha, r_\beta) + \sum_i f_{\alpha i}(v_\alpha, v_i, r_\alpha, r_i) \tag{4.2}$$

Il est à noter que le comportement "stop" qui spécifie que le piéton modélisé est immobile se traduit dans ce modèle par une vitesse nulle et une accélération nulle (l'équation 4.1 doit être égale à 0). $\xi_\alpha(t)$ étant un paramètre interne au piéton, il est facile de le rendre nul. Cela signifie que le piéton ne doit plus tenir compte des différentes forces qui pourraient s'appliquer à lui. Le comportement "stop" doit alors inhiber les autres comportements basiques en annulant $f_\alpha(t)$.



(a) Un piéton est mû par une (ou plusieurs) destination(s) à atteindre.

(b) Un piéton évite d'autres piétons.

(c) Un piétons suit certains piétons.

(d) Un piéton évite les obstacles.

FIGURE 4.7 – Modèle de piéton [Helbing, 1991]. Description des différentes règles composant le comportement d'un piéton

4.4.4 Avantages et limites de l'approche multi-agent pour la modélisation de la mobilité

Le premier avantage d'un tel modèle multi-agent de mobilité est qu'il permet de satisfaire les différentes propriétés énoncées par [Bai and Helmy, 2004]. La dépendance temporelle est atteinte par le fait de tenir compte d'une (ou de plusieurs) destination(s). Le piéton ainsi modélisé ne change pas abruptement au cours du temps de direction et de vitesse. La dépendance spatiale est atteinte par la possibilité d'éviter les collisions avec des piétons et de définir des ensembles de piétons à suivre. Enfin, la restriction géographique est atteinte par le fait d'inclure directement l'environnement dans le modèle et par la possibilité d'éviter les obstacles. Concernant la catégorie des mouvements aléatoires, il est aussi possible de reproduire des modèles comme la marche aléatoire, par exemple, en choisissant aléatoirement différentes destinations au cours du temps.

Le deuxième avantage d'un tel modèle est d'être facilement extensible. En effet, l'approche agent permet aisément d'ajouter de nouveaux comportements que ce soit pour l'ensemble des agents ou pour une population donnée. De plus, l'environnement est modélisé dès le départ. Il est donc possible, par exemple, d'ajouter une zone dans laquelle un piéton s'arrête pendant quelques temps. Dans le cadre de notre étude sur les phénomènes d'inter-influences, l'approche agent nous autorise à définir de nouveaux comportements comme le fait que l'usager réagisse au fonctionnement du réseau. Il est tout à fait possible, par exemple, de représenter des usagers qui s'arrêtent pour utiliser les services du réseau. Pour ce modèle particulier, la composition des différents comportements basiques ne pose pas de problème. Tant que ces comportements de base s'apparentent à des forces, la composition revient à effectuer la somme des forces présentes.

La principale limite de ce modèle vient du paramétrage [Bourgois et al., 2010]. En effet, le nombre de paramètres est relativement élevé comparé aux modèles plus classiques et de mauvais paramètres peuvent provoquer des résultats inattendus. Par exemple, une trop faible force d'évitement des obstacles peut conduire à un modèle de mobilité dans lequel les piétons traversent les obstacles. À l'inverse, une force trop élevée va conduire à un modèle de mobilité dans lequel les piétons ne pourront pas approcher d'obstacles ou passer dans un couloir. Il est donc nécessaire de s'appuyer sur les différents travaux de la littérature du domaine de la modélisation urbaine pour pouvoir utiliser ce type de modèle (voir par exemple [Helbing et al., 2001; Gaud, 2007; Bourgois et al., 2010]).

4.5 Synthèse

Le domaine des réseaux informatiques accorde une place importante à la démarche de modélisation et de simulation. Il en résulte que de nombreux modèles et simulateurs ont vu le jour dans cette communauté. Ces outils reposent sur une forte tradition de modélisation des réseaux informatiques et représentent les différentes couches du modèle OSI. Avec l'arrivée ces dernières années des réseaux pair-à-pair et le développement des réseaux mobiles ad hoc, les outils de simulation incorporent désormais des aspects liés aux applications et certains représentent même les comportements des usagers.

Dans ce travail de thèse, nous souhaitons représenter à la fois dynamique du réseau et la dynamique des comportements des usagers. Mais surtout, nous cherchons à modéliser les phénomènes d'inter-influences qui peuvent avoir lieu entre ces deux dynamiques. L'objectif principal des outils de modélisation et de simulation des réseaux ambiants est de modéliser la dynamique et le fonctionnement de ces réseaux. Les comportements des usagers sont alors vus comme des paramètres d'entrée du modèle de réseau et les modèles utilisés ne permettent pas de représenter des usagers qui réagissent au fonctionnement du réseau. À l'opposé, la modélisation multi-agent permet de représenter des usagers dont les comportements sont hétérogènes, dynamiques et qui peuvent réagir à des évènements extérieurs. Cependant, dans ces outils, le réseau est vu comme un paramètre d'entrée du modèle des usagers.

Il nous apparaît donc comme nécessaire de faire interagir à la fois un modèle du réseau et un modèle des usagers pour pouvoir représenter les phénomènes d'inter-influences qui nous intéressent. Dans le chapitre suivant (chapitre 5), nous nous intéressons aux aspects scientifiques propres au couplage de modèles. Le couplage de modèle soulève un ensemble de questions aussi bien d'un point de vue purement technique

que conceptuel. Parmis ces questions, il nous a semblé pertinent de prendre du recul par rapport à notre cas d'étude et de nous intéresser aux aspects les plus conceptuels du couplage de modèles. Le cas d'étude des réseaux ambiants sert alors de cadre applicatif, pour évaluer nos contributions théoriques.

Chapitre 5

Couplage de modèles et interopérabilité de simulations

Sommaire

5.1 Introduction

Ce chapitre présente les différentes problématiques rencontrées lors de la démarche de couplage de modèles (encore appelée multi-modélisation, voir section 5.2). La multi-modélisation faisant appel à de nombreux domaines, nous structurons l'ensemble des questions en plusieurs niveaux : le niveau conceptuel, le niveau sémantique, le niveau syntaxique, le niveau dynamique et le niveau technique. Cette classification a pour but premier de rendre notre travail plus intelligible et comparable aux travaux existants (voir section 5.4).

Nous passons ensuite en revue les différentes contraintes ou besoins inhérents à notre cas d'étude. Étant donné la spécificité des phénomènes étudiés et la quantité d'outils de modélisation et de simulation présents dans le domaine de l'informatique ambiante, nous posons des contraintes en terme de réutilisation, d'interopérabilité, de modularité et de passage à l'échelle (voir section 5.3).

Nous présentons par la suite les approches existantes de couplage de modèles, leurs avantages et leurs limites. Nous distinguons les approches de multi-modélisation ad hoc qui sont mises en place pour traiter un problème particulier et qui n'ont pas pour vocation d'être des approches génériques et les méthodes de multi-modélisation dédiée (voir section 5.5). Nous montrons que ces méthodes apportent des réponses techniques aux problèmes liés à la multi-modélisation. Bien que ces solutions techniques soient indispensables à la mise en œuvre de la multi-modélisation, ces approches manquent de solutions aux niveaux conceptuels. Nous pensons d'autres solutions sont à rechercher du côté de la modélisation multi-agent.

5.2 Démarche de multi-modélisation

Dans la partie précédente, nous avons montré que l'étude des phénomènes d'inter-influences nécessite de prendre en compte simultanément plusieurs modèles. D'une manière générale, modéliser un système complexe par un seul modèle n'est pas toujours la solution la plus simple. En effet, dans ce type de système il est nécessaire de représenter différentes dynamiques et leurs interactions. Ces dynamiques peuvent être représentées par des domaines et des niveaux d'abstraction particuliers et requérir des techniques et des connaissances provenant de domaines scientifiques différents. La solution consiste à représenter chacune de ces dynamiques élémentaires par un modèle spécifique et de représenter le système complexe par un ensemble de modèles en interaction. Cette approche se nomme la multi-modélisation.

La multi-modélisation *est la démarche qui consiste à coupler différents modèles.*

La démarche de multi-modélisation est un cas particulier de modélisation. Tout comme cette dernière, la multi-modélisation cherche à répondre à une question sur un système étudié en créant un modèle de ce système. À la différence de la modélisation "classique" qui ne crée qu'un seul modèle, la multi-modélisation représente le système étudié par un ensemble plusieurs modèles couplés.

L'ensemble des modèles couplés se nomme le multi-modèle *[Zeigler and Ören, 1986; Fishwick and Zeigler, 1992].*

Nous appelons graphe de dépendances*, l'ensemble des relations qui lient les différents modèles.*

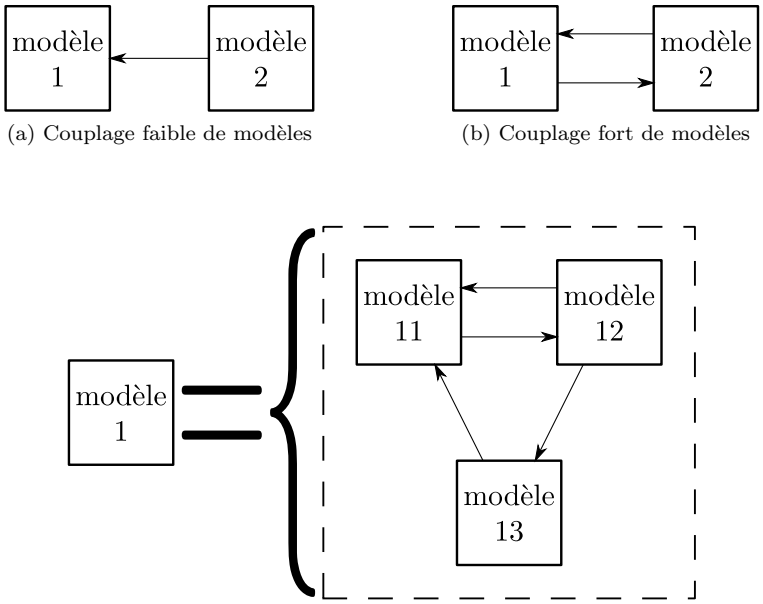
On distingue généralement trois types de couplages [Ramat, 2006] : le couplage faible, le couplage fort et le couplage hiérarchique. Le **couplage faible**, aussi appelé paramétrage, définit une relation unidirectionnelle entre deux modèles. Les résultats d'un modèle sont utilisés en entrée (comme paramètres) d'un deuxième modèle (voir figure 5.1a) mais l'inverse n'est pas vrai (le graphe de dépendance est orienté et acyclique). Dans le cas du **couplage fort**, aussi appelé couplage dynamique, les deux modèles s'influencent l'un et l'autre (voir figure 5.1b). Le **couplage hiérarchique** définit, quant à lui, une relation de décomposition. Un modèle ou une partie de ce modèle est représenté par des sous-modèles interconnectés (voir figure 5.1c). Ce dernier cas est une généralisation du couplage fort.

Nous laissons ici de côté le couplage faible de modèles. En effet, celui-ci représente pour nous peu d'intérêt dans la mesure ou nous cherchons à modéliser les phénomènes d'inter-influences qui sont présents dans des systèmes complexes comme les réseaux ambiants. Nous nous focalisons sur les couplages de modèles de type forts et hiérarchique. D'une manière générale, nous affirmons que la modélisation des systèmes complexes implique automatiquement ce genre d'approche.

5.3 Contraintes propres à notre cas d'étude

5.3.1 Réutilisation et interopérabilité

Nous avons vu dans la partie précédente que les outils de modélisation existent à la fois dans le domaine des réseaux informatiques et dans le domaine des sciences sociales. Nous posons donc une première contrainte forte qui est de pouvoir réutiliser les modèles et les simulateurs présents dans ces



(a) Couplage faible de modèles

(b) Couplage fort de modèles

(c) Couplage hiérarchique de modèles

FIGURE 5.1 – Différents types de couplages de modèles. D'après [Ramat, 2006].

domaines. Nous posons comme corollaire de cette première contrainte que les outils réutilisés doivent pouvoir être utilisés à la fois de manière unique (*standalone*) et dans le multi-modèle.

Il est intéressant de réutiliser les outils de modélisation et de simulation à la fois pour accélérer la mise en place du multi-modèle et pour bénéficier d'outils vérifiés et de modèles réalistes et valides. Cependant, il ne faut pas perdre de vue que, d'une part, la plupart du temps, ces outils n'ont pas été conçus pour interagir avec d'autre. D'autre part, ces outils sont à différents niveaux de leurs cycles de vie : certains sont abandonnés, d'autres sont maintenus par une communauté ou encore en plein développement. Ainsi, il nous paraît important d'apporter le minimum de modifications aux outils existants que l'on souhaite réutiliser.

5.3.2 Modularité

Nous avons également vu que, dans le domaine des réseaux ambiants, les problématiques liées aux phénomènes d'inter-influences entre les comportements des usagers et le fonctionnement des réseaux se retrouvent dans différentes technologies (réseaux de capteurs sans fil, réseaux mobiles ad hoc, réseaux pair-à-pair). Plus simplement, il peut être intéressant de comparer plusieurs protocoles de réseaux face à différents cas d'usage. Cela principalement dans une optique de *benchmarking*. Les modèles de protocoles peuvent être modélisés par différentes personnes voire implantés dans des simulateurs différents. Idem pour les modèles des cas d'usages. Nous posons donc comme deuxième contrainte forte le fait de pouvoir facilement changer nos modèles élémentaires dans le multi-modèle.

5.3.3 Passage à l'échelle

Il nous faut définir ce que nous entendons par passage à l'échelle. Les réseaux ambiants sont des systèmes qui comprennent généralement un grand nombre d'usagers (de l'ordre du million d'usagers pour les réseaux pair-à-pair, actuellement de l'ordre de la centaine, voire du millier pour les réseaux mobiles ad hoc). Modéliser de tels systèmes implique de pouvoir représenter un ordre de grandeur suffisamment grand pour que les résultats de simulation soient satisfaisants. Nous parlerons dans ce cas de *size-up* [12].

12. Terme emprunté à l'informatique parallèle. Le *size-up* désigne la capacité à résoudre des problèmes de taille de plus en plus grande [Bouzid, 2001]

Ensuite, au vu de la taille du phénomène étudié, il est aussi nécessaire que la simulation se fasse en un temps satisfaisant. Nous parlerons dans ce cas de *speed-up*[13]. Nous ajouterons une nouvelle définition dans le passage à l'échelle. Les phénomènes que nous cherchons à modéliser nous obligent à considérer simultanément différents niveaux d'abstraction (microscopique, mésoscopiques, macroscopique). Ces niveaux d'abstraction interagissent ensemble. Il est donc nécessaire de pouvoir représenter les niveaux d'abstraction, voire de pouvoir en ajouter ou en supprimer selon les besoins de l'étude. Nous parlerons ici de passage à l'échelle de type *level-up*.

Le passage à l'échelle de type level-up *est la capacité d'une approche de modélisation à permettre l'ajout (ou la suppression) de niveaux d'abstraction.*

Nous ajoutons, comme dernière contrainte forte, le passage à l'échelle de type *level-up* de la multi-modélisation. Les deux autres types de passage à l'échelle (*size-up* et *speed-up*) nous semblent moins important au regard des phénomènes étudiés.

5.4 Structuration des problèmes rencontrés en niveaux

Créer un multi-modèle pose de nouvelles questions par rapport à l'approche de modélisation et de simulation classique. Nous classons ces dernières en différents niveaux. Ces niveaux nous permettent simplement de structurer les familles de problèmes auxquels le modélisateur doit faire face durant la multi-modélisation. Cette structuration n'est pas sans rappeler les niveaux de spécifications introduits par Zeigler [Zeigler et al., 2000] et les niveaux de connaissances introduits par Klir [Klir, 1984]. Ces derniers permettent de structurer méthodologiquement l'approche de modélisation et de simulation. Pour notre part nous proposons une structuration plus proche de celle proposée par Tolk (*Levels of Conceptual Interoperability Models*) présenté dans [Tolk and Muguira, 2003; Cantot et al., 2009] dans le domaine de l'interopérabilité des simulations.

5.4.1 Niveau conceptuel

Le niveau conceptuel est le niveau le plus élevé. Ici, les modèles sont vus comme des boîtes noires et la discussion porte sur la manière de représenter le multi-modèle. Les questions qui se posent à ce niveau sont, par exemple : "quelle représentation du multi-modèle doit être utilisée pour répondre aux questions portant sur un système complexe particulier ?" ou encore "une représentation du multi-modèle est-elle assez générique pour s'appliquer à d'autre type de systèmes complexes ?". En fonction de l'étude menée, le modélisateur peut avoir besoin de définir les notions de système ouvert, d'environnement, d'échelle ou de niveau d'abstraction. De même, il peut être intéressant de voir le multi-modèle comme un système de système ou encore comme un système multi-agent.

Le niveau conceptuel *porte sur les questions de méta-modélisation du multi-modèle.*

5.4.2 Niveau sémantique

Le niveau sémantique porte sur la signification et le rôle du multi-modèle face aux questions posées sur le système étudié. Les modèles élémentaires utilisés dans le multi-modèle sont développés par les thématiciens des domaines concernés. Dans notre cas ce sont les modèles d'usagers et de réseaux. On se pose ici des questions comme "que doivent représenter chacun des modèles ?", "comment interagissent les modèles ?", "comment est représenté un même objet dans des modèles différents ?" ou encore "que signifie passer d'un niveau d'abstraction à un autre ?".

D'une manière générale, le multi-modèle doit répondre à une question sur le phénomène étudié. Le modélisateur ne conçoit pas seulement un modèle pour répondre à une question sur le système étudié mais couple un ensemble de modèles élémentaires pour ce faire. Il faut dès lors définir le graphe d'interaction du multi-modèle puis définir ce que signifient chacune des flèches de ce graphe. Ensuite, il faut faire en sorte

13. Terme là aussi emprunté à l'informatique parallèle. Le *speed-up* désigne la capacité à accélérer les traitements informatiques [Bouzid, 2001]

que l'ensemble des objets et phénomènes représentés dans chacun des modèles élémentaires s'accordent afin que le multi-modèle soit cohérent sémantiquement (voir l'exemple donné par la figure 5.2).

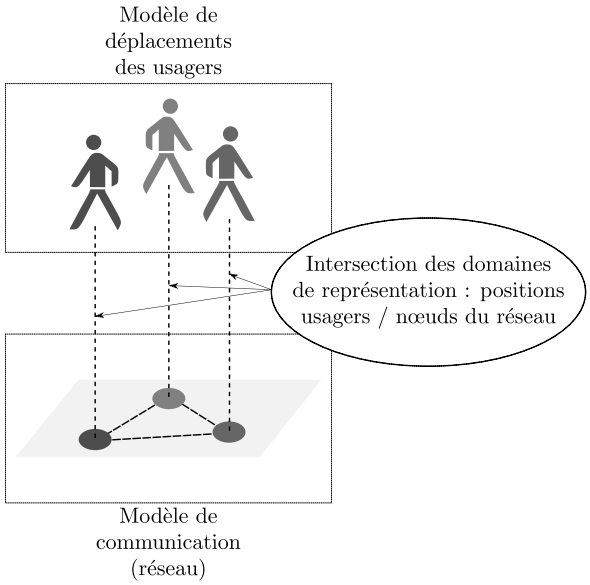


FIGURE 5.2 – Exemple d'intersection de domaines de représentation de différents modèles. Cet exemple illustre la multi-modélisation d'un réseau mobile représenté par un modèle de déplacement des usagers (en haut) et un modèle de réseau (en bas). Ces deux modèles partagent une information commune (on dit que leurs domaines d'abstraction s'intersectent). En effet, la position spatio-temporelle des usagers correspond à la position des nœuds du réseau.

Sachant que les modèles sont peut-être représentés à des échelles spatio-temporelles différentes, le modélisateur doit définir comment une information d'un modèle doit être perçue par un autre. Par exemple, la qualité de services du réseau au cours du temps dépend notamment d'un certain nombre de messages qui sont relativement rapides et fréquents (de l'ordre de la milliseconde). Si l'on souhaite modéliser des utilisateurs qui perçoivent la qualité de services et qui modifient leurs comportements en fonction de celle-ci, l'échelle temporelle ne sera pas la même (de l'ordre de la minute ou de l'heure) car un comportement humain n'évolue pas aussi rapidement et fréquemment. Il faut représenter la qualité de services perçue pour un usager, par exemple le nombre de messages d'erreur survenus dans les dernières 10 minutes (voir figure 5.3). Le modélisateur doit donc réifier et spécifier les interactions entre les modèles.

Outre les questions de modélisation qui se posent pour chacun des modèles élémentaires, se posent ici des questions de modélisation quant à l'interaction de ces modèles élémentaires au sein du multi-modèle. La définition du graphe d'interaction et la gestion de la cohérence sémantique sont autant de choix faits par le modélisateur dans le but de répondre aux questions qu'il se pose sur le système étudié. Il est donc nécessaire que ce dernier puisse les expliciter, les spécifier et les valider.

Les choix de multi-modélisation *sont les choix faits par le modélisateur qui concernent la sémantique de l'interaction des différents modèle élémentaires constituants le multi-modèle.*

5.4.3 Niveau syntaxique

Le niveau syntaxique concerne les questions propres à la formalisation du multi-modèle. Tout d'abord, chaque modèle élémentaire utilisé dans le multi-modèle est formalisé. Étant donné que ces modèles peuvent provenir de différents domaines scientifiques, il est fort probable que les formalismes utilisés soient différents.

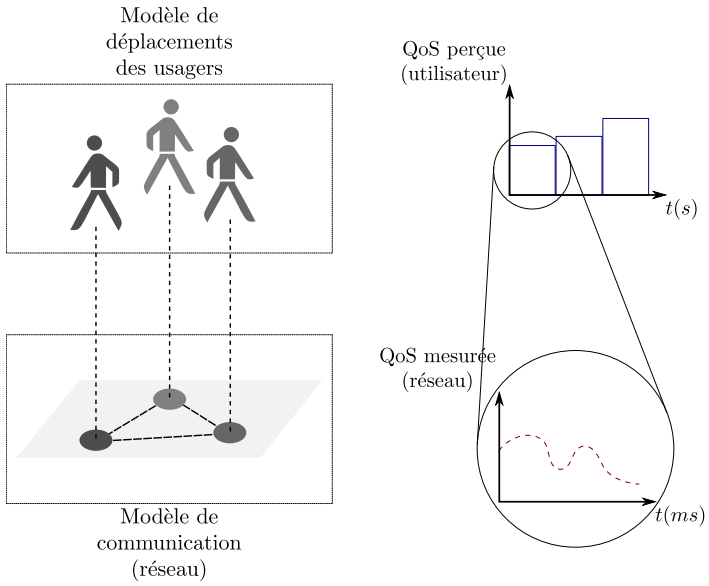


FIGURE 5.3 – Exemple d'interaction de différents modèles. Différence d'échelles entre les différents modèles.

> *On parle de* multi-modélisation homogène *quand tous les modèles élémentaires entrant en jeu possèdent le même formalisme ; et de* multi-modélisation hétérogène *sinon. D'après [Fishwick, 2007].*

La multi-modélisation homogène ne pose généralement pas de soucis à ce niveau. La multi-modélisation hétérogène nécessite de gérer les différences entre les formalismes comme par exemple, les représentations des changements d'états, de l'espace ou encore du temps. Il est important de noter que la formalisation du multi-modèle se doit de prendre en compte à la fois les formalismes de chacun des modèles élémentaires présents et la spécification des choix de multi-modélisation.

5.4.4 Niveau dynamique

Le niveau dynamique concerne les questions propres à l'exécution du multi-modèle. Tout d'abord, on retrouve ici les problématiques classiques du domaine de la simulation distribuée. Il est nécessaire de gérer l'ordre dans lequel les modèles sont exécutés, si la simulation est distribuée ou non (problèmes de respect de la causalité de la simulation) ou encore si les modèles nécessitent des politiques d'exécutions identiques ou différentes (voir chapitre 2).

Ensuite, on retrouve des problématiques classiques du domaine de la simulation multi-agent. Faire interagir des modèles va entraîner la possibilité pour différents modèles de modifier simultanément un même paramètre. Il faut alors gérer la simultanéité des actions et la résolution des conflits potentiels. Il a été démontré, dans la communauté de la simulation multi-agent, que cette gestion se doit d'être spécifiée. En effet, même si cet aspect n'est pas directement lié au niveau sémantique du multi-modèle, les différents choix liés au niveau dynamique vont avoir un impact sur les résultats des simulation [Michel, 2004; Chevrier and Fates, 2010].

5.4.5 Niveau technique

Le niveau technique concerne l'ensemble des questions liées aux aspects implantation de la simulation du multi-modèle. La discussion à ce niveau porte sur la distribution (ou la centralisation) de l'implantation, sur la robustesse, la fiabilité et l'efficacité de la simulation.

5.5 Approches existantes

Dans cette section, nous passons en revue les différentes méthodes de multi-modélisation existantes. Tout d'abord, nous faisons la distinction entre les approches ad hoc qui sont utilisées pour répondre à une problématique particulière et qui n'ont pas pour vocation d'être générique (voir section 5.5.1), et les méthodes dédiées à la multi-modélisation qui se veulent plus génériques. Dans les approches dédiées, nous distinguons l'approche formelle d'intégration de modèles (voir section 5.5.2) et l'approche de couplage de simulateurs (voir section 5.5.3). Dans la section 6.1 nous nous intéressons à l'approche de modélisation et de simulation multi-agent en tant que forme de multi-modélisation.

5.5.1 Approches ad hoc

Les approches de multi-modélisation dites ad hoc sont utilisées pour répondre aux besoins d'une étude spécifique et n'ont pas pour vocation d'être générique. Il est relativement fréquent de rencontrer ce type d'approches car elles présentent une grande souplesse en pratique. Le modélisateur se concentre sur les questions du niveau sémantique qui sont en relation directe avec le but de son étude. Ensuite, les niveaux technique, dynamique et syntaxique sont vus comme autant d'aspects d'ingénierie qui sont faciles à résoudre lorsqu'il s'agit de mettre sur pied une solution logicielle spécifique à l'étude en cours. L'avantage de ces méthodes ad hoc est de mettre sur pied rapidement des outils de multi-modélisation qui répondent exactement aux besoins du modélisateur. Cependant, la spécificité se fait généralement au détriment des spécifications et donc de la possibilité pour la communauté scientifique de comparer et de reproduire les résultats de simulation obtenus par ces méthodes ad hoc. Nous distinguons deux types d'approches ad hoc. La démarche d'intégration opérationnelle de modèles et la démarche de couplage ad hoc de simulateurs.

Intégration opérationnelle de modèles

> *L'*intégration opérationnelle *consiste à coupler ensemble des modèles dans un seul et même logiciel de simulation.*

Dans cette démarche, chaque modèle est conçu dans son propre formalisme. Ensuite, ces modèles sont tous implantés dans un même langage informatique puis intégrés dans un simulateur (voir figure 5.4). Le niveau syntaxique est ici complètement caché puisque les différences entre les formalismes sont gérées via le langage informatique, donc au niveau technique.

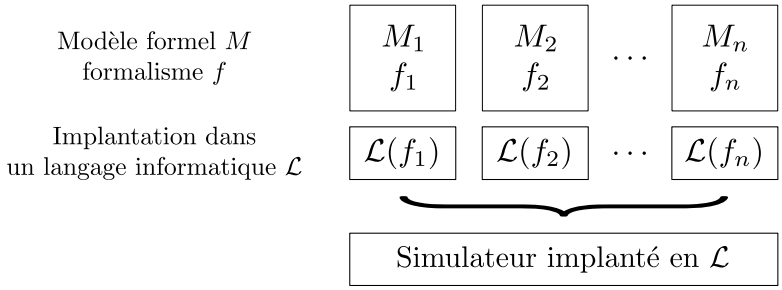


FIGURE 5.4 – Approche d'intégration opérationnelle de modèles. Les modèles sont implantés dans un outil et un langage informatique commun.

Cette démarche est certainement la plus répandue. Dans le domaine des réseaux informatiques, les plates-formes comme NS2, son successeur NS3 [14], GTNetS [Riley, 2003] ou encore Omnet++ [Pongor, 1993] permettent d'utiliser simultanément plusieurs modèles représentant les différents aspects des réseaux. Au niveau conceptuel, le multi-modèle obtenu suit généralement le modèle OSI. Chaque modèle

14. `http://www.nsnam.org/`

représente les services d'une couche réseau particulière et l'interaction entre les modèles suit le schéma classique en couche (voir chapitre 4). Au niveau dynamique, cette approche ne nécessite qu'un seul simulateur. Dans le domaine des réseaux informatiques, par exemple, la simulation est souvent événementielle. Le simulateur peut être centralisé ou distribué selon l'outil utilisé et les problèmes présents à ce niveau sont ceux rencontrés et maîtrisés par la communauté de la simulation distribuée. Au niveau technique, les plates-formes de simulation proposent généralement d'implanter les modèles sous forme de composants légers (*plugin*) et les outils peuvent s'exécuter de manière concurrente pour augmenter les performances en terme de *size-up* et de *speed-up*.

Pour notre part, nous avons nous même appliqué cette première approche pour étudier le phénomène du *free-riding* dans les réseaux pair-à-pair. Sans entrer dans les détails, nous avons utilisé un simulateur existant (PeerFactSim.Kom). Étant donné que cette plate-forme n'intégrait pas de modèle des usagers satisfaisant pour notre étude, nous y avons ajouté un modèle multi-agent du comportement des usagers [Siebert, 2007a]. Au delà des résultats expérimentaux (présentés dans [Julien Siebert et al., 2008, 2009]), ces travaux nous ont permis de débattre de la démarche d'intégration opérationnelle de modèles.

La première limite que nous mettons en avant pour la démarche d'intégration opérationnelle de modèles est l'impossibilité d'utiliser simultanément plusieurs simulateurs existants. À notre niveau, cette démarche ne répond pas à notre contrainte de réutilisation.

Couplage ad hoc de simulateurs

Le deuxième type de méthode ad hoc est le couplage de simulateur. Par rapport à la précédente démarche, cette approche utilise plusieurs simulateurs qui sont couplés ensemble pour les besoins d'une étude. Dans le domaine des réseaux informatiques, la démarche de co-simulation est courante au niveau des couches basses du réseau (à la croisée des domaines de l'électronique et des réseaux informatiques). Le couplage de simulateurs est également présent dans le domaine des réseaux ambiants. Par exemple, la plate-forme iTetris utilise simultanément le simulateur de réseaux informatiques NS3 avec le simulateur de mobilité urbaine SUMO afin de simuler les réseaux véhiculaires ad hoc (VANET) [Kumar et al., 2010]. Dans le domaine des réseaux ambiants, les plates-formes Ubiwise [Disz et al., 1997; Barton and Vijayaraghavan, 2003] et Tatus [O'Neil, 2004] couplent chacune un simulateur de réseaux ambiants avec un simulateur 3D de l'environnement (un moteur graphique de jeux vidéo). Le but est de pouvoir tester en simulation une application ambiante. L'utilisateur de la plate-forme peut se déplacer dans l'environnement 3D virtuel et utiliser le réseau ambiant simulé. Cela génère différents scénarios d'usage à partir desquels, il est possible d'évaluer l'application ambiante.

Le couplage ad hoc de simulateurs répond plus à nos contraintes que l'intégration opérationnelle à la fois en terme de réutilisation et d'interopérabilité. Cependant, au niveau des contraintes de modularité et de passage à l'échelle de type *level-up*, nous atteignons les limites des approches ad hoc. En effet, celles-ci ont pour but de répondre aux besoins spécifiques d'une étude donnée et n'ont pas pour vocation d'être génériques. Cette généricité peut être atteinte mais seulement au niveau technique.

Synthèse et critique des approches ad hoc

Les solutions ad hoc présentent l'avantage de répondre rapidement aux besoins d'une étude donnée. Il est possible, via cette démarche, de se focaliser sur un cas d'étude particulier et de proposer une plate-forme expérimentale dédiée. Pour notre part, nos premiers travaux expérimentaux portaient sur le phénomène de *free-riding* dans les réseaux pair-à-pair [15]. Cependant, nous avons délibérément choisi de prendre du recul par rapport à ce cas d'étude.

D'une part, la problématique de l'évaluation des phénomènes d'inter-influences dans les réseaux ambiants n'est pas spécifique à une seule technologie. D'autre part, les questions que nous avons soulevées à la section 5.4 ne sont pas spécifiques aux réseaux ambiants et peuvent s'appliquer d'une manière générale à la multi-modélisation des systèmes complexes.

15. Le phénomène du *free-riding* n'est pas l'objet de cette thèse. Cependant, le lecteur intéressé par les expériences que nous avons menées dans ce domaine pourra se référer à [Julien Siebert et al., 2008; Navarrete Gutierrez et al., 2010].

Les approches ad hoc de couplage de modèles ne nous satisfont pas dans la mesure où l'ensemble des questions posées aux niveaux conceptuels, sémantiques et syntaxiques se gèrent uniquement au niveau technique. Cela leur fait perdre leur généricité. Au niveau conceptuel peu de choses sont définies dans les approches ad hoc. En général, cela dépend du système à modéliser. Par exemple, dans le domaine des réseaux informatiques, le modèle OSI représente le fonctionnement des réseaux au niveau conceptuel. Cependant celui-ci n'a pas pour vocation de représenter les comportements des usagers.

D'une manière générale, les approches ad hoc ne définissent pas l'ensemble des choix de multi-modélisation, comme par exemple les changements d'échelles. De même, le nombre de niveaux d'abstraction et les liens entre eux sont souvent figés. Les approches ad hoc sont actuellement relativement contraignantes lorsque l'on cherche à coupler des modèles qui représentent des niveaux d'abstraction et des échelles différents, notamment quand cette différence atteint ou dépasse un ordre de grandeur.

Dans ce travail de thèse, nous avons fait le choix de nous intéresser aux approches génériques de multi-modélisation. Nous pensons que ce travail contribue au domaine des réseaux ambiants, dans la mesure où ce domaine fait face à des phénomènes d'inter-influences qui nécessitent de tenir compte simultanément de plusieurs niveaux d'abstraction dans leur évaluation. Nous pensons également que ce travail contribue au domaine des systèmes complexes dans la mesure où les réseaux ambiants en sont un cas particulier qui présente des contraintes intéressantes que ce soit en terme de modularité, de réutilisation, d'interopérabilité et de passage à l'échelle.

5.5.2 Intégration formelle de modèles

La première approche générique de multi-modélisation que nous présentons ici est la démarche d'intégration formelle de modèles. Nous appelons intégration formelle de modèles l'approche qui consiste à coupler les différents modèles du multi-modèle en utilisant un formalisme plus générique que ceux des modèles élémentaires (voir figure 5.5).

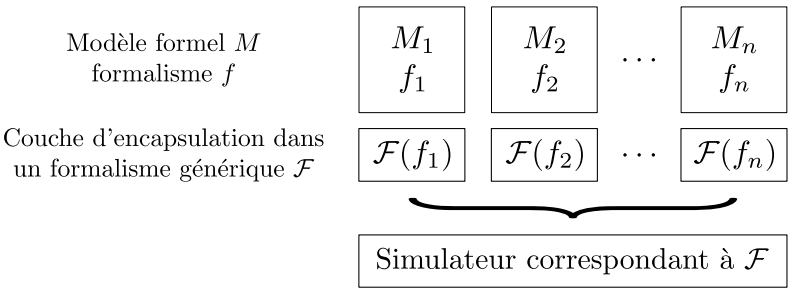


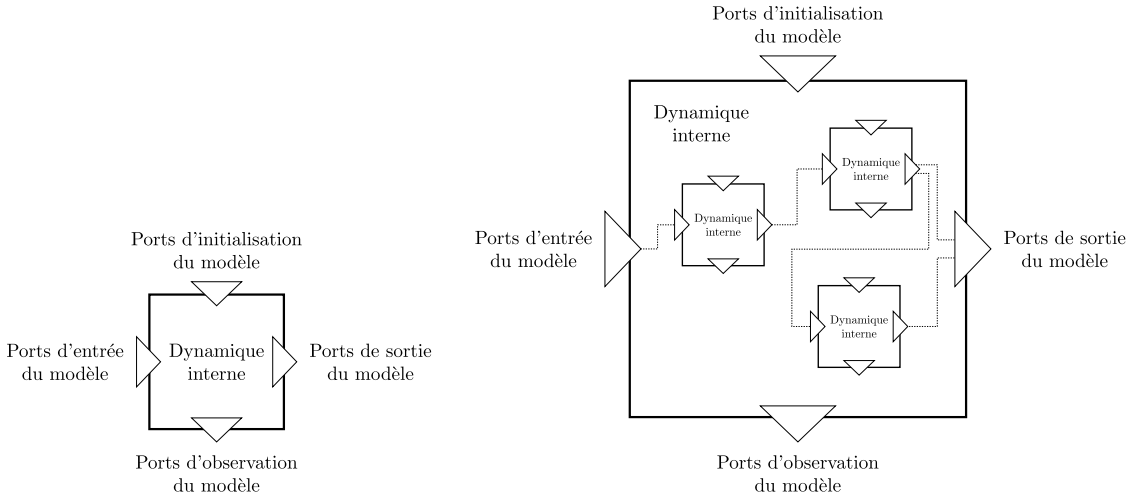
FIGURE 5.5 – Approche d'intégration formelle de modèles. Les modèles sont couplés ensemble via un formalisme commun.

Cette approche peut prendre deux formes. La première consiste à réécrire complètement les modèles que l'on souhaite coupler dans un formalisme plus générique. La deuxième consiste à construire une interface pour passer du formalisme particulier de chaque modèle au formalisme plus générique. C'est ce qu'on appelle l'encapsulation de formalismes [Ramat, 2006]. Cette approche nécessite également l'existence d'un formalisme assez général pour pouvoir englober tous les autres formalismes . C'est actuellement le cas du formalisme DEVS (*discrete event system specification*) et de ses déclinaisons. Les travaux théoriques actuels montrent que des modèles déjà formalisés peuvent être réécrits dans le formalisme DEVS sans perdre de leur pouvoir explicatif [Zeigler et al., 2000; Ramat, 2006].

Sans aller dans les détails du formalisme on peut présenter un couplage de modèles spécifié en DEVS de la manière suivante. Tout d'abord, un modèle DEVS est une boîte qui possède des ports d'entrée et de sortie, une dynamique interne et des ports d'observation (voir figure 5.6a) [16]. Le couplage de modèle est

16. Plus formellement, un modèle DEVS est représenté par un tuple $(X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$ dans lequel X et Y représentent les ports d'entrée et de sortie du modèle, S représente l'ensemble des états du modèle, ta définit la fonc-

composé de modèles DEVS couplés ensemble. Le couplage de modèle est lui-même un modèle DEVS. Il est appelé modèle couplé dans [Zeigler et al., 2000]. C'est lui aussi une boîte qui possède des ports d'entrée et de sortie, une dynamique interne et des ports d'observation (voir figure 5.6b). Le lecteur intéressé par les détails de l'intégration formelle de modèles peut se référer aux travaux menés autour des plates-formes VLE (*Virtual Laboratory Environment*) [Quesnel et al., 2009a] et JAMES II (*Java-based Agent Modeling Environment for Simulation*) [Himmelspach and Uhrmacher, 2007; Himmelspach et al., 2008].



(a) Modèle DEVS élémentaire (b) Multi-modèle DEVS (modèle couplé)

FIGURE 5.6 – Intégration formelle de modèles : modèles couplés DEVS

Critique de l'approche d'intégration formelle de modèles

L'avantage d'une telle approche se situe au niveau syntaxique car la formalisation du multi-modèle est forte. En effet, les modèles, les liens entre les modèles sont tous spécifiés. Cela permet une bonne reproduction des résultats de simulation donc une meilleure validation ou invalidation de ces derniers. De plus, au niveau dynamique, de nombreux algorithmes existent pour simuler de manière distribuée les modèles basés sur le formalisme DEVS. Ainsi cette démarche d'encapsulation des modèles dans DEVS présente aussi l'avantage de faciliter la mise en place d'un simulateur et la gestion des contraintes liées à la simulation.

La première limite de cette approche vient de notre contrainte de réutilisation. En effet, il ne paraît pas possible de réutiliser facilement des outils de modélisation et de simulation existants au travers de cette approche. En effet, pour réutiliser un modèle existant, il est nécessaire de soit l'encapsuler en DEVS, soit de le réécrire. La deuxième limite provient du fait que cette approche n'a pas pour vocation de répondre aux questions situées aux niveaux conceptuel et sémantique. Même s'il est toujours possible d'expliciter les choix de multi-modélisation, de définir les différents niveaux d'abstraction via des modèles DEVS, les solutions proposés dans cette approche sont du ressort des niveaux syntaxique, dynamique ou technique.

5.5.3 Couplage de simulateurs

Le pendant de l'approche d'intégration de modèles est l'approche de couplage de simulateurs. Celle-ci est aussi appelée fédérations de simulateurs, multi-simulation ou encore co-simulation. Il est à noter que cette dernière appellation réfère originellement au domaine de l'électronique et de la simulation

tion d'avancement du temps de simulation, δ_{int} définit comment les états interne du modèle changent au cours du temps et δ_{ext} comment les évènements externes changent les états internes du modèle. Pour plus de détails sur ce formalisme et ses variations, le lecteur intéressé peut se référer à l'ouvrage de référence [Zeigler et al., 2000] et aux travaux menés dans ce domaine [Ramat, 2006; Himmelspach and Uhrmacher, 2007; Himmelspach et al., 2008; Quesnel et al., 2009a].

simultanée du *hardware* et du *software*. Ce sont là des techniques largement utilisées dans des domaines comme l'automobile, l'armée et l'électronique [Luzeaux and Ruault, 2008; Cantot et al., 2009].

Couplage de simulateurs et interaction de modèles

Cette approche de multi-modélisation consiste à faire interagir les modèles en couplant leurs simulateurs. Les modèles ne sont plus intégrés et encapsulés dans un formalisme plus générique mais gardent leurs formalismes respectifs et interagissent ensemble via le couplage de leurs simulateurs (voir figure 5.7). Cette méthode n'opère donc pas comme les précédentes au niveau des modèles, mais au niveau des simulateurs.

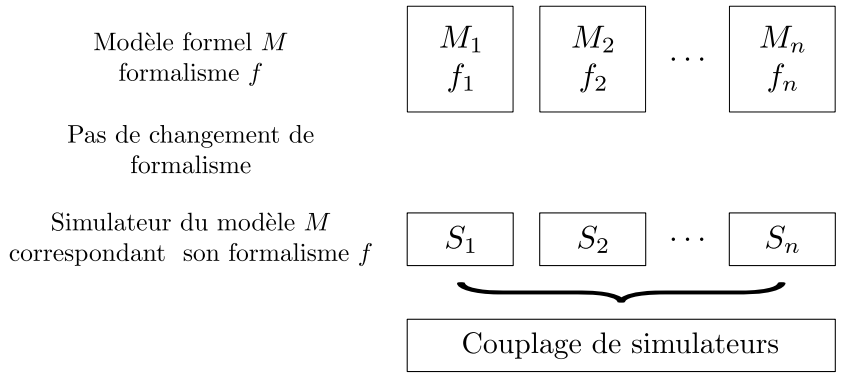


FIGURE 5.7 – Approche de couplage de simulateurs et d'interaction de modèles. Ce ne sont plus les modèles qui sont directement couplés mais les simulateurs via une fédération de simulation.

L'approche de couplage de simulateurs et d'interaction de modèles permet donc de réutiliser et de rendre interopérables les outils de modélisation et de simulation afin de concevoir et de simuler un multi-modèle.

Exemple des fédérations de simulateurs

Le couplage de simulateurs est étroitement lié aux travaux de simulations militaires distribuées et aux travaux de standardisation de l'interopérabilité des simulations distribuées (voir à cet effet, le chapitre 7 dans [Cantot et al., 2009]). Le standard HLA (*High Level Architecture*) [Kuhl et al., 1999] est certainement l'exemple le plus représentatif du domaine du couplage de simulateurs. Le standard HLA propose un cadre conceptuel et une architecture logicielle pour créer une fédération de simulateurs. Dans ce dernier, chaque simulateur est appelé un "fédéré". Une entité appelée RTI (*Run Time Infrastructure*) est en charge de la gestion de la fédération, c'est-à-dire à la fois de la simulation de l'ensemble des fédérés mais aussi des liens entre les fédérés, de la distributions des données, *etc.* L'infrastructure RTI contrôle l'ensemble des simulateurs via un interfaçage spécifique. La manière d'interfacer des simulateurs au RTI est standardisée par la *specification interface*.

Ensuite, pour que l'ensemble des fédérés puissent interopérer, il est nécessaire de décrire et de documenter à la fois les objets représentés par chaque modèle (par exemple, les nœuds et les liens du réseau), les objets partagés par plusieurs modèles (par exemple les positions des usagers qui correspondent aux positions des nœuds du réseau) et les interactions entre les modèles. Ces descriptions et ces documentations sont standardisées par l'*Object Model Template*.

Enfin, l'ensemble de la fédération et son utilisation sont régies pas un ensemble de règles (10 au total), appelées *HLA rules*. On peut citer en guise d'exemple la règle numéro 3 qui stipule que durant l'exécution de la fédération (s'entend l'exécution de l'ensemble du multi-modèle) tout échange de données entre les simulateurs doit se faire au travers de la RTI [HLA Working Group and Symington, 2000]. HLA est défini en détail par le standard IEEE 1516 et est fourni avec un ensemble de méthodologies de développement.

HLA est avant tout un standard et il n'existe pas une unique plate-forme de couplage de simulateurs HLA. Plusieurs implantations de RTI sont disponibles et le choix d'utiliser telle ou telle implantation revient au modélisateur. L'avantage de HLA et des solutions respectant ce standard est de mettre en lumière l'ensemble des problèmes liés à l'interopérabilité des simulations distribuées et de proposer un ensemble de solutions opérationnelles.

Critique de l'approche de couplage de simulateurs

L'avantage de cette approche est de permettre la réutilisation et l'interopérabilité des simulateurs. Les réponses apportées se situent majoritairement aux niveaux technique et dynamique. La présence de standards et de méthodologies unanimement reconnus confère à ses solutions une crédibilité scientifique forte.

La limite actuelle de cette approche se situe aux niveaux sémantique et conceptuel [Fowler and Rose, 2004; Straßburger et al., 2002; Cantot et al., 2009]. Par exemple, les notions de niveaux d'abstraction ou de systèmes ouverts ne sont pas explicitées. Les choix de multi-modélisation apparaissent mais seulement au niveau technique.

5.6 Synthèse

5.6.1 Résumé du chapitre

Dans ce chapitre, nous avons montré que la multi-modélisation apporte de nouvelles questions par rapport à l'approche de modélisation et de simulation "classique". Nous proposons de classer cet ensemble de questions en différents niveaux (voir la figure 5.8).

Niveau	**Domaine concerné**
Conceptuel	Représentation du multi-modèle (méta-modélisation)
Sémantique	Signifiaction et rôle du multi-modèle (face au système étudié)
Syntaxique	Formalisation du multi-modèle
Dynamique	Exécution du multi-modèle (simulation)
Technique	Implantation, distribution, fiabilité

FIGURE 5.8 – Résumé des différents niveaux de problèmes rencontrés en multi-modélisation

Nous avons également énoncé les contraintes propres à notre cas d'étude. Celles-ci concerne la réutilisation et l'interopérabilité de modèles et de simulateurs existants ; la modularité (le fait de pouvoir facilement changer un modèle ou un simulateur par un autre) et le passage à l'échelle. Nous avons également introduit une notion nouvelle qui est le passage à l'échelle de type *level-up*. C'est la possibilité de prendre en compte plusieurs niveaux d'abstraction et de pouvoir facilement gérer leur nombre. Nous avons ensuite passé en revue les solutions existantes dans le domaine de la multi-modélisation.

5.6.2 Limite des approches existantes de multi-modélisation

Les solutions que nous venons de passer en revue apportent des réponses aux niveaux technique, syntaxique et dynamique. Les approches génériques présentent en plus l'avantage de la formalisation et de la standardisation (voir figure 5.9).

Actuellement, nous pensons que le problème n'est pas de faire des simulations distribuées, de les répartir sur différentes machines, d'optimiser les algorithmes de simulation ou de gérer les problèmes d'interopérabilité aussi bien aux niveaux technique, dynamique que syntaxique. Si tel avait été notre choix, nous aurions pu nous focaliser sur un cas d'étude particulier, utiliser ces solutions et créer un multi-modèle pour étudier les phénomènes d'inter-influences. Ce travail de thèse aurait pris une tournure beaucoup plus expérimentale et beaucoup plus spécifique à ce cas d'étude.

Niveaux	**Approches ad hoc**		**Approches dédiées**	
	Itg. O	Cp. Sim ad hoc	Itg. F	Féd. Sim
Conceptuel	✗	✗	✗	✗
Sémantique	✗	✗	✗	patrons HLA
Syntaxique	✗	✗	✓	✗
Dynamique	✗	✓	✓	✓
Technique	✓	✓	✓	✓

Contraintes	**Approches ad hoc**		**Approches dédiées**	
	Itg. O	Cp. Sim ad hoc	Itg. F	Féd. Sim
Réutilisation	✗	✗	✗	✓
Modularité	✗	✗	modèles DEVS	✓
Level-up	✗	✗	✗	✗

Légende :	
Itg .O	Intégration Opérationnelle de modèles
Cp. Sim ad hoc	Couplage ad hoc de simulateurs
Itg.F	Intégration Formelle de modèles
Féd. Sim	Fédération (couplage) de simulateurs
✓	Solutions satisfaisantes ou contrainte respectée
✗	Solutions non satisfaisantes ou contrainte non respectée
texte	Solution ou contrainte partiellement satisfaisante

FIGURE 5.9 – Résumé des solutions apportées par les approches de multi-modélisation existantes

Cela n'a pas été notre choix. Nous avons délibérément choisi de nous intéresser aux niveaux sémantique et conceptuel. La problématique sur laquelle nous nous positionnons ne concerne pas ce que nous pouvons faire techniquement en multi-modélisation mais ce que nous pouvons représenter (niveaux d'abstraction, échelles, *etc.*) et comment nous pouvons le représenter (méta-modèle). Pour répondre à cette problématique, nous considérons la modélisation multi-agent comme une voie à explorer. Pour nous, la modélisation multi-agent est en soi une forme de multi-modélisation. C'est, de plus, la forme de multi-modélisation la plus adaptée pour représenter les systèmes complexes.

Nous cherchons donc dans les travaux de la communauté multi-agent, des solutions aux niveaux conceptuel et sémantique. Dans le chapitre suivant, nous passons en revue les différentes plates-formes multi-agents existantes et cherchons une solution conceptuelle qui nous permette de concevoir un multi-modèle à partir de modèles existants, sachant que chaque modèle va représenter un niveau d'abstraction particulier, qu'il nous faut pouvoir modéliser les interactions entre ces différents niveaux d'abstraction et qu'il nous faut respecter l'ensemble des contraintes énoncées en section 5.3.

Chapitre 6

Paradigme multi-agent et multi-modélisation

Sommaire

6.1 Introduction

Nous avons montré, dans le chapitre précédent, que la multi-modélisation pose de nombreuses questions à différents niveaux. Les outils issus du domaine de la multi-modélisation et de la simulation distribuée apportent des réponses aux niveaux technique, dynamique et syntaxique. Ils présentent des limites aux niveaux conceptuels et sémantique lorsqu'il s'agit de représenter des systèmes complexes qui nécessitent de prendre en compte simultanément plusieurs niveaux d'abstraction.

Dans le chapitre 3 nous avons montré que le paradigme de modélisation multi-agent permet de représenter un système complexe. Aussi, le paradigme de modélisation multi-agent ne présuppose rien quant à la dynamique interne d'un agent ou d'un environnement. D'une manière générale, les agents et les environnements sont des entités qui englobent des modèles particuliers. On peut dès lors affirmer qu'un modèle multi-agent est une forme de multi-modèle.

Un modèle multi-agent est en soi un multi-modèle d'un système complexe.

6.2 Intérêt de l'approche multi-agent pour la multi-modélisation

La définition précédente peut sembler naïve mais elle a l'avantage d'expliciter clairement la nature de la modélisation multi-agent. Au niveau conceptuel, un modèle multi-agent est une représentation d'un multi-modèle. C'est la représentation la plus adaptée pour modéliser les systèmes complexes. Il est donc nécessaire de faire ressortir les choix liés à la multi-modélisation dès ce niveau. Rappelons que c'est le méta-modèle multi-agent qui guide la conception du modèle multi-agent et qui définit les rôles et les relations de chaque entité. Ainsi, pour être adapté à la multi-modélisation, le méta-modèle multi-agent se doit de forcer la spécification à la fois des modèles d'agents et d'environnements utilisés mais aussi des mécanismes d'interactions propres à la dynamique du système multi-agent.

L'utilisation du paradigme multi-agent peut apporter des solutions au niveau sémantique. Cette approche peut être vue comme une manière de distribuer la connaissance du modélisateur [Drogoul et al., 2002]. On peut dès lors créer un système multi-agent dont le rôle ne va être d'aider le modélisateur au niveau sémantique. Par exemple, les auteurs de [Servat et al., 1998; Drogoul et al., 2002] proposent d'utiliser le paradigme multi-agent afin d'aider à l'interprétation des résultats de simulation. L'approche multi-agent peut aussi servir au niveau de la calibration du multi-modèle. En effet, concevoir un multi-modèle d'un système complexe engendre souvent un nombre de paramètres très élevé. Les auteurs de [Drogoul et al., 2002; Pereira and Reis, 2004a,b] proposent d'utiliser des agents de calibration pour changer automatiquement les paramètres des modèles afin que les résultats de simulation obtenus soient proches des observations menées sur le système réel étudié.

Aux niveaux dynamique et technique, le paradigme multi-agent peut aussi être utilisé pour gérer les problèmes de distribution de la simulation, d'optimisation des performances [Sebastien, 2009] ou de fiabilité de la simulation quand celle-ci est distribuée [Drogoul et al., 2002]. De même, l'architecture logicielle agent permet une bonne modularité [Gleizes, 2008].

Nous voyons ici, que le paradigme multi-agent présente de nombreux avantages pour la multi-modélisation des systèmes complexes. Dans les sections suivantes, nous passons en revue les outils de la communauté multi-agent. Pour notre part, nous cherchons surtout à mettre en avant des solutions aux niveaux conceptuel et sémantique. Nous cherchons également à respecter les contraintes que nous nous sommes fixés en section 5.3. Nous distinguons les plates-formes de programmation multi-agent pour lesquelles l'objectif principal n'est pas la simulation multi-agent en tant que tel mais la conception d'applications multi-agents (voir section 6.3). Ensuite viennent les plates-formes dédiées à la modélisation multi-agent (voir section 6.4). Enfin, nous nous intéressons plus spécifiquement aux approches multi-agent pour la multi-modélisation (voir section 6.5).

6.3 Plates-formes de programmation multi-agent

Les plates-formes de programmation multi-agent sont des outils qui permettent de concevoir des applications multi-agent. D'un point de vue conceptuel, l'application à développer est décrite comme un système multi-agent. Le rôle de la plate-forme de programmation est de mettre à disposition du développeur un ensemble de solutions techniques pour concevoir son application multi-agent. Le but premier de ces plates-formes n'est pas de cibler la modélisation multi-agent. Cependant, la simulation multi-agent peut faire partie des applications de ces plates-formes et certaines proposent directement des extensions dédiées à la simulation.

6.3.1 JADE

JADE (*Java Agent DEvelopment Framework*[17]) est une plate-forme de programmation multi-agent totalement implantée en Java. Son rôle est de simplifier l'implantation d'applications multi-agents grâce aux technologies de type *middleware*, à la spécification FIPA (the Foundation for Intelligent Physical Agents[18]) et à un ensemble d'outils de développement logiciel [Bellifemine et al., 2003]. Le rôle de JADE est d'apporter un ensemble de solutions techniques. Au niveau conceptuel, l'application est vue comme un ensemble d'agents et d'un environnement. Les niveaux sémantique, syntaxique et dynamique sont quant à eux inexistants car cette plate-forme n'est pas dédiée à la simulation multi-agent et encore moins à la multi-modélisation.

6.3.2 Madkit

MadKit[19] est une plate-forme de programmation multi-agent développée en Java et pensée pour être modulaire et passer à l'échelle [Gutknecht et al., 2000]. Contrairement à JADE, MadKit propose une extension dédiée à la simulation multi-agent : TurtleKit[20] [Michel et al., 2005].

Au niveau conceptuel la plate-forme MadKit se base sur un méta-modèle appelé AGRE (Pour agent groupes rôles et environnement) [Stratulat et al., 2009] qui permet de représenter des organisations d'agents. Une séparation est également faite entre le corps des agents (la partie immergée dans l'environnement : capteurs et effecteurs) et l'esprit de l'agent (la partie décisionnelle). Un agent peut ainsi être présent dans plusieurs environnements simultanément. La plate-forme suit aussi le modèle influence-réaction initialement proposé dans [Ferber and Müller, 1996] puis adapté pour la simulation dans [Michel, 2007]. Ce modèle spécifie que les agents ne peuvent modifier directement l'environnement et que la résolution des conflits doit être effectuée au niveau de l'environnement.

À notre connaissance, la plate-forme MadKit ne cible pas la multi-modélisation. Les notions d'échelle, de niveaux d'abstraction et les choix de multi-modélisation ne sont pas présents. Au niveau dynamique, le modèle influence-réaction permet de spécifier comment se passe la résolution des conflits (actions simultanées) et la simulation peut se faire de manière distribuée via un ordonnanceur global (centralisé). Au niveau syntaxique, l'extension TurtleKit s'appuie sur un langage de modélisation multi-agent de type Logo. Il est également possible de passer par une approche d'intégration opérationnelle de couplage de modèles.

6.3.3 Janus

Comme MadKit, Janus est une plate-forme de programmation multi-agent qui propose des extensions dédiées à la simulation : Jaak[21] et JaSIM[22].

Contrairement à MadKit, la plate-forme Janus a, entre autre, été pensée pour la simulation multi-niveau. Le but est de pouvoir tenir compte de plusieurs niveaux d'abstraction simultanément dans la modélisation [Gaud, 2007]. Au niveau conceptuel, la plate-forme Janus s'appuie sur le méta-modèle CRIO (*Capacity, Role, Interaction and Organization*). Ce dernier se base sur la notion de système multi-agent holonique. Un système multi-agent holonique est un système multi-agent dans lequel les agents peuvent aussi être des systèmes multi-agents. Le système ainsi obtenu forme ce qu'on appelle une holarchie. Il est ainsi possible de faire cohabiter dans le même modèle des niveaux d'abstraction différents (simulation multi-niveau).

À notre connaissance, les travaux concernant la plate-forme Janus et son utilisation autour de la simulation multi-niveau ne prennent pas en compte les problématiques liées aux choix de multi-modélisation. Au niveau conceptuel, ceux-ci ne sont pas explicités. Au niveau sémantique, il est fait une séparation entre l'ensemble des agents et l'environnement qui sont chacun représentés par deux modèles holoniques séparés. Cela présente l'avantage de bien séparer les deux modèles. Au niveau syntaxique, Janus propose,

17. `http://jade.tilab.com/`
18. `http://www.fipa.org/`
19. `http://www.madkit.org/`
20. `http://www.lirmm.fr/~fmichel/TurtleKit/`
21. Jaak : `http://www.janus-project.org/Jaak`
22. JaSIM : `http://www.multiagent.fr/Jasim_Platform`

comme Madkit, soit d'utiliser un langage de modélisation multi-agent de type logo soit de passer par une démarche d'intégration opérationnelle de modèles.

La plate-forme Janus offre des solutions spécifiques aux niveaux dynamique et technique. Au niveau dynamique, la politique d'exécution du modèle est adaptable. Cela permet notamment d'optimiser la simulation temps réel de systèmes complexes [Gaud et al., 2008]. Au niveau technique, la plate-forme Janus se repose sur une méthodologie d'ingénierie logicielle agent appelée ASPECTS (*Agent-Oriented Software Process for Engineering Complex Systems*). Elle fournit également un ensemble complet de fonctionnalités pour développer, exécuter, afficher et contrôler un système multi-agents. De plus les applications Janus peuvent être distribuées sur un réseau (JXTA) [23].

6.3.4 Limite des plates-formes de programmation multi-agent

Les plates-formes de programmation multi-agent proposent un ensemble de solutions techniques pour implanter des applications multi-agents. Bien que celles-ci ne soient originellement pas dédiées à la simulation multi-agent, des extensions propres à la simulation multi-agent ont vu le jour. Celles-ci permettent de tirer partie des solutions techniques apportées par de telles plates-formes de programmation. Cependant, ces extensions ne sont pas spécifiquement orientées multi-modélisation. Ainsi, ces dernières présentent des limites au niveau conceptuel notamment sur la spécification des choix de multi-modélisation. L'autre limite que nous retenons est que ces approches ne ciblent pas du tout la réutilisation et le couplage de simulateurs.

6.4 Plates-formes dédiées à la simulation multi-agent

Les plates-formes dédiées à la simulation multi-agent ont pour but de proposer au modélisateur des solutions aux niveaux technique, dynamique et conceptuel. Les agents et le (ou les) environnement(s) sont autant de cases vides dans lesquelles le modélisateur peut venir implanter ses modèles. Ces plates-formes se basent sur une démarche d'intégration opérationnelle de modèles et n'ont pas pour vocation de réutiliser des modèles et des simulateurs existants. Nous passons en revue ces outils pour voir les solutions conceptuelles qui sont utiles et mises en place par la communauté multi-agent. Pour une revue des aspects technique et dynamique, le lecteur intéressé peut se référer à [Railsback et al., 2006] et au premier chapitre de [Sebastien, 2009].

6.4.1 Swarm

Swarm [24] est une plate-forme de simulation multi-agent créée en 1996 et développée par le Santa Fe Institute [Minar et al., 1996]. Au niveau conceptuel, Swarm utilise la notion de *swarm* équivalente à la notion de holon présentée précédemment. Au niveau dynamique, chaque *swarm* possède sa propre horloge et l'ensemble de la simulation est géré par un ordonnanceur global. Au niveau technique, l'ensemble des outils de visualisation et d'analyse de la simulation ainsi que le modèle et son ordonnanceur forment également un *swarm*.

6.4.2 Gama

Gama est une plate-forme de simulation multi-agent spécifique à la modélisation d'agents situés [Amouroux et al., 2009]. Cette plate-forme intègre un certain nombre d'outils pour décrire et manipuler l'environnement. Au niveau conceptuel, tout est considéré comme un agent que ce soient l'environnement, les objets de l'environnement, les agents eux même ou les groupes d'agents. Cette plate-forme est utilisée notamment pour faire de la modélisation multi-échelle. Plusieurs niveaux d'abstraction peuvent être décrits et chaque niveau d'abstraction représente une échelle particulière (par exemple : un bâtiment, un quartier, une ville) [Amouroux et al., 2009]. La création ou la destruction de modèles à des niveaux

23. `http://www.janus-project.org`
24. `http://www.swarm.org`

d'abstraction différents se fait par la création ou la destruction d'agents correspondants. On retrouve dans Gama les idées présentes dans Janus ou Swarm.

6.4.3 Repast-Symphony

Repast-Symphony est une plate-forme de simulation multi-agent écrite entièrement en Java et développée par le Argonne National Laboratory [North et al., 2007]. Cette plate-forme permet d'implanter des modèles soit via un langage de programmation multi-agent de type Logo (ReLogo) soit par une approche d'intégration opérationnelle de modèles. Au niveau conceptuel, Repast-Symphony propose la notion de contexte en lieu et place de l'environnement. Un contexte peut contenir un ensemble d'agents, un état et une dynamique. À un contexte peuvent être associées une ou plusieurs projections qui sont des représentations de l'espace (espace continu, grille, graphe). Un contexte peut lui-même être décomposé en une hiérarchie de sous-contextes. L'intérêt principal de cette décomposition est d'associer des comportements aux agents en fonction du contexte dans lequel ce dernier se trouve. Ainsi, Repast-Symphony permet de représenter des environnements correspondants à différents niveaux d'abstraction.

6.4.4 Jedi

Jedi est une plate-forme de simulation multi-agent qui, au contraire des précédentes, est centrée sur la notion d'interaction [Kubera et al., 2008]. Au niveau conceptuel, l'interaction, qu'elle soit entre plusieurs agents ou entre un agent et l'environnement, est vue comme un objet à part entière. Cette approche de réification de l'interaction a été initialement développée dans [Desmeulles, 2006] pour représenter les systèmes biologiques par une approche multi-modèle. Cette approche est bien adaptée dans les cas où on souhaite représenter simultanément plusieurs systèmes qui ont certains éléments en commun (on parle alors de couplage structurel) et quand l'organisation et la structure du système modélisé peut être amenée à changer au cours de la simulation (voir [Desmeulles, 2006] chapitre 4).

6.4.5 Limites des plates-formes de simulation multi-agent

Tout d'abord, ces plates-formes de simulation multi-agent ne ciblent pas la réutilisation de modèles et de simulateurs existants. Nous retrouvons ici les limites de l'approche d'intégration opérationnelle ou formelle de modèles déjà présentées dans les sections précédentes. Aux niveaux sémantique et conceptuel, ces approches ne ciblent pas explicitement la multi-modélisation. Ainsi, les problèmes liés aux choix de multi-modélisation sont rarement explicités. Cependant, nous constatons que dans les travaux passés en revue, il existe un besoin fort de représenter différents niveaux d'abstraction. Les travaux de la communauté multi-agent apportent donc des solutions conceptuelles intéressantes pour représenter au sein d'un même modèle multi-agent plusieurs niveaux d'abstraction. Nous proposons de classer les manières de représenter différents niveaux d'abstraction au sein d'un même modèle multi-agent en trois catégories.

- Méta-modèle organisationnel (MadKit, Janus).
- Méta-modèle holonique (Janus, Swarm, Gama).
- Méta-modèle centré sur les interactions (Jedi).

L'utilisation d'un méta-modèle organisationnel force le modélisateur à définir comment s'organise le système étudié, quels sont les groupes, les rôles que peuvent jouer les agents et quelles règles ils doivent respecter. Ce type de méta-modèle est intéressant, par exemple, dans les sciences humaines pour représenter des groupes, des sociétés, *etc.* Le méta-modèle holonique permet, quant à lui, de remplacer un ensemble d'agents par un agent de niveau supérieur. L'ensemble des niveaux d'abstraction sont représentés par une hiérarchie (une holarchie). On parle alors de niveaux d'agrégation. Cette approche est intéressante si on cherche à représenter avec plus ou moins de finesse (granularité) un aspect du système étudié.

Les méta-modèles organisationnel et holonique sont des méta-modèles centrés sur les agents. Ces derniers ne sont pas toujours utilisables en pratique. Dans notre cas par exemple, nous cherchons à représenter à la fois le réseau ambiant et ses usagers. Ce sont là deux niveaux d'abstraction différents qui sont couplés structurellement : les usagers correspondent aux nœuds du réseau (voir figure 6.1). Dans ce cas, le méta-modèle centré interaction est intéressant car on ne fait pas a priori d'hypothèse sur l'organisation des agents ou leur éventuelle agrégation à un niveau supérieur. Dans la section suivante,

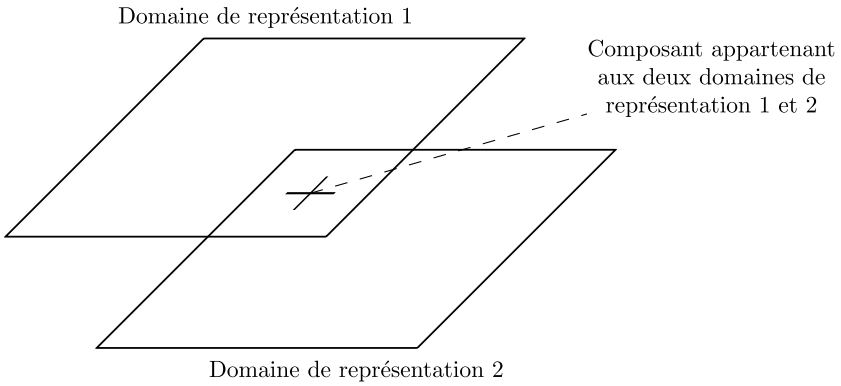


FIGURE 6.1 – Couplage Structurel (Source [Desmeulles, 2006])

nous passons en revue les approches multi-agent spécifiquement dédiées à la multi-modélisation. Nous verrons en détail l'intérêt d'un méta-modèle orienté interaction.

6.5 Approche multi-agent pour la multi-modélisation et le couplage de simulations

Dans cette section, nous présentons deux approches, les travaux menés par l'équipe SMART [25] du LIM/IREMIA [26] autour de la plate-forme GEAMAS-NG et les travaux menés par l'équipe EBV [27] du LISyC [28] autour de la modélisation orientée patterns et du méta-modèle Agents et Artefacts. À notre connaissance, ce sont les seules approches multi-agent qui visent explicitement la multi-modélisation et le couplage de simulations. Toutes les deux utilisent des méta-modèles que l'on peut qualifier d'orientés interaction.

6.5.1 Plate-forme GEAMAS-NG

Description

La plate-forme GEAMAS-NG est une plate-forme de modélisation et de simulation multi-agent qui permet de prendre en compte simultanément plusieurs modèles d'environnements. Celle-ci est utilisée notamment pour modéliser et simuler l'aménagement énergétique du territoire de l'île de la Réunion [Gangat et al., 2009]. Au niveau conceptuel, cette plate-forme s'appuie sur une démarche de modélisation orientée dynamique (*dynamic oriented modeling*). Le but de cette démarche est de modéliser les agents en identifiant leurs interactions. Une dynamique est définie comme un ensemble d'interactions qui participent à la spécification d'une caractéristique majeure du phénomène étudié [Conruyt et al., 2009]. Un agent peut tout à fait participer à plusieurs dynamiques. Dès lors le modèle multi-agent peut devenir très complexe et donc difficile à concevoir, implanter et simuler. La démarche orientée dynamique à pour but d'éviter cette complexité et donc de faciliter la conception, l'implantation, *etc.*

Chaque agent possède un ensemble d'états et de comportements. Les comportements peuvent modifier les états de l'agent et les états peuvent influencer les comportements. Ces deux ensembles sont ensuite décomposés selon les dynamiques auxquelles participe l'agent (voir figure 6.2). Une fois ces distinctions faites, chaque dynamique est représentée par un modèle particulier. Un agent qui participe à plusieurs dynamiques voit ses états et ses comportements répartis dans les dynamiques (voir figure 6.3). Enfin, il est

25. Systèmes Multi-Agents et Réseaux de Télécommunications
26. Laboratoire d'Informatique et de Mathématiques / Institut de Recherche en Mathématiques et Informatique Appliquées
27. Ecosystémique et Biologie Virtuelle
28. Laboratoire d'Informatique des Systèmes Complexes

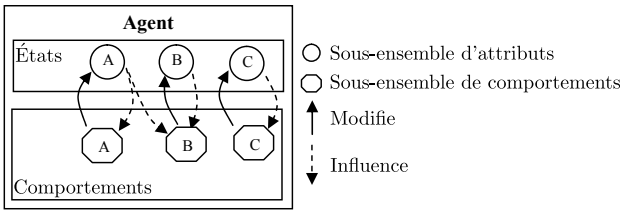


FIGURE 6.2 – Modélisation d'un agent selon les différentes dynamiques (A, B et C) auxquelles il participe (Source [Conruyt et al., 2009]).

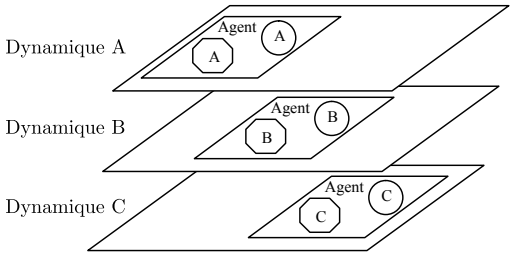


FIGURE 6.3 – Répartition des états et comportements d'un agent dans les différentes dynamiques (Source [Conruyt et al., 2009]).

nécessaire d'ajouter un environnement. Celui-ci est vu comme l'élément de couplage entre les différentes dynamiques (voir figure 6.4).

Avantages et limites de la plate-forme GEAMAS-NG

Au niveau conceptuel, cette approche permet de représenter des systèmes qui présentent des niveaux d'abstraction couplés structurellement. Les choix de multi-modélisation sont gérés par l'environnement. Cependant, à notre connaissance, il n'est fait aucun état des choix de multi-modélisation dans les travaux passés en revue. De même, il n'est pas évident de comprendre comment cette plate-forme gère l'hétérogénéité des modèles de dynamiques. Que se passe-t-il quand une dynamique est modélisée par un modèle à équations différentielles et une autre par un modèle multi-agent ? par exemple. Il semble que ces aspects doivent être implicitement gérés par le modélisateur. Au niveau syntaxique, la démarche retenue est l'intégration opérationnelle de modèles. Aux niveaux dynamique et technique, la plate-forme se base sur un système multi-agent de distribution (voir figure 6.5). Cette démarche présente l'avantage de pouvoir réutiliser des modèles et des noyaux de simulation tout en permettant l'utilisation de ceux-ci de manière *standalone* ou couplée.

De manière synthétique, le système multi-agent de distribution a pour rôle de garantir le respect de la contrainte de causalité (appelé qualité de la simulation dans [Sebastien, 2009]), de permettre la flexibilité

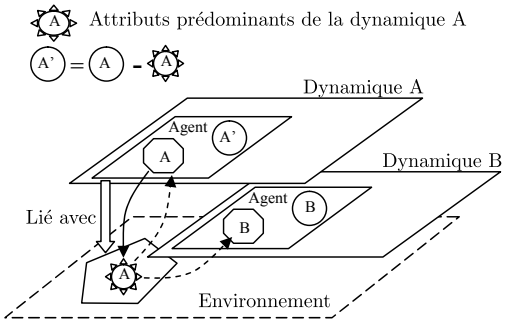


FIGURE 6.4 – Couplage des différentes dynamiques via l'environnement (Source [Conruyt et al., 2009]).

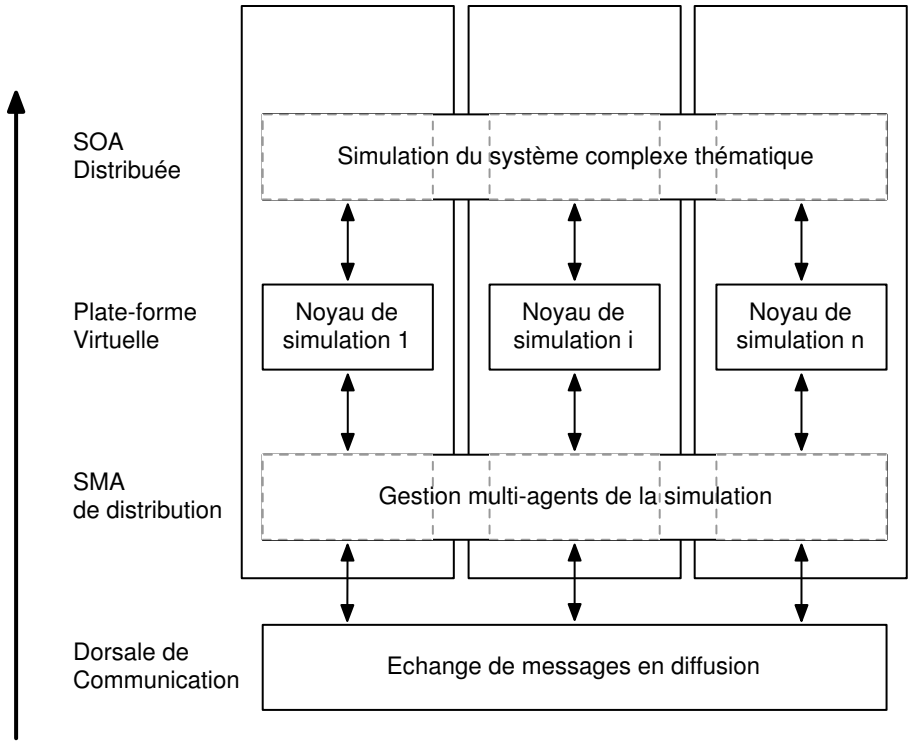


FIGURE 6.5 – Architecture multi-agent de couplage de simulation. Source [Sebastien, 2009]. Plusieurs plates-formes de simulation sont couplées ensemble via un système multi-agent de distribution.

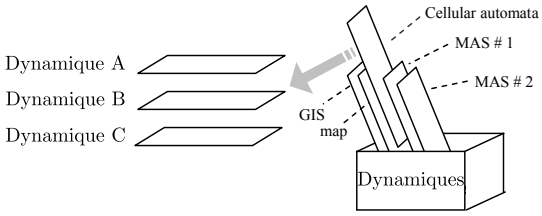


FIGURE 6.6 – Modularité de l'approche orientée dynamique (Source [Conruyt et al., 2009]).

de la simulation (ajout, suppression ou échange d'un modèle ou d'un simulateur de manière aisée) et d'optimiser le déroulement de la simulation.

Par rapport à nos contraintes, la modélisation orientée dynamique et l'utilisation du système multi-agent de distribution permettent une grande modularité dans le couplage de modèles et un passage à l'échelle de type *level-up* relativement bon (voir figure 6.6). Cependant, la plate-forme GEAMAS-NG n'a pas comme but la réutilisation de simulateurs existants. Même s'il est possible de coupler plusieurs noyaux de simulation via le système multi-agent de distribution, il ne nous semble pas que ces noyaux soient des simulateurs existants, ni même que la plate-forme GEAMAS-NG ait été conçue dans cette optique. GEAMAS-NG ne satisfait donc pas totalement notre contrainte de réutilisation.

6.5.2 Méta-modèle agents et artefacts pour la multi-modélisation

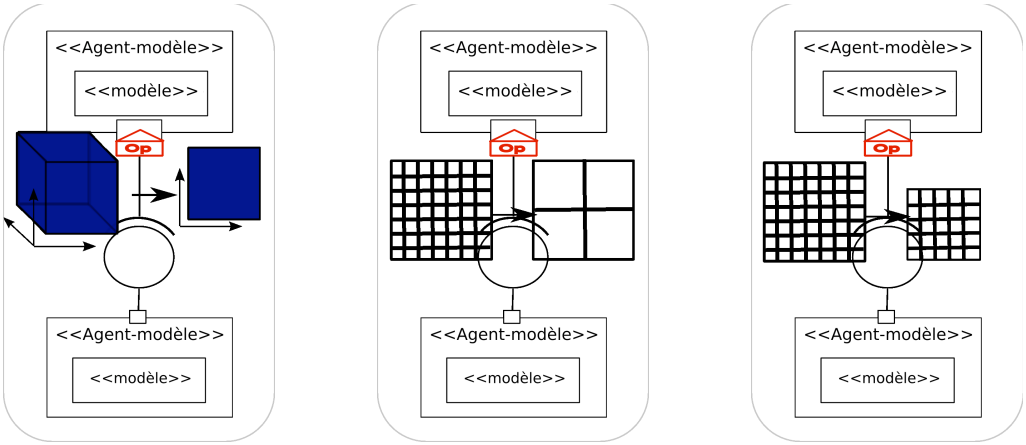
La deuxième approche multi-agent de multi-modélisation que nous passons en revue provient de la thèse de Stéphane Bonneaux [Bonneaud, 2008]. Le cas d'étude de ces travaux est de réaliser la simulation de systèmes de pêcherie en prenant en compte à la fois les aspects écosystémiques (dynamiques des population des poissons) et socio-économiques (pêcheries, marché, quotas). L'approche utilisée est

explicitement une approche multi-modèle et multi-échelle. Au niveau conceptuel, ces travaux se basent sur le méta-modèle Agents et Artefacts (A&A).

Le méta-modèle Agents et Artefact est un méta-modèle multi-agent orienté interactions. Pour représenter un système multi-agent, A&A propose de décomposer l'environnement en un ensemble d'entités passives appelées artefacts [Omicini et al., 2008]. Les artefacts sont les entités de l'environnement que les agents peuvent voir et utiliser. Ils peuvent être vus comme des outils pour les agents.

Application du méta-modèle A&A à la multi-modélisation

Le méta-modèle A&A est utilisé dans [Bonneaud, 2008] pour construire une société de modèles en interaction (un multi-modèle multi-agent). A&A permet de répondre aux questions du niveau conceptuel. Dans ces travaux, le multi-modèle est vu comme un ensemble de modèles élémentaires autonomes en interaction : le multi-modèle est vu comme un système multi-agent. Chaque agent a la charge d'un modèle particulier. Les interactions entre les modèles et les choix liés à la multi-modélisation (changement d'échelle, cohérence, *etc.*) sont explicitement décrits au travers d'artefacts. Chaque lien entre deux modèles est symbolisé par un artefact particulier. Des opérations permettent de spécifier les choix du modélisateur quant aux problèmes de changement d'échelle, de cohérence sémantique ou de compatibilité des formalismes. Dans [Bonneaud, 2008], trois opérations sont présentées : l'opération de projection qui permet de réduire le nombre de dimensions des données (voir figure 6.7a), l'opération de discrétisation qui permet de réduire ou d'augmenter le pas de discrétisation d'une donnée (voir figure 6.7b) et l'opération de réduction qui permet de sélectionner seulement une partie des données (voir figure 6.7c).



(a) Opération de projection (b) Opération de discrétisation (c) Opération de réduction

FIGURE 6.7 – Types d'opérations qu'un agent peut manipuler pour satisfaire aux contraintes de multi-modélisation. Source [Bonneaud, 2008] chapitre 4.

Avantages et limites de ces travaux

L'utilisation du méta-modèle A&A faite dans [Bonneaud, 2008] apporte des réponses aux niveaux conceptuel et sémantique. Le multi-modèle est vu comme une société de modèles en interaction et l'ensembles des choix de multi-modélisation sont spécifiés par des artefacts. Ensuite, A&A permet une grande modularité dans la conception du multi-modèle. Cette vision orientée interaction facilite l'ajout, l'échange et la suppression d'agents dans le système. L'orientation multi-échelle des travaux permet aussi d'atteindre un bon passage à l'échelle de type *level-up*.

La principale limite de ce travail provient du fait qu'il ne cible pas la réutilisation de simulateurs existants. Au niveau syntaxique, l'approche actuellement retenue est celle d'intégration opérationnelle de modèles. Au niveau dynamique, l'ensemble du multi-modèle est exécuté grâce à un ordonnanceur global. Ces travaux ne répondent donc pas à notre contrainte de réutilisation.

6.5.3 Limite de l'approche multi-agent pour la multi-modélisation

L'avantage des approches que nous venons de présenter provient du fait qu'elles ciblent explicitement la multi-modélisation. Ainsi, les méta-modèles utilisés proposent des solutions adaptées aux niveaux conceptuel et sémantique. Nos contraintes de modularité et de passage à l'échelle (de type *level-up*) sont satisfaites.

À notre connaissance, le principal inconvénient des approches existantes provient du fait qu'elles ne ciblent pas l'interopérabilité et la réutilisation de simulateurs déjà existants. Notre contrainte de réutilisation n'est pas satisfaites avec ces approches.

6.6 Synthèse

Dans ce chapitre nous avons passé en revue les approches de modélisation et de simulation multi-agents dans le but de mettre en lumière les solutions adaptées à la multi-modélisation que ce domaine apporte aux niveaux conceptuel et sémantique. Nous avons séparé les approches de programmation multi-agent (Jade, MadKit, Janus), des approches dédiées à la modélisation simulation multi-agent (Swarm, Gama, Repast-Symphony, Jedi) et des approches ciblant explicitement la multi-modélisation.

Notre premier constat porte sur le choix du type de méta-modèle à utiliser. L'approche agent est en soi l'approche idéale pour représenter des systèmes complexes. Dans ce chapitre, nous avons mis en lumière deux grandes catégories de méta-modèles multi-agents : les méta-modèles orientés agents (méta-modèles holonique et organisationnel) et les méta-modèles orientés interaction (A&A, modélisation orientée dynamique, plate-forme Jedi). Pour notre part, il est plus intéressant d'utiliser une méta-modèle orienté interaction. Cette vision permet de représenter des niveaux d'abstraction qui sont couplés structurellement, ce qui est typiquement le cas dans notre cas d'étude où les usagers correspondent aux nœuds du réseaux.

Notre deuxième constat porte sur le fait que relativement peu de travaux considèrent dès le départ qu'un modèle multi-agent est en soi un multi-modèle. En conséquence, les choix de multi-modélisation sont souvent peu exprimés et les problèmes liés aux changements d'échelle ou de niveaux d'abstraction sont résolus de manière implicite au niveau technique lors de l'implantation du multi-modèle. Nous pensons, comme [Bonneaud, 2008] que les choix de multi-modélisation doivent être explicités dès le niveau conceptuel (dans le méta-modèle). Cela permet d'augmenter le degré de spécification du multi-modèle et donc la possibilité de reproduire ce multi-modèle et les résultats de simulation.

Notre troisième constat porte sur le fait que les approches passées en revue ne ciblent pas la réutilisation et l'interopérabilité de simulateurs existants. Les approches multi-agent ne satisfont donc pas notre contrainte de réutilisation. La figure 6.8 résume les solutions apportées par les approches multi-agent pour les différents niveaux de questions et la satisfaction de nos contraintes.

Les solutions du domaine de la multi-modélisation et du domaine multi-agent présentent chacune leurs avantages et leurs limites. Pour satisfaire à nos contraintes, il nous faudrait une approche de multi-modélisation qui mélange à la fois les solutions techniques, dynamiques et syntaxiques proposées par des approches de couplage de simulateurs (comme HLA par exemple) avec les solutions conceptuelles et sémantiques des approches multi-agents dédiées à la multi-modélisation (comme A&A par exemple). Dans le chapitre suivant, nous passons en revue les problématiques qu'introduirait une telle approche.

Plates-formes	Niveaux		
	Conceptuel	**Sémantique**	**Syntaxique**
Jade			
MadKit	✗(Org.)	✗	✗(Itg. Op / Lang.)
Janus	✗(Org. / Hln.)	✗	✗(Itg. Op / Lang.)
Swarm	✗(Hln.)	✗	✗(Itg. Op)
Gama	✗(Hln.)	✗	✗(Itg. Op)
Repast-Symphony	✗	✗	✗(Itg. Op / Lang.)
Jedi	Int.	✗	✗(Itg. Op)
A&A	✓(Int. / A&A)	✓	✗(Itg. Op)
Geamas-ng	Int. / DOM	Env.	✗(Itg. OP)

Plate-forme	Contraintes		
	Réutilisation	**Modularité**	**Level-up**
Jade			
MadKit	✗	✗	✗
Janus	✗	✗	✓
Swarm	✗	✗	✓
Gama	✗	✗	✓
Repast-Symphony	✗	✗	✗
Jedi	✗	✗	✗
A&A	✗	modèles	✓
Geamas-ng	✗	modèles	✓

Légende :

Conceptuel	Org.	Méta-modèle organisationnel
	Hln.	Méta-modèle holonique
	Int.	Méta-modèle orienté interaction
	A&A	Méta-modèle Agents et Artefacts
	DOM	Modélisation orientée dynamique
Sémantique	Env.	Couplage des modèles par l'environnement, spécification des choix de multi-modélisation pas explicite
	✓	Solutions satisfaisantes ou contrainte respectée
	✗	Solutions non satisfaisantes ou contrainte non respectée
	texte	Solution ou contrainte partiellement satisfaisante

FIGURE 6.8 – Résumé des solutions proposées par les approches multi-agent concernant la multi-modélisation.

Chapitre 7

Positionnement et problématique

Sommaire

7.1 Introduction

Les démarches de multi-modélisation et de réutilisation de modèles et de simulateurs posent des contraintes relativement bien identifiées. Depuis les contraintes purement conceptuelles de prise en compte de plusieurs modèles hétérogènes, jusqu'aux contraintes techniques d'interopérabilité des outils de simulation, le processus de multi-modélisation est en soi un véritable travail pluridisciplinaire qui nécessite des connaissances à la fois dans les domaines thématiques du multi-modèle (dans notre cas les réseaux ambiants et les sciences humaines), dans le domaine de la théorie de la modélisation et de la simulation et dans le domaine de l'ingénierie logicielle.

Dans les chapitres précédents (chapitre 5 et 6) nous avons passé en revue les différentes solutions existantes concernant la multi-modélisation et le couplage de simulateurs. Ces solutions prises une à une ne nous satisfont pas. En effet, une des problématiques majeures du domaine des systèmes complexes est la nécessité de fournir une solution simple qui permette à un modélisateur (qui n'est généralement pas expert en multi-modélisation) de concevoir un modèle qui capte toute la complexité du système étudié [Bourgine et al., 2008]. Nous pensons qu'il est important de pouvoir réutiliser les outils de modélisation et de simulation qui sont disponibles dans les domaines scientifiques concernés. Il est aussi nécessaire que le multi-modèle et les résultats obtenus par simulation soient reproductibles afin de valider la crédibilité scientifique du travail du modélisateur.

Les outils comme VLE, JAMESII ou encore HLA apportent des solutions aux niveaux technique, dynamique et syntaxique (voir la figure 7.1). Ces solutions satisfont nos contraintes de réutilisation et d'interopérabilité des outils de modélisation et de simulation existants. Cependant, ces approches ne proposent pas de solutions conceptuelles qui satisfassent nos besoins en terme de représentation des différents niveaux d'abstractions (passage à l'échelle type *level-up*).

Les approches multi-agents, quant à elles, proposent des solutions aux niveaux conceptuel et sémantique satisfaisantes par rapport à nos contraintes de représentation des différents niveaux d'abstraction. Cependant, les solutions proposées aux niveaux technique, dynamique et syntaxique ne satisfont pas nos contraintes de réutilisation et d'interopérabilité des outils existants de modélisation et de simulation (voir la figure 7.1).

Nous avons donc besoin d'une approche de multi-modélisation adaptée à nos contraintes. Celle-ci se doit d'apporter à la fois les avantages d'une approche multi-agent aux niveaux conceptuel et sémantique et

Approches	Contraintes		
	Réutilisation	**Modularité**	**Level-up**
Itg. O	✗	✗	✗
Cp. Sim ad hoc	✗	✗	✗
Itg. F	✗	modèles DEVS	✗
Féd. Sim	✓	✓	✗
Prog. SMA	✗	✗	✗
Sim. Sma	✗	✗	✗
SMA MM.	✗	modèles	✓

Légende :

Approches	Itg .O	Intégration Opérationnelle de modèles
	Cp. Sim ad hoc	Couplage ad hoc de simulateurs
	Itg.F	Intégration Formelle de modèles
	Féd. Sim	Fédération (couplage) de simulateurs
	Prog. SMA	Plates-formes de programmation multi-agent
	Sim. SMA	Plates-formes de modélisation et de simulation multi-agent
	SMA MM.	Approches multi-agents pour la multi-modélisation
	✓	Solutions satisfaisantes ou contrainte respectée
	✗	Solutions non satisfaisantes ou contrainte non respectée
	texte	Solution ou contrainte partiellement satisfaisante

FIGURE 7.1 – Satisfaction des contraintes

les avantages des approches de couplage de modèles et de simulations aux niveaux technique, dynamique et syntaxique. Nous proposons une approche de multi-modélisation. Celle-ci est présentée au chapitre suivant. Ce chapitre ci a pour but de montrer les questions auxquelles notre proposition doit répondre et comment nous comptons évaluer cette proposition.

7.2 Problématiques abordées

Nous gardons ici la structure en niveaux présentée au chapitre 5. Pour chacun des niveaux, nous exprimons les problématiques auxquelles notre proposition doit répondre.

Au niveau conceptuel nous souhaitons une approche qui nous permette de définir les modèles comme des entités hétérogènes et autonomes. Nous souhaitons également pouvoir exprimer les notions de niveau d'abstraction et d'échelle. Pour que cette approche soit scientifiquement enrichissante, il nous faut pouvoir faire ressortir les choix de multi-modélisation (le passage d'une échelle à l'autre par exemple). Au niveau conceptuel, nous ne voulons pas introduire d'entité centrale en charge du contrôle du multi-modèle. Ce choix se justifie d'abord par l'ensemble des contraintes de réutilisation, de modularité et de passage à l'échelle (aussi bien en terme de *size-up*, de *speed-up* que de *level-up*). Ensuite, la séparation en différents niveaux facilite l'utilisation de solutions existantes qu'elles soient centralisées ou non (pour des questions techniques notamment). Cependant, le niveau conceptuel étant le plus générique, il nous paraît beaucoup plus facile d'avoir une vision conceptuelle totalement décentralisée utilisant des solutions de plus bas niveau centralisées ou décentralisées que l'inverse, à savoir, une vision conceptuelle centralisée qui utiliserait des solutions de plus bas niveau décentralisées. Ainsi, au niveau conceptuel, nous nous efforcerons de garder ce point de vue décentralisé.

Au niveau sémantique, il est important de noter que chaque modèle élémentaire composant le multi-modèle est issu de domaines différents et n'a pas obligatoirement été conçu pour être couplé. Il est donc nécessaire de pouvoir donner du sens aux liens entre les différents modèles. Le couplage entre plusieurs modèles est un choix de modélisation qu'il faut pouvoir expliciter et valider au même titre que les modèles

élémentaires. Au niveau conceptuel, nous devons nous donner les moyens de faire ressortir les choix de multi-modélisation afin de leur donner du sens au niveau sémantique.

Au niveau syntaxique, nous faisons l'hypothèse que les modèles réutilisés n'ont pas obligatoirement les mêmes formalismes. Le fait de les coupler introduit donc la contrainte de compatibilité des modèles formalisés. Il faut pouvoir faire interagir des modèles dont les formalismes sont différents.

Au niveau dynamique, nous faisons l'hypothèse que les modèles réutilisés n'ont pas obligatoirement les mêmes politiques d'exécution. L'exécution du multi-modèle doit donc se faire en tenant compte de ces différentes politiques. Elle doit également respecter la contrainte de causalité présentée au chapitre 2. À ce niveau, notre volonté de proposer une approche décentralisée va nous amener à proposer un nouvel algorithme de simulation que nous présenterons au chapitre 9.

Au niveau technique, nous souhaitons faire interagir différents modèles et réutiliser leurs propres simulateurs. Un travail d'ingénierie logicielle est donc nécessaire pour rendre interopérable et modulaires les différentes briques logicielles de base réutilisées. Il nous faut aussi admettre qu'au niveau technique, les performances en terme de passage à l'échelle de type *speed-up* et *size-up* ne sont pas la priorité de ce travail. La satisfaction de la contrainte de passage à l'échelle de type *level-up* se passe, quant à elle, aux niveaux précédents.

7.3 Synthèse

Il nous parait scientifiquement intéressant de proposer une approche de multi-modélisation qui satisfait simultanément nos contraintes de réutilisation, de modularité et de passage à l'échelle. En effet, la modélisation des systèmes complexes fait souvent appel à plusieurs domaines scientifiques et nous pensons qu'il est intéressant de pouvoir réutiliser les outils de modélisation et de simulation présents dans les domaines concernés.

Après avoir passé en revue les approches existantes de multi-modélisation et de couplage de simulations, nous souhaitons proposer une approche novatrice de multi-modélisation et de couplage de simulation pour les systèmes complexes. En plus de répondre à nos contraintes de réutilisation et d'interopérabilité, de modularité et de passage à l'échelle, cette approche se doit de répondre aux différents problèmes rencontrés à la fois aux niveaux conceptuel, sémantique, syntaxique, dynamique et technique. Pour ce faire, nous proposons d'utiliser la vision conceptuelle issue du monde multi-agent associée aux solutions plus techniques offertes par le domaine de la multi-modélisation et des fédérations de simulations.

Une des contraintes de ce travail est d'avoir une approche homogène et cohérente afin de ne pas compliquer inutilement la tâche de multi-modélisation. En effet, en ciblant explicitement les systèmes complexes, la réutilisation et le couplage d'outils de modélisation et de simulation, nous nous plaçons dans le cas où plusieurs modélisateurs issus de domaines scientifiques différents doivent interagir afin de créer un multi-modèle. La méthode de multi-modélisation que nous proposons se doit, autant que faire ce peut, de ne pas ajouter de difficultés en plus de celles inhérentes à l'interdisciplinarité de ce travail.

Nous proposons une approche de multi-modélisation et de couplage de simulation appelée AA4MM dans la partie suivante. Celle-ci est basée sur le paradigme Agents et Artefacts et s'inscrit dans la lignée des travaux menés dans la communauté de la simulation multi-agent. L'originalité de notre contribution vient du fait que nous ciblons explicitement la réutilisation et l'interopérabilité d'outils de modélisation et de simulation existants tout en conservant une vision purement multi-agent. Nous proposons également un ensemble de démonstrations théoriques, conceptuelles et expérimentales qui nous permettent d'évaluer les avantages et les limites de notre proposition.

Chapitre 8

Le méta-modèle AA4MM : agents et artefacts pour la multi-modélisation

Sommaire

8.1 Introduction

Dans ce chapitre, nous proposons le méta-modèle appelé AA4MM (*Agents and Artifact for Multimodeling*). Celui-ci permet de concevoir un multi-modèle comme une "société de modèles en interaction" (l'expression est empruntée à [Bonneaud, 2008]). Le multi-modèle est vu comme un système multi-agent. L'originalité de notre proposition est que nous appliquons l'approche multi-agent à tous les niveaux. Le méta-modèle AA4MM définit un ensemble de concepts qui s'appliquent aux niveaux sémantique et syntaxique – pour voir le multi-modèle comme un ensemble de modèles autonomes, hétérogènes et en interaction ; au niveau dynamique – pour voir la simulation du multi-modèle comme un ensemble de simulations autonomes, hétérogènes et en interaction ; et au niveau technique – pour voir le logiciel en charge du multi-modèle comme un ensemble de logiciels autonomes, hétérogènes et en interaction.

Nous présentons d'abord une vue générale de AA4MM. Nous donnons les grandes lignes et présentons les quelques hypothèses propre à ce travail (voir section 8.2). Ensuite, nous présentons pour chaque niveau

comment s'applique la vision multi-agent (voir section 8.3). Cette démarche relativement répétitive- on applique les mêmes concepts à des problèmes différents- est néanmoins nécessaire pour ancrer rigoureusement notre travail à la théorie de la modélisation et de la simulation. Enfin, notre démarche pour chaque niveau étant homogène, nous présentons une version simplifiée mais toutefois valide à tous les niveaux de AA4MM qui correspond à l'implantation que nous en avons fait (voir section 8.4).

8.2 Principe du méta-modèle AA4MM

8.2.1 Vue générale

L'idée directrice qui a présidé à la proposition de AA4MM est de concevoir un multi comme une "*société de modèles en interaction*" et de créer un système multi-agent dont le rôle va être de gérer l'ensemble des problèmes liés à la multi-modélisation, au couplage de simulateur et à nos contraintes. À chaque modèle, à chaque simulateur et à chaque logiciel, nous associons un agent.

Dans la pratique, il n'existe que très rarement des objets appelés modèles, modèles formalisés et simulateurs abstraits. Ces derniers sont présents à l'intérieur de logiciels de simulation. Il arrive parfois que plusieurs modèles ou plusieurs mécanismes de simulation soient présents dans un seul et même logiciel de simulation. Pour notre part, nous avons fait l'hypothèse qu'à un logiciel de simulation n'est associé qu'un seul simulateur abstrait et un seul modèle. Nous appelons cet ensemble un *bloc MSL* (pour modèle, simulateur et logiciel). L'agent associé à un *bloc MSL* est appelé *m-agent*.

Niveaux	**Concepts manipulés**
Sémantique	Modèle
Syntaxique	Modèle formalisé
Dynamique	Simulateur abstrait
Technique	Logiciel de simulation

FIGURE 8.1 – Concepts et niveaux de questions manipulés par un *m-agent*.

Chaque agent doit pouvoir contrôler et gérer le modèle, le simulateur ou le logicel auquel il est associé. Comme ceux-ci sont des outils qui peuvent être préexistants, il est nécessaire de concevoir une interface générique entre un agent et l'outil qu'il gère. Cela nous permet de répondre à nos contraintes de réutilisation et de modularité.

Ensuite, l'ensemble des agents doivent interagir et se coordonner afin de mener à bien l'ensemble de la simulation du multi-modèle. Que ce soit au niveau sémantique (changement d'échelle d'un modèle à un autre), syntaxique (changement de formalisme), dynamique (coordination des simulations) ou technique (échange de messages entre plusieurs logiciels), l'ensemble des interactions entre les agents se déroule via leur environnement. Il n'y a donc pas d'agent particulier qui centraliserait le contrôle et la gestion de la simulation du multi-modèle.

La figure suivante (figure 8.2) illustre l'idée générale du méta-modèle AA4MM.

8.2.2 Méta-modèle utilisé

Nous utilisons un méta-modèle multi-agent centré sur les interactions. Plus particulièrement, nous nous inspirons du méta-modèle Agents et Artefact (A&A) défini dans [Ricci et al., 2007a]. Pour rappel, le méta-modèle A&A propose que les interactions entre les agents soient modélisées par des artefacts. Ces derniers sont des objets passifs de l'environnement. Ils ont pour but de permettre aux agents d'interagir. On peut les voir comme des ensembles d'outils mis à disposition des agents afin que ceux-ci puissent réaliser leurs buts (voir figure 8.3).

Ce méta-modèle ne fait aucune supposition sur la dynamique interne des agents et de l'environnement. La seule hypothèse qui est faite est que les agents peuvent et savent comment utiliser les artefacts. Au niveau modularité, il est facile d'ajouter des agents, d'en supprimer ou d'échanger un agent par un

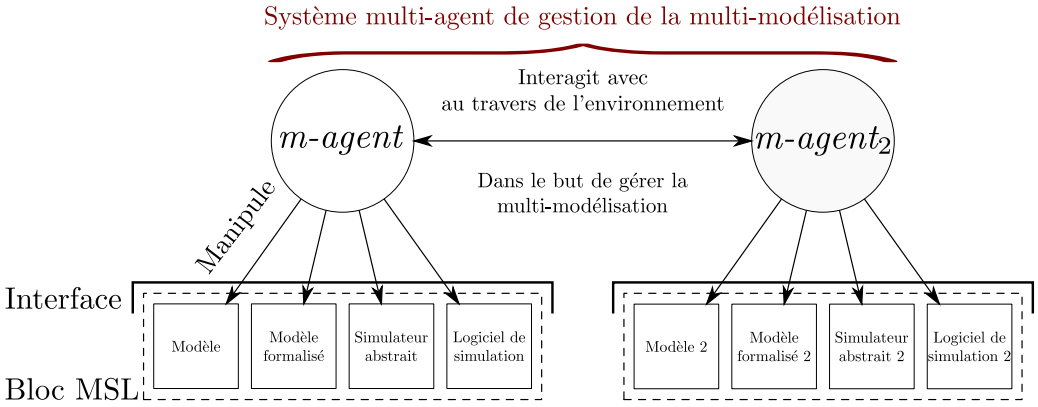


FIGURE 8.2 – Vue générale de AA4MM. Notre proposition peut se résumer à un système multi-agent qui manipule des modèles, des simulateurs abstraits et des logiciels de simulation afin de gérer les problèmes liés à la multi-modélisation.

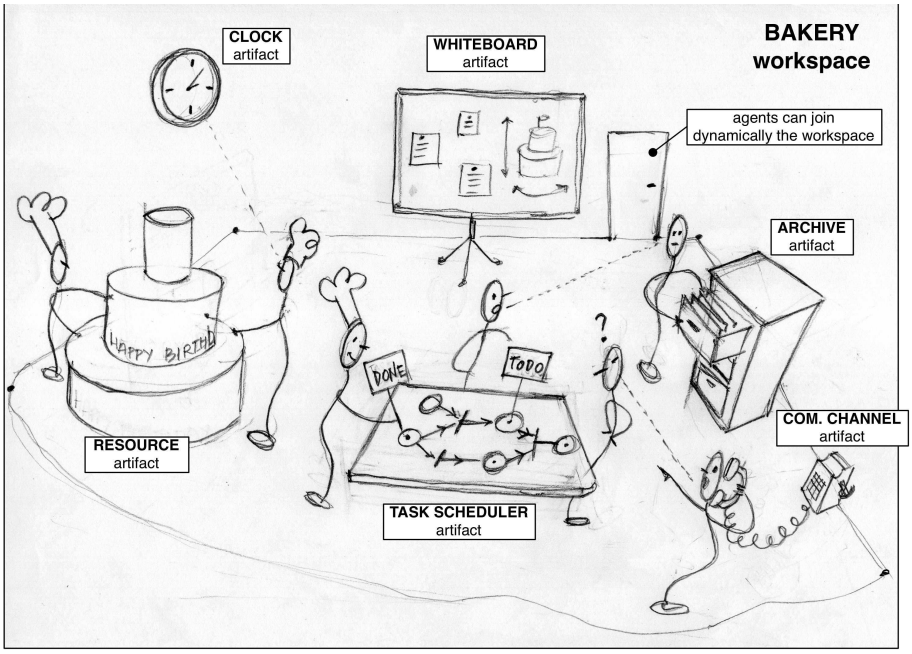


FIGURE 8.3 – Représentation métaphorique d'un système multi-agent selon le méta-modèle Agents et Artefacts (A&A). Les agents, ici des cuisiniers, coordonnent leur activité grâce à un ensemble d'artefacts présents dans leur environnement. Source [Piunti and Ricci, 2008]

autre. Il est également plus facile d'ajouter et de supprimer un artefact ou d'échanger deux artefacts qui remplissent la même fonction mais de manière différente. Ensuite, le fait de décomposer l'environnement en un ensemble d'artefacts supprime l'aspect centralisateur de l'environnement dans le système multi-agent. Enfin, l'avantage d'un tel méta-modèle est de pouvoir expliciter les interactions entre les agents et de les réifier.

8.2.3 Concepts et hypothèses

Il nous faut faire quelques hypothèse sur les concepts définis. Nous souhaitons réutiliser et faire interagir des modèles, des simulateurs abstraits et des logiciels de simulation. Il est donc nécessaire de définir quels sont les moyens que l'on s'autorise à utiliser pour aboutir à ces interactions. La contrainte principale ici est la réutilisation d'outils existants. Nous ne pouvons faire aucune supposition sur la sémantique et le corps d'un modèle, ni sur le formalisme utilisé, ni sur la politique d'exécution utilisée par un simulateur abstrait, ni sur le langage utilisé et la façon dont est codé un logiciel de simulation. Cependant, il nous faut définir des points d'entrée relativement génériques pour pouvoir avancer.

Nous considérons qu'un modèle possède un ensemble de ports d'entrée et de ports de sortie qui nous sont accessibles. Nous faisons l'hypothèse qu'il nous est possible, pour chaque valeur du temps de simulation, de donner un ensemble de paramètres au modèle et d'en tirer un ensemble de données. Nous postulons également que les formalismes d'un modèle élémentaire à un autre ne sont pas forcément identiques.

Au niveau du simulateur abstrait, nous faisons l'hypothèse que l'on peut récupérer le temps courant de simulation et que l'on peut contrôler le simulateur. Par contrôle, nous entendons qu'il est possible de demander au simulateur abstrait d'exécuter son modèle pour une quantité de temps de simulation donnée. Par exemple, exécuter le prochain pas de temps, les cinq prochains pas de temps, le prochain événement de simulation ou les évènements de simulation jusqu'à une date donnée. Nous faisons également l'hypothèse que le simulateur ne possède pas de mécanisme de retour arrière (*rollback*, voir chapitre 2) et qu'il nous est impossible d'ajouter au simulateur abstrait des événements de simulation qui proviennent d'autres simulateurs abstraits.

Enfin, au niveau du logiciel de simulation, nous faisons l'hypothèse qu'il est possible de programmer les points précédents, par exemple via une interface de programmation ou directement par le code source du logiciel.

Niveaux	**Concepts manipulés**	**Hyptohèses**
Sémantique	Modèle	Possède un ensemble de ports d'entrée et de ports de sortie
Syntaxique	Modèle formalisé	Formalismes différents d'un modèle à l'autre
Dynamique	Simulateur abstrait	Politiques d'exécutions différentes d'un simulateur à l'autre
		Possibilité de connaître le temps courant de simulation
		Possibilité de contrôler le simulateur abstrait
Technique	Logiciel de simulation	Possibilité d'interfacer le logiciel

FIGURE 8.4 – Hypothèse posées.

8.3 Application du méta-modèle aux différents niveaux de question

Dans les sections suivantes, nous expliquons comment, pour chaque niveau, nous appliquons la vision A&A aux problèmes posés par la multi-modélisation et nos contraintes de réutilisation, modularité et de passage à l'échelle de type *level-up*. Bien que cette partie fasse apparaître un bon nombre de répétitions, elle est néanmoins nécessaire afin d'ancrer rigouresement notre démarche dans la théorie de la modélisation et de la simulation telle qu'elle est présentée par Bernard P. Zeigler dans [Zeigler et al., 2000].

8.3.1 Niveau sémantique

Le niveau sémantique porte sur la signification et le rôle du multi-modèle face aux questions posées. Chaque modèle élémentaire utilisé dans le multi-modèle est développé par rapport à une question particulière. La dynamique du multi-modèle provient des dynamiques des modèles élémentaires et de leurs interactions. Les interactions entre les modèles font intervenir un certain nombre de choix de multi-modélisation qu'il est nécessaire d'expliciter (voir chapitre 5).

À ce niveau, le but du système multi-agent est de gérer la cohérence sémantique du multi-modèle. Cela se passe au niveau de l'échange des données entre les modèles. Le *m-agent* doit être capable de récupérer les données émises par le modèle, via ses ports de sortie. Ensuite le *m-agent* doit savoir à qui envoyer ces données. Lors de l'échange, il est nécessaire d'appliquer un certain nombre d'opérations à ces données. Cela est du aux différences d'échelles ou de dimensions entre les modèles. Ensuite le *m-agent* qui récupère ces données doit être capable de les fournir au modèle, via ses ports d'entrée (voir figure 8.5).

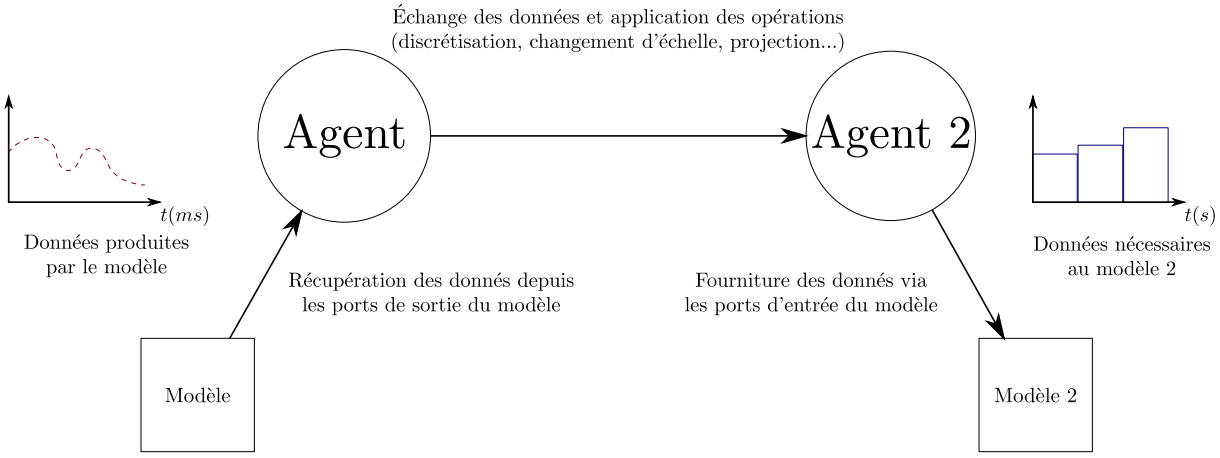


FIGURE 8.5 – Au niveau sémantique, le rôle du système multi-agent est de gérer les choix de multi-modélisation.

Les *m-agents* ne peuvent entreprendre des actions qu'au travers de leurs artefacts. Ici, nous proposons d'utiliser un artefact pour faire l'interface entre le *m-agent* et le modèle. Le rôle de celui-ci est de permettre à le *m-agent* de récupérer les données provenant des ports de sortie du modèle et d'injecter des données dans les ports d'entrée du modèle.

Ensuite, de la même manière que dans [Bonneaud, 2008], nous proposons d'utiliser un artefact pour donner corps à l'interaction entre deux modèles. Ce dernier a pour rôle de permettre à deux *m-agents* d'échanger des données provenant des modèles et d'appliquer les opérations nécessaires aux modifications des données (changement d'échelle, projection, discrétisation, *etc.*). Un certain nombre d'opérations sont déjà décrites dans [Bonneaud, 2008]. Étant donné, que ces opérations vont dépendre du système étudié et des questions posées au multi-modèle, il n'est pas intéressant de lister ici toutes les opérations possibles. Ce qui est important, c'est d'avoir une entité particulière au travers de laquelle le modélisateur peut (et doit) faire apparaître ces choix de multi-modélisation.

Si on reprend la notation graphique utilisée dans [Bonneaud, 2008] pour représenter les artefacts et que nous appliquons notre méta-modèle à l'exemple de la figure précédente (figure 8.5), nous obtenons la figure 8.6.

8.3.2 Niveau syntaxique

Pour nous, le niveau syntaxique est le pendant du niveau sémantique. Nous postulons que les modèles qui interagissent ont été décrits dans des formalismes différents. Les interactions entre les modèles formalisés impliquent des échanges de données qui sont décrites dans des formalismes différents. Le rôle du

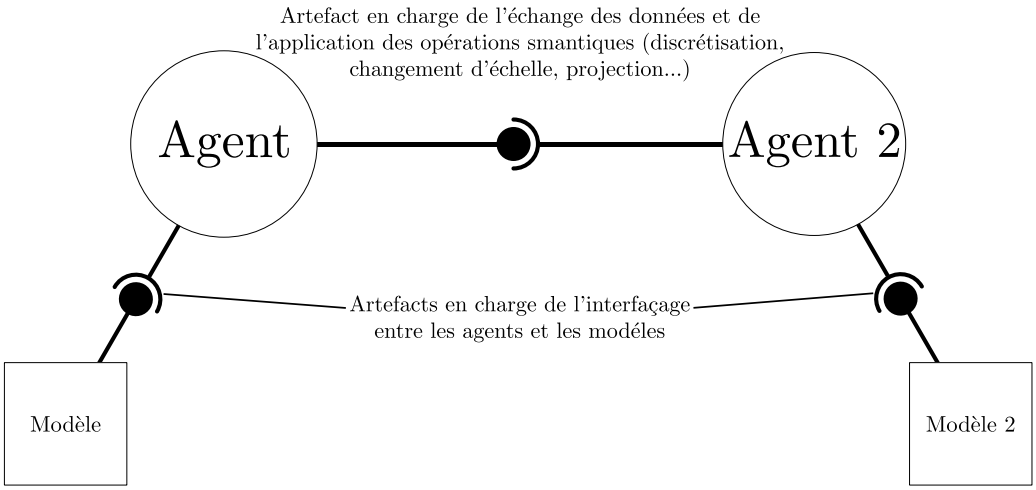


FIGURE 8.6 – Au niveau sémantique, nous définissons deux types d'artefacts : les artefacts en charge de l'interfaçage entre un *m-agent* et le modèle qu'il gère et les artefacts qui s'occupent des échanges de données entre plusieurs modèles.

système multi-agent de gestion de la multi-modélisation est de faire en sorte que chaque modèle formalisé récupère des données dans son propre formalisme. De la même manière que l'on peut définir des opérations qui consistent à modifier la sémantique des données échangées, on peut définir des opérations qui vont modifier le formalisme des données échangées. Comme précédemment on peut définir un ensemble d'artefacts dont les rôles vont être de permettre aux *m-agents* d'échanger des données et de modifier au besoin le formalisme de ces données (voir figure 8.7).

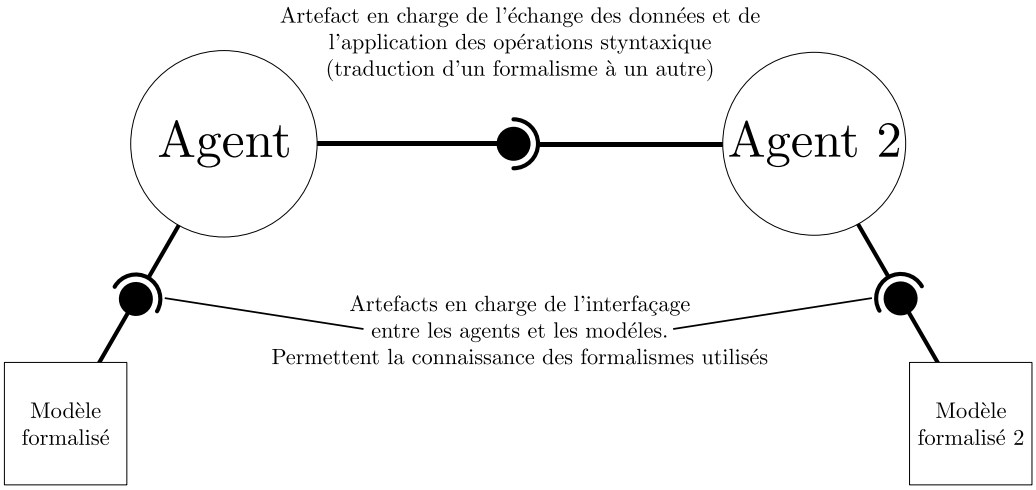


FIGURE 8.7 – Rôle des artefacts au niveau syntaxique.

8.3.3 Niveau dynamique

Au niveau dynamique, la simulation du multi-modèle est vue comme un ensemble de simulateurs abstraits autonomes qui doivent interagir afin de faire en sorte que la simulation globale respecte la contrainte de causalité (voir chapitre 2). Nous postulons que chaque simulateur abstrait peut utiliser une politique d'exécution de modèle qui lui est propre (simulation par pas de temps, simulation événementielle, *etc.*).

Ici, nous proposons que le *m-agent* qui manipule le simulateur abstrait puisse observer ce simulateur afin de connaître la valeur courante du temps de simulation et puisse contrôler le simulateur. Par contrôle, le *m-agent* peut seulement agir sur le simulateur abstrait pour que celui-ci exécute son modèle pour une quantité donnée de temps de simulation (un pas de temps, le prochain événement de simulation, *etc.*). De la même manière que précédemment, nous proposons de définir un artefact qui va permettre à le *m-agent* de récupérer le temps de simulation et de contrôler les simulateurs abstraits (voir figure 8.8).

Ensuite, l'ensemble des simulateurs abstraits doivent se coordonner pour que la simulation du multi-modèle global respecte la contrainte de causalité. Cette coordination entre simulateurs abstraits se traduit par une coordination entre *m-agents*. Nous avons pris le parti d'une approche décentralisée. Nous n'utilisons pas d'agent spécifique pour coordonner l'ensemble des simulateurs abstraits. C'est par l'échange d'informations via leur environnement (les artefacts) que se passe la coordination. De plus, nous avons fait les hypothèses que les simulateurs abstraits ne possèdent pas de mécanisme de retour arrière (*roll-back*) et que les simulateurs abstraits possèdent des politiques de simulations qui peuvent être différentes. Ces contraintes nous ont amené à chercher des solutions dans le domaine de la simulation distribuée, au niveau des algorithmes conservatifs de simulation (voir le chapitre 2). Cependant, ne trouvant pas d'algorithme satisfaisant l'ensemble de nos contraintes, nous proposons un algorithme de simulation distribuée qui nous est propre et qui s'adapte à notre vision conceptuelle. La présentation de celui-ci fait l'objet du chapitre suivant (chapitre 9).

Nous ne donnons dans cette section que quelques points de repère afin d'illustrer le rôle des *m-agents* et des artefacts au niveau dynamique. Le principe de notre algorithme de simulation distribuée est d'associer aux données échangées une information concernant le temps de simulation du modèle qui a produit cette donnée. Ce principe est classique dans la simulation distribuée. L'originalité de notre approche provient de notre vision multi-agent et des contraintes que nous nous imposons. Un *m-agent* ne connaît de son simulateur que le temps courant de simulation. Lors d'une exécution du modèle, le simulateur abstrait change l'état du modèle, il produit des données (ports de sortie du modèle) et augmente le temps de simulation d'une valeur t à une valeur $t + \Delta t$. Ces deux valeur t et $t + \Delta t$ déterminent un intervalle de temps de simulation que nous appelons intervalle de validité (noté Γ). En associant les données issues des ports de sortie du modèle avec cet intervalle de validité, nous affirmons chaque *m-agent* possède une information suffisante pour respecter la contrainte de causalité et se coordonner avec les autres *m-agents*. Pour respecter la contrainte de causalité, un *m-agent* filtre simplement les données qu'il reçoit en fonction de son propre temps de simulation et des intervalles de validité associés aux données (voir le détail de l'algorithme donné au chapitre suivant).

Nous proposons de définir un artefact dont le rôle va être de permettre aux *m-agents* de filtrer les données échangées afin de respecter la contrainte de causalité. La figure suivante (figure 8.8) résume les rôles des *m-agents* et des artefacts au niveau dynamique.

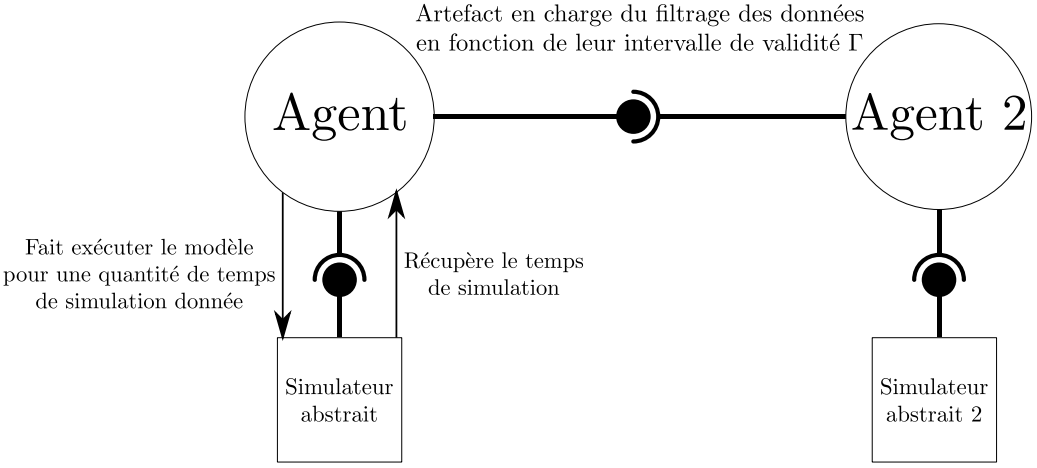


FIGURE 8.8 – Rôles des artefacts au niveau dynamique

8.3.4 Niveau technique

Au niveau technique, nous souhaitons que le logiciel qui gère la simulation du multi-modèle soit composé des logiciels de simulation des différents modèles élémentaires réutilisés. Nous souhaitons également que les logiciels réutilisés soient utilisables à la fois de manière couplée et de manière non-couplée (en version *standalone*). Il est donc nécessaire de créer une interface entre un *m-agent* et le logiciel de simulation. Cette interface doit permettre à le *m-agent* de concrétiser l'ensemble des actions qu'il doit effectuer sur le logiciel de simulation.

Au niveau des choix techniques, il nous paraît intéressant suivre l'architecture conceptuelle décentralisée que nous proposons afin de satisfaire nos contraintes de modularité et de passage à l'échelle. À court terme, nous souhaitons surtout satisfaire les contraintes de modularité et de passage à l'échelle de type *level-up*. Nous pensons qu'il est également important que les choix techniques ne remettent pas en cause la possibilité d'un passage à l'échelle de type *size-up* et *speed-up*. Pour cela, nous proposons que chaque entité du méta-modèle (que ce soient les *m-agents* ou les artefacts) soit vue comme une entité logicielle à part entière et que les interactions entre ces entités puisse se faire de manière distante, concurrente et asynchrone.

Faire interagir plusieurs logiciels de simulation peut nécessiter des ressources en terme de mémoire et de puissance de calcul relativement grande. Il est alors intéressant de faire s'exécuter les logiciels de simulations sur des machines distinctes reliées en réseau. Ensuite, plusieurs *m-agents* peuvent interagir avec un artefact donné de manière concurrente (lors de l'échange de données par exemple). Enfin, chaque *m-agent* manipule son propre logiciel de simulation. Le temps de calcul pour chacun peut donc être différent. L'interaction entre plusieurs *m-agents* doit donc pouvoir se faire de manière asynchrone. Cela afin d'éviter, d'une part, l'introduction à un niveau technique d'une entité centrale qui synchronise l'ensemble des calculs. D'autre part, pour éviter, quand c'est possible, que le calcul global de la simulation soit contraint par le plus lent des logiciels de simulation.

Le méta-modèle A&A stipule que les agents sont des entités proactives et les artefacts des entités passives. Ce sont donc les *m-agents* qui initient les interactions. Les artefacts ne font que réagir aux demandes des *m-agents*.

8.3.5 Synthèse

Notre proposition consiste à créer un système multi-agent dont le rôle est de gérer l'ensemble de la multi-modélisation et du couplage des logiciels de simulations. Cette proposition doit respecter les contraintes de réutilisation, de modularité et de passage à l'échelle que nous avons posées. Dans cette section, nous avons listé l'ensemble des niveaux de question auxquels nous faisons face et, pour chacun de ces niveaux, nous appliquons un point de vue multi-agent particulier : le méta-modèle A&A.

Le résultat est que chaque *m-agent* doit gérer un ensemble de concepts qui sont un modèle, un modèle formalisé, un simulateur abstrait et un logiciel de simulation. Les *m-agents* interagissent entre eux pour gérer l'ensemble des problèmes liés à la multi-modélisation. Nous avons défini un ensemble d'artefacts - qui sont des entités passive appartenant à l'environnement du système multi-agent - dont le rôle va être de permettre aux *m-agents* de mener à bien leurs actions. À chaque niveau de question, nous avons défini un artefact dont le rôle est de faire l'interface entre un *m-agent* et le concept manipulé (modèle, simulateur abstrait, *etc.*) et un artefact dont le rôle est de gérer l'interaction entre plusieurs *m-agents*. Pour des raisons de simplicité de l'illustration, nous n'avons donné à chaque fois que des exemples avec seulement deux *m-agents*. Cependant, nous ne mettons aucune contrainte sur le nombre d'*m-agents* impliqués dans des interactions. La seule contrainte que nous imposons à notre système multi-agent est qu'un *m-agent* ne gère qu'un seul modèle, un seul modèle formalisé, un seul simulateur abstrait et un seul logiciel de simulation. La figure 8.9 résume les concepts mis en œuvre pour chaque niveau de question.

8.4 Méta-modèle AA4MM, spécifications opérationnelles

Dans la section précédente, nous décrivons notre proposition dans les grandes lignes. Dans cette section nous décrivons notre proposition de manière plus formelle.

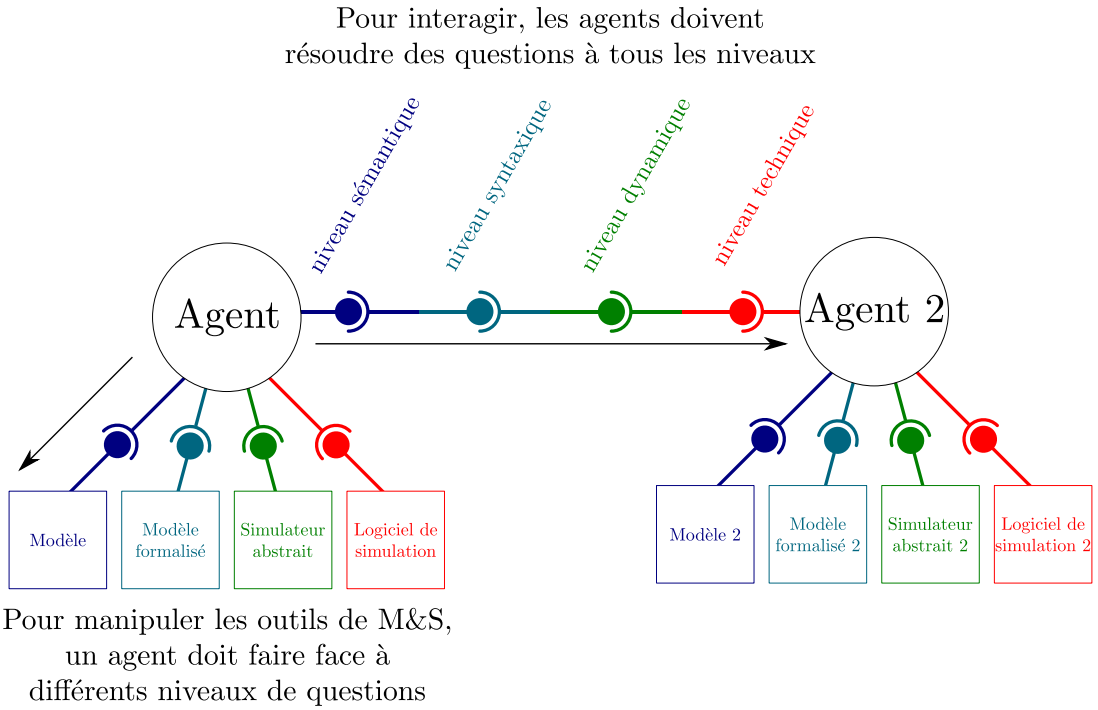


FIGURE 8.9 – Pour chaque niveau de question, nous proposons un artefact en charge de l'interface entre le *m-agent* et le concept manipulé et un artefact en charge de l'interaction entre plusieurs *m-agents*. Ici nous avons représenté le niveau sémantique en bleu foncé, le niveau syntaxique en bleu clair, le niveau dynamique en vert et le niveau technique en rouge.

8.4.1 Notations et simplifications du méta-modèle

Nous avons fait l'hypothèse qu'à un logiciel de simulation n'est associé qu'un seul modèle, un seul modèle formalisé et un simulateur abstrait. Il est donc possible de regrouper l'ensemble des concepts en un seul et même bloc. Ainsi, en considérant, i comme étant l'indice du logiciel de simulation utilisé, on note ls_i un logiciel de simulation, m_i un modèle, mf_i un modèle formalisé, s_i un simulateur abstrait et $\mathcal{A}_i$ le *m-agent* en charge de l'ensemble (m_i, mf_i, s_i, ls_i).

De même l'ensemble des artefacts qui font office d'interfaces entre le *m-agent* et chacun des concepts peut être simplifié en un seul et même artefact que nous appelons *artefact-d'interface* et que nous notons $\mathcal{I}_i$ (voir figure 8.10). La seule contrainte est que cette nouvelle interface implante l'ensemble des opérations définies à chaque niveau. Il est important de noter que nous faisons cette simplification uniquement par rapport à notre hypothèse de départ. Dans le cas où cette hypothèse ne serait pas valide, si, par exemple différents modèles ou différents simulateurs abstraits sont présents dans un même logiciel de simulation, alors il faudrait utiliser plusieurs artefacts (un pour chaque niveau) comme vu précédemment. Cette simplification nous est utile pour ne pas surcharger inutilement les spécifications opérationnelles.

Chaque modèle possède un ensemble de ports d'entrée noté X_i et un ensemble de ports de sortie notés Y_i. On note x_i^k le k$^{\text{ième}}$ port d'entrée du modèle m_i ($0 < k \leq \text{card}(X_i)$). De même on notera y_i^n le n$^{\text{ième}}$ port de sortie du modèle m_i ($0 < n \leq \text{card}(Y_i)$). Ensuite, un simulateur abstrait s_i possède une variable appelée temps de simulation et notée t_i. L'exécution d'un modèle m_i au temps de simulation t_i signifie quatre choses :

- tenir compte des données d'entrée du modèle : pour chaque port d'entrée donné x_i^k, on note $\mathsf{d_{in}}_i^k(t_i)$ la donnée d'entrée ;
- modifier l'état du modèle m_i ;
- produire les données de sortie du modèle : pour chaque port de sortie donné y_i^n, on note $\mathsf{d_{out}}_i^n(t_i)$

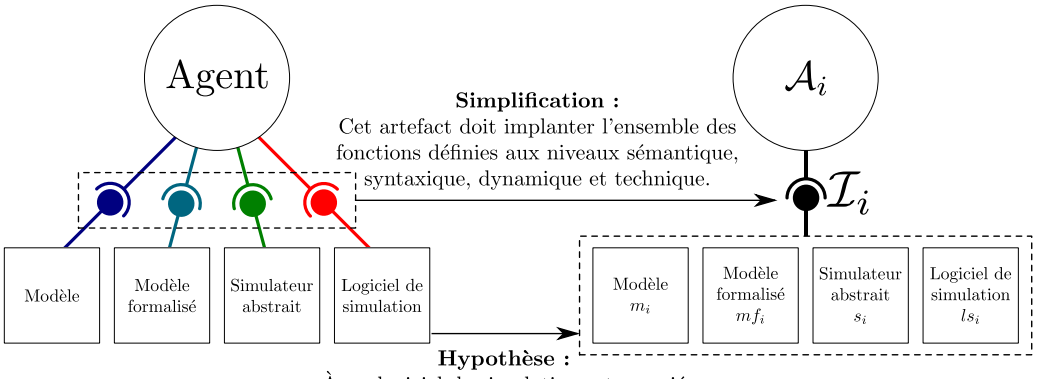


FIGURE 8.10 – Il est possible de simplifier le schéma en remplaçant les quatre interfaces par une seule. Cette interface plus générique doit permettre les mêmes actions que les quatre précédentes.

la donnée produite ;
- augmenter le temps de simulation : $t_i \leftarrow t_i + \delta t_i$ avec $\delta t_i \geq 0$.

Par la suite, on s'autorisera à simplifier les notations des ports d'entrée et de sortie ainsi que les notations des données d'entrée et de sortie quand cela ne nuit pas à la compréhension.

Il nous faut également formaliser le graphe de dépendance. On notera G le graphe de dépendance représenté par l'ensemble des quadruplets (j, n, i, k) tel que la donnée $\mathsf{d}_{\mathsf{out}_j}^{\,n}(t_j)$ est nécessaire à l'exécution du modèle m_i et tel que cette donnée doit être envoyée au modèle m_i au travers de son port d'entrée x_i^k (avec $0 < n \leq \text{card}(Y_j)$ et $0 < k \leq \text{card}(X_i)$). Il est à noter que G contient toute l'information globale du graphe de dépendance. Pour respecter le méta-modèle AA4MM, il faut que cette information soit répartie à la fois sur les *m-agents* et sur les *artefacts-de-couplage*.

Chaque *m-agent* ne possède qu'une information partielle et locale du graphe de dépendance G. Le *m-agent* $\mathcal{A}_i$ connaît seulement ses ports d'entrée x_i^k, ses ports de sortie y_i^n et les *artefact-de-couplage* $\mathcal{C}_i^j$ depuis lesquels il peut récupérer ses données d'entrée et les *artefact-de-couplage* $\mathcal{C}_{j'}^i$ vers lesquels il peut envoyer ses données de sortie. En aucun cas le *m-agent* $\mathcal{A}_i$ n'a connaissance des ports d'entrée $x_{j'}^{k'}$ et de sortie $y_j^{n'}$ des autres *m-agents* $\mathcal{A}_j$ et $\mathcal{A}_{j'}$.

Ainsi, pour $\mathcal{I}_i$, on note IN_i l'ensemble des liens d'entrée du modèle m_i et OUT_i l'ensemble des liens de sortie du modèle m_i. IN_i comprend les couples (j, k) tels que les données d'entrée fournies par l'*artefact-de-couplage* $\mathcal{C}_i^j$ correspondent au port d'entrée x_i^k. OUT_i comprend les couples (n, j) tels que les données de sortie issues du port de sortie x_i^n doivent être envoyées à l'*artefact-de-couplage* $\mathcal{C}_j^i$.

Chaque *artefact-de-couplage* doit faire le lien entre deux *m-agents* mais aussi faire en sorte qu'une donnée nécessaire à l'exécution du modèle m_i : $\mathsf{d}_{\mathsf{out}_j}^{\,n}(t_j)$ issue du port de sortie y_j^n soit envoyée à m_i au travers de son port d'entrée x_i^k. Un *artefact-de-couplage* $\mathcal{C}_i^j$ possède alors un ensemble de couples (n, k) faisant le lien entre le port de sortie y_j^n et le port d'entrée x_i^k. On note cet ensemble L_i^j. De cette manière, toute l'information du graphe de dépendance est répartie et respecte le méta-modèle AA4MM.

En terme d'interaction entre deux *m-agents* $\mathcal{A}_i$ et $\mathcal{A}_j$, il est également possible de simplifier le méta-modèle. Une interaction est dirigée (par exemple de $\mathcal{A}_i$ vers $\mathcal{A}_j$) et comprend une suite d'opérations au niveau sémantique, syntaxique, dynamique, et technique. Nous plaçons dans le cas simple mais relativement générique où chaque interaction (chaque lien du graphe de dépendance) n'implique que deux *m-agents*, nous remplaçons l'ensemble des artefacts définis dans chaque niveau par un seul plus générique appelé *artefact-de-couplage* et noté $\mathcal{C}_j^i$. Ce dernier se doit de remplir les mêmes fonctions que ceux définis précédemment (voir figure 8.11).

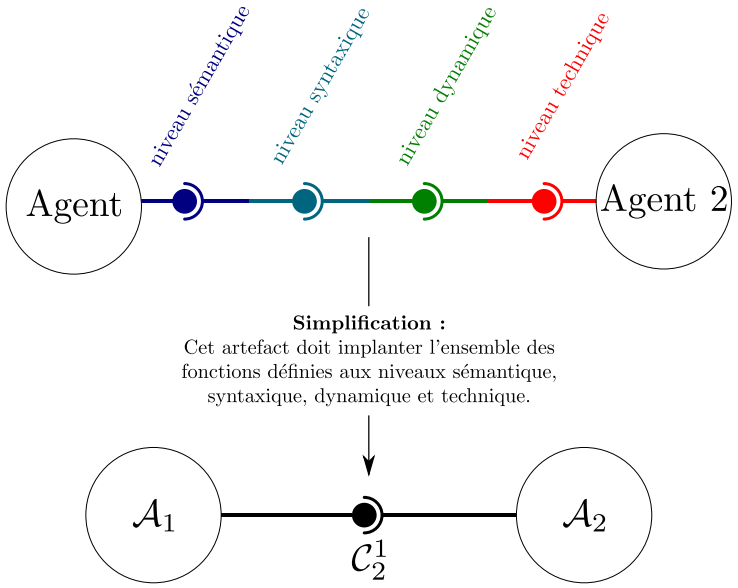


FIGURE 8.11 – Il est possible de simplifier le schéma en remplaçant les quatre artefacts gérant les interactions par un seul. Cet artefact plus générique doit permettre les mêmes actions que les quatre précédentes.

D'une manière pratique, on peut implanter chaque artefact dans un composant logiciel et lier les artefacts les uns à la suite des autres. Le couplage d'artefacts est prévu initialement et décrit formellement dans le méta-modèle A&A [Ricci et al., 2007a]. C'est d'ailleurs la meilleure solution en terme d'ingénierie logicielle pour respecter notre contrainte de modularité. La manière dont sont couplés les artefacts entre eux n'entre pas dans l'objet de ce manuscrit de thèse. Nous nous contenterons de spécifier les différents types d'entités : *m-agents*, *artefacts-d'interface* et *artefacts-de-couplage*.

Niveaux	**Description**	**Notation**
conceptuel	*m-agent*	$\mathcal{A}_i$
	artefact-d'interface	$\mathcal{I}_i$
	artefact-de-couplage entre $\mathcal{A}_i$ et $\mathcal{A}_j$	$\mathcal{C}_j^i$
sémantique	modèle	m_i
	liens d'entrée du modèle m_i	IN_i
	liens de sortie du modèle m_i	OUT_i
	liens entre la sortie du modèle m_j et l'entrée du modèle m_i	L_i^j
syntaxique	modèle m_i formalisé	mf_i
	ensemble des ports d'entrée de m_i	X_i
	ensemble des ports de sortie de m_i	Y_i
	k$^{\text{ième}}$ port d'entrée	x_i^k $(0 < k \leq \text{card}(X_i))$
	n$^{\text{ième}}$ port de sortie	y_i^n $(0 < n \leq \text{card}(Y_i))$
dynamique	simulateur abstrait	s_i
	temps de simulation	t_i
	donnée d'entrée (port x_i^k)	$\mathsf{d}_{\mathsf{in}_i^k}(t_i)$
	donnée de sortie (port y_i^n)	$\mathsf{d}_{\mathsf{out}_i^n}(t_i)$
technique	logiciel de simulation	ls_i

FIGURE 8.12 – Ensemble des notations utilisées

8.4.2 Exemple de multi-modèle

Nous présentons un exemple de multi-modèle relativement simple pour faciliter l'illustration de notre méta-modèle (voir figure 8.13). Cet exemple comprend deux logiciels de simulation distincts (ls_1 et ls_2). Chaque modèle ne possède qu'un seul port d'entrée et un seul port de sortie (x_1^1 et y_1^1 pour m_1 et x_2^1 et y_2^1 pour m_2). Pour cette raison, nous simplifions la notation et nous les notons x_1 et y_1 pour m_1 et x_2 et y_2 pour m_2. Chaque modèle lors de son exécution produit une donnée notée $\mathsf{d}_{\mathsf{out}1}{}^1(t_1)$ pour m_1 et $\mathsf{d}_{\mathsf{out}2}{}^1(t_2)$ pour m_2 que l'on simplifie en $\mathsf{d}_{\mathsf{out}1}$ et $\mathsf{d}_{\mathsf{out}2}$. Le graphe de dépendance entre les modèles m_1 et m_2 est le suivant : le modèle m_1 fourni la donnée $\mathsf{d}_{\mathsf{out}1}$ au modèle m_2 et le modèle m_2 fourni la donnée $\mathsf{d}_{\mathsf{out}2}$ au modèle m_1. On note alors $\mathsf{IN}_1 = (2,1)$, $\mathsf{OUT}_1 = (1,2)$, $\mathsf{IN}_2 = (1,1)$, $\mathsf{OUT}_2 = (1,1)$, $\mathsf{L}_2^1 = (1,1)$ et $\mathsf{L}_1^2(1,1)$.

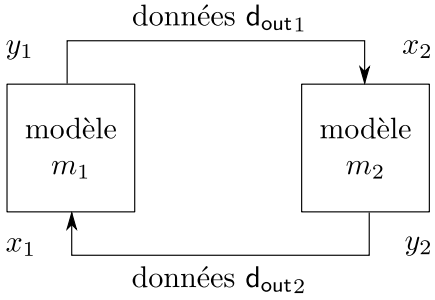


FIGURE 8.13 – Exemple d'un multi-modèle composé de deux modèles distincts m_1 et m_2 et dont le graphe de dépendance implique un couplage fort.

Le système multi-agent de gestion de la multi-modélisation comprend deux *m-agents* notés $\mathcal{A}_1$ et $\mathcal{A}_2$, deux *artefact-d'interface* notés $\mathcal{I}_1$ et $\mathcal{I}_2$ et deux *artefact-de-couplage* notées $\mathcal{C}_2^1$ (de $\mathcal{A}_1$ vers $\mathcal{A}_2$) et $\mathcal{C}_1^2$ (de $\mathcal{A}_2$ vers $\mathcal{A}_1$). La figure 8.14 présente le schéma simplifié obtenu.

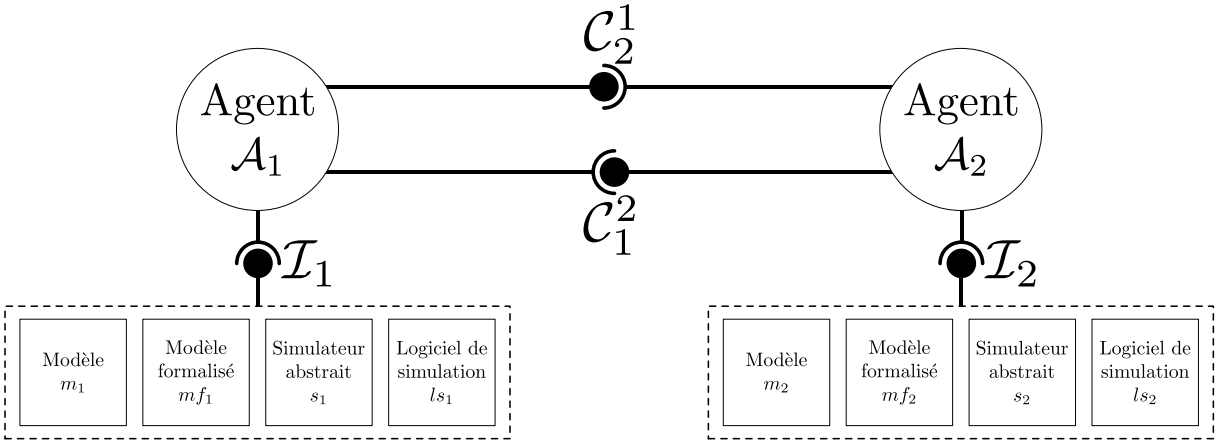


FIGURE 8.14 – Exemple précédent. Nous présentons ici les *m-agents* et les artefacts utilisés de manière simplifiée. Ce schéma comprend aussi les notations.

8.4.3 Gestion de l'interfaçage : l'*artefact-d'interface*

L'*artefact-d'interface* met à disposition du *m-agent* un ensemble d'opérations et d'informations pour que ce dernier puisse interagir avec les concepts manipulés (modèle, modèle formalisé, simulateur abstrait et logiciel de simulation). Celui-ci est noté $\mathcal{I}_i$. L'ensemble des opérations qu'autorise cet artefact est décrit par la figure 8.15.

Opérations	**Rôles**
init()	Instancie le logiciel de simulation ls_i, initialise le simulateur s_i et paramètre le modèle m_i
run(qt)	Exécute le simulateur s_i pour certaine quantité de temps de simulation qt
getCurrentTime()	Retourne le temps courant de simulation t_i du simulateur s_i
getOutputData(y_i^n)	Retourne la donnée $\mathsf{d}_{\mathsf{out}_i}^{n}(t_i)$ présente dans le port de sortie y_i^n du modèle m_i
setInputData(x_i^k, $\mathsf{d}_{\mathsf{out}_j}^{k'}(t_j)$)	Paramètre le modèle m_i avec la donnée $\mathsf{d}_{\mathsf{out}_i}^{k'}(t_j)$
stop()	Gère la fin de la simulation (stockage des données, analyse, *etc.*)

FIGURE 8.15 – Opérations permises par l'*artefact-d'interface*

Il est à noter que ces opérations sont le minimum nécessaire pour interfacer un logiciel de simulation. Cela permet à la fois de faciliter la réutilisation d'outils de modélisation et de simulation existants mais aussi de pouvoir utiliser ceux-ci soit de manière couplée soit de manière isolée avec un minimum de modifications.

Initialisation : opération init()

L'opération init() est la toute première opération que doit effectuer un *m-agent*. Le but de cette opération est d'instancier le logiciel de simulation ls_i, d'initialiser le simulateur s_i et de paramétrer le modèle m_i. La phase d'initialisation et d'instanciation ne pose pas de problème en soi. Cela peut se faire soit en instanciant le logiciel de simulation et en chargeant le modèle depuis une bibliothèque (comme dans le cas de NetLogo par exemple), soit, s'il est directement codé dans le logiciel de simulation, en instanciant ce dernier.

La configuration du modèle peut se faire directement, si les paramètres sont donnés à l'avance ou indirectement, si les paramètres sont récupérés depuis un autre modèle. Dans ce cas, il faut que les autres *m-agents* aient partagé les données initiales qui servent de paramètres (voir figure 8.16).

> *Nous appelons* données initiales *les données* $\forall n \in [1, \text{card}(X_i)]$ $\mathsf{d}_{\mathsf{out}_i}^{n}(t_i^\bullet)$ *qui sont produites par un modèle* m_i *au temps de simulation initial noté* $t_i^\bullet$ *avant la phase d'exécution* [a].
>
> a. la plupart du temps $t_i^\bullet = 0$

La phase d'initialisation pose un problème : la possibilité d'introduire une situation bloquante lors de l'initialisation. Reprenons notre exemple. On voit qu'il peut y avoir une situation de blocage si, pour être paramétré, le modèle m_1 attend les données initiales provenant de m_2 et si à l'opposé, m_2 attend les données initiales provenant de m_1. Ce genre de situation d'initialisation bloquante est dépendante à la fois du graphe d'interaction et du système étudié. Il n'est pas possible de débloquer la situation automatiquement sans remettre en cause la sémantique du multi-modèle. C'est le rôle du modélisation de gérer la situation d'initialisation. La seule chose que peut faire le méta-modèle est de s'apercevoir de la situation de blocage et de faire remonter cette information au modélisateur.

Il est important de noter que cette situation bloquante est propre à la phase d'initialisation. Pour l'exécution du multi-modèle, nous montrerons que ce genre de situation bloquante ne peut arriver (voir le détail au chapitre suivant).

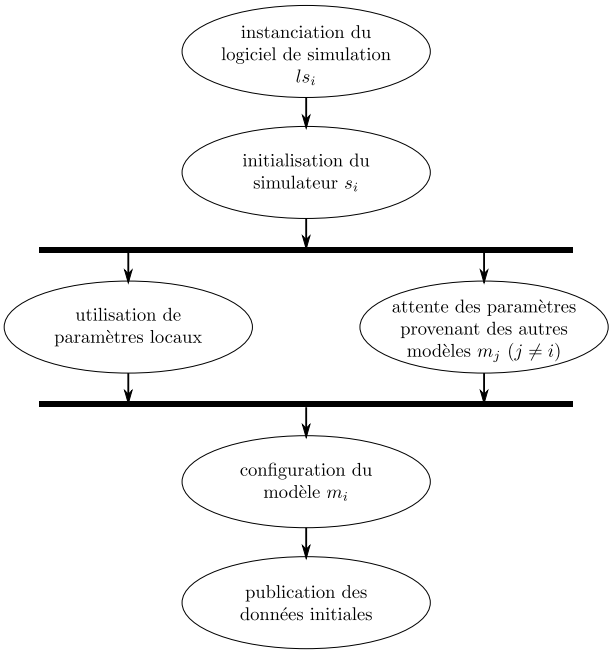


FIGURE 8.16 – Opération init() : diagramme d'activité.

Exécution du modèle m_i par le simulateur abstrait s_i : opération run(qt)

L'opération run(qt) permet à le *m-agent* de demander à son simulateur abstrait l'exécution du modèle et de faire avancer le temps local de simulation. Ces deux actions vont dépendre explicitement de la politique d'exécution utilisée par le simulateur abstrait. Le paramètre qt sert à préciser la quantité de temps de simulation que le simulateur peut avancer. Le modèle est exécuté jusqu'à ce que t_i prenne la valeur de $t_i + qt$.

Par convention, l'opération run() *(sans paramètre qt) provoque le comportement suivant :*
- *Si le simulateur utilise une politique par pas de temps, l'opération run() doit permettre d'exécuter le prochain pas de temps de simulation.*
- *Si le simulateur utilise une politique par évènements discrets, l'opération run() doit permettre d'exécuter le prochain événement de simulation.*

Récupération du temps courant de simulation. Opération getCurrentTime()

L'opération getCurrentTime() doit retourner à le *m-agent* la valeur du temps courant de simulation de son simulateur abstrait s_i.

Flux des données en entrée et en sortie. Opérations setInputData(x_i, $\mathsf{d}_{\mathsf{out}\,j}$) et getOutputData($y_i$)

L'opération setInputData(x_i^k, ${\mathsf{d}_{\mathsf{out}}}_j^{k'}(t_j)$) a pour but de permettre à le *m-agent* de paramétrer le modèle m_i avec la donnée ${\mathsf{d}_{\mathsf{out}}}_j^{k'}(t_j)$. C'est-à-dire envoyer la donnée ${\mathsf{d}_{\mathsf{out}}}_j^{k'}(t_j)$ au port d'entrée $x_i^{k'}$ du modèle m_i correspondant (avec $(0 < k \leq \text{card}(X_i))$ et $(0 < k' \leq \text{card}(Y_j))$. On rappel que le lien entre k et k' est donnée par IN_i. L'opération getOutputData(y_i^n) a pour but de récupérer la donnée ${\mathsf{d}_{\mathsf{out}}}_i^n(t_i)$ dans le port de sortie y_i^n du modèle m_i correspondant.

8.4.4 Gestion des interactions : l'*artefact-de-couplage*

Le rôle de l'*artefact-de-couplage* est mettre à disposition des *m-agents* un ensemble d'opérations et d'informations pour que ces derniers puissent interagir et gérer, de manière décentralisée, l'ensemble de la multi-modélisation. L'*artefact-de-couplage* représente un lien du graphe de dépendance du multi-modèle. Il est orienté et nous le notons $\mathcal{C}_j^i$ lorsque, dans le graphe de dépendances, le sens de la flèche va du modèle m_i au modèle m_j. L'ensemble des opérations permises par un *artefact-de-couplage* est résumé par la figure 8.17.

Opérations	**Rôles**
post($\mathsf{d}_{\mathsf{out}_i}^{\ n}(t_i)$*,*$\Gamma_i$*)*	Envoie les données à échanger $\mathsf{d}_{\mathsf{out}_i}^{\ n}$ associées à l'intervalle de validité Γ_i
read(t_i*, k)*	Retourne les données $\mathsf{d}_{\mathsf{out}_j}^{\ n'}(t_j)$ valides pour le temps courant de simulation t_i (avec $0 < n' \leq \text{card}(Y_j)$). Données qui correspondent au port d'entrée x_i^k.
trans($\mathsf{d}_{\mathsf{out}_i}^{\ n}$*)(*$t_i$*)*	Transforme les données valides $\mathsf{d}_{\mathsf{out}_i}^{\ n}(t_i)$ en données transformées $\mathsf{d}^*_{\mathsf{out}_i}^{\ n}(t_i)$

FIGURE 8.17 – Opérations autorisées par l'*artefact-de-couplage*.

Écrire des données à échanger. Opération post($\mathsf{d}_{\mathsf{out}_i}$,$\Gamma_i$)

L'*artefact-de-couplage* doit permettre aux *m-agents* d'écrire des données afin que celles ci puissent être échangées. Tel est le but de l'opération *post(*$\mathsf{d}_{\mathsf{out}_i}$,$\Gamma_i$*)*. Il est à noter que le méta-modèle AA4MM spécifie qu'au niveau dynamique, les données échangées $\mathsf{d}_{\mathsf{out}_i}$ doivent être associées à un intervalle de temps de simulation noté Γ_i (plus de détails au chapitre suivant).

Lire des données valides. Opération read(t_i, k)

Lorsqu'un *m-agent* cherche à obtenir des données, le méta-modèle AA4MM spécifie que ces données doivent être valides. Par valide, nous entendons que l'utilisation de ces données par le *m-agent* ne remet pas en cause le principe de causalité. L'opération *read(*t_i*, k)* doit donc retourner l'ensemble des données valides pour le temps t_i (comme précédemment, plus de détails au chapitre suivant). Le paramètre k signifie juste que cette donnée doit être transmise au modèle m_i via le port d'entrée x_i^k. Par convention s'il n'existe qu'un seul port d'entrée (comme dans notre exemple) nous omettons le paramètre k.

Traitement des données

Au niveau sémantique et au niveau syntaxique, les données échangées peuvent subir des opérations. Des opérations sémantiques sont proposées dans [Bonneaud, 2008], comme par exemple la discrétisation, le changement de dimension, *etc.*. Nous reprenons le même principe et nous y ajoutons les opérations nécessaires au changement de formalisme. Ces opérations étant dépendantes de la simulation voulue, on se contentera de préciser qu'il est possible de définir de nouvelles opérations permettant ce traitement des données. Nous présentons des exemples d'opérations au chapitre 10. D'une manière générale, nous notons *trans(*$\mathsf{d}_{\mathsf{out}_i}^{\ n}(t_i)$*)* l'opération qui transforme un ensemble de données $\mathsf{d}_{\mathsf{out}_i}^{\ n}(t_i)$ en un ensemble de données $\mathsf{d}^*_{\mathsf{out}_i}^{\ n}(t_i)$.

8.4.5 Caractéristiques techniques de l'*artefact-de-couplage*

Accès concurrent

Nous avons fait l'hypothèse que deux *m-agents* peuvent utiliser simultanément un même *artefact-de-couplage*. Les opérations proposées par le *artefact-de-couplage* doivent tenir compte de cette concurrence. Ce sont des opérations dites critiques, dans le sens où elles agissent sur une donnée partagée entre plusieurs *m-agents* : $\mathsf{d}_{\mathsf{out}_i}^{\ n}(t_i)$. Le fait qu'un *m-agent* propose des données - par l'opération post($\mathsf{d}_{\mathsf{out}_i}^{\ n}(t_i)$,$\Gamma_i$) - ne doit pas empêcher un autre *m-agent* de lire des données ou de transformer des données - via les opération read(t_j, k) ou trans($\mathsf{d}_{\mathsf{out}_i}^{\ n}(t_i)$). De même, lorsque plusieurs *m-agents* proposent des données, l'opération post($\mathsf{d}_{\mathsf{out}_i}^{\ n}(t_i)$,$\Gamma_i$) doit gérer l'accès concurrent et éviter toute corruption de données.

Mémoire

Afin de permettre l'interaction asynchrone de plusieurs *m-agents*, mais aussi à des fins de log et d'analyse *a posteriori*, l'*artefact-de-couplage* se doit d'assurer une certaine persistance des données échangées. Cette persistance des données doit durer, au minimum le temps de permettre à tous les *m-agents* d'interagir entre eux et au maximum le temps d'une simulation complète du multi-modèle.

8.4.6 Agents en charge de la multi-modélisation : les *m-agents*

Le rôle des *m-agents* est de gérer le processus complet de multi-modélisation. Dans cette section, nous nous sommes focalisés sur un comportement du *m-agent* réduit à son strict minimum. Non pas, parce que des comportements plus évolués sont difficiles à mettre en place, mais parce que ce comportement typique permet de valider notre approche et aussi d'expliquer le plus clairement possible le fonctionnement du méta-modèle AA4MM. La question de la mise en place automatique d'un multi-modèle par le système multi-agent de gestion de la multi-modélisation n'est pas abordé dans ce travail de recherche. Le comportement d'un *m-agent* peut être décomposé en trois grandes phases : initialisation, exécution et finalisation.

L'initialisation se passe via l'opération init() de l'*artefact-d'interface*. Nous avons vu précédemment que cette opération est cruciale pour la bonne mise en route de la multi-modélisation. Nous ne reviendrons donc pas dessus.

Une fois les composants du multi-modèle mis en place et paramétrés, il est nécessaire d'exécuter l'ensemble du multi-modèle. À ce niveau, chaque *m-agent* doit gérer à la fois le flux de données entrant et sortant de son modèle et l'exécution de son modèle. Le comportement du *m-agent* se déroule ici selon une boucle lecture des données, exécution du modèle et écriture des données. Cette phase du comportement de l'agent est décrite plus en détail dans le chapitre suivant.

Lorsque la simulation du multi-modèle est terminée, l'ensemble des *m-agents* peuvent demander à leurs logiciels de simulation respectif d'appliquer quelques opérations nécessaires à la récupération et la sauvegarde des résultats de simulation, à leur analyse, *etc.* Ces opérations vont dépendre à la fois du problème posé et des capacités des logiciels de simulation. L'*m-agent* utilise alors l'opération stop() du *artefact-d'interface*.

8.5 Synthèse

Dans ce chapitre, nous avons proposé un méta-modèle multi-agent pour coupler des modèles et des simulateurs existants : AA4MM. Le but est de pouvoir créer une "société de modèles" en interaction à partir de logiciels de simulation déjà existants. Ce méta-modèle se base sur le paradigme agents et artefact et le résultat obtenu est un système multi-agent de gestion de la multi-modélisation où chaque agent, appelé *m-agent*, gère un logiciel de simulation, un modèle, un modèle formalisé et un simulateur. Ce méta-modèle définit aussi deux type d'artefacts les *artefacts-d'interface* et les *artefacts-de-couplage* qui permettent aux *m-agents* de gérer à la fois les problèmes liés à la réutilisation de modèles et de simulateurs existants, de modularité et de gérer les problèmes liés à la multi-modélisation et à la simulation distribuée. Il est à noter que AA4MM offre une vue homogène aux aspects liés aux différents niveaux : sémantique, syntaxique, dynamique et technique.

Dans ce chapitre, nous avons proposé les spécifications formelles d'une version simplifiée du méta-modèle. Dans le chapitre suivant, nous décrivons plus en détail l'algorithme de simulation distribué adapté à notre approche. Nous proposons par la suite d'établir des preuves de concepts en appliquant le méta-modèle AA4MM à la fois à un exemple jouet d'une modélisation d'un écosystème composé de loups de moutons et de bergers (voir chapitre 10) et à un cas d'étude dans le domaine des réseaux ambiants (chapitre 11).

Chapitre 9

Gestion des contraintes de causalité : un algorithme de multi-simulation

Sommaire

9.1 Introduction

Dans ce chapitre, nous nous focalisons sur le niveau dynamique de la multi-modélisation. Nous abordons la question : "comment se déroule la coordination des *m-agents* afin que ceux-ci puissent exécuter leurs modèles sans enfreindre la contrainte de causalité ?" Nous proposons un algorithme conservatif pour la multi-simulation pour notre méta-modèle. Cet algorithme est une version adaptée de l'algorithme CMB (présenté au chapitre 2), un algorithme classique de la simulation événementielle distribuée (PDES). L'avantage de notre algorithme consiste en la possibilité de coupler des simulateurs dont les politiques d'exécution sont hétérogènes. De plus, dans le méta-modèle AA4MM, l'exécution des modèles et le respect des contraintes de causalité sont vus comme la coordination des *m-agents*. Cette coordination a lieu au travers des interactions entre *m-agents* au sein de leur environnement : les *artefacts-de-couplage*. Il est donc nécessaire que l'algorithme de simulation distribuée s'intègre dans cette vision agent où aucun contrôle centralisé n'est souhaitable.

9.2 Notions préliminaires

Avant de décrire l'algorithme de simulation que nous proposons, il est nécessaire d'introduire et de rappeler quelques notions ayant trait à la simulation distribuée.

9.2.1 Simulation

Nous commençons par rappeler que la simulation est le processus qui fait changer le modèle d'état en fonction de son état précédent, des paramètres d'entrée du modèle et d'une variable appelée temps de simulation. Nous appellerons ces paramètres l'ensemble des paramètres propres au simulateur abstrait s_i. Nous réutiliserons également la représentation graphique introduite dans le chapitre 2. Pour la suite, il est nécessaire de bien saisir le comportement d'un simulateur abstrait. Prenons un exemple dans lequel un simulateur abstrait s_i est initialisé au temps de simulation $t_i = t_i^{\bullet} = 0$ (voir la figure 9.1). À cet instant, le modèle m_i est dans l'état A. L'exécution du modèle signifie trois choses :

- le changement d'état du modèle m_i, de l'état A celui-ci passe dans l'état B ;
- la production de données dans les ports de sortie y_i du modèle ;
- l'augmentation du temps de simulation, le simulateur abstrait passe d'une valeur de temps de simulation $t_i = 0$ à $t_i = 5$, par exemple.

En d'autres termes, cela signifie que le modèle était dans l'état A au temps $t_i = 0$ et dans l'état B à partir d'un temps $t_i > 0$ jusqu'au temps $t_i = 5$.

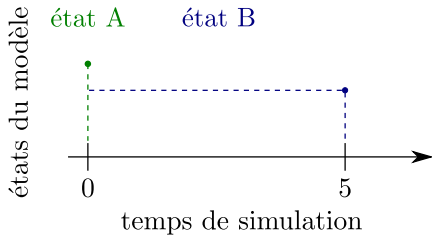


FIGURE 9.1 – Exemple de simulation.

9.2.2 Exécution distribuée et contrainte de causalité

Reprenons l'exemple du multi-modèle du chapitre précédent. Dans ce multi-modèle, deux simulateurs abstraits (s_1 et s_2) doivent exécuter simultanément leurs modèles (respectivement m_1 et m_2). D'après le graphe de dépendance, pour pouvoir exécuter son modèle m_1, le simulateur abstrait s_1 doit prendre en compte à la fois ses paramètres propres (l'état de m_1, le temps de simulation t_1 et un ensemble de paramètres d'entrée) et un certain nombre de données produites par l'exécution du modèle m_2. Autrement dit, le simulateur s_1 dépend également de l'état du modèle m_2 (voir figure 9.2).

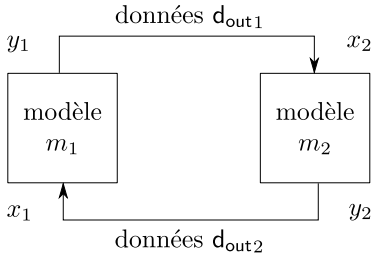
Lorsque plusieurs simulateurs abstraits exécutent leurs modèles respectifs, il faut veiller à ce que la simulation globale respecte la contrainte de causalité (voir section 2.4.3). Pour rappel, respecter la contrainte de causalité, c'est faire en sorte que l'exécution distribuée et parallèle des différents modèles donne les mêmes résultats qu'une exécution centralisée et séquentielle [Fujimoto, 2001]. La question est de savoir quelles sont les conditions qui permettent aux simulateurs abstraits de respecter cette contrainte. De manière plus résumée, on cherche l'ordre dans lequel chaque modèle peut être exécuté.

9.2.3 Notion d'intervalle de validité

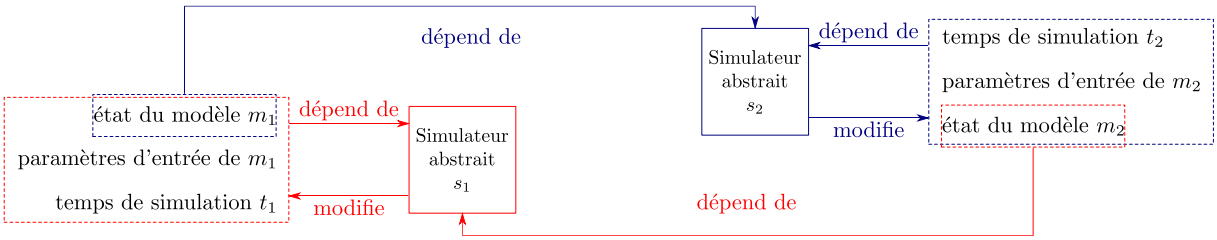
Nous appelons condition de validité, la condition qui permet à un simulateur s_i de respecter la contrainte de causalité. Cette dernière repose sur une notion particulière que nous appelons intervalle de validité et notée Γ_i. Cette notion est directement inspirée de la notion de *lookahead* dans le domaine de la simulation événementielle distribuée.

> *L'*intervalle de validité Γ_i *est un intervalle de temps de simulation pendant lequel un simulateur s_i garantit qu'il ne modifie pas l'état du modèle m_i et donc ne produit pas de nouvelles données.*

L'intervalle de validité prend la forme $\Gamma_i =]ct_i, nt_i]$, avec ct_i et nt_i des valeurs de temps de simulation (avec $ct_i \leq nt_i$).



(a) Rappel : graphe de dépendance des modèles.



(b) Chaque simulateur abstrait dépend à la fois de son propre modèle et du modèle exécuté par l'autre simulateur abstrait.

FIGURE 9.2 – Exemple d'un multi-modèle composé de deux modèles. Graphe de dépendance des modèles et représentation des dépendances au niveau dynamique.

On dit qu'une donnée issue du modèle m_j au temps t_j (notée $\mathsf{d}_{\text{out}_j}^n(t_j)$) est valide pour le simulateur abstrait s_i si et seulement si le simulateur s_i peut utiliser la donnée $\mathsf{d}_{\text{out}_j}^n(t_j)$ pour exécuter son propre modèle m_i sans enfreindre la contrainte de causalité.

Une donnée $\mathsf{d}_{\text{out}_j}^n(t_j)$ est valide pour un simulateur abstrait s_i au temps de simulation t_i *si et seulement si :*

– $\mathsf{d}_{\text{out}_j}^n(t_j)$ *a été produite avant* t_i *(dans le référentiel du temps de simulation) :*

$$t_j < t_i$$

– *la donnée produite par* s_j directement après [a] $\mathsf{d}_{\text{out}_j}^n(t_j)$ ($\mathsf{d}_{\text{out}_j}^n(t_{j'})$ *avec* $t'_j > t_j$) *correspond à un temps de simulation supérieur ou égal à* t_i *:*

$$t_i \leq t'_j$$

*On note alors l'*intervalle de validité *associé à la donnée* $\mathsf{d}_{\text{out}_j}^n(t_i)$ $\Gamma_j =]t_j, t'_j]$.

a. Que la donnée $\mathsf{d}_{\text{out}_j}^n(t_{j'})$ soit produite directement après $\mathsf{d}_{\text{out}_j}^n(t_j)$ signifie qu'il n'existe aucune donnée $\mathsf{d}_{\text{out}_j}^n(t_{j''})$ tel que $t_j < t''_j < t'_j$.

On peut résumer la condition de validité comme suit. La donnée $\mathsf{d}_{\text{out}_j}^n(t_j)$ associée à l'intervalle de validité Γ_j est valide pour un simulateur abstrait s_i au temps de simulation t_i si et seulement si $t_i \in \Gamma_j$. Il s'en suit qu'un modèle m_i peut être exécuté par son simulateur abstrait s_i sans violer la contrainte de causalité si toutes les données dont dépend son exécution sont valides.

9.3 Algorithme de simulation

9.3.1 Principe

Afin d'exécuter leurs modèles tout en respectant la contrainte de causalité, les *m-agents* échangent les issues des modèles associées à l'intervalle de validité correspondant. Nous considérons qu'à chaque

exécution du modèle m_i, le simulateur abstrait s_i produit des données de sorties notées $\mathsf{d_{out}}_i^n(t_i)$ (qui peut prendre la valeur *NULL* si aucune donnée n'est produite à ce moment). Celle-ci est associée à l'intervalle de validité Γ_i par le *m-agent* $\mathcal{A}_i$. Ensuite ce dernier fournit le couple donnée, intervalle de validité $(\mathsf{d_{out}}_i^n(t_i), \Gamma_i)$ aux *artefacts-de-couplage*. Enfin, les autres *m-agents* qui ont besoin de cette donnée pour faire exécuter leurs modèles respectifs, demandent aux *artefacts-de-couplage* de leurs fournir les données qui leurs sont valides. Un *m-agent* $\mathcal{A}_j$ peut récupérer une donnée valide $\mathsf{d_{out}}_i^n(t_i)$ si $t_j \in \Gamma_i$. La figure 9.3 présente le principe de ce mécanisme de coordination.

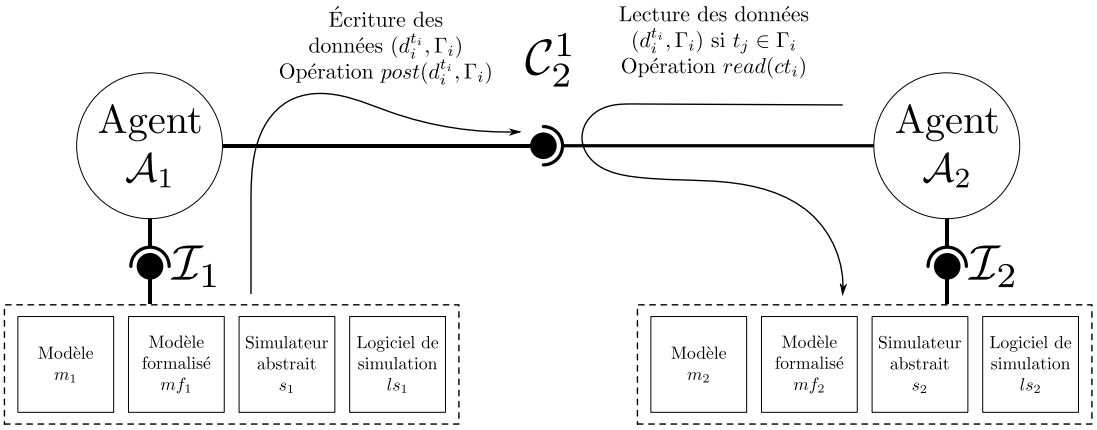


FIGURE 9.3 – Représentation schématique du mécanisme de coordination des *m-agents*.

9.3.2 Algorithme de simulation

L'algorithme de simulation décentralisé que nous proposons correspond au comportement du *m-agent* durant la phase de simulation du modèle. L'algorithme complet est décrit ci-après (voir l'algorithme 1).

Durant la première étape, le *m-agent* doit d'abord récupérer le temps courant de simulation via l'opération getCurentTime() de son *artefact-d'interface* (ligne 2). Ensuite, le *m-agent* doit récupérer les données qui sont nécessaires à l'exécution de son modèle (des lignes 3 à 6). Ces données doivent également ne pas remettre en cause la contrainte de causalité. Cela se passe via l'opération read(t_i, k) de l'*artefact-de-couplage*. Il est à noter qu'à ce moment, le *m-agent* peut effectuer des opérations au niveau sémantique et syntaxique sur les données récupérées si celles-ci ne conviennent pas (par exemple changement de formalisme, discrétisation, *etc.*). Cela se passe via l'opération trans($\mathsf{d_{out}}_i^n(t_i)$) de l'*artefact-de-couplage* (celle-ci n'est pas décrite dans l'algorithme 1).

La deuxième étape est l'exécution du modèle. Une fois l'ensemble des données récupérées, le *m-agent* paramètre son modèle et demande au simulateur abstrait d'exécuter son modèle pour une certaine quantité de temps de simulation qt (lignes 7 et 8). Le choix de ce paramètre va dépendre explicitement du problème étudié, du graphe de dépendance et des politiques d'exécution de chacun des simulateurs abstraits. Ici, nous présentons la version la plus générique, à savoir que l'exécution du modèle se fait pour une quantité minimale de temps de simulation $qt = 1$. Le *m-agent* exécute son modèle via l'opération run() du *artefact-d'interface*.

Durant la troisième étape, le *m-agent* récupère les données produites par le modèle lors de son exécution. Il calcul l'intervalle de validité Γ et envoie l'ensemble à l'*artefact-de-couplage* afin d'échanger ces données. Cela se déroule via l'opération post($\mathsf{d_{out}}_i^n(t_i)$,$\Gamma_i$) de l'*artefact-de-couplage* (des lignes 9 à 16).

9.3.3 Initialisation

Les données échangées $\mathsf{d_{out}}_i^n(t_i)$ sont associées à un intervalle de temps de simulation appelé intervalle de validité et noté Γ_i. Au moment de l'initialisation, il n'existe qu'une seule valeur de temps de simulation : $t_i^\bullet$. Il n'est donc pas possible de créer un intervalle de temps de simulation Γ_i tel quel car il manque une

Algorithme 1 – Algorithme de simulation distribuée du méta-modèle AA4MM.

COMPORTEMENT(IN_i, OUT_i)
Description: comportement du *m-agent* $\mathcal{A}_i$ durant la phase d'exécution de son propre modèle m_i
Entrée: IN_i, OUT_i
Sortie:

(1) **tant que** simulation non finie
(2) $ct_i \leftarrow \mathcal{I}_i$.getCurrentTime()
(3) $\forall (j,k) \in \mathsf{IN}_i : {\mathsf{d}_{\mathsf{in}}}_i^k(ct_i) \leftarrow \varnothing$
(4) **tant que** $\exists (j,k) \in \mathsf{IN}_i : {\mathsf{d}_{\mathsf{in}}}_i^k(ct_i) = \varnothing$
(5) ${\mathsf{d}_{\mathsf{in}}}_i^k(ct_i) \leftarrow \mathcal{C}_i^j$.read($ct_i$, k)
(6) **fin tant que**
(7) $\forall (j,k) \in \mathsf{IN}_i : \mathcal{I}_i$.setInputData($x_i^k$, ${\mathsf{d}_{\mathsf{in}}}_i^k(ct_i)$)
(8) $\mathcal{I}_i$.run()
(9) $nt_i \leftarrow \mathcal{I}_i$.getCurrentTime()
(10) $\Gamma_i \leftarrow]ct_i, nt_i]$
(11) **pour tout** $n' \in [1, \text{card}(Y_i)]$
(12) ${\mathsf{d}_{\mathsf{out}}}_i^{n'} \leftarrow \mathcal{I}_i$.getOutputData($y_i^{n'}$)
(13) **pour tout** $(n', j') \in \mathsf{OUT}_i$
(14) $\mathcal{C}_{j'}^i$.post(${\mathsf{d}_{\mathsf{out}}}_i^{n'}(ct_i)$,$\Gamma_i$)
(15) **fin pour**
(16) **fin pour**
(17) **fin tant que**

$\mathcal{C}_i^j$.READ(ct_i, k)
Description: Opération read(ct_i, k) de l'*artefact-de-couplage* $\mathcal{C}_i^j$
Entrée: L_i^j, ct_i le temps courant de simulation du *m-agent* $\mathcal{A}_i$
Sortie: ${\mathsf{d}_{\mathsf{out}}}_j^n$ valide ou $\varnothing$ sinon

(1) **pour tout** $({\mathsf{d}_{\mathsf{out}}}_j^n(t_j)$,$\Gamma_j)$ tel que $(j,k) \in \mathsf{L}_i^j$
(2) **si** $ct_i \in \Gamma_j$
(3) **alors retourne** ${\mathsf{d}_{\mathsf{out}}}_j^n(t_j)$
(4) **sinon retourne** $\varnothing$
(5) **fin pour**

des deux bornes. À l'initialisation, nous proposons d'utiliser l'intervalle de validité initial suivant :

$$\Gamma_i^\bullet =]-\infty, t_i^\bullet]$$

En réalité la borne inférieure de $\Gamma_i^\bullet$ peut être remplacée par toute valeur strictement inférieure au $\min_i(t_i^\bullet)$. Cette condition et le fait d'éviter une situation bloquante à l'initialisation sont suffisantes pour démarrer l'exécution du multi-modèle. La preuve de cette assertion est donnée en section 9.4.

9.3.4 Exemple

Afin d'illustrer le fonctionnement de l'algorithme de simulation décentralisé que nous proposons, nous reprenons l'exemple du multi-modèle composé de deux modèles m_1 et m_2 déjà présenté précédemment.

Tout d'abord, il faut procéder à l'initialisation. Chaque *m-agent* doit publier les données initiales de son modèle dans les *artefact-de-couplage* concernés. Dans notre exemple, le *m-agent* $\mathcal{A}_1$ doit récupérer le couple $(\mathsf{d}_{\mathsf{out}1}(t_1^\bullet),]-\infty, 0])$ et le faire stocker par l'*artefact-de-couplage* $\mathcal{C}_2^1$. Le *m-agent* $\mathcal{A}_2$ doit, quant à lui, récupérer le couple $(\mathsf{d}_{\mathsf{out}2}(t_2^\bullet),]-\infty, 0])$ et le faire stocker par l'*artefact-de-couplage* $\mathcal{C}_1^2$ (voir la figure 9.4).

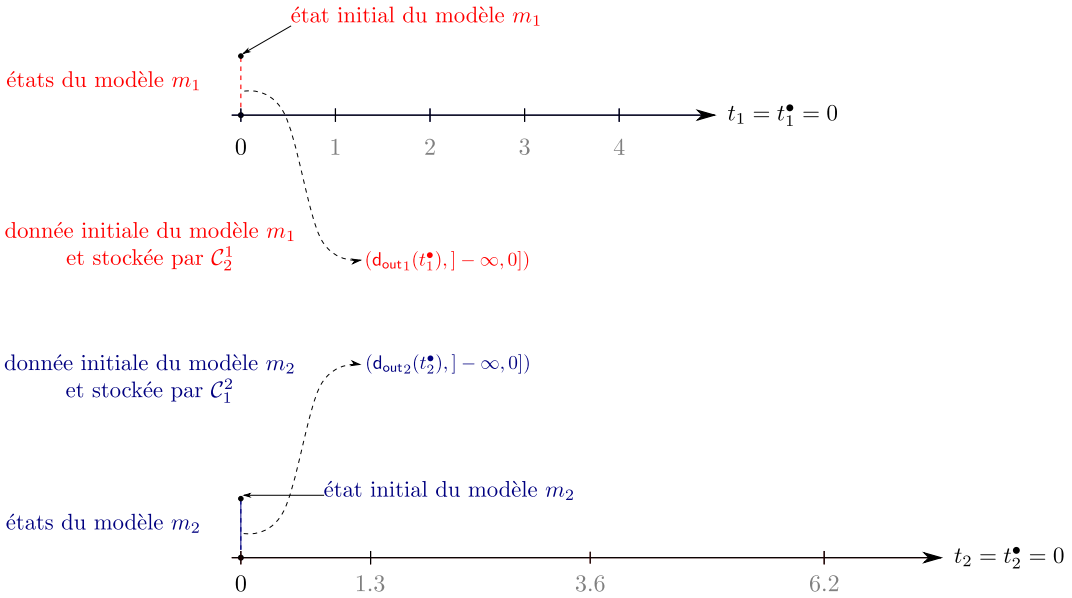


FIGURE 9.4 – Initialisation : chaque *m-agent* doit publier les données initiales de son modèle. $(\mathsf{d}_{\mathsf{out}1}(t_1^\bullet),]-\infty, 0])$ pour le *m-agent* $\mathcal{A}_1$ et $(\mathsf{d}_{\mathsf{out}2}(t_2^\bullet),]-\infty, 0])$ pour le *m-agent* $\mathcal{A}_2$

Après l'initialisation, chaque *m-agent* entre dans la phase de simulation. Nous prenons ici le cas du *m-agent* $\mathcal{A}_1$. Il est à noter que l'ordre d'exécution n'importe pas. Le *m-agent* $\mathcal{A}_1$ doit d'abord lire les données nécessaires à l'exécution de son modèle et paramétrer ce dernier.

- Lecture : $\mathcal{C}_1^2$.read(0) $= (\mathsf{d}_{\mathsf{out}2}(t_2^\bullet),]-\infty, 0])$ car $t_1 = 0 \in \Gamma_2 =]-\infty, 0]$.
- Paramétrage : $\mathcal{I}_1$.setInputData($\mathsf{d}_{\mathsf{out}2}(t_2^\bullet)$).

Une fois le modèle m_1 paramétré, le *m-agent* $\mathcal{A}_1$ peut faire exécuter son modèle.

- Exécution : $\mathcal{I}_1$.run()
- Remarque : le temps de simulation augmente. $t_1 \leftarrow 1$

Le modèle m_1 exécuté, le *m-agent* $\mathcal{A}_1$ doit récupérer des données de sortie, calculer de l'intervalle de validité Γ_1 et écrire ceux-ci dans l'*artefact-de-couplage* $\mathcal{C}_2^1$.

- Calcul de l'intervalle de validité : $\Gamma_1 \leftarrow]0, 1]$
- Récupération de la donnée de sortie : $\mathsf{d}_{\mathsf{out}1}(0) \leftarrow \mathcal{I}_1$.getOutputData()

– Écriture du couple : $\mathcal{C}_2^1$.post($\mathsf{d}_{\mathsf{out}1}(0)$,$\Gamma_1$)

La figure suivante illustre l'exécution du modèle m_1 par son *m-agent* $\mathcal{A}_1$ (voir figure 9.5).

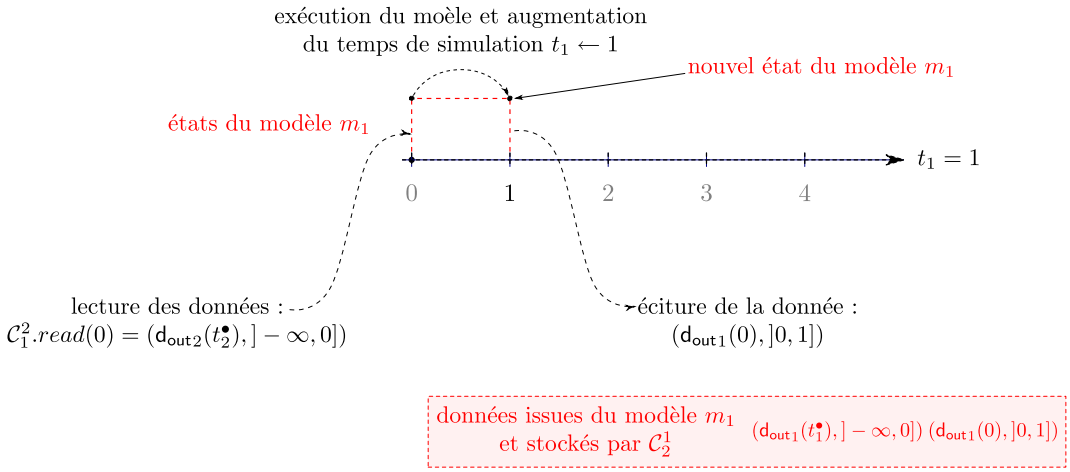


FIGURE 9.5 – Le *m-agent* $\mathcal{A}_1$ exécute son modèle m_1.

De la même manière le *m-agent* $\mathcal{A}_2$ peut faire exécuter son modèle m_2. La simulation se retrouve dans l'état suivant (voir la figure 9.6) : $t_1 = 1$, $t_2 = 1.3$. L'*artefact-de-couplage* $\mathcal{C}_2^1$ stocke les données $(\mathsf{d}_{\mathsf{out}1}(t_1^\bullet),]\infty, 0])$ et $(\mathsf{d}_{\mathsf{out}1}(0),]0, 1])$ respectivement la donnée initiale de m_1 et celle produite à la première exécution. L'*artefact-de-couplage* $\mathcal{C}_1^2$ stocke quant à lui les données $(\mathsf{d}_{\mathsf{out}2}(t_2^\bullet),]\infty, 0])$ et $(\mathsf{d}_{\mathsf{out}2}(0),]0, 1.3])$ respectivement la donnée initiale de m_2 et celle produite à la première exécution.

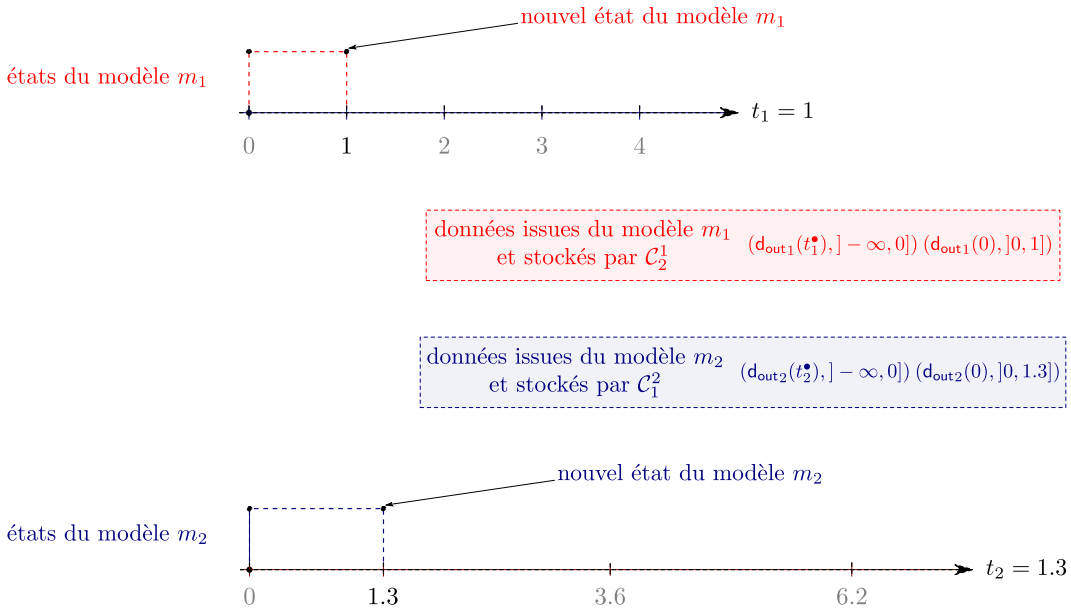


FIGURE 9.6 – Les *m-agents* $\mathcal{A}_1$ et $\mathcal{A}_2$ ont exécuté leurs modèles respectifs m_1 et m_2.

Prenons maintenant le *m-agent* $\mathcal{A}_2$. Celui-ci recommence le cycle de simulation. Il doit tout d'abord lire les données nécessaires à l'exécution de son modèle m_2. Il utilise pour cela l'opération de lecture de l'*artefact-de-couplage* $\mathcal{C}_2^1$: $\mathcal{C}_2^1$.read(1.3). Cette opération lui retourne comme valeur $\varnothing$. En effet, l'*artefact-de-couplage* $\mathcal{C}_2^1$ compare tout d'abord $t_2 = 1.3$ avec $\Gamma_1 =]-\infty, 0]$. La condition de validité n'est pas

remplie. L'*artefact-de-couplage* compare ensuite $t_2 = 1.3$ avec $\Gamma_1 =]0, 1]$. Là encore, la condition de validité n'est pas remplie. Aucune donnée valide n'est disponible pour le *m-agent* $\mathcal{A}_2$. Celui-ci doit donc attendre que l'autre *m-agent* produise de nouvelles données $\mathsf{d}_{\mathsf{out}1}$.

Le *m-agent* $\mathcal{A}_1$ quant à lui n'a pas de problème pour faire exécuter son modèle. En effet, la donnée $\mathsf{d}_{\mathsf{out}2}(0)$ associée à l'intervalle $\Gamma_2 =]0, 1.3]$ est valide pour le temps de simulation $t_1 = 1$. Il peut donc lire la donnée nécessaire à l'exécution de son modèle m_1, paramétrer celui-ci avec, exécuter m_1 et récupérer la donnée de sortie $\mathsf{d}_{\mathsf{out}1}(1)$ associée à l'intervalle $\Gamma_1 =]1, 2]$. Nous nous trouvons alors dans l'état décrit par la figure 9.7. Dans cet état, c'est le *m-agent* $\mathcal{A}_1$ qui doit attendre la production d'une donnée valide et le *m-agent* $\mathcal{A}_2$ peut alors exécuter son modèle. La figure 9.8 retrace l'ensemble des exécutions jusqu'aux temps de simulation $t_1 = 4$ et $t_2 = 6.2$.

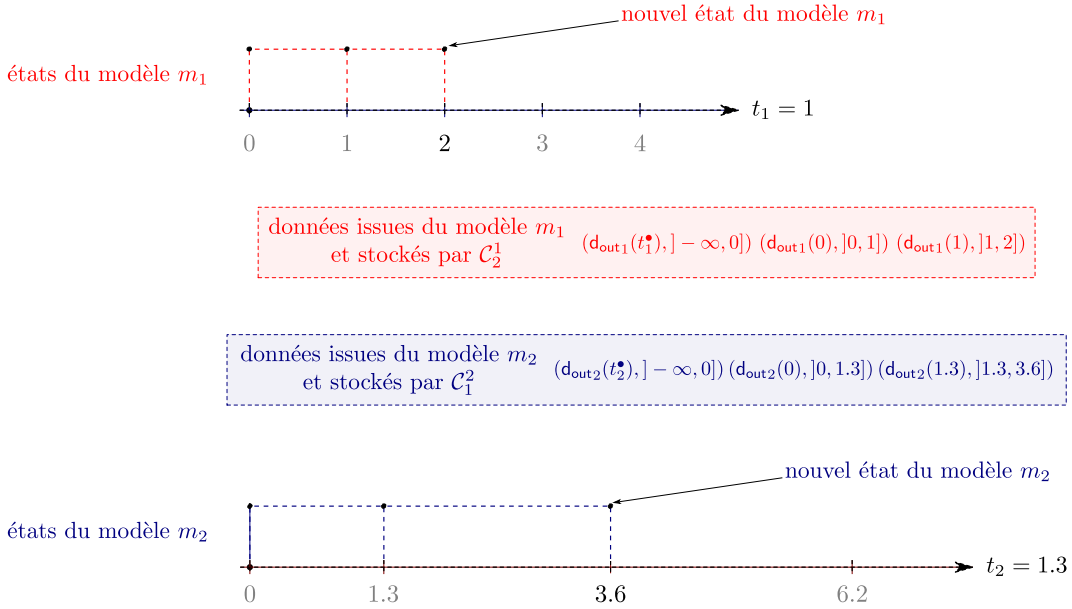


FIGURE 9.7 – Le *m-agent* $\mathcal{A}_2$ à pu exécuter son modèle m_2 seulement après que le *m-agent* $\mathcal{A}_1$ ait exécuté son modèle m_1 et produit une donnée valide $\mathsf{d}_{\mathsf{out}1}(1)$.

9.4 Propriétés et preuves

Cette section a pour but d'exposer les preuves formelles associées à l'algorithme de simulation décentralisé. Nous démontrons d'abord que l'utilisation d'un intervalle de validité initial de la forme $\Gamma_i^\bullet =]infty, t_i^\bullet]$ est une condition suffisante pour faire démarrer l'algorithme de simulation. Ensuite, nous montrons que cette algorithme ne provoque pas d'attente infinie (*deadlock*). Nous faisons l'hypothèse que la communication entre les différents composants du multi-modèle est fiable. Le fait que le processus de communication entre les différents composants puisse interférer dans la bonne marche de la simulation est une autre problématique qui ne fait pas partie de l'objet de cette thèse.

9.4.1 Intervalle de validité initial

Dans cette section, nous montrons que l'utilisation d'un intervalle de validité initial $\Gamma_i^\bullet =]-\infty, t_i^\bullet]$ et le fait d'éviter une situation bloquante à l'initialisation sont une condition suffisante pour que l'algorithme de simulation démarre.

Prenons $m_1, m_2 \ldots m_n$, un ensemble fini de modèles. Les temps initiaux $t_1^\bullet, t_2^\bullet \ldots t_n^\bullet$ sont tous différents deux à deux. On pose $t_i^\bullet = \min_k t_k^\bullet$ (avec i et $k \in [0, n]$). On pose que tous les modèles sont couplés deux

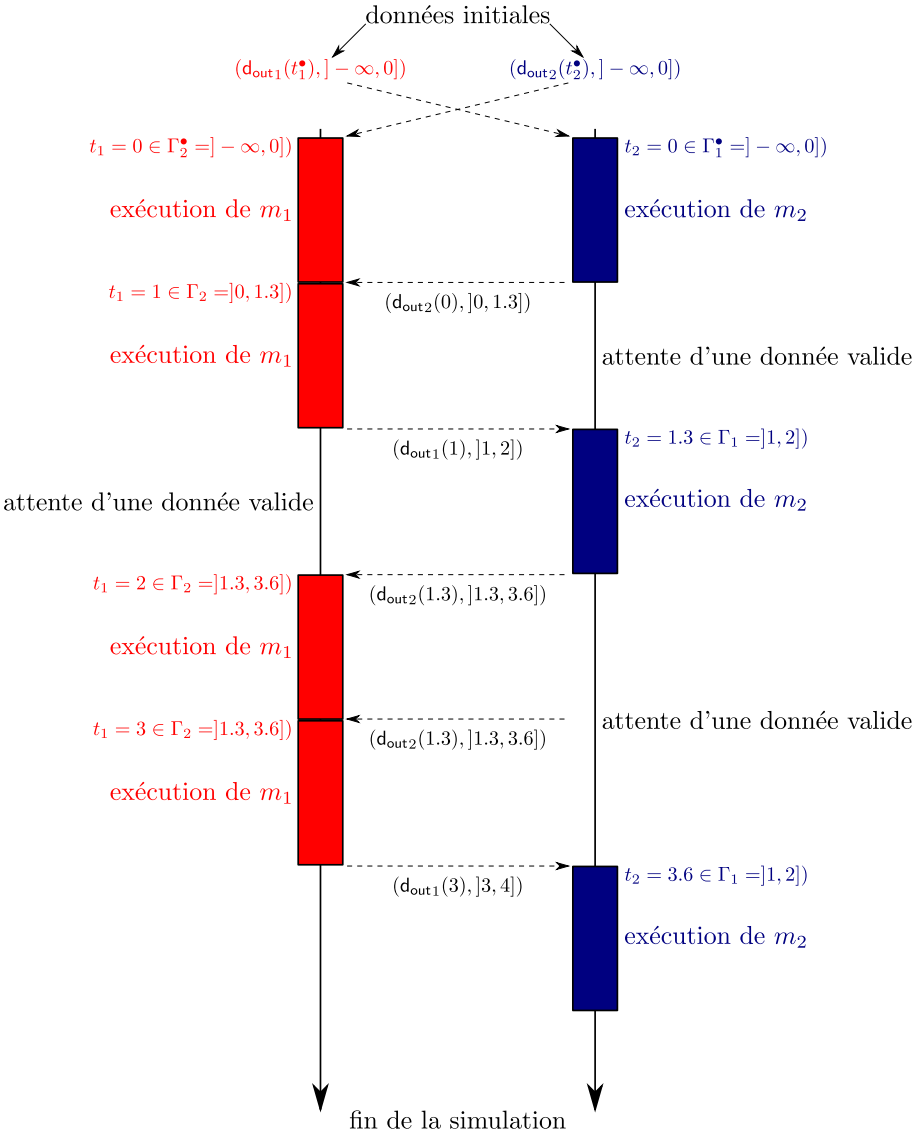


FIGURE 9.8 – Exemple précédent : diagramme de séquence. À gauche sont représentées les différentes exécutions du modèle m_1. À droite sont représentées les différentes exécutions du modèle m_2. Les deux axes, orientés vers le bas, représentent le temps réel de calcul au niveau du logiciel de simulation et non le temps de simulation (qui lui est virtuel).

à deux (le graphe de dépendance est complet). On pose également comme hypothèse qu'il n'y a pas de situation bloquante à l'initialisation. Chaque modèle est initialement paramétré et chaque donnée initiale est publiée dans l'*artefact-de-couplage* correspondant.

Le premier modèle à pouvoir être exécuté est le modèle m_i. En effet, c'est le seul qui satisfait la condition de validité (voir 9.1a). Une fois m_i exécuté, le temps de simulation est augmenté (voir 9.1b). L'intervalle de validité est calculé (voir 9.1c).

$$\forall k, t_i^\bullet \in \Gamma_k^\bullet \qquad \text{avec } k \in [0,n]\backslash\{i\} \tag{9.1a}$$
$$t_i \leftarrow t_i^\bullet + \delta t_i \qquad \text{avec } \delta t_i > 0 \tag{9.1b}$$
$$\Gamma_i \leftarrow]t_i^\bullet, t_i^\bullet + \delta t_i] \tag{9.1c}$$

Deux possibilité s'offrent à nous :
- soit $t_i = \min_k t_k^\bullet$,
- soit $t_i > \min_k t_k^\bullet$ (avec dans les deux cas $k \in [0,n]\backslash\{i\}$).

Dans le premier cas, le modèle m_i peut de nouveau être exécuté. On retombe alors sur la situation précédente (voir l'équation 9.1). Dans le deuxième cas, il existe un modèle m_j qui vérifie la condition de validité et peut être exécuté (voir 9.2a). Le temps de simulation est ensuite augmenté (voir 9.2b) et l'intervalle de validité calculé (voir 9.2c).

$$\exists j : \forall k, t_j^\bullet \in \Gamma_k \text{ ET } t_j^\bullet \in \Gamma_i \qquad \text{avec } j \in [0,n]\backslash\{i\} \text{ et } k \in [0,n]\backslash\{j,i\} \tag{9.2a}$$
$$t_j \leftarrow t_j^\bullet + \delta t_j \qquad \text{avec } \delta t_j > 0 \tag{9.2b}$$
$$\Gamma_j \leftarrow]t_j^\bullet, t_j^\bullet + \delta t_j] \tag{9.2c}$$

De nouveau, deux possibilités s'offrent à nous :
- soit $t_j = \min_k t_k^\bullet$,
- soit $t_j > \min_k t_k^\bullet$ (avec dans les deux cas $k \in [0,n]\backslash\{j\}$).

Dans le premier cas, le modèle m_j peut de nouveau être exécuté. On retombe alors sur la situation précédente (voir l'équation 9.2). Dans le deuxième cas, de la même manière que précédemment, il existe un modèle qui peut être exécuté. On reproduit la même démonstration pour l'ensemble des modèles et on montre que l'utilisation d'un intervalle de validité initial $\Gamma_i^\bullet =]-\infty, t_i^\bullet]$ est bien une condition suffisante pour que l'algorithme de simulation démarre.

9.4.2 Pas d'attente infinie

Dans cette section nous montrons que l'algorithme n'implique pas d'attente infinie quelque soit le graphe de dépendance utilisé. La démonstration s'appuie sur le raisonnement par l'absurde. Nous partons d'une hypothèse qui est qu'un simulateur donné est bloqué (en attente de données d'entrée) et nous dérivons l'ensemble des causes du blocage pour arriver à la conclusion que l'hypothèse initiale (le blocage) est impossible. Nous avons également développé une spécification formelle en event-B de l'algorithme de simulation. Le but de cette formalisation est de montrer que la synchronisation des simulateurs se fait toujours et ce, sans créer d'attente infinie. Cette dernière est présentée en annexe.

Prenons $m_1, m_2 \ldots m_n$, un ensemble fini de modèles. Les modèles sont tous couplés deux à deux (graphe de dépendance complet). Nous faisons l'hypothèse que l'ensemble des modèles ont été paramétrés et que la simulation à démarré. Autrement dit, $\forall k \in [0,n], t_k > t_k^{\bullet}$.

Nous considérons, ensuite, qu'il existe au moins un *m-agent* bloqué. On notera B l'ensemble des indices i tels que $\mathcal{A}_i$ est bloqué (avec $\mathsf{B} \neq \varnothing$ et $\subset[0,n]$). On dit que le *m-agent* $\mathcal{A}_i$ est bloqué au temps de simulation $t_i = t_i^{stop}$ si et seulement si il n'obtient pas toutes les données valides nécessaires à l'exécution du modèle m_i.

Le m-agent $\mathcal{A}_i$ est dit bloqué si et seulement si $\forall (j,k) \in \mathsf{IN}_i,\ \exists j : \mathcal{C}_i^j.read(t_i^{stop}, k) = \varnothing$

Notre hypothèse initiale postule que le modèle m_i a été exécuté jusqu'au temps t_i^{stop}. Cela signifie que les données ${\mathsf{d}_{\mathsf{out}}}_i^n(t_i)$ ont été produites et stockées dans les *artefact-de-couplage* $\mathcal{C}_j^i$ correspondants $(\forall (j,k,n) \in \mathsf{OUT}_i)$ jusqu'à ce que le *m-agent* $\mathcal{A}_i$ soit stoppé à $t_i = t_i^{stop}$.

Un *m-agent* $\mathcal{A}_i$ ne peut faire exécuter son modèle et augmenter le temps de simulation si et seulement s'il possède toutes les données valides dont il a besoin. Si ce dernier est bloqué, cela veut dire qu'il attend les données d'un *m-agent* $\mathcal{A}_j$ dont le temps de simulation est inférieur au sien (avec $t_j < t_j$ et $i \neq j$). Soit ce dernier est en train de faire exécuter son modèle et donc de produire les données attendues et dans ce cas, le *m-agent* $\mathcal{A}_i$ n'est pas bloqué. Soit le *m-agent* $\mathcal{A}_j$ est lui aussi bloqué mais à un temps de simulation $t_j^{stop} < t_i^{stop}$. En conséquence, nous proposons le lemme suivant :

Lemme 9.4.2.1 *Un m-agent $\mathcal{A}_i$ est dit bloqué ($i \in \mathsf{B}$) si et seulement si :*

1. *il dépend des données ${\mathsf{d}_{\mathsf{out}}}_j^n$ produites par un sous-ensemble fini de m-agent $\mathcal{A}_j$ $\exists k : ((j,n,k) \in \mathsf{OUT}_i)$;*
2. *parmi ces m-agents, il existe au moins un m-agent $\mathcal{A}_{j'}$ bloqué $\exists n', k' : ((j',n',k') \in \mathsf{OUT}_i$ et $j' \in \mathsf{B})$;*
3. $t_{j'}^{stop} < t_i^{stop}$;
4. $j' \neq i$.

Le *m-agent* $\mathcal{A}_j$ est bloqué signifie qu'il existe un troisième *m-agent* $\mathcal{A}_k$ bloqué avec $t_k^{stop} < t_j^{stop} < t_i^{stop}$ et $i \neq j \neq k$. Étant donné que l'ensemble des *m-agents* bloqués et le nombre de données échangées sont finis, on montre que si un *m-agent* $\mathcal{A}_i$ est bloqué au temps de simulation t_i^{stop} alors c'est l'ensemble des *m-agents* qui sont bloqués. Il existe alors au moins un *m-agent* dont le temps de simulation est minimum.

$$\forall i \in \mathcal{B}, \exists z \in \mathcal{B} \text{ tel que } t_z^{stop} = \min_i(t_i^{stop})$$

La contradiction apparaît avec ce dernier *m-agent* $\mathcal{A}_z$. En effet, considérant le lemme 9.4.2.1, si $\mathcal{A}_z$ est bloqué, cela signifie qu'il doit exister un simulateur $\mathcal{A}_{z'}$ lui aussi bloqué mais à un temps de simulation $t_{z'}^{stop} < t_z^{stop}$. Cela n'est pas possible car t_z^{stop} est déjà un minima.

Cela signifie que si ce dernier *m-agent* $\mathcal{A}_z$ est effectivement bloqué ce n'est pas à cause d'autres *m-agents* dont le temps de simulation serait inférieur à t_z^{stop}. Le blocage peut provenir du fait que les simulateurs bloqués n'ont pas été initialisés correctement et ne peuvent donc pas exécuter leurs modèles respectifs et ce dès le départ de la simulation ce qui est en contradiction avec notre hypothèse initiale. Dans ce cas, ce n'est pas l'algorithme de simulation en lui même qui est incriminé mais bien l'initialisation et le paramétrage des modèles élémentaires. D'autre part, le blocage peut provenir du fait que la communication entre les composants du méta-modèle n'est pas fiable. Dans ce cas, l'algorithme de simulation n'est pas non plus à incriminer.

9.5 Synthèse

Dans cet chapitre, nous avons proposé un algorithme de simulation distribué adapté au méta-modèle AA4MM. Ce dernier s'inspire des algorithmes conservatifs classiques (tel que l'algorithme de Chandy,

Misra et Bryant [Chandy and Misra, 1979; Adam et al., 1981]) tout en apportant des solutions pour gérer à la fois la différence de politiques d'exécution (lorsque la représentation du temps est soit discrète soit continue) et la vision multi-agent apportée par AA4MM. Nous avons démontré que cet algorithme permet aux *m-agents* de respecter la contrainte de causalité entre les simulateurs et qu'il n'introduit pas d'attente infinie (*deadlock*). Cet algorithme, en plus de permettre l'interaction d'un ensemble de modèles ayant des politiques d'exécution différentes, s'inscrit en plein dans l'architecture multi-agent proposée. En effet, la coordination des *m-agents* s'effectue au travers des *artefacts-de-couplage* par les données échangées.

Les chapitres suivants (chapitres 10 et 11) présentent, quant à eux, deux exemples d'utilisation de l'architecture AA4MM pour la conception d'une "société de modèles". Ces deux exemples permettent à la fois d'établir des preuve de concept et illustrent les propriétés d'une telle approche.

Chapitre 10

Preuves de concepts

Sommaire

10.1 Introduction

La démarche utilisée dans ce chapitre est la suivante. Le but recherché est de se mettre à la place d'une personne qui construit un multi-modèle à partir de modèles et simulateurs existants qu'elle ne maîtrise pas totalement. C'est-à-dire, par exemple, que le code source des simulateurs n'est accessible qu'au travers une API (*Application Programming Interface*), que les modèles utilisés sont hors de son champ disciplinaire, etc.

Ainsi, les modèles utilisés dans ce chapitre sont délibérément simples. Cependant ceux-ci sont issus de travaux existants. Ensuite, le domaine duquel proviennent les modèles est, là aussi, délibérément en dehors du champ des réseaux ambiants. Enfin, la seule interaction possible avec les simulateurs réutilisés se fait via l'API qui nous est fournie.

Les preuves de concept apportées par ce chapitre se font au travers de la démarche de conception et d'utilisation d'un multi-modèle particulier. Ces dernières ciblent les points suivants :

- La réutilisation de simulateurs et de modèles existants.
- L'expression des choix de multi-modélisation.
- La prise en compte de niveaux d'abstraction différents (espace et temps).
- La modularité de l'approche (ajout, suppression et échange de modèles et de simulateurs).

Pour des raisons de clarté, nous ne ciblons pas dans ce chapitre l'interaction de simulateurs possédant des politiques d'exécution différentes. En effet, le chapitre précédent démontre de manière théorique que le méta-modèle AA4MM respecte les contraintes de causalité quels que soient les simulateurs impliqués. De plus, une telle situation est complètement transparente pour le méta-modèle proposé. Le chapitre suivant et les travaux expérimentaux faits dans le domaine des réseaux ambiants montrent un exemple d'un tel cas.

10.2 Exemple utilisé : un multi-modèle proie-prédateur composé de simulateurs NetLogo en interaction

10.2.1 Utilisation de NetLogo

Nous avons choisi d'utiliser le logiciel de simulation NetLogo [Wilensky, 1999]. Celui-ci est bien connu de la communauté multi-agent. En effet, ce dernier est souvent utilisé de manière pédagogique pour introduire la modélisation et la simulation multi-agent ou pour concevoir des prototypes.

En plus de sa grande diffusion au sein de la communauté, nous avons choisi cet outil pour deux raisons. Tout d'abord, la plate-forme NetLogo possède un grand nombre de modèles qui vont de la biologie aux mathématiques en passant par les sciences sociales. Ensuite, cet outil possède une API relativement bien documentée qui permet d'interfacer cet outil avec un autre programme (dans notre cas l'implantation du méta-modèle AA4MM).

10.2.2 Cas d'étude utilisé : un multi-modèle proie-prédateur

Le modèle proie-prédateur décrit la dynamique de deux populations : les proies et les prédateurs. Ce modèle est un exemple devenu classique dans le domaine de l'écologie et de la modélisation et simulation des phénomènes collectifs (voir les utilisations faites dans [Michel, 2004; Grimm, 2005]). Le but visé dans cette section n'est pas de concevoir un modèle le plus réaliste possible mais de montrer, sur un exemple connu, l'application du méta-modèle AA4MM.

Nous souhaitons concevoir un multi-modèle proie-prédateur qui représente les interactions entre une population de moutons, de loups et de bergers. Dans un premier temps, nous considérons des interactions relativement basiques entre ces différentes populations : les loups mangent les moutons, les moutons mangent de l'herbe et les bergers tentent d'attrouper les moutons (voir la figure 10.1).

Pour construire notre multi-modèle, nous disposons des modèles fournis avec la plate-forme NetLogo. Cette dernière propose un modèle de dynamique de population dans lequel des moutons [29] mangent de l'herbe [Wilensky, 2001], un modèle proie-prédateur représentant des loups qui mangent des moutons [Wilensky, 1997b] et un modèle d'agrégation dans lequel des bergers attroupent des moutons [Wilensky, 1998b].

10.3 Conception du multi-modèle

Cette section a pour but d'illustrer les différentes étapes nécessaires à la conception et la mise en place, via le méta-modèle AA4MM, d'un multi-modèle composé de modèles élémentaires existants.

29. Originellement des lapins mais nous avons changé l'espèce pour les besoins de l'illustration

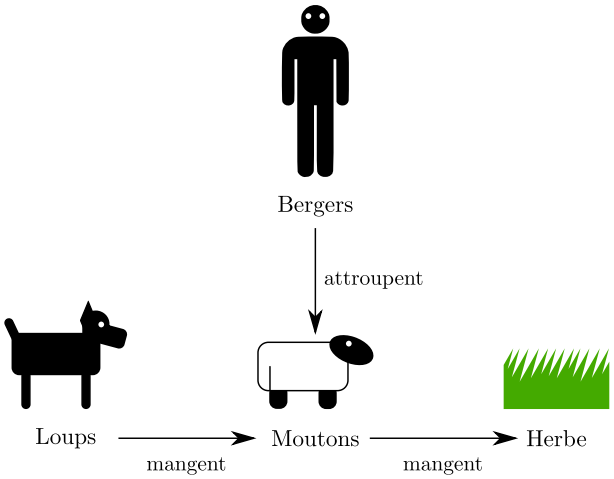


FIGURE 10.1 – Phénomène à modéliser et à simuler

10.3.1 Modèles

La première étape de conception du multi-modèle consiste à définir au niveau sémantique quels sont les modèles et à spécifier les entrées et sorties de chaque modèle élémentaire.

Le premier modèle représente des moutons qui mangent de l'herbe. La dynamique de ce modèle est la suivante : le modèle détermine la quantité d'herbe présente par unité de surface (patchs) et, à partir de la position de chaque mouton, calcule la quantité d'énergie ingérée par chaque ovidé. Les données d'entrée de ce modèle sont donc les positions des moutons PM et les données de sortie, la quantité d'énergie gagnée pour chaque mouton EM (voir figure 10.2). On notera ce modèle m_1, il possède un port d'entrée x_1 et un port de sortie y_1. Les données d'entrée sont notées $\mathsf{d}_{\mathsf{in}1}(t_1)$ et $\mathsf{d}_{\mathsf{out}1}(t_1)$, $\mathsf{d}_{\mathsf{in}1}(t_1) \in PM$ et $\mathsf{d}_{\mathsf{out}1}(t_1) \in EM$ (avec $\mathsf{d}_{\mathsf{out}2}(t_2) \geq 0$)..

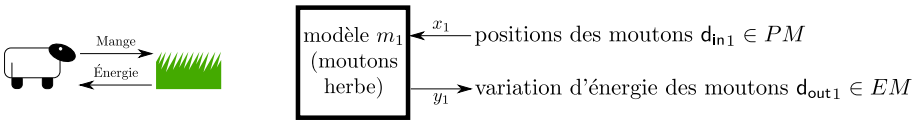


FIGURE 10.2 – Modèle m_1 représentant la dynamique entre les moutons et l'herbe

Le deuxième modèle représente des loups qui mangent des moutons. La dynamique de ce modèle est la suivante : le modèle détermine le comportement des loups et, à partir des positions des moutons, ce dernier détermine si un mouton est attaqué. Pour des raisons d'illustration, lorsqu'un mouton se fait attaquer par un loup, celui-ci n'est pas dévoré mais perd de l'énergie. De cette manière, les entrées du modèle sont donc les positions des moutons. On noter PM le type "position des moutons". Les sorties du modèle sont la quantité d'énergie perdue pour chaque mouton. On note EM le type "variation d'énergie des moutons" (voir figure 10.3). On notera ce modèle m_2, il possède un port d'entrée x_2 et un port de sortie y_2. Les données d'entrée sont notées $\mathsf{d}_{\mathsf{in}2}(t_2)$ et $\mathsf{d}_{\mathsf{out}2}(t_2)$, $\mathsf{d}_{\mathsf{in}2}(t_2) \in PM$ et $\mathsf{d}_{\mathsf{out}2}(t_2) \in EM$ (avec $\mathsf{d}_{\mathsf{out}2}(t_2) \leq 0$).

Le dernier modèle représente des bergers qui attroupent des moutons. La dynamique de ce modèle est la suivante : les moutons bougent en fonction de leur énergie et les bergers tentent de les rassembler. Les entrées du modèle sont donc la quantité d'énergie EM des moutons et les sorties sont l'ensemble des positions des moutons PM (voir figure 10.4). On notera ce modèle m_3, il possède un port d'entrée x_3

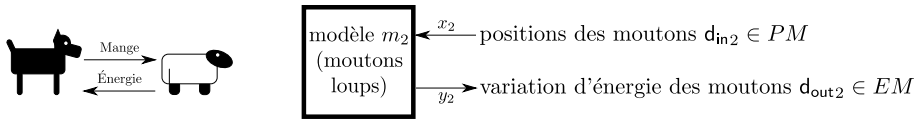


FIGURE 10.3 – Modèle m_2 représentant la dynamique entre les loups et les moutons

et un port de sortie y_3. Les données d'entrée sont notées $\mathsf{d}_{\mathsf{in}3}(t_3)$ et $\mathsf{d}_{\mathsf{out}3}(t_3)$, $\mathsf{d}_{\mathsf{in}3}(t_3) \in EM$ et $\mathsf{d}_{\mathsf{out}3}(t_3) \in PM$.

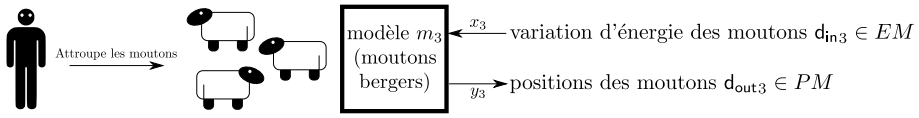


FIGURE 10.4 – Modèle représentant la dynamique entre les bergers et les moutons

On fera l'hypothèse que les données échangées sont décrites dans le même formalisme et que le nombre de moutons (noté $\#_{moutons}$) est identique dans chaque modèle. Le type de donnée "positions des moutons" (noté PM) est une matrice $2 \times \#_{moutons}$ qui met en correspondance le numéro du mouton avec sa position. Le type de donnée "variation d'énergie des moutons" (noté EM) est une matrice $2 \times \#_{moutons}$ qui met en correspondance le numéro du mouton avec la variation de son énergie.

description	**notation**
premier modèle (moutons + herbe)	m_1
port d'entrée	x_1
port de sortie	y_1
donnée d'entrée	$\mathsf{d}_{\mathsf{in}1}(t_1) \in PM$
donnée de sortie	$\mathsf{d}_{\mathsf{out}1}(t_1) \in EM$
deuxième modèle (loups + moutons)	m_2
port d'entrée	x_2
port de sortie	y_2
donnée d'entrée	$\mathsf{d}_{\mathsf{in}2}(t_2) \in PM$
donnée de sortie	$\mathsf{d}_{\mathsf{out}2}(t_2) \in EM$
troisième modèle (bergers + moutons)	m_3
port d'entrée	x_3
port de sortie	y_3
donnée d'entrée	$\mathsf{d}_{\mathsf{in}3}(t_3) \in EM$
donnée de sortie	$\mathsf{d}_{\mathsf{out}3}(t_3) \in PM$
nombre de moutons	$\#_{moutons}$
positions des moutons	PM
variation d'énergie des moutons	EM

FIGURE 10.5 – Notations des modèles

10.3.2 Graphe de dépendance

À partir des modèles ainsi spécifiés, le graphe de dépendance peut aisément se construire. La figure 10.6 illustre le graphe de dépendance tandis que la figure 10.7 donne l'ensemble des notations formelles. Il est à noter que les données représentant les positions des moutons $\mathsf{d}_{\mathsf{out}3}(t_3)$ sont nécessaires à l'exécution des modèles m_1 et m_2. Ensuite les données représentant la variation d'énergie des moutons $\mathsf{d}_{\mathsf{out}1}(t_1)$ et $\mathsf{d}_{\mathsf{out}2}(t_2)$ doivent être combinées pour pouvoir être consommées par le troisième modèle. Nous explicitons la composition dans la section suivante.

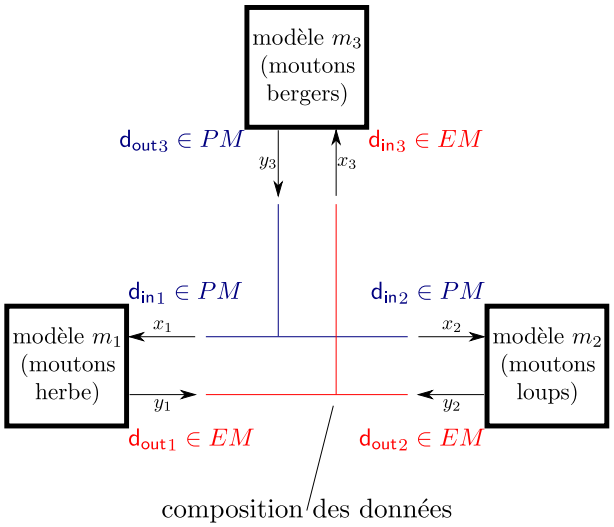


FIGURE 10.6 – Graphe de dépendance du multi-modèle moutons-loups-bergers

description	notation
entrées du modèle m_1	$\mathsf{IN}_1 = \{(3,1)\}$
sorties du modèle m_1	$\mathsf{OUT}_1 = \{(1,3)\}$
entrées du modèle m_2	$\mathsf{IN}_2 = \{(3,1)\}$
sorties du modèle m_2	$\mathsf{OUT}_2 = \{(1,3)\}$
entrées du modèle m_3	$\mathsf{IN}_3 = \{(1,1);(2,1)\}$
sorties du modèle m_3	$\mathsf{OUT}_3 = \{(1,1);(1,2)\}$
Lien de m_1 à m_3	$\mathsf{L}_3^1 = \{(1,1)\}$
Lien de m_2 à m_3	$\mathsf{L}_3^2 = \{(1,1)\}$
Lien de m_3 à m_1	$\mathsf{L}_1^3 = \{(1,1)\}$
Lien de m_3 à m_2	$\mathsf{L}_2^3 = \{(1,1)\}$

FIGURE 10.7 – Notation du grape de dépendance

10.3.3 Composition de données

Dans cette section, nous définissions ce que signifie composer les données échangées et comment cela est pris en compte dans le méta-modèle. C'est le cas, dans notre exemple, avec l'énergie des moutons. En effet, les gains et pertes d'énergies issus de m_1 et m_2 doivent être combinés afin de fournir au modèle m_3 une donnée représentant la variation d'énergie de chaque mouton.

Comme nous avons fait l'hypothèse de réutiliser des modèles existants, nous considérons que ceux-ci ne peuvent composer eux même les données. La composition doit donc se faire en dehors de tous les modèles. Celle-ci sera effectuée par un *artefact-de-couplage*.

Au niveau sémantique, la composition est relativement simple à définir puisqu'il suffit d'additionner les gains et les pertes d'énergie : $\mathsf{d}_{\mathsf{in}3}(t_3) = \mathsf{d}_{\mathsf{out}2}(t_2) + \mathsf{d}_{\mathsf{out}1}(t_1)$. On dira que l'opérateur de composition est l'opérateur $+$.

Au niveau syntaxique, si les données à composer sont dans des formalismes différents, il faut résoudre les conflits avant la composition. Par exemple, si les données issues des modèles m_1 et m_2 sont des réels et la donnée d'entrée du modèle m_3 est entière ($\mathsf{d}_{\mathsf{out}2}(t_2) \in \mathbb{R}_-$, $\mathsf{d}_{\mathsf{out}1}(t_1) \in \mathbb{R}_+$ et $\mathsf{d}_{\mathsf{out}3}(t_3) \in \mathbb{Z}$) alors il faut transformer le résultat de la somme qui est réel en un entier. Dans notre exemple, nous faisons l'hypothèse qu'il n'y a pas de conflit au niveau syntaxique.

Enfin au niveau dynamique, il faut tenir compte des intervalles de validité lors de la composition des données. Prenons comme exemple deux données $\mathsf{d}_{\mathsf{out}1}(t_1 = 1)$ et $\mathsf{d}_{\mathsf{out}2}(t_2 = 1.5)$ associées à leurs intervalles de validité Γ_1 et Γ_2. Ces intervalles Γ_1 et Γ_2 ne sont pas toujours de même taille. Par exemple la donnée $\mathsf{d}_{\mathsf{out}1}(t_1 = 1)$ peut être associée à un intervalle $\Gamma_1 =]1, 3]$ et la donnée $\mathsf{d}_{\mathsf{out}2}(t_2 = 1.5)$ peut être associée à un intervalle de validité $\Gamma_2 =]1.5, 5]$. Pour le troisième modèle m_3 seule la donnée $\mathsf{d}_{\mathsf{out}1}(t_1 = 1)$ est valide pour $t_3 \in]1, 1.5]$. Ensuite, les deux données sont valides en même temps pour $t_3 \in]1.5, 3]$. Enfin, seule la donnée $\mathsf{d}_{\mathsf{out}2}(t_2 = 1.5)$ est valide pour $t_3 \in]3, 5]$.

Afin de ne pas remettre en question la contrainte de causalité et d'utiliser l'algorithme de simulation présenté au chapitre précédent, nous proposons de composer les intervalles de validité en même temps que les données.

> *Soit* $\circ$ *l'opérateur de composition. Soit* $(\mathsf{d}_{\mathsf{out}1}^{n}(t_1), \Gamma_1)$ *et* $(\mathsf{d}_{\mathsf{out}2}^{n'}(t_1 = 1), \Gamma_2)$ *les données à composer.* Composer les données $\mathsf{d}_{\mathsf{out}1}^{n}(t_1)$ *et* $\mathsf{d}_{\mathsf{out}2}^{n'}(t_2)$ *signifie produire un nouvel ensemble de données comprenant :*
>
> $$(\mathsf{d}_{\mathsf{out}1}^{n}(t_1) \circ \mathsf{d}_{\mathsf{out}2}^{n'}(t_2), \Gamma_1 \bigcap \Gamma_2),\ (\mathsf{d}_{\mathsf{out}1}^{n}(t_1), \Gamma_1 \backslash \{\Gamma_1 \bigcap \Gamma_2\})\ \textit{et}\ (\mathsf{d}_{\mathsf{out}2}^{n'}(t_2), \Gamma_2 \backslash \{\Gamma_1 \bigcap \Gamma_2\}).$$

Il est à noter que cette opération de composition devra être automatiquement effectuée par les *artefact-de-couplage* et que celle-ci ne change en rien l'algorithme de simulation proposé. La figure suivante (figure 10.8) illustre cette opération de composition.

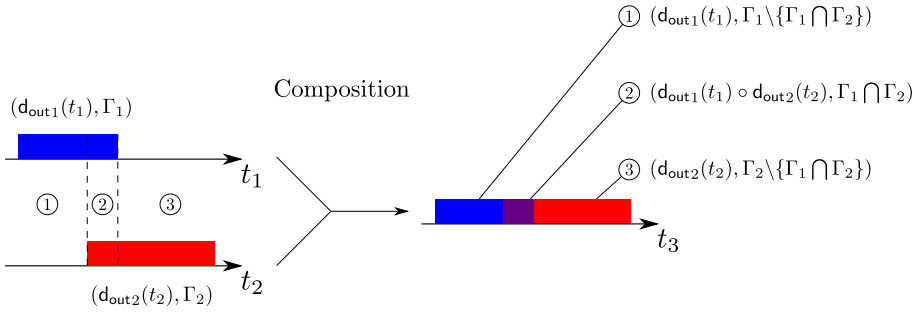


FIGURE 10.8 – Composition de deux données $(\mathsf{d}_{\mathsf{out}1}^{n}(t_1), \Gamma_1)$ et $(\mathsf{d}_{\mathsf{out}2}^{n'}(t_1 = 1), \Gamma_2)$ par un *artefact-de-couplage*. Chaque intervalle de validité est représenté par un rectangle sur l'axe du temps de simulation (rouge pour Γ_1 et bleu pour Γ_2). Les données composées sont affectées à un nouvel intervalle de validité : $\Gamma_1 \bigcap \Gamma_2$ (représenté par un rectangle violet).

10.3.4 Création des entités propres au méta-modèle AA4MM

Une fois le graphe de dépendance établi, il est relativement simple de créer les entités propres au méta-modèle AA4MM. Il faut tout d'abord créer pour chaque logiciel de simulation un *m-agent*. Ensuite, il faut établir l'interface entre cet *m-agent* et le logiciel de simulation géré : l'*artefact-d'interface*. Enfin, pour chaque lien du graphe de dépendance, il faut définir un *artefact-de-couplage*. La figure 10.9 donne une vue générale des entités composant le multi-modèle multi-agent (un exemple de code est donné en annexe).

L'*artefact-d'interface* fait office d'interface entre l'*m-agent* et le logiciel de simulation. Nous définissons trois *artefacts-d'interface* (un pour chaque logiciel de simulation) notés $\mathcal{I}_1$, $\mathcal{I}_2$ et $\mathcal{I}_3$. Différentes opérations doivent être implantées au sein de chaque *artefact-d'interface* (voir chapitre 8). Nous ne donnons pas ici le détail complet de l'implantation de chaque opération mais seulement les grandes lignes. Il est important de noter que chaque opération utilise uniquement les fonctions fournies par l'API NetLogo (en Java). En aucun cas, les *artefacts-d'interface* ne modifie le code interne de ce dernier.

1. L'opération *init()* crée une instance de NetLogo, charge le modèle (le fichier .nlogo correspondant) et paramètre celui-ci.

2. L'opération *run()* lance l'exécution du modèle pour un pas de simulation. En effet, la plate-forme NetLogo exécute ses modèles pas par pas. Cette opération est implantée en invoquant la méthode *"go"* de NetLogo.
3. L'opération *getCurrentTime()* retourne le nombre de pas de simulation effectués (le nombre de *"ticks"*).
4. Les opérations *getOutputData()* et *setInputData(d_i)* se chargent de récupérer les données de sortie du modèle et d'injecter à ce dernier les données d'entrée.
5. L'opération *stop()*, quant à elle, termine l'instance de NetLogo.

Chaque lien du graphe de dépendance donne naissance à un *artefact-de-couplage*. Du modèle m_3 aux modèles m_1 et m_2, nous définissons deux *artefacts-de-couplage*. Nous les notons $\mathcal{C}_1^3$ et $\mathcal{C}_2^3$. Ceux-ci ont pour rôle respectif de faire le lien entre la donnée de sortie du modèle m_3 $\mathsf{d}_{\mathsf{out}3}(t_3)$ et les données d'entrée des modèles m_1 et m_2 : $\mathsf{d}_{\mathsf{in}1}(t_1)$ et $\mathsf{d}_{\mathsf{in}2}(t_2)$. L' *artefact-de-couplage* $\mathcal{C}_1^3$ possède l'information L_1^3 et l' *artefact-de-couplage* $\mathcal{C}_2^3$ L_2^3.

Pour le transfert des gains et des pertes d'énergie des moutons provenant des deux modèles m_1 et m_2 vers la variation d'énergie au modèle m_3, nous définissons un *artefact-de-couplage*. Celui-ci est noté $\mathcal{C}_3^{1,2}$ et fait le lien entre $\mathsf{d}_{\mathsf{out}1}(t_1)$, $\mathsf{d}_{\mathsf{out}2}(t_2)$ et $\mathsf{d}_{\mathsf{in}3}(t_3)$. Celui-ci effectue également la composition des données selon le principe présenté précédemment (l'opérateur de composition correspond à l'opérateur d'addition $+$).

Il ne reste plus qu'à définir les *m-agents*. Il y en a un pour chaque logiciel de simulation. On les notes $\mathcal{A}_1$, $\mathcal{A}_2$ et $\mathcal{A}_3$. En ce qui concerne la phase d'exécution des modèles, le comportement de ces *m-agents* suit l'algorithme de simulation présenté au chapitre précédent.

Pour la phase d'initialisation, il est nécessaire de spécifier quelque peu les choses. L'initialisation des modèles m_1 et m_2 dépend des positions générées par le modèle m_3. Les *m-agents* $\mathcal{A}_1$ et $\mathcal{A}_2$ doivent, pendant la phase d'initialisation, attendre que l'*m-agent* $\mathcal{A}_3$ leur ait fourni les positions des moutons ($\mathsf{d}_{\mathsf{out}3}(t_3) \in PM$) avant de pouvoir paramétrer leurs modèles. L'*m-agent* $\mathcal{A}_3$ doit, quant à lui, attendre que les *m-agents* $\mathcal{A}_1$ et $\mathcal{A}_2$ lui aient envoyé les variations d'énergie initiales de chaque mouton. On se retrouve ici dans une situation bloquante à l'initialisation. Pour résoudre cet inter-bloquage, on considérera que le modèle m_3 prend en paramètre initial une variation d'énergie nulle pour chaque mouton. Ce que l'on peut traduire par $\mathsf{d}_{\mathsf{out}1}(t_1^\bullet)+\mathsf{d}_{\mathsf{out}2}(t_2^\bullet) = 0$. Cette variation d'énergie sera donnée directement à l'*m-agent* $\mathcal{A}_3$ par nos soins.

Ainsi, le modèle m_3 est paramétré puis l'*m-agent* $\mathcal{A}_3$ envoie les positions des moutons ($\mathsf{d}_{\mathsf{out}3}(t_3^\bullet)$) aux autres *m-agents* $\mathcal{A}_1$ et $\mathcal{A}_2$ via les *artefacts-de-couplage* $\mathcal{C}_1^3$ et $\mathcal{C}_2^3$. De cette manière les *m-agents* $\mathcal{A}_1$ et $\mathcal{A}_2$ peuvent initialiser et paramétrer leurs modèles respectifs (voir figure 10.10).

10.3.5 Choix d'implantation

Il est important de noter que le méta-modèle AA4MM n'impose pas l'utilisation d'un même langage de programmation pour tous ses composant et permet plus facilement, de par son essence distribuée (un ensemble d'entités en interaction), l'utilisation de langages hétérogènes. Dans cet exemple, notre approche nous permet d'avoir trois instances de la plate-forme NetLogo fonctionnant de manière concurrente. Étant donné que le logiciel NetLogo fournit une API en Java, nous avons implanté l'ensemble des entités du méta-modèle AA4MM en Java.

Nous avons fait deux implantations de cet exemple. La première a consisté simplement à implanter l'ensemble des *m-agents*, des *artefacts-d'interface* et des *artefacts-de-couplage* dans un programme concurrent (à base de threads) qui est exécuté par une seule JVM (*java virtual machine*). Le but de cette première implantation était de vérifier expérimentalement le comportement du méta-modèle AA4MM. La deuxième implantation a consisté à utiliser un *middleware* existant (JMS : *Java Messaging Service*) pour implanter les *artefacts-de-couplage*. Le but de cette implantation est de démontrer la possibilité d'utiliser AA4MM de manière distribuée sur plusieurs machines. Bien que le choix de JMS est suffisant pour établir une preuve de concepts, celui-ci s'est montré très mauvais en terme de performance. Le temps de calcul variant du simple à plus du double de la première à la deuxième implantation.

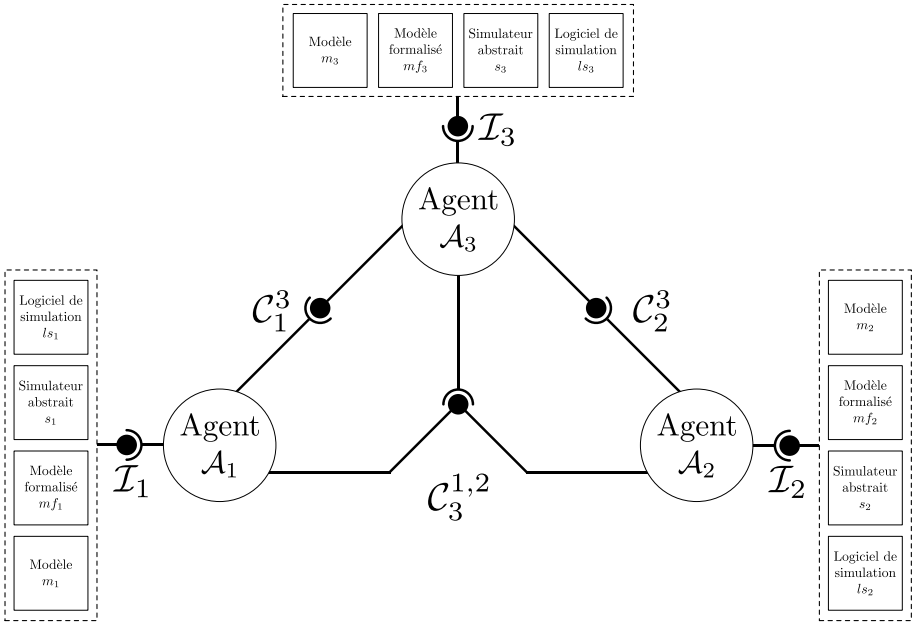


FIGURE 10.9 – Multi-modèle multi-agent moutons-loups-bergers : vue générale

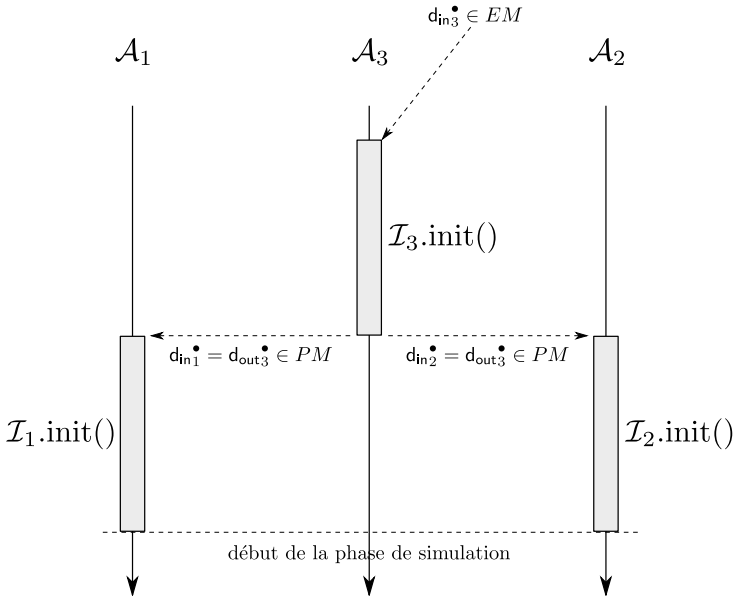


FIGURE 10.10 – Initialisation : diagramme de séquence.

10.4 Exécutions des modèles

Un fois les modèles initialisés, la simulation se déroule suivant l'algorithme décrit dans le chapitre précédent. Afin d'éviter toute redondance avec les exemples déjà présentés, nous ne donnerons ici qu'un aperçu rapide de la phase d'exécution des modèles. La principale différence avec les exemples précédents vient du fait que nous avons introduit des opérations sur les données échangées. Le fait d'avoir plus de deux modèles ne pose aucun problème puisque nous avons démontré que l'algorithme de simulation respecte la contrainte de causalité et n'introduit pas d'attente infinie pour un nombre quelconque et fini d'*m-agents*.

10.4.1 Phase de simulation

Nous avons vu précédemment comment résoudre la situation de bloquage à l'initialisation. Chaque modèle étant initialisé et paramétré, chaque *m-agent* peut faire exécuter son modèle, augmenter le temps de simulation et récupérer les données de sortie. La simulation dans NetLogo s'effectue par pas de temps (ou "ticks"). De ce fait, les temps de simulation après la première exécution sont $t_1 = t_2 = t_3 = 1$. Les données de sortie et les intervalles de validité associés pour chaque modèle sont :

- les gains d'énergie des moutons : $\mathsf{d}_{\mathsf{out}1}(t_1 = 1)$, $\Gamma_1 =]0, 1]$;
- les pertes d'énergie des moutons : $\mathsf{d}_{\mathsf{out}2}(t_2 = 1)$, $\Gamma_2 =]0, 1]$;
- les positions des moutons : $\mathsf{d}_{\mathsf{out}3}(t_3 = 1)$, $\Gamma_3 =]0, 1]$.

À cet instant, les *m-agents* $\mathcal{A}_1$ et $\mathcal{A}_2$ peuvent de nouveau faire exécuter leurs modèles respectifs car, la donnée $\mathsf{d}_{\mathsf{out}3}(t_3 = 1)$ est valide pour le temps de simulation $t_1 = t_2 = 1$ (voir figure 10.11). Concernant l'*m-agent* $\mathcal{A}_3$, celui-ci doit récupérer la composition des données $\mathsf{d}_{\mathsf{out}1}(t_1 = 1)$ et $\mathsf{d}_{\mathsf{out}2}(t_2 = 1)$. Cela se déroule via l'*artefact-de-couplage* $\mathcal{C}_3^{1,2}$: quand celui-ci a récupéré les données $\mathsf{d}_{\mathsf{out}1}(t_1 = 1)$ et $\mathsf{d}_{\mathsf{out}2}(t_2 = 1)$, il les compose et retourne à l'*m-agent* $\mathcal{A}_3$ la donnée : $\mathsf{d}_{\mathsf{in}3}(1) = \mathsf{d}_{\mathsf{out}1}(1) + \mathsf{d}_{\mathsf{out}2}(1)$ associée à l'intervalle de validité $\Gamma_1 \bigcap \Gamma_2 =]0, 1]$ (voir figure 10.12). Après, exécution de m_3, les modèles m_1 et m_2 peuvent de nouveau être exécutés, *etc.* (voir figure 10.13).

Il est à noter que la composition des données est plus simple que ce que nous avions définis à la section 10.3.3 car les intervalles de validité Γ_1 et Γ_2 sont ici égaux.

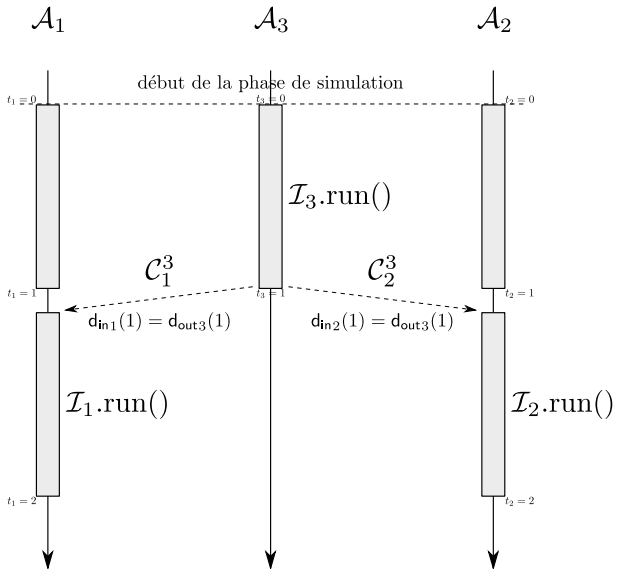


FIGURE 10.11 – Exécution des modèles m_1 et m_2.

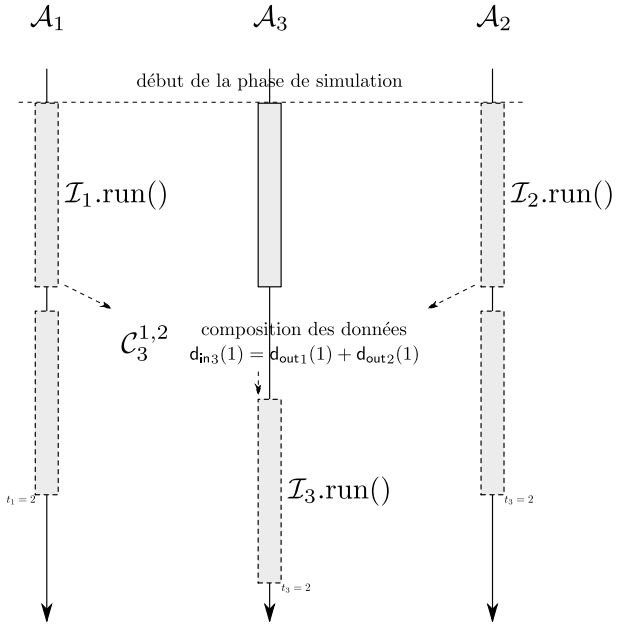


FIGURE 10.12 – Exécution du modèle m_3 après composition des variations d'énergie des moutons issues de m_1 et $m2$.

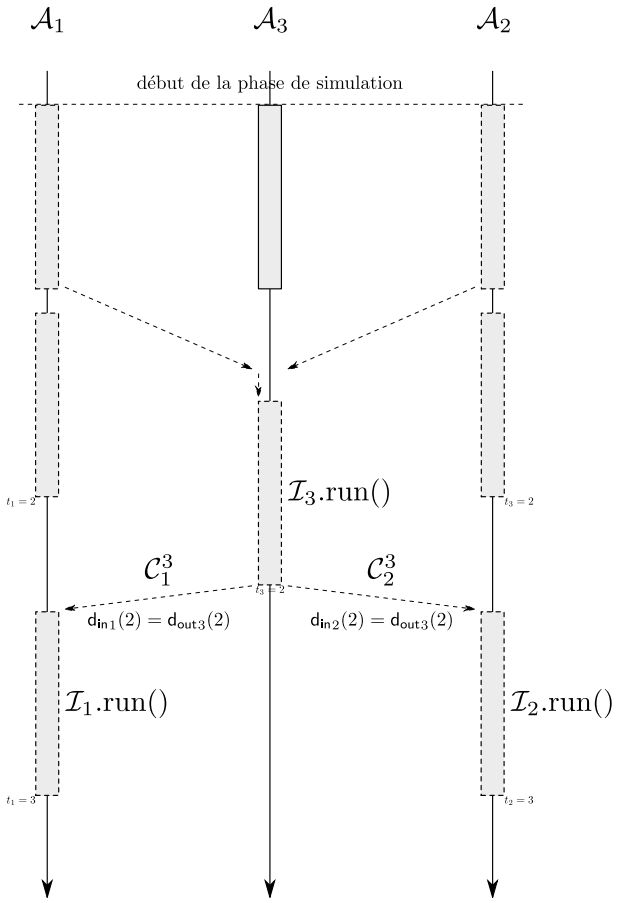


FIGURE 10.13 – Nouvelle exécution des modèles m_1 et m_2 après récupération des positions des moutons issues de m_3.

10.4.2 Récupération et analyse des résultats

Une fois la phase de simulation passée, il est possible de récupérer les résultats de simulation produits par le multi-modèle afin de les analyser et de les visualiser. Chaque *m-agent* peut demander à son logiciel de simulation de sauvegarder les résultats qui lui sont propres ou de les afficher quand cela est possible. Il peut également être intéressant de récupérer les données qui ont été échangées au cours de la simulation. Dans ce cas, il faut demander aux *artefacts-de-couplage* de restituer l'ensemble des données qui ont transité par eux.

Dans cette exemple, nous avons récupéré les positions des moutons dans chaque modèle et dans les *artefacts-de-couplage* $\mathcal{C}_1^3$ et $\mathcal{C}_2^3$ plus les variations d'énergie des moutons dans chaque modèle et dans l'*artefact-de-couplage* $\mathcal{C}_3^{1,2}$ afin de vérifier expérimentalement si les échanges d'information étaient corrects. Cela nous a permis de vérifier notre implantation et de vérifier le fonctionnement du méta-modèle AA4MM.

10.5 Prise en compte d'échelles différentes

Jusqu'à présent, nous n'avons fait aucune hypothèse concernant les échelles de temps et d'espace dans chacun des modèles élémentaires. Nous avons supposé qu'elles étaient identiques. Dans cette section nous illustrons comment le méta-modèle AA4MM facilite la prise en compte des modèles qui utilisent des échelles différentes.

10.5.1 Différentes échelles spatiales

Imaginons que la représentation de l'espace qui soit différente d'un modèle à l'autre. L'espace est représenté dans les modèles NetLogo par une grille de *patchs*. Mettons que dans le modèle m_1, un *patch* d'herbe correspond à un carré de 2×2 patchs des autres modèles (voir figure 10.14).

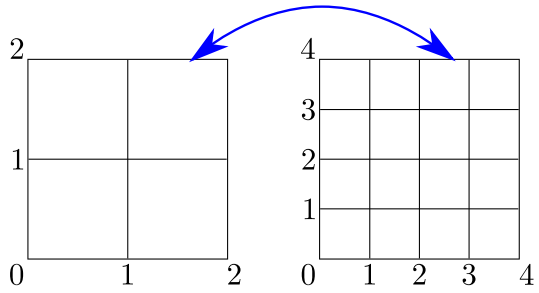


FIGURE 10.14 – Différences de représentations spatiales. À gauche, la représentation de l'espace dans le modèle m_1. À droite, la représentation de l'espace dans les autres modèles m_2 et m_3.

Dans ce cas, la position des moutons $\mathsf{d}_{\mathsf{out}_3}(t_3)$ fournie par le modèle m_3 n'est plus compatible avec celle nécessaire au fonctionnement du modèle $m_1 : \mathsf{d}_{\mathsf{in}_1}(t_1)$. Il est donc nécessaire d'introduire une fonction de conversion pour passer de $\mathsf{d}_{\mathsf{out}_3}(t_3)$ à $\mathsf{d}_{\mathsf{in}_1}(t_1)$. Celle-ci consiste simplement à diviser les coordonnées fournies par le modèle m_3 par deux.

Cette fonction de conversion peut être simplement ajoutée à notre multi-modèle déjà en place. En effet, c'est le rôle de l'*artefact-de-couplage* $\mathcal{C}_1^3$ que de gérer les conversions des flux de données. Il suffit donc d'ajouter une opération qui implante cette fonction de conversion : $\mathcal{C}_1^3.\mathrm{trans}(\mathsf{d}_{\mathsf{out}_3}(t_3))$. Quand l'*m-agent* $\mathcal{A}_1$ va lire les données depuis l'*artefact-de-couplage* $\mathcal{C}_1^3$, celui-ci doit faire attention à bien appliquer l'opération de conversion.

10.5.2 Échelle temporelle

De la même manière que précédemment, imaginons que la représentation du temps de simulation ne soit pas la même dans le modèle m_2 que dans les autres. Par exemple, un pas de simulation dans le

modèle m_2 correspond à deux dans les autres modèles m_1 et m_3 (voir figure 10.15). Dans le logiciel de simulation NetLogo, on peut dire que le modèle m_2 est exécuté pour un seul tick (on appelle une fois la procédure "go") alors que les deux autres modèles m_1 et m_3 sont exécutés pour deux ticks (on appelle deux fois la procédure "go"). Dans le méta-modèle AA4MM, cela se traduit par le fait que les *m-agents* $\mathcal{A}_1$ et $\mathcal{A}_3$ n'utilisent pas l'opération run() de leurs *artefacts-d'interface* respectifs mais l'opération run(2) qui appelle deux fois la procédure "go" dans NetLogo.

Dans ce cas, le méta-modèle AA4MM se charge automatiquement des problèmes de causalité. En effet, l'algorithme de gestion du temps de simulation sur lequel il repose (voir chapitre 9) est fait de tel manière que ce genre de problème est totalement transparent. Il n'y a donc pas d'autre modification à apporter à notre multi-modèle.

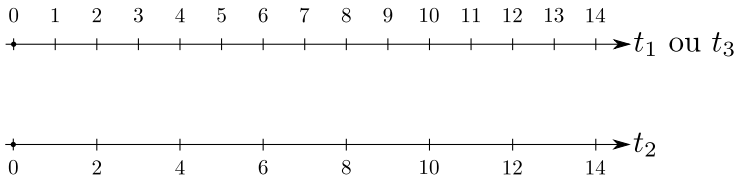


FIGURE 10.15 – Différentes politiques d'exécution entre le modèle m_2 et les deux autres modèles.

10.6 Modification du multi-modèle

Le méta-modèle AA4MM a été conçu en gardant à l'esprit que les logiciels de simulation utilisés peuvent aussi être utilisés en dehors du multi-modèle (en version *standalone*). Ainsi, les aspects conceptuels, sémantiques, syntaxiques, dynamiques et techniques propres à un modèle particulier sont séparés de ceux propres à la multi-modélisation.

10.6.1 Modification ou échange d'un modèle par un autre

La modification d'un modèle m_1 ou son remplacement par un autre modèle (m'_1) se fait facilement dans AA4MM. Nous faisons l'hypothèse que le modèle modifié ou le modèle remplaçant possède les mêmes ports d'entrée et de sortie que le modèle original.

Dans ce cas, il n'est pas nécessaire de modifier le graphe de dépendances du multi-modèle. L'utilisation d'*artefacts-d'interface* permet de facilement gérer la modification ou le remplacement d'un modèle. En effet, il est possible d'intervenir sur le modèle m_1 sans changer l'ensemble des entités propres à AA4MM. On peut par exemple changer la dynamique du modèle NetLogo en ne changeant que le contenu de la procédure "go" ou changer les paramètres intrinsèques au modèle comme la vitesse à laquelle repousse l'herbe par exemple. L'*artefact-d'interface* $\mathcal{A}_1$ et l'*m-agent* $\mathcal{A}_1$ ne sont pas modifiés.

On peut également changer le logiciel de simulation NetLogo par un autre. Dans ce cas, et sous l'hypothèse de ne pas modifier le graphe de dépendance, l'utilisation d'*artefacts-d'interface* permet de facilement changer de logiciel de simulation. Il suffit de créer un *artefact-d'interface* $\mathcal{I}_{1'}$ associé au modèle $m_{1'}$. Ensuite pour échanger les deux modèles, il suffit de spécifier à l'*m-agent* $\mathcal{A}_1$ que son *artefact-d'interface* n'est plus $\mathcal{I}_1$ mais $\mathcal{I}_{1'}$ (voir la figure 10.16). Une fois les *artefacts-d'interface* conçus, les *m-agents* peuvent gérer n'importe quel logiciel de simulation.

Dans le cas où le nouveau modèle n'a pas les mêmes ports d'entrée et de sortie que le modèle précédent, il est nécessaire de refaire le graphe de dépendance. Cela se traduit par des changements au niveau des entités du méta-modèle AA4MM comme la modification, l'ajout voire la suppression d'*artefacts-d'interface* et de *artefacts-de-couplage*.

10.6.2 Ajout et suppression d'un modèle

Concernant l'ajout et la suppression d'un modèle, il faut également distinguer deux cas. Dans cette section, nous nous plaçons dans le cas où l'ensemble des modèles gardent les mêmes ports d'entrée et

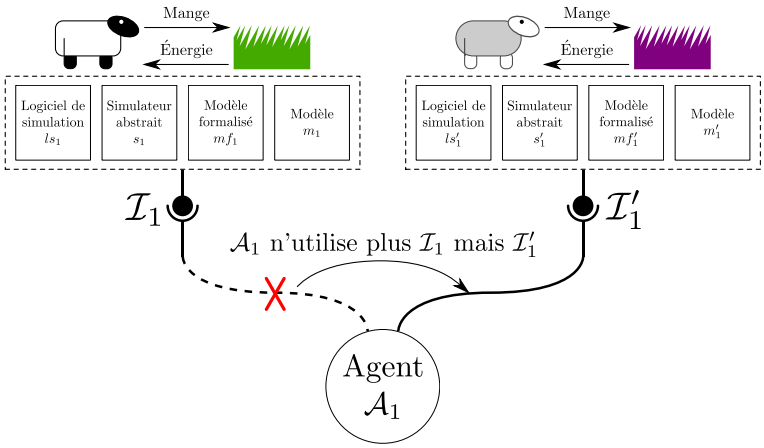
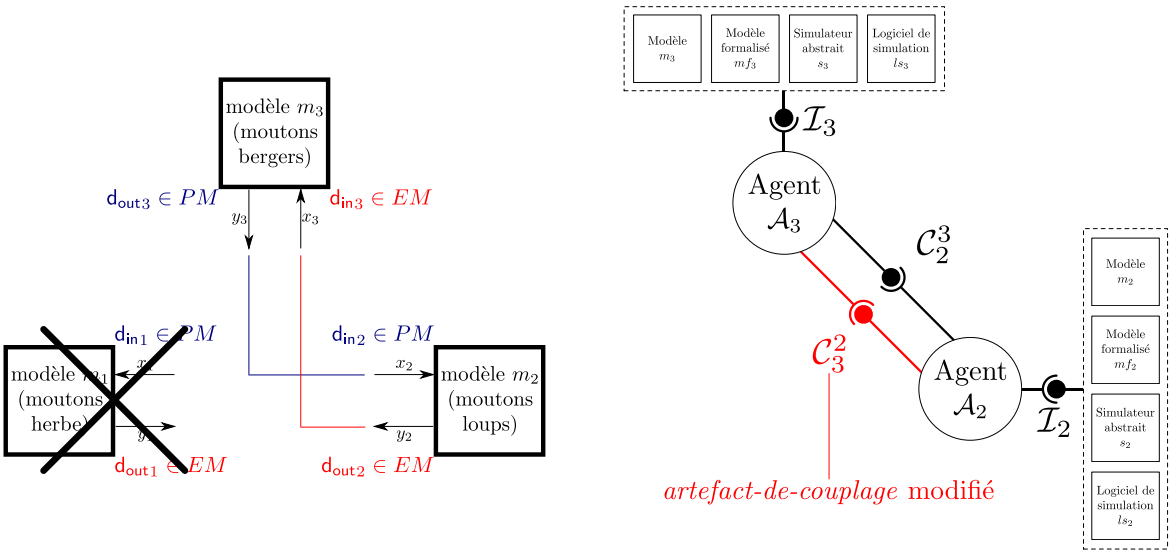


FIGURE 10.16

de sortie et l'échange des données n'est pas interrompu. C'est par exemple le cas où on supprimerait le modèle m_1 du multi-modèle. Les deux modèles restant m_2 et m_3 peuvent toujours s'échanger les positions des moutons et les variations d'énergie des moutons. Dans ce cas, il faut prendre garde à modifier le *artefact-de-couplage* $\mathcal{C}_3^{1,2}$ afin de ne plus composer les données (voir figure 10.17).

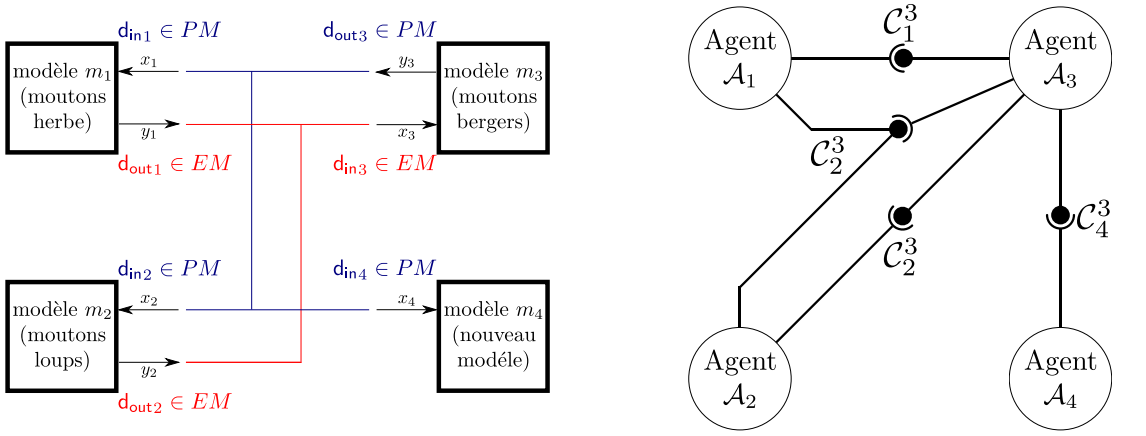


(a) Graphe de dépendance modifié.

(b) Modification de l'*artefact-de-couplage*.

FIGURE 10.17 – Exemple de suppression du modèle m_1.

On peut également ajouter un nouveau modèle m_4 qui prendrait en entrée la position des moutons par exemple. Dans ce cas, il est nécessaire de créer un *m-agent* $\mathcal{A}_4$ et un *artefact-d'interface* $\mathcal{I}_4$. Concernant l'échange des données, il est nécessaire de créer un nouvel *artefact-de-couplage* $\mathcal{C}_4^3$ qui gère l'échange des positions des moutons entre le modèle m_3 et m_4 (voir figure 10.18). Dans chacun des exemples donnés, il est nécessaire de mettre à jour l'ensemble des informations propres au graphe de dépendance (IN_1, OUT_2, L_2^1, *etc.*) afin que chaque *m-agent* sache avec quel *artefact-de-couplage* échanger les données de leurs modèles.



(a) Graphe de dépendance.

(b) Nouveaux *m-agents* et *artefacts-de-couplage*.

FIGURE 10.18 – Exemple d'ajout d'un modèle m_4. Il est à noter que dans la figure de droite nous n'avons pas ajouté les *artefacts-d'interface*, les modèles, les simulateurs et les logiciels de simulation pour plus de lisibilité.

10.6.3 Ajout d'un lien entre deux modèles

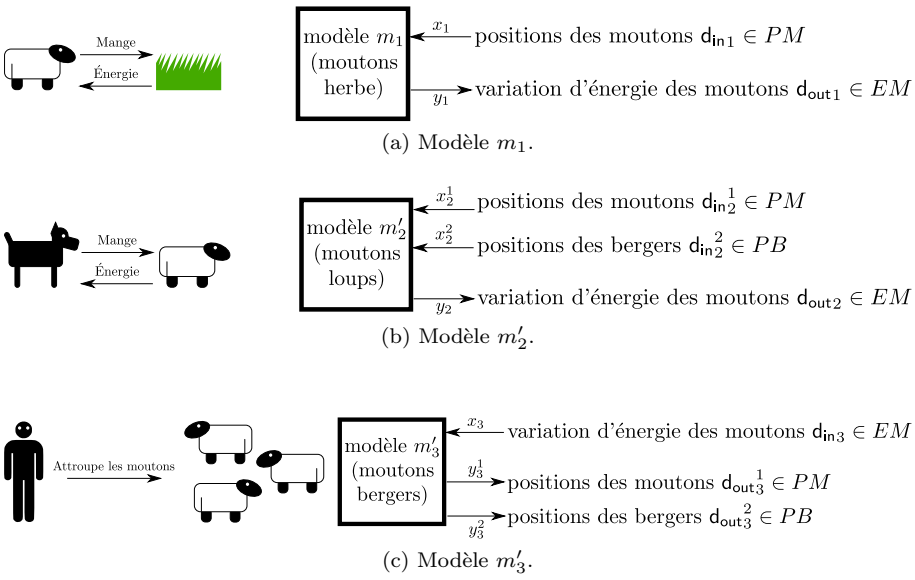
Dans les sections précédentes, nous avons fait l'hypothèse que les modifications des modèles, l'ajout et la suppression de modèles ou de logiciels de simulation n'engendrait que des modifications mineures du graphe de dépendances. Si on cherche à modifier plus en profondeur le multi-modèle, il est nécessaire de revoir le graphe de dépendance, et l'ensemble des entités du méta-modèle AA4MM.

Dans cette section, nous reprenons l'exemple des moutons, loups et bergers. Nous introduisons un nouveau phénomène qui est que les loups ont peur de la présence des bergers. Nous avons donc trois modèles : le modèle m_1 des moutons qui mangent de l'herbe et qui reste inchangé, le modèle m_2' des loups qui et une modification de m_2 et le modèle m_3' le modèle de bergers qui est une modification du modèle m_3 (voir figure 10.19).

Si on reprend le graphe de dépendance du multi-modèle, cela signifie que le modèle m_3' (moutons + bergers) envoie la position des bergers (que l'on notera PB) au modèle m_2' (loups + moutons). Ces deux modèles possèdent donc respectivement un port de sortie et un port d'entrée en plus par rapport aux modèles m_2 et m_3 précédents (voir figure 10.20).

10.7 Synthèse

Bien que relativement simple, l'exemple introduit dans ce chapitre nous a permis de montrer comment le méta-modèle AA4MM permet de construire un multi-modèle à partir de logiciels de simulation, de simulateurs et de modèles existants. L'utilisation de AA4MM impose une certaine démarche dans la multi-modélisation : définition des modèles élémentaires utilisés, définition d'un graphe de dépendances puis création des entités propres à AA4MM. La décomposition en différents niveaux (sémantique, syntaxique, dynamique, technique) permet de cibler l'ensemble des problèmes liés à cette approche de couplage de modèles. Loin de proposer un méthodologie complète de multi-modélisation, nous pensons que cet exemple peut servir de guide lors de l'utilisation de AA4MM.

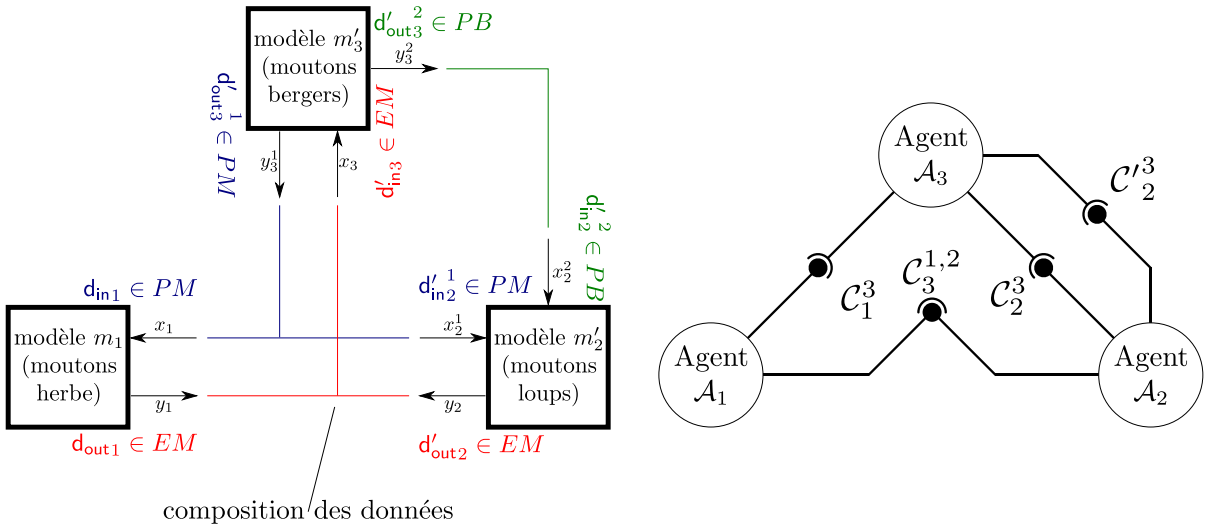


(a) Modèle m_1.

(b) Modèle m'_2.

(c) Modèle m'_3.

FIGURE 10.19 – Nouveaux modèles utilisés.



(a) Nouveau graphe de dépendance.

(b) Nouveaux *m-agents* et *artefacts-de-couplage*.

FIGURE 10.20 – Nouveau graphe de dépendance et entités du méta-modèle AA4MM. Il est à noter que le graphe de dépendance implique de créer deux *artefacts-de-couplage* entre les *m-agents* $\mathcal{A}_3$ et $\mathcal{A}_2$. Par soucis de lisibilité nous n'avons pas fait figurer les *artefacts-d'interface* ainsi que les modèles, les simulateurs, *etc.*

L'exemple présenté dans ce chapitre, nous a donné l'occasion de définir et d'illustrer ce que l'on entend par compositions des données et comment celle-ci est gérée par AA4MM à la fois aux niveaux sémantique, syntaxique et dynamique. Cet exemple a donné lieu à plusieurs implantations ce qui nous a permis de vérifier expérimentalement le déroulement de la simulation. À ce niveau, il est important de noter que les implantations effectuées possèdent leurs limites : en terme de passage à l'échelle pour la version centralisée et en terme de performances pour la version décentralisée utilisant JMS.

Une fois le multi-modèle défini et les entités propres au méta-modèle AA4MM créées, nous avons montré comment il est possible de prendre en compte des différences d'échelles aussi bien spatiales que temporelles. Nous avons également montré comment modifier le multi-modèle sans changer l'ensemble des entités du multi-modèle. À ce sujet, il est important de noter que l'utilisation d'*artefacts-d'interface* et d'*artefacts-de-couplage* rend la phase de multi-modélisation relativement modulaire. On peut par exemple changer facilement de logiciel de simulation, ajouter un nouveau modèle, *etc.*

Au chapitre suivant, Le méta-modèle AA4MM est utilisé pour concevoir un multi-modèle de réseau dynamique (chapitre 11). Ainsi, nous décrivons une partie du travail expérimental effectué durant cette thèse. Ce dernier chapitre porte sur les problématiques d'influences entre le fonctionnement des réseaux mobiles ad hoc et les comportements mobiles de leurs usagers.

Chapitre 11

Application du méta-modèle AA4MM au domaine des réseaux ambiants

Sommaire

11.1 Introduction

Notre question initiale portait sur l'étude des influences mutuelles entre le fonctionnement des réseaux ambiants et les comportements de leurs usagers. Le développement du méta-modèle AA4MM provient de notre besoin d'étudier les systèmes complexes que sont les réseaux ambiants en faisant interagir plusieurs modèles hétérogènes et en couplant des logiciels de simulation existants. L'application de ce méta-modèle au domaine des réseaux ambiants a pour but de clore ce travail de thèse.

Dans ce chapitre, les résultats apportés se situent sur deux niveaux. Au niveau applicatif, AA4MM nous a permis de mettre en place une plate-forme de multi-modélisation des réseaux ambiants. Cette plate-forme, encore au stade expérimental, nous a déjà permis de mener quelques expériences originales pour le domaine. Au niveau de notre démarche de multi-modélisation, l'application du méta-modèle AA4MM au domaine des réseaux ambiants traduit une volonté de se mesurer à la réalité applicative. Les preuves de concepts présentées précédemment se sont déroulées dans un environnement relativement favorable (présence d'une API, modèles simples, politiques d'exécutions identiques). Pour concevoir une plate-forme de multi-modélisation des réseaux ambiants, nous avons du réutiliser un logiciel de simulation qui n'avait jamais été conçu ni conceptuellement ni techniquement pour être couplé à d'autres logiciels et manipulé depuis l'extérieur.

Nous présentons d'abord le contexte et le cas d'étude que nous avons choisi afin d'appliquer AA4MM (voir section 11.2). Ensuite nous présentons un multi-modèle que nous avons conçu (voir section 11.3). Nous terminons ce chapitre par la présentation de quelques résultats expérimentaux (voir section 11.4).

11.2 Contexte et cas d'étude

Après avoir exploré les réseaux pair-à-pair dans des travaux préliminaires [Siebert et al., 2008b,a; Julien Siebert et al., 2008], le domaine des réseaux mobiles ad hoc s'est avéré présenter des cas d'étude plus abordables à la fois en terme de validité des modèles élémentaires mais aussi en terme de taille et de facilité de mise en place des simulations et des expériences réelles.

Les réseaux mobiles ad hoc se composent d'un ensemble de nœuds mobiles interconnectés par des liaisons sans fil. Ce type de réseaux à la particularité de ne pas nécessiter d'infrastructure préexistante (pas d'antenne relais). Les nœuds peuvent se connecter à des réseaux préexistants si ceux-ci existent (on parle alors de réseau *mesh*) ou s'interconnecter entre eux et créer leur propre réseau de manière émergente. Les communications se font alors de proche en proche et chaque nœud du réseau doit participer au transfert des informations en fonction de ses capacités. Les contraintes autour de ces réseaux sont relativement fortes. En terme d'énergie, les appareils utilisés (souvent des téléphones, des *netbooks*, *etc.*) utilisent des batteries et doivent donc gérer leur énergie afin de participer au réseau le temps nécessaire. En terme de communication, les appareils doivent réussir à transférer l'information d'un point A à un point B en tenant compte de la mobilité des autres nœuds et de la possibilité de déconnexions et de connexions. En terme de contrôle et de sécurité, l'ensemble des nœuds doit faire en sorte que les services et les applications fonctionnent et donnent des résultats satisfaisants malgré la mobilité, les interférences radio, les pannes, la possibilité que des nœuds corrompus [30] puissent se connecter et participer au réseau.

L'évaluation et l'ingénierie de tels réseaux sont difficiles. Les expériences grandeur nature sont relativement coûteuses à mettre en place : ce sont des réseaux qui peuvent faire intervenir un grand nombre de nœuds, dans des environnements qui sont rarement contrôlés. Il s'en suit que les expériences ne sont quasiment jamais reproductibles. Ainsi, la modélisation et la simulation sont des outils qui deviennent indispensables à la fois pour évaluer des réseaux existants mais aussi pour en concevoir de nouveaux. Cependant, ces réseaux sont complexes à modéliser car plusieurs aspects s'entremêlent et s'influencent les uns et les autres : mobilité, liaisons radio, environnement physique, protocoles, *etc.* Tenir compte de chacun de ces aspects est un travail difficile qui nécessite de faire interagir des connaissances et des modèles de plusieurs disciplines scientifiques.

Nous nous intéressons à la question des influences entre le fonctionnement des réseaux mobiles ad hoc et le comportement de leurs usagers. La mobilité des usagers joue sur les performances des réseaux mobiles ad hoc. Ce sont les usagers qui modifient la topologie du réseau via leurs mouvements. Cette mobilité n'est pas de nature aléatoire, elle dépend à la fois des buts que se sont fixés les usagers et de l'environnement qui les entoure [Song et al., 2010]. Nous introduisons une nouvelle hypothèse : nous postulons que les usagers sont également influencés par les performances du réseaux. Le manque de connexion pousse les utilisateurs de téléphonie mobile à bouger à la recherche d'un signal de meilleure intensité, l'utilisation d'un téléphone au volant d'un véhicule réduit l'attention du conducteur, *etc.* Au delà de ces exemples plutôt anecdotiques en téléphonie mobile, ces questions sont intéressantes dans le cadre des réseaux mobiles ad hoc où la mobilité des usagers joue un rôle majeur. Par exemple, "que se passe-t-il si un pourcentage des utilisateurs ont le même comportement ?", "peut-on faire cette hypothèse ?", "peut-on s'appuyer sur ce fait pour démontrer qu'un protocole va fonctionner (ou fonctionner mieux qu'un autre) ?". À l'opposé, on peut également se demander "que se passe-t-il si un pourcentage des usagers ne suivent pas le scénario prévu par les concepteurs du réseau ?", "le réseau est-il robuste ?", *etc.*

Ce type de questions nécessite de modéliser à la fois le comportement mobile des usagers et les aspects du réseau que l'on souhaite évaluer. AA4MM nous permet de coupler plusieurs modèles et de réutiliser des logiciels de simulation existants. Ainsi, l'application de AA4MM à ce cas d'étude peut se traduire par

30. On parlera de nœud corrompu quand celui-ci présente un comportement qui nuit aux performances du réseau.

le fait de coupler à la fois un modèle de la mobilité des usagers avec un modèle du réseau et de faire en sorte que ceux-ci puissent interagir.

Il est à noter que ce cas d'étude et l'application de AA4MM au domaine des réseaux mobiles ad hoc ont fait l'objet d'une collaboration avec un doctorant de l'équipe MADYNES travaillant sur ce type de réseaux : Tom Leclerc. Ainsi, les protocoles de réseaux évalués, les modèles et le logiciel de simulation du réseau (JANE [Gorgen et al., 2007]) proviennent directement du travail de recherche de Tom Leclerc. Ensuite, pour la modélisation des comportements mobiles des usagers nous nous sommes inspirés du modèle multi-agent de piéton proposé par Dirk Helbing [Helbing et al., 2001]. Nous avons déjà présenté les grandes lignes de ce modèle et ses avantages au chapitre 4. Nous avons développé un logiciel de simulation multi-agent afin d'implanter ce modèle (ainsi que des variantes) dans le but de représenter les comportements des usagers des réseaux mobiles ad hoc. Cet outil se nomme MASDYNE (pour *Multi-Agent Simulator of DYnamic NEtwork users*).

L'application du méta-modèle AA4MM consiste donc à coupler les logiciels JANE et MASDYNE et de faire interagir les modèles de réseau et des comportements des usagers. Le but expérimental est de pouvoir cibler les questions présentées précédemment. Le but pragmatique est de confronter le méta-modèle AA4MM à une réalité applicative forte.

La question qui va guider la conception d'un multi-modèle faisant interagir un modèle de réseau et des usagers est la suivante. Nous cherchons à voir ce qui se passe dans l'hypothèse où les usagers peuvent changer de comportement en fonction des performances du réseau. Contrairement aux approches courantes en terme d'évaluation des réseaux mobiles ad hoc où le comportement des usagers est globalement défini, notre but est d'évaluer le fonctionnement du réseau sous des scénarios d'usage particuliers. Par scénario d'usage nous entendons à la fois l'ensemble des comportements des usagers et l'environnement dans lequel prend place le réseau. Ces scénarios sont particuliers dans la mesure où les comportements des usagers peuvent être hétérogènes et évoluer en fonction de la situation.

Nous nous plaçons dans le cas où les usagers réagissent à la connectivité du réseau. Nous considérons que la connectivité représente le fait d'être connecté à un nœud dit "source" et de recevoir les messages qui sont émis depuis cette source. Un utilisateur va se mouvoir en fonction de son environnement et des autres usagers (évitement d'obstacles, suivi d'une personne, déplacements en groupes) mais également en fonction de la connectivité qu'il obtient. Par exemple, si la connectivité est bonne, on peut imaginer que les usagers ralentissent leur mouvement voire s'arrêtent.

À l'opposé, la topologie du réseau mobile ad hoc provient directement de la position des usagers. Les performances du réseaux vont dépendre plus ou moins de la topologie. Par exemple, une topologie éparse (avec une densité faible de nœuds) qui change rapidement va demander au réseau plus d'efforts pour maintenir une communication et des services efficaces. Les performances auront tendance à être moins bonnes qu'un réseau dont la topologie est plus dense et relativement statique. Ainsi, on postule que le comportement des usagers va influencer (en partie) les performances du réseau et que celles-ci vont en retour influencer (en partie) le comportement des usagers.

11.3 Vue générale du multi-modèle

Le multi-modèle que nous proposons est relativement simple. C'est la première pierre d'un travail exploratoire. Ainsi, bien que s'appuyant sur des modèles de réseau et des comportements des usagers tirés de la littérature, ces modèles ne correspondent peut-être pas à la réalité. Nous postulons que les usagers réagissent à la connectivité. Cette hypothèse mériterait que l'on s'y intéresse à plein temps afin de déterminer si oui ou non les usagers réagissent aux performances du réseau, dans quelle(s) condition(s) et à quelle(s) métrique(s) ils réagissent. Ceci est un travail de modélisation complet qui sort du cadre de ce travail de thèse.

11.3.1 Mutli-modèle et modèles utilisés

Nous proposons ici un premier multi-modèle dont le but est de nous permettre d'appliquer AA4MM au domaine des réseaux mobiles ad hoc et de proposer une plate-forme de multi-modélisation de ces réseaux.

Celui-ci se compose de deux modèles : un modèle des usagers (noté m_1) et un modèle de réseau (noté m_2). Nous faisons interagir ces deux modèles de la manière suivante : le modèle des usagers m_1 fournit la position des usagers au modèle du réseau m_2 et le modèle m_2 fournit l'information de connectivité au modèle m_1 (voir figure 11.1).

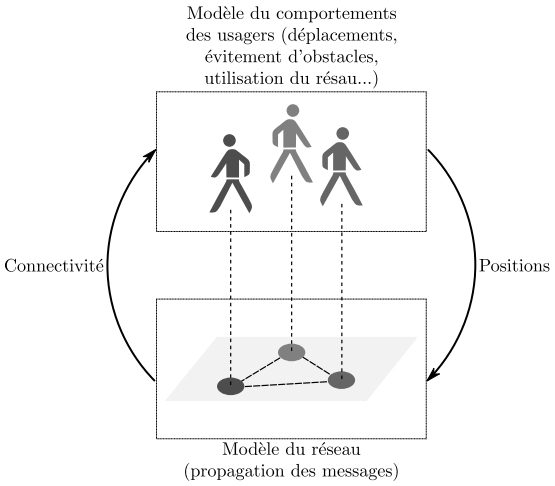


FIGURE 11.1 – Graphe de dépendance

Pour décrire les usagers, nous utilisons un modèle multi-agent inspiré du modèle de piétons proposé par Dirk Helbing [Helbing et al., 2001] et présenté au chapitre 4. Ce modèle comprend l'ensemble des usagers, leurs comportements et l'environnement dans lequel ils se déplacent. Chaque usager (noté μ) est représenté par un agent. Il possède un champ de vision (un secteur circulaire centré sur l'usager, de rayon R_μ et d'angle Θ_μ). Chaque usager possède une direction et une vitesse toutes deux représentées par un vecteur v_μ. Chaque usager possède également une vitesse maximum (un scalaire) notée v_μ^{max} et une vitesse minimum égale à 0.

En terme de comportements, les interactions usager–usager et usager–environnement sont modélisées sous forme de forces. Quand un usager approche d'un obstacle, on applique une force de répulsion sur l'agent afin que celui-ci évite l'obstacle, idem pour éviter les autres usagers. Pour suivre un autre utilisateur ou aller vers une zone donnée de l'environnement, on applique une force d'attraction. Pour appliquer plusieurs comportements en même temps il suffit d'additionner l'ensemble des forces qui s'appliquent sur l'agent (voir la présentation faite au chapitre 4).

Le modèle m_1 possède un port d'entrée x_1 dans lequel on doit fournir la connectivité de chaque nœud. Cette donnée fait correspondre à chaque utilisateur μ l'information booléenne "être connecté ou non". Ce modèle m_1 possède également un port de sortie y_1 qui fournit la position des utilisateurs. Cette donnée fait correspondre à chaque utilisateur μ sa position (x, y).

En terme d'implantation, nous avons réutilisé un logiciel de simulation propre à l'équipe MAIA (utilisé notamment dans la thèse de François Klein [Klein, 2009]) que nous avons modifié afin d'implanter les différents comportements souhaités. Il utilise une politique d'exécution par pas de temps.

Pour le réseau, nous avons utilisé des modèles de réseaux mobiles ad hoc développés par Tom Leclerc durant sa thèse et implanté dans le logiciel de simulation JANE [Gorgen et al., 2007]. Au niveau de la couche physique, ce modèle utilise un modèle de type *disc modèle* : chaque nœud (noté η) possède un disque centré sur lui même à l'intérieur duquel il peut communiquer avec les autres nœuds. Au niveau des couches supérieures, les protocoles modélisés sont ceux présents initialement dans JANE (OLSR, AODV, *etc.*) et ceux développés par Tom Leclerc (SLSF [Leclerc et al., 2010a]). Il n'existe pas de modèle d'application dans cet outil.

Le modèle représentant le réseau est noté m_2. Celui-ci possède un port d'entrée x_2 dans lequel on doit fournir la position de chaque nœud η et un port de sortie y_2 qui fournit la connectivité de chaque nœud.

Dans les expériences suivantes, on fera l'hypothèse qu'à un utilisateur μ du modèle m_1 correspond un nœud η du modèle m_2.

JANE utilise une politique d'exécution par événements. C'est un outil qui n'a pas été conçu ni pour être couplé avec d'autres logiciels ni pour être interfacé. Il n'existait pas d'API pour interfacer le logiciel. Nous avons donc développé une API pour interfacer cet outil avec les *artefacts-d'interface* du méta-modèle AA4MM. Nous avons implanté :

- la fonction run() qui permet d'exécuter un événement de simulation,
- la fonction getCurrentTime() qui récupère le temps de simulation,
- les fonction getOutputData() et setInputData() qui permettent respectivement de récupérer la connectivité de chaque nœud et d'injecter dans le modèle les positions des nœuds,
- la fonction init() qui permet d'initialiser le modèle et le simulateur,
- la fonction stop() qui permet de sauvegarder les mesures effectuées par JANE durant la simulation.

Une fois ces fonctions implantées dans JANE, il a été possible de l'intégrer grâce à AA4MM dans une plate-forme de multi-modélisation des réseaux mobiles ad hoc.

11.3.2 Application du méta-modèle AA4MM

Pour chacun des logiciels de simulation MASDYNE (ls_1) et JANE (ls_2), nous avons développé un *m-agent* et un *artefact-d'interface* (notés $\mathcal{A}_1$ et $\mathcal{I}_1$ pour MASDYNE et $\mathcal{A}_2$ et $\mathcal{I}_2$ pour JANE).

Les *artefacts-de-couplage* se chargent des données transférées entre les modèles. L'*artefact-de-couplage* $\mathcal{C}_2^1$ gère l'échange des positions des usagers depuis MASDYNE vers JANE. L'*artefact-de-couplage* $\mathcal{C}_1^2$ gère l'échange de la connectivité de chaque nœud depuis JANE vers MADYNES (voir figure 11.2). Dans ce cas d'étude, nous n'avons pas eu besoin d'appliquer des opérations de composition particulières.

Les logiciels MASDYNE et JANE utilisent des unités de longueur et de temps qui sont différentes. Une unité de longueur dans MASDYNE est équivalente à dix unités de longueur dans JANE. De même pour le temps, une unité de temps de simulation dans MASDYNE correspond à dix unités de temps dans JANE. Ainsi, la seule transformation des données qui intervient au niveau des *artefacts-de-couplage* est le changement de l'unité de longueur et de l'unité de temps d'un modèle à l'autre. La position des usagers qui provient de m_1 doit être transformée en une position correcte pour m_2. Avec le ratio une unité de longueur de m_1 égale dix unités de longueur de m_2, il suffit de multiplier les distances obtenues dans m_1 par dix. Idem pour le temps de simulation.

Un autre point important provient du fait que chaque usager présent dans le modèle de comportements doit correspondre à un nœud dans le modèle de réseau. Pour cela, les informations échangées (connectivité et positions) sont accompagnées de l'identifiant du nœud ou de l'usager. En faisant l'hypothèse qu'un usager correspond à un seul nœud, il est facile de transférer les informations entre les deux modèles.

Concernant les *m-agents*, leur comportement suit l'algorithme de simulation présenté au chapitre 9. Pour la phase d'initialisation, il est nécessaire de spécifier quelque peu les choses. En effet, le graphe de dépendance du multi-modèle peut engendrer une situation de bloquage à l'initialisation. Pour être initialisé, m_1 a besoin de la connectivité issue de m_2 alors que m_2 a besoin des positions issues de m_1. Pour résoudre cette situation bloquante, nous avons fait l'hypothèse qu'au temps initial de la simulation, l'ensemble des usagers n'est pas connecté. Cela nous permet de définir l'ensemble des paramètres du modèle m_1. Ensuite, le modèle m_2 doit attendre que le modèle m_1 soit initialisé pour obtenir l'ensemble de ses paramètres (voir figure 11.3).

11.4 Expériences menées et résultats

Le multi-modèle et le système multi-agent de gestion de la multi-modélisation décrits précédemment sont les mêmes pour chaque série d'expérience. Seuls changent les modèles des comportements des usagers m_1 et les modèles des réseaux m_2. Pour modifier ces derniers, nous changeons directement de modèle dans le logiciel de simulation concerné. Comme le graphe de dépendance n'est pas modifié, il n'est pas nécessaire de modifier le système multi-agent de gestion de la multi-modélisation.

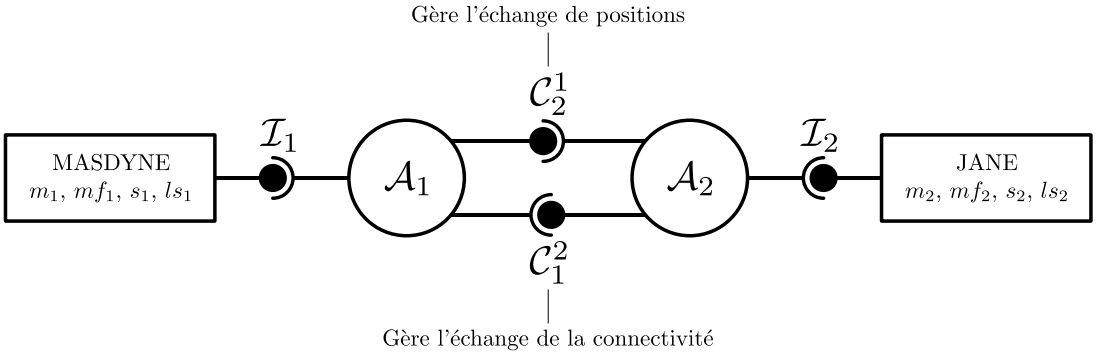


FIGURE 11.2 – Application du méta-modèle AA4MM à notre cas d'étude.

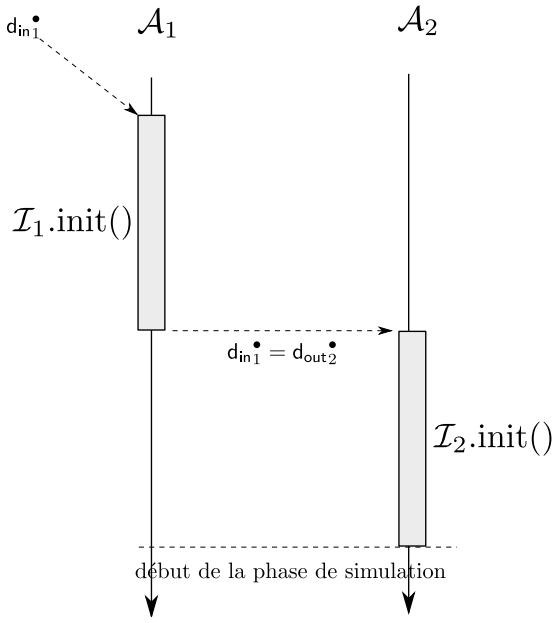


FIGURE 11.3 – Phase d'initialisation.

11.4.1 Premières séries d'expériences

Les premières séries d'expériences menées grâce au couplage MASDYNE–JANE ont eu pour but d'évaluer les protocoles proposés par Tom Leclerc. Nous ne présenterons pas ici de résultats expérimentaux, toujours en cours, qui sont le fruit du travail de Tom Leclerc et sont donc hors du champ de cette thèse. Nous donnerons ici quelques exemples de modèles et d'expériences qui sont possibles grâce à notre plate-forme de multi-modélisation.

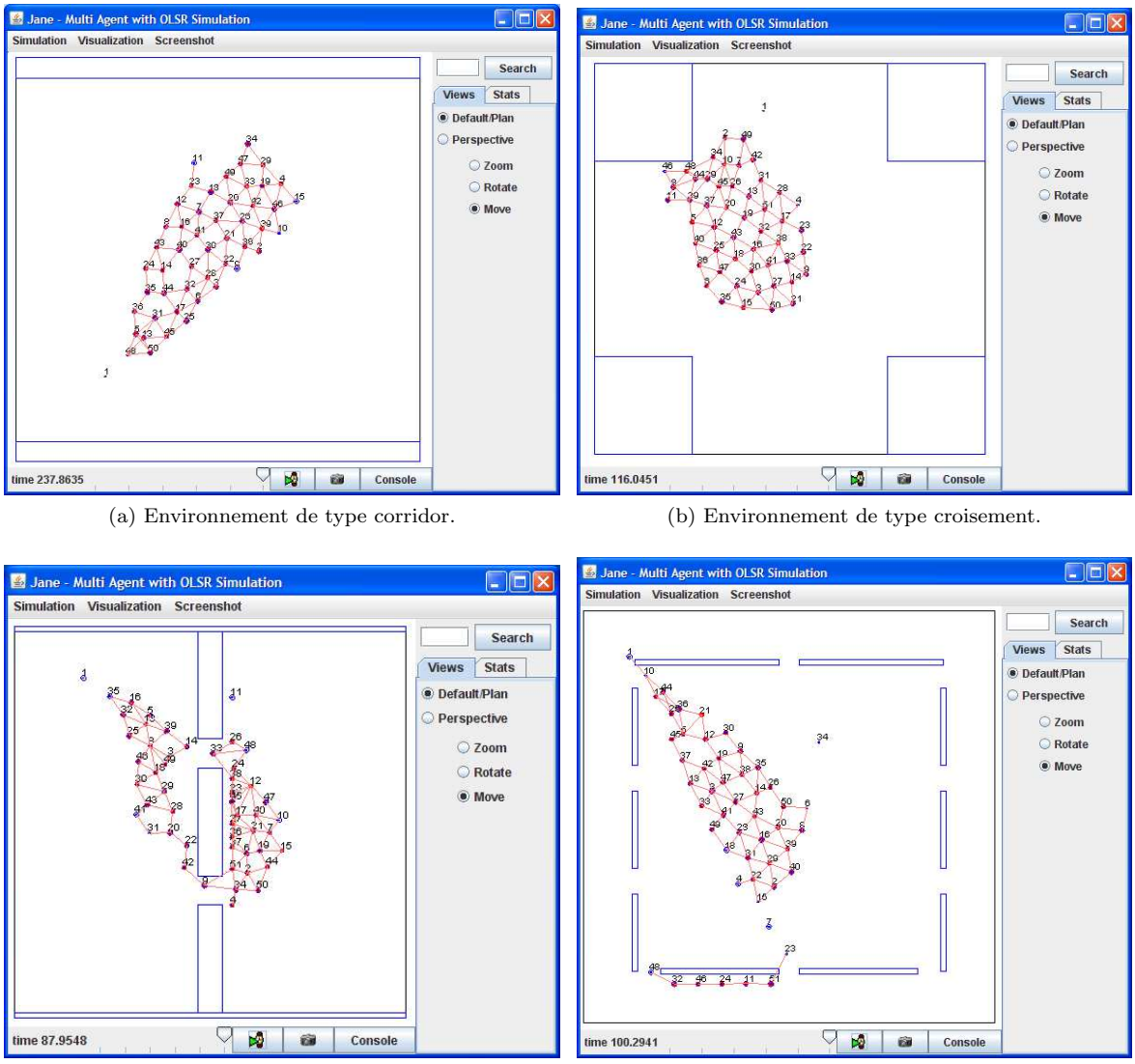
Le contexte est le projet ANR-SARAH. Dans ce projet, il était prévu d'effectuer des expériences grandeur nature de déploiement et d'utilisation d'un réseau mobile ad hoc. Le but de ces expériences est d'instrumenter la visite d'un musée : chaque usager se voit remettre en début de visite un appareil de type *smartphone* ou *PDA* (*Personal Digital Assistant*) et pendant la visite se crée un réseau mobile ad hoc qui permet à tout un chacun de bénéficier d'informations et de services propres au musée. Une partie des protocoles de routage et de découverte de services qui sont utilisés ont été mis au point et implantés par Tom Leclerc.

Au niveau de la modélisation des scénarios d'usages, les mouvements des visiteurs du musée sont particuliés : déplacements en groupes, attente devant des œuvres, suivi d'un guide, passage de portes, de corridors, *etc.* L'environnement est également particulier : salles d'exposition, bâtiment, œuvres, *etc.* Nous avons implanté au sein du simulateur multi-agent MASDYNE, un ensemble de modèles de comportements des usagers inspirés du modèle de piéton de Helbing [Helbing et al., 2001] ainsi que des modèles de mobilité plus classiques en simulation des réseaux mobiles ad hoc (marche aléatoire, suivi de chemin, *etc.*). L'avantage de la simulation multi-agent est de pouvoir représenter l'ensemble des modèles de mobilité classique plus d'en proposer de nouveaux (voir à ce sujet le chapitre 4). Le logiciel MASDYNE possède une bibliothèque de modèles de mobilité qui permet de représenter des mobilités de type : aléatoires, suivi de buts, évitement d'obstacles, déplacements en lignes, en groupes, suivi d'une personne en particulier, sortir d'une zone particulière de l'environnement, *etc.*. L'environnement se compose d'obstacles, de zones particulières et de buts. Nous avons défini quelques environnements type : corridor, croisement, portes, bâtiment, mais il est également possible d'en définir des nouveaux.

Concernant les expériences, voici un scénario d'usage qui a été testé : le guide du musée est une source qui propose un service réseau. Le comportement des usagers est le suivant : ils ne bougent pas si ils sont connectés au guide ; sinon ils se dirigent vers le guide en évitant les obstacles et les autres usagers jusqu'à ce que la connexion soit de nouveau établie. La figure 11.4 montre quelques captures d'écran tirées de ce scénario d'usage. Il est possible de facilement définir d'autres scénarios d'usage en configurant MASDYNE : changement d'environnement, changement des comportements, *etc.* De même, on peut évaluer plusieurs protocoles ou applications en utilisant les différents modèles présents dans JANE. Dans le cas où on souhaiterait ajouter ou modifier un logiciel de simulation, il faut redéfinir le graphe de dépendance et modifier ou concevoir des *m-agents*, des *artefacts-d'interface* et des *artefacts-de-couplage* (voir le chapitre 10). La seule réelle difficulté dans l'application de AA4MM est l'interfaçage d'outils qui n'ont pas été conçus pour être interfacés.

11.4.2 Deuxièmes séries d'expériences

La deuxième série d'expériences illustre un scénario d'usage où la population des usagers est hétérogène. Le but est de montrer les types d'expériences qu'il est possible de faire lorsque l'on cherche à évaluer le fonctionnement du réseau quand les usagers ne suivent pas tous le même comportement. Le contexte n'est plus ici un musée où le but est de suivre le guide mais celui-ci est un peu plus général dans le sens où il existe des sources qui émettent des messages et l'ensemble des usagers doivent diffuser l'information. Nous introduisons différents types de comportements au niveau des usagers. Le but est de voir comment l'approche de multi-modélisation permet d'étudier l'impact de l'hétérogénéité de la population sur les performances du réseau.



(a) Environnement de type corridor.

(b) Environnement de type croisement.

(c) Environnement de type portes.

(d) Environnement de type place ou bâtiment.

FIGURE 11.4 – Captures d'écran de JANE couplé avec MASDYNE (ici nous présentons l'interface graphique de JANE). Chaque nœud est représenté par un point, les liens entre les nœuds sont représentés en rouge et les obstacles en bleu.

Modèles utilisés

Le modèle représentant les usagers est un modèle multi-agent. Chaque usager est représenté par un agent. Comme précédemment, chaque usager possède un champ de vision, une vitesse courante, une vitesse maximum et une vitesse minimum égale à 0 (voir figure 11.5). Nous proposons un ensemble de cinq types de comportements différents que l'on note : source, 1, 2, 2bis et 3.

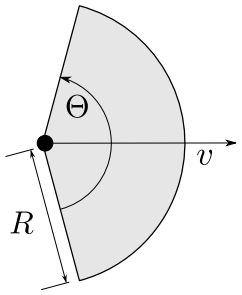


FIGURE 11.5 – Représentation d'un utilisateur

Un usager peut être de type source. Dans ce cas, il reste immobile et envoie des messages avec une certaine fréquence notée T_{msg}. Un usager qui n'a pas un comportement de type source est mobile. Le modèle de mobilité adopté ici est de type aléatoire. L'usager choisit régulièrement une nouvelle vitesse et une nouvelle direction aléatoirement (de manière uniforme) avec une période notée T_{alea}. Si un usager rencontre un obstacle, le bord de l'environnement ou un autre usager, celui-ci l'évite en appliquant une force de répulsion. Le comportement de ces usagers peut changer en fonction de la connectivité du réseau. Les usagers de type 1 s'immobilisent lorsqu'ils sont connectés au réseau. Les usagers de types 2 et 2bis divisent leurs vitesses respectivement d'un facteur 6 et 3 lorsqu'ils sont connectés au réseau. Les usagers de type 3 ne réagissent pas à la connectivité du réseau et continuent leurs mouvements. Le tableau suivant (voir figure 11.6) résume ces différents comportements.

Type de comportements		**si réseau déconnecté**	**si réseau connecté**
Comportement	source	immobilisation	immobilisation
Comportement	1	bouge aléatoirement	immobilisation
Comportement	2	bouge aléatoirement	ralentissement (vitesse $v_\alpha = \frac{v_\alpha}{6}$)
Comportement	2bis	bouge aléatoirement	ralentissement (vitesse $v_\alpha = \frac{v_\alpha}{3}$)
Comportement	3	bouge aléatoirement	

FIGURE 11.6 – Différents types de comportements des usagers

En terme de notations, le nombre total d'usagers que l'on souhaite représenter est noté n. Le nombre d'usagers possédant le comportement de type 1 est noté n_1, de type 2 est noté n_2, *etc.* Le nombre total d'usagers est égal à la somme du nombre d'usagers de chaque type $n = n_1 + n_2 + n_{2bis} + n_3 + n_{source}$.

Le modèle de réseau que nous utilisons est relativement simple. Au niveau de la couche physique, nous utilisons un modèle de type *disc model*. Ensuite, au niveau protocolaire, nous utilisons un modèle de *broadcast*. Les sources diffusent des messages périodiquement. L'ensemble des nœuds récepteurs appliquent l'algorithme suivant. Quand un message est nouveau, le nœud le retransmet sinon il le laisse tomber.

Protocole expérimental

Une expérience correspond à une simulation numérique. Elle est définie par un environnement Σ, par l'ensemble des comportements affectés à chacun des usagers et par une condition initiale (notée *cond*). Cette dernière correspond au placement de l'ensemble des usagers dans l'environnement et à la graine

utilisée pour initialiser le générateur aléatoire des simulateurs. La figure suivante (figure 11.7) récapitule l'ensemble des paramètres à fixer lors d'une expérience.

Paramètre		**Notation**
Paramètres globaux	Nombre total d'utilisateurs	n
	Nombre de sources	$n_{sources}$
	Nombre d'utilisateurs de type 1	n_1
	Nombre d'utilisateurs de type 2	n_2
	Nombre d'utilisateurs de type 2bis	n_{2bis}
	Nombre d'utilisateurs de type 3	n_3
Usagers	Champ de vision des usagers (rayon, angle)	(R, Θ)
	Vitesse maximale des usagers	v_{max}
	Période de changement de vitesse et de direction	T_{alea}
Réseau	Période d'envoi de messages	T_{msg}
	Rayon d'émission et de réception	$radius$

FIGURE 11.7 – Récapitulation des paramètres à fixer pour une expérience

Une série d'expériences correspond à un ensemble d'expériences dans lesquelles seule change la condition initiale. Une série d'expériences est donc définie par un environnement Σ, par l'ensemble des comportements affectés à chacun des usagers et par un ensemble de conditions initiales (noté $\mathbb{C}_{init}$). On note N_{exp} le nombre d'expériences dans une série. Nous avons mené plusieurs séries d'expériences. Pour chacune, nous avons fait varier les différents types de comportements possibles pour les usagers. Dans chaque série, il y a 100 utilisateurs et 4 sources qui envoient des messages toutes les secondes. Initialement, les nœuds sont placés dans l'environnement de manière aléatoire. Chaque série d'expériences contient 50 conditions initiales différentes. La figure 11.8 présente la valeurs des paramètres expérimentaux.

Paramètre		**Notation**	**Valeur**
Paramètres globaux	Nombre d'expériences par série	N_{exp}	50
	Nombre total d'utilisateurs	n	104
	Nombre de sources	$n_{sources}$	4
Usagers	Champ de vision des usagers (rayon, angle)	(R, Θ)	$R = 100m$, $\Theta = 2\pi$
	Vitesse maximale des usagers	v_{max}	$3ms^{-1}$
	Période de changement de vitesse et de direction	T_{alea}	$10s$
Réseau	Période d'envoi de messages	T_{msg}	$1s$
	Rayon d'émission et de réception	$radius$	$50m$

FIGURE 11.8 – Valeur des paramètres expérimentaux

Dans chaque série d'expériences, nous avons mesuré le nombre de nœuds connectés au cours du temps de simulation (toutes les secondes).

Usagers de types 1 et 3

Dans cette série d'expérience, nous mettons en scène des usagers dont les comportements sont de type 1 (qui s'arrêtent en cas de connexion) et de type 3 (qui ne réagissent pas en cas de connexion). Nous faisons varier le nombre d'usagers de chaque catégorie (n_1 et n_3). Le nombre total d'usager étant toujours fixe, on a l'équation 11.1.

$$n = n_{source} + n_1 + n_3 \tag{11.1}$$

Avec les valeurs de $n_{source} = 4$ et $n = 104$, on obtient l'équation 11.2.

$$\frac{n_1}{100} = 1 - \frac{n_3}{100} \tag{11.2}$$

Quand le ratio $n_1/100 = 0\%$, tous les usagers, hormis les sources, ne réagissent pas à la connectivité et continuent de bouger aléatoirement. On obtient alors l'équivalent d'une expérience dans laquelle tous les usagers sont modélisés par un modèle de mobilité de type *random waypoint*. À l'opposé, quand le ratio $n_1/100 = 100\%$, tous les usagers, hormis les sources, réagissent à la connectivité et s'immobilisent lors de la connexion. Étant donné le modèle du réseau utilisé, une fois la connexion établie et une fois l'usager immobilisé, celui-ci reste immobilisé. On obtient alors un cas particulier où la topologie du réseau se stabilise de plus en plus jusqu'à ce que tous les nœuds soient connectés.

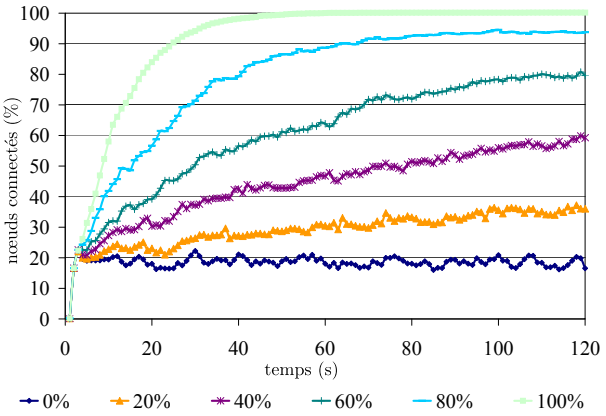


FIGURE 11.9 – $n_1/100$ varie de 0 à 100% par tranche de 20 points.

Cet exemple est relativement simple mais il illustre parfaitement le type de résultats que l'on cherche à produire. On observe ici que si $n_1/100 = 0$, l'ensemble des usagers bougent de manière aléatoire et environ 20% des usagers sont connectés au cours du temps. Si $n_1/100 = 100$, on obtient l'extrême inverse où tous les utilisateurs s'immobilisent lorsqu'ils sont connectés. À ce moment, le réseau n'est plus mobile. Ces deux extrêmes sont faciles à prédire et ces résultats n'apportent que peu de connaissance sur le fonctionnement du réseau.

Là où cette expériences est intéressante c'est dans le cas où une certaine proportion des usagers ne suit pas le comportement moyen (la marche aléatoire par exemple). Sur cet exemple, on observe que si une personne sur cinq suit le comportement "s'arrêter lorsque l'on est connecté", alors le nombre de nœuds connectés aux réseau passe de moins de 20% à 35% (presque le double).

Usagers de types 2 et 3

Cette série d'expérience est la suite de la précédente. Au lieu d'avoir une catégorie d'usagers qui s'arrêtent lorsqu'ils sont connectés, ceux-ci ralentissent (comportement de type 2, la vitesse est divisée par six). Le reste des usagers sont de type 3 : ils ne réagissent pas en cas de connexion. Nous faisons varier le nombre d'usagers de chaque catégorie (n_2 et n_3). De la même manière que précédemment, le nombre total d'usagers étant toujours fixe, on obtient l'équation 11.3.

$$\frac{n_2}{100} = 1 - \frac{n_3}{100} \tag{11.3}$$

Quand le ratio $n_2/100 = 0\%$, tous les usagers, hormis les sources, ne réagissent pas à la connectivité et continuent de bouger aléatoirement. On obtient, de nouveau, l'équivalent d'une expérience dans laquelle tous les usagers sont modélisés par un modèle de mobilité de type *random waypoint*. À l'opposé, quand le ratio $n_2/100 = 100\%$, tous les usagers, hormis les sources, réagissent à la connectivité et ralentissent leurs mouvements lors de la connexion (la vitesse est divisée par six). L'usager continue de se mouvoir pendant le temps où celui-ci est connecté. Si durant ce mouvement, il se produit une déconnexion alors l'usager reprend son comportement initial.

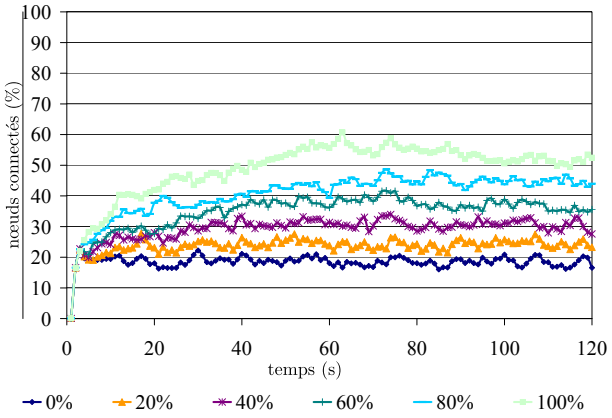
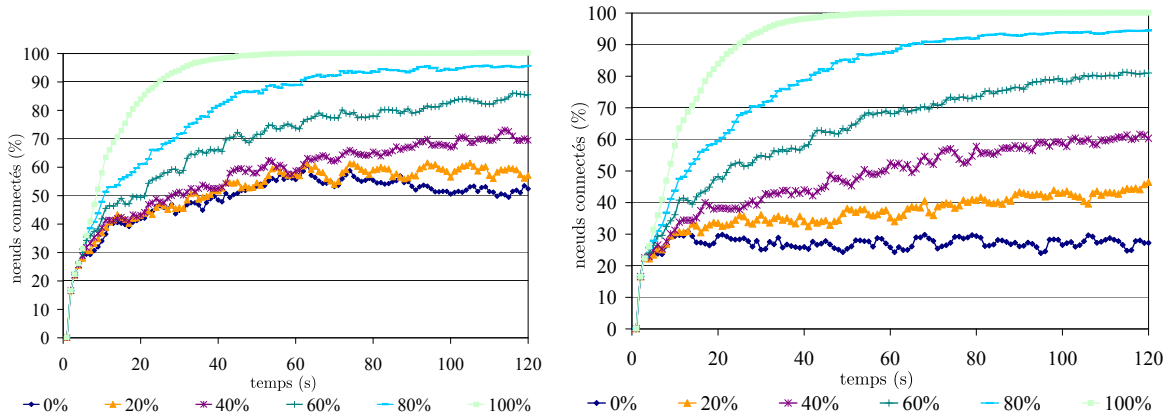


FIGURE 11.10 – $n_2/100$ varie de 0 à 100% par tranche de 20 points.

On observe ici que le gain en terme de nombre de nœuds connecté est moins important que dans l'expérience précédente. Cela paraît logique car ici le réseau est toujours mobile. La différence est que la topologie évolue plus ou moins rapidement en fonction des décisions que prennent les usagers. Dans cette expérience, nous avons seulement modifié le modèle de comportement des usagers. En aucun cas, les logiciels de simulation, le multi-modèle et le système multi-agent de gestion de la multi-modélisation proposé par AA4MM n'ont été modifiés.

Autres expériences

Nous avons également fait d'autres expériences qui prennent la suite des précédentes. Nous avons mis en scène des usagers dont les comportements sont de types 1 et 2 et, dans d'autres expériences de type 1 et 2bis (voir figure 11.11). L'idée est de parcourir l'ensemble des cas possibles en utilisant les modèles de comportements présentés précédemment.



(a) $n_1/100$ varie de 0 à 100% par tranche de 20 points. (b) $n_{2bis}/100$ varie de 0 à 100% par tranche de 20 points.

FIGURE 11.11 – Autres résultats

Bilan des expériences

Certes l'exemple présenté est simple : le modèle de réseau est une simple diffusion de l'information sans construction de tables de routages ou d'optimisation du trafic. De même les comportements des usagers sont de l'ordre de la spéculation et mériteraient de faire l'objet d'une étude plus approfondie.

Cependant, au travers de ces expériences, nous avons montré que d'une part le méta-modèle AA4MM est effectivement appliqué au domaine des réseaux mobiles ad hoc et est utilisé en dehors du seul cadre de cette thèse. D'autre part, cet outil nous permet d'aborder notre question initiale de l'étude des influences mutuelles entre les performances des réseaux ambiants et les comportements des leurs usagers.

Il est à noter que lorsque le multi-modèle est défini, il est facile, moyennant l'interfaçage des logiciels de simulation, d'implanter les différentes entités définies par AA4MM (*artefacts-de-couplage*, *artefacts-d'interface* et *m-agents*). Les problèmes liées à la multi-modélisation sont gérés une fois pour toute par AA4MM et le modélisateur peut se concentrer sur le système étudié. Par exemple, dans les expériences présentées le modèle de réseau n'est jamais modifié et seuls sont modifiés les modèles des usagers qui sont implantés dans MASDYNE.

Dans cet exemple, le fait de faire interagir un modèle du réseau avec un modèle multi-agent des usagers nous permet de mesurer à la fois des métriques "classiques" du réseau (nombre de nœuds connectés, nombre de sauts jusqu'à la source, nombre de messages reçus, *etc.*), des métriques propres à la mobilité (densité en fonction du temps, vitesse moyenne des usagers, vitesse instantanée, *etc.*) mais aussi des métriques propres aux comportements des utilisateurs comme la qualité d'expériences.

Ce type d'approche ouvre la voie à des expériences originales qui devront permettre d'obtenir une meilleure compréhension de la complexité engendrée par les réseaux mobiles ad hoc. Concevoir des expériences de simulation pour répondre aux questions comme "que se passe-t-il si un pourcentage des utilisateurs ont le même comportement ?", "peut-on faire cette hypothèse ?", "peut-on s'appuyer sur ce fait pour démontrer qu'un protocole va fonctionner (ou fonctionner mieux qu'un autre) ?", "que se passe-t-il si un pourcentage des usagers ne suivent pas le scénario prévu par les concepteurs du réseau ?", "le réseau est-il robuste ?", *etc.* n'est plus un problème en soi.

Le véritable problème qui se pose provient maintenant de la prise en compte des différents aspects (réseau, environnement, usagers) et de leurs influences mutuelles. Si la mise en place du multi-modèle est simplifiée, le protocole expérimental et l'analyse des résultats produits deviennent plus compliqués. Il est alors nécessaire de bien poser le problème et le système étudié et de valider les modélisations par des expériences réelles.

11.5 Synthèse

Dans ce chapitre nous avons appliqué le méta-modèle AA4MM au domaine des réseaux mobiles ad hoc. Le but était de revenir à la question initiale qui nous préoccupait : l'étude des influences mutuelles entre réseaux ambiants et comportements des usagers ; et de montrer comment AA4MM peut faire face à une réalité objective. Nous nous sommes placés dans le cas d'étude des influences entre la mobilité des usagers et les performances des réseaux mobiles ad hoc. Nous avons posé les premières pierres d'une plate-forme de multi-modélisation des réseaux mobiles ad hoc en couplant deux logiciels de simulation : MASDYNE pour la modélisation et les simulations des comportements des usagers et JANE pour la modélisation et la simulation des réseaux mobiles ad hoc. Cette plate-forme, bien qu'encore au stade d'un travail de recherche, est utilisée dans le cadre des travaux de recherches menées par l'équipe MADYNES sur les réseaux mobiles ad hoc. Il est également important de noter que ce travail expérimental n'aurait pu avoir lieu sans la collaboration de Tom Leclerc.

11.5.1 Bilan

Les résultats des travaux expérimentaux effectués dans le contexte des réseaux mobiles ad hoc sont de l'ordre à la fois du cadre applicatif et de la démarche de multi-modélisation dans le cadre des réseaux ambiants.

En terme d'approche de multi-modélisation, AA4MM apporte la possibilité de facilement réutiliser et de coupler des outils de modélisation et de simulation existants qui n'ont pas forcément été pensés pour interagir. AA4MM gère l'ensemble des problèmes liés à la multi-modélisation, notamment les problèmes de simulation distribuée. La seule réelle difficulté concerne l'interfaçage de logiciels de simulation qui ont été pensés comme un seul bloc. Cependant, cette difficulté est toute relative car l'interface nécessaire au bon fonctionnement d'AA4MM ne contient que six fonctions relativement génériques.

Dans le contexte de l'évaluation des réseaux mobiles ad hoc, AA4MM nous a permis de coupler les logiciels JANE et MASDYNE et de proposer des expériences relativement originales pour le domaine. L'approche multi-agent utilisée par MASDYNE nous a permis de proposer des modèles de comportements hétérogènes qui peuvent réagir à leur environnement ainsi que des environnements variés. La stricte séparation entre le modèle et le multi-modèle et la modularité de AA4MM nous ont également offert la possibilité de composer des scénarios d'usages de plus en plus évolués. De ce fait les expériences peuvent devenir plus expressives et plus nombreuses. Il est donc nécessaire de bien cibler le problème et le système étudié ainsi que les expériences réelles nécessaires à la validation des résultats de simulation.

En terme applicatif, il est à noter que la plate-forme de multi-modélisation des réseaux ambiants est utilisée en dehors du simple cadre de cette thèse. Nous avons proposé quelques modèles de comportements d'usagers et quelques modèles d'environnement. Au niveau des expériences menées, nous avons pris le parti d'explorer des questions qui ne ciblent pas un comportement moyen du système mais des cas particuliers, avec des comportements hétérogènes et des influences mutuelles entre le réseaux et ses usagers.

11.5.2 Limites

Au niveau des limites, nous ne reviendrons pas sur les performances des implantations (voir chapitre 10). En effet, au jour d'aujourd'hui, en terme de performance, la version centralisée de AA4MM et nettement plus performante que la version décentralisée basée sur JMS. Cela est un réel problème lorsque l'on aborde des problèmes qui nécessitent des modèles détaillés et de grande taille.

Au niveau des expériences, si l'application de AA4MM nous permet d'envisager de nombreuses expériences, nous avons du nous contenter d'exemples relativement simples. En effet, un travail expérimental ciblé sur les influences mutuelles entre la mobilité des usagers et les performances des réseaux mobiles ad hoc demande, en plus d'outils comme AA4MM, de valider les modèles utilisés et les résultats de simulation face à des expériences réelles, ce que par manque de temps nous n'avons pu faire. Ainsi, les expériences que nous avons menées sur les influences mutuelles entre le réseau et les usagers se sont arrêtées au stade de la preuve de concepts.

11.5.3 Perspectives

Les perspectives au niveau des réseaux mobiles ad hoc sont nombreuses. Tout d'abord, il est à signaler une véritable envie de proposer une plate-forme de *benchmark* des réseaux mobiles ad hoc basée sur AA4MM. Il serait alors intéressant de définir des scénarios d'usages standards.

Nous pensons également, intégrer d'autres logiciels de simulation comme NS3 ou Omnet qui sont des outils moins spécifiques que JANE. De même, il serait également intéressant d'utiliser d'autres modèles d'environnement qui intégreraient des informations issues de système d'information géographique, voire des modèles de propagation d'onde radio.

En terme de cadre applicatif, nous pensons que notre démarche de multi-modélisation peut également présenter des contributions intéressantes dans le domaine des réseaux de capteurs sans fil, domaine relativement proche des réseaux mobiles ad hoc.

Chapitre 12

Conclusion

Sommaire

12.1 Bilan

12.1.1 Résumé

Dans cette thèse, nous nous sommes initialement intéressés aux phénomènes d'influences mutuelles qui peuvent avoir lieu entre un réseau ambiant et les comportements de leurs usagers. Cette problématique nous a poussé à définir des besoins pratiques forts : réutiliser des modèles et des logiciels de simulation existants, les rendre interopérables et les faire interagir, et à recentrer notre travail de recherche dans le cadre de la multi-modélisation des systèmes complexes.

Il y a donc deux volets dans ce manuscrit. Le premier concerne les aspects théoriques en lien avec la multi-modélisation, la simulation multi-agent et les systèmes complexes. Le second concerne plus l'évaluation des réseaux ambiants et les phénomènes d'influences mutuelles entre les performances de ces réseaux et les comportements de leurs usagers.

Cette dichotomie provient à la fois du contexte pluridisciplinaire et quelque peu particulier de cette thèse (ce travail a été mené au sein de deux équipes MAIA et MADYNES) mais également d'une hypothèse scientifique forte et sous-jacente qui est que l'étude des systèmes complexes se doit de prendre en compte simultanément plusieurs points de vue sur le système étudié.

Dans ce travail de thèse cela s'est traduit par la volonté et le besoin de réutiliser des modèles, des simulateurs et des logiciels de simulations issus de différents domaines et pas forcément pensés pour être couplés ensembles. Nous avons exploré les réponses conceptuelles que peut apporter le paradigme des systèmes multi-agents dans ce contexte et nous les avons couplées avec les solutions plus techniques apportées par le domaine de la multi-modélisation et par le domaine du couplage de simulation. Nous avons également appliqué ces réponses au problème de l'évaluation des réseaux ambiants.

12.1.2 Contributions

Une première contribution de ce manuscrit est la proposition du méta-modèle AA4MM qui aborde la multi-modélisation d'un phénomène complexe comme la construction d'une société de modèles, de

simulateurs et de logiciels en interaction. L'originalité de AA4MM est de facilement intégrer les réponses hétérogènes aux différents niveaux de questions liés à la multi-modélisation (conceptuel, sémantique, syntaxique, dynamique et technique) dans une approche multiagent homogène.
En s'appuyant sur le méta-modèle Agents & Artefacts, AA4MM offre la possibilité de facilement réutiliser, de rendre interopérable et modulaire des outils de modélisation et de simulation existant et hétérogènes. AA4MM s'inscrit alors directement dans la lignées des approches de modélisation et de simulation orientées composants, la vision multi-agent en plus.
Au niveau méthodologique, AA4MM s'accompagne d'une décomposition des questions liées à la multi-modélisation en cinq niveaux qui permet au modélisateur de clairement spécifier chacun des aspects du multi-modèle notamment les changements d'échelles entre différents modèles.
En terme de simulation, nous avons proposé un algorithme de simulation distribuée totalement décentralisé adapté à la vision multi-agent (coordination via l'environnement).
Nous avons également illustré et prouvé les propriétés du méta-modèle que ce soit au travers d'exemples relativement simples ou grâce à des preuves formelles.

Une deuxième contribution intervient au niveau des réseaux ambiants. Nous avons appliqué le méta-modèle AA4MM au problème des influences entre la mobilité des usagers et le fonctionnement des réseaux mobiles ad hoc.
AA4MM nous a servi pour poser les premières pierres d'une plate-forme générique de multi-modélisation de ces réseaux. Celle-ci prend en compte à la fois un modèle du réseau issus du domaine des réseaux informatiques ubiquitaires et un modèle du comportement des usagers issus du domaine des simulations sociales urbaines.
Il est à noter que cette application fait interagir des logiciels qui existaient au préalable et qui n'étaient pas initialement conçus pour interagir ensemble.
Notre démarche a également été de proposer une approche d'évaluation qui tient compte des aspects connexes au réseau, d'évaluer des phénomènes complexes jusqu'à présent très difficiles à aborder et de proposer les outils qui permettent construire de telles modélisations.

12.2 Discussions

12.2.1 Processus de multi-modélisation et AA4MM

Le méta-modèle AA4MM permet de réutiliser des logiciels de simulation existants, de les rendre interopérables et d'en faire une "société de modèles et de simulateurs en interaction". Bien qu'il réponde aux contraintes que nous nous sommes fixées, il est des points que nous n'avons pas abordé dans ce manuscrit et qu'il nous semble important de discuter ici, afin de faire ressortir encore d'avantage les possibilités offertes par la vision multi-agent de la multi-modélisation.

Visualisation et analyse des résultats de simulation

La premier point que nous souhaitons éclairer est l'utilisation d'outils annexes au multi-modèle afin de visualiser et d'interpréter les résultats de simulation. En effet, le processus de multi-modélisation ne comprend pas seulement un ensemble de modèles en interaction. Pour que la simulation ait du sens par rapport au système étudié, il faut y ajouter des outils de visualisation, d'analyse des résultats et aussi de stockage.

En réutilisant des logiciels de simulation existants, il est probable que certains de ces outils soient déjà présents. Cependant rien ne nous garantit que de tels outils soient présent pour l'ensemble du multi-modèle, que ces outils puissent fonctionner de concert et que les résultats de simulation émis par chaque logiciel soient compatibles aux niveaux sémantique, syntaxique et dynamique. Il est par exemple nécessaire de formater les données, de les combiner ensemble en fonction des échelles de temps et d'espaces qu'elles utilisent.

L'approche multi-agent du méta-modèle AA4MM nous permet sans trop de difficulté de considérer chaque outil de visualisation, d'analyse et de stockage comme une entité autonome à part entière. Ainsi,

on peut appliquer le même raisonnement qu'avec les logiciels de simulation et considérer que les outils annexes doivent être interfacés chacun à un *m-agent* particulier qui va récupérer et échanger des données soit avec les *m-agents* en charge de la simulation ou avec les *m-agents* en charge des opérations de visualisation, stockage ou analyse.

Pour ce faire, il est nécessaire de définir pour chaque modèle des ports de visualisation (ou d'observation). Cela nécessite de modifier les *artefacts-d'interface* existants car ceux-ci devront permettre au *m-agent* en charge de la simulation du modèle de récupérer les données intéressantes (en terme de visualisation ou d'analyse). Ces *m-agents* pourront alors transférer ces données aux *m-agents* en charge des opérations de visualisation, stockage ou analyse. Le comportement de ces nouveaux *m-agents* sera différent de ceux qui gèrent la simulation. De plus, il faudra définir de nouveaux *artefacts-de-couplage* pour gérer ces nouveaux flux de données. L'utilisation des intervalles de validité est utile lorsqu'il s'agit de combiner des données issues de deux logiciels de simulation différents qui utilisent des échelles temporelles différentes.

On voit ici que l'ajout d'outils annexes à la multi-modélisation ne pose pas de problème fort au méta-modèle AA4MM. La figure 12.1 illustre un exemple d'application potentiel de AA4MM dans cette optique.

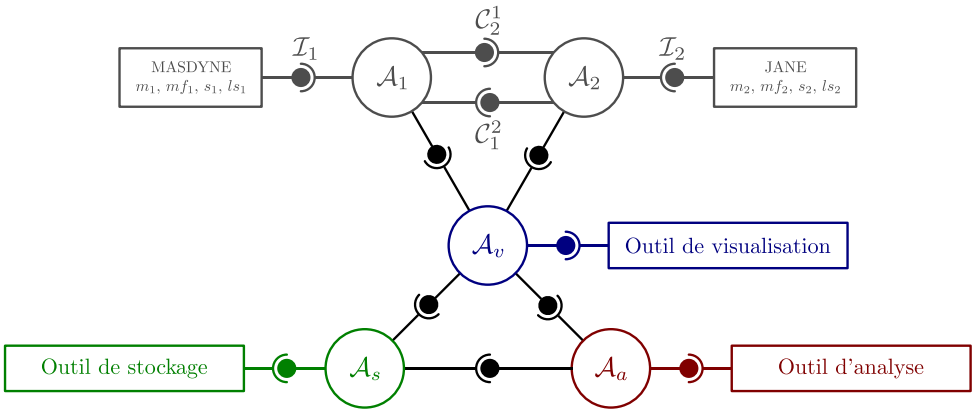


FIGURE 12.1 – Exemple d'application du méta-modèle AA4MM aux outils d'analyse, de visualisation et de stockage des résultats de simulation

Validité du multi-modèle

Dans ce manuscrit, nous n'avons pas évoqué la validation du multi-modèle. Pour être crédible, il faut que les résultats de simulation issus du multi-modèle soient validés. Cela se fait par comparaison avec les observations du système étudié et en fonction de la question posée. En réutilisant et en couplant des modèles, la question de la validation de pose à plusieurs niveaux. Au niveau du multi-modèle dans sa globalité, il est nécessaire de le valider par rapport au système étudié dans sa totalité quand cela est possible. Au niveau de chaque modèle élémentaire utilisé, il est nécessaire de valider chaque modèle par rapport à la partie du système représentée.

Il est probable que chaque modèle élémentaire soit valide alors que le multi-modèle ne l'est pas. C'est par exemple le cas où les interactions entre les modèles font que les modèles élémentaires sortent de leur domaine de validité et produisent des résultats localement faux. Il également probable que des modèles non localement valides produisent globalement, au niveau du multi-modèle des résultats valides. C'est le cas si on admet une marge d'erreur relativement souple au niveau global et une marge d'erreur plus stricte au niveau local.

Comme on le voit ici, cela dépend grandement du système étudié et de la question qu'on se pose. Cependant, il est utile de contrôler les écarts et les erreurs faites pendant la multi-modélisation. Si on se place dans l'hypothèse où tous les modèles élémentaires possèdent un domaine de validité, on peut via le méta-modèle AA4MM avoir un aperçu des erreurs faites durant l'exécution du multi-modèle et pourquoi pas, contrôler celles-ci en changeant les modèles en court de simulation.

Pour cela il est nécessaire de décrire au niveau sémantique le domaine de validité de chaque modèle. Ensuite chaque *m-agent* en charge d'un logiciel de simulation doit vérifier que les données d'entrée et de sortie du modèle sont bien comprises dans ce domaine de validité. Si ces dernières sortent de ce domaine alors l'agent peut tout à fait le signaler au modélisateur ou prendre la décision, si l'erreur devient trop importante, d'arrêter la simulation ou d'échanger son modèle par un autre dont le domaine serait plus approprié.

Implantation

Nous avons quelque peu discuté de l'implantation de AA4MM dans le manuscrit. À ce jour, nous possédons deux implantations. La première est centralisée : tous les logiciels de simulation, les *artefacts-d'interface*, les *artefacts-de-couplage* et les *m-agents* s'exécutent sur une même machine. Dans la seconde, nous utilisons un middleware basé sur l'API JMS (*java messaging service*) afin d'implanter les *artefacts-de-couplage* et de distribuer l'exécution des logiciels de simulation, des *artefacts-d'interface* et des *m-agents* sur différentes machines en réseau. Ces deux solutions ont été utilisées pour construire les multi-modèles présentés aux chapitres 10 et 11. Ces deux implantations nous servent de preuves de concepts et leurs performances n'ont pas été notre priorité

Nous avons cependant mené des réflexions sur les choix techniques. L'implantation idéale serait une implantation distribuée et décentralisée. Il serait également intéressant de disposer d'une méthode de déploiement automatisée. En terme de robustesse, un point que nous n'avons pas abordé, il serait intéressant de pouvoir répliquer les logiciels de simulation afin de palier aux pannes ainsi que de fiabiliser les échanges de messages. Au terme de ces réflexions, la plate-forme multi-agent JADE nous est apparue comme un potentiel candidat à l'implantation de AA4MM. Nous avons tenté cette implantation. Malheureusement des problèmes techniques et le manque de temps nous ont contraint à laisser cette implantation inachevée.

12.2.2 AA4MM : directions futures

Vers une société auto-organisée de modèles et de simulateurs en interaction

La version du méta-modèle AA4MM que nous avons présenté est relativement statique, notamment au niveau de la mise en place du multi-modèle. Or cette phase de la multi-modélisation nécessite une connaissance accrue de la théorie de la modélisation et de la simulation. Comme le souligne Jean Louis Igarza dans le chapitre 7 de [Cantot et al., 2009], le domaine de la simulation distribuée manque d'outils au niveau sémantique pour mettre en place de façon relativement simple, automatisée et intelligente des société de modèles et de simulateurs en interaction.

Nous pensons que l'approche multi-agent est un premier pas vers cette "mise en place intelligente" d'une société de modèles et de simulateurs en interaction. Le méta-modèle AA4MM s'est avant tout focalisé sur la faisabilité d'une telle société. Nous pensons que l'avantage de cette approche est la possibilité d'ajouter de la connaissance à nos agents afin d'orienter le processus de multi-modélisation vers plus d'autonomie et d'auto-organisation. Une voie de réflexion possible est la prise en compte d'ontologies au niveau des *m-agents*. Nous citerons à ce propos les quelques travaux en la matière : [Miller et al., 2004; Benjamin et al., 2006]. Il reste aujourd'hui à voir comment les résultats de ces recherches peuvent être mis à profit dans une approche multi-agent telle que AA4MM.

Modélisation des systèmes ouverts

Dans ce manuscrit, nous n'avons pas évoqué la modélisation des systèmes ouverts. Notre cas d'étude et les hypothèses que nous nous sommes fixées ne nous ont pas confronté à ce type de systèmes. Cependant, nous pensons que la simulation des systèmes complexes ne peut se passer de la représentation de systèmes ouverts.

Créer un multi-modèle représentant un système ouvert signifie que des objets représentés dans un modèle vont pouvoir apparaître, disparaître voire passer d'un modèle à un autre. Cela pose de nombreuses questions aussi bien au niveau sémantique (que signifie faire passer un objet d'un modèle à un autre ?), syntaxique (que ce passe-t-il si cet objet doit changer de formalisme lors de son passage d'un modèle à un

autre ?), dynamique (comment traite-t-on de tels événements ?) et technique (comment implanter un tel changement ?). En l'état actuel, AA4MM est trop limité pour pouvoir réaliser ce genre de modélisation. Nous pensons que la vision multi-agent adoptée par AA4MM est une voie à explorer pour créer un tel multi-modèle. Cependant, ce méta-modèle comporte des manques au niveau conceptuel qu'il faudra combler pour arriver à une "société de modèles et de simulateurs en interaction" capable de modéliser les systèmes ouverts.

12.2.3 Étude des réseaux ambiants et phénomènes d'inter-influences

Le fil directeur de ce travail de thèse est la prise en compte dans l'évaluation des réseaux ambiants des comportements de leurs utilisateurs. Nous avons abordé ce problème en étudiant d'abord les réseaux pair-à-pair de partage de fichiers puis les réseaux mobiles ad hoc. Dans les deux cas, nous avons retrouvé des problématiques génériques en terme de modélisation et de simulation. C'est ce qui nous a poussé à proposer AA4MM. L'application de AA4MM au domaine des réseaux ambiants en est resté, par manque de temps, au stade de preuve de concepts. Dans cette section, nous souhaitons évoquer les différentes voies qui s'ouvrent à nous maintenant que nous possédons de nouveaux outils de modélisation et de simulation.

Vers une plate-forme d'évaluation des réseaux ambiants

Tout d'abord en terme d'outils, notre souhait est de mettre sur pied une plate-forme d'évaluation des réseaux mobiles ad hoc. Cette plate-forme, grâce à AA4MM, peut être composée de plusieurs logiciels de simulation existants en interaction afin de représenter les différents aspects des réseaux mobiles ad hoc (environnement, usagers et réseaux).

Nous avons également constaté que le domaine des réseaux ambiants manque d'outils standardisés pour pouvoir comparer à la fois les résultats le plus souvent obtenus par simulation mais aussi les outils qui permettent d'obtenir de tels résultats. Dans ce manuscrit, nous avons évoqué la notion de scénario d'usage (un environnement et un ensemble de comportements des usagers). Nous pensons qu'il est nécessaire que la communauté se mette d'accord sur un certain nombre de scénarios d'usage standards. Cela permettra une meilleure comparaison des résultats de simulation.

Validation des comportements des usagers

Les modèles des comportements des usagers utilisés par la communauté des réseaux ambiants sont souvent très (voire trop) simplistes. Il n'existe que peu de travaux qui lient à la fois le domaine des réseaux ambiants et les sciences humaines. Cela a pour conséquence d'introduire un doute quant à la validité des expériences menées en simulation. Cela introduit aussi une difficulté supplémentaire pour passer de la simulation à l'implantation réelle. Nous pensons qu'un travail pluridisciplinaire est nécessaire entre scientifiques des deux domaines. Pour avoir touché du doigt cet aspect pluridisciplinaire, nous savons qu'un tel travail n'est pas aisé et qu'il faut savoir satisfaire les besoins de chaque domaine.

À ce propos, nous pensons qu'une plate-forme de multi-modélisation telle que nous la décrivons peut être l'outil à la base de ce genre de collaboration. En effet, il est tout à fait possible d'évaluer les aspects réseaux face à des cas d'usages de plus en plus réalistes et il est tout à fait envisageable de tester les théories sur le comportement des utilisateurs des réseaux ambiants ou sociaux face à de véritables modèles de réseaux et de comparer les résultats obtenus avec l'observation des réseaux réels. Nous pensons que le méta-modèle AA4MM peut être une première étape vers ce genre de collaborations pluridisciplinaires.

Passage du modèle au réel

Nous ne saurions terminer ce chapitre de discussions sans évoquer les expériences réelles qui constituent pour nous un passage obligé que ce soit pour valider les modèles (ou les multi-modèles) ou pour concevoir et tester de nouveaux protocoles et applications dans les réseaux ambiants. Dans ce manuscrit, nous avons principalement évoqué la simulation pour l'évaluation et la compréhension des réseaux ambiants. La simulation peut également être utilisée comme une étape dans le développement et la conception des systèmes complexes. On parle alors de simulation pour l'ingénierie des systèmes complexes.

Dans ce domaine, nous retiendrons comme problématique le fait de passer d'un modèle de simulation où tout est maîtrisé à une implantation réelle dans un environnement inconnu et où on ne maîtrise que localement quelques parties du système. Comment peut-on alors contrôler le système et comment peut-on mettre en relation les observations de la réalité avec les résultats de simulation obtenus ?

Avec le méta-modèle AA4MM, il est possible d'envisager de coupler à la fois les modèles de simulation avec le système réel. En effet, on pourrait remplacer un modèle du multi-modèle par le système réel en remplaçant l'*artefact-d'interface* par un ensemble de capteurs et d'effecteurs. À long terme, nous envisageons qu'AA4MM apporte un certain nombre de solutions pour l'ingénierie des systèmes complexes. Bien sûr cette vision ne se fera pas sans aborder de nouvelles problématiques comme les aspects temps réels de la simulation ou l'apparition de nouveaux comportements dans les *m-agents*. Cependant, nous pensons que ce dialogue constant entre théorie et pratique est nécessaire pour avancer dans la compréhension et la conception de systèmes aussi complexes que les réseaux ambiants.

Pour finir

AA4MM permet la conception et la simulation d'un modèle d'un système complexe comme une société de modèles, de simulateur et de logiciels hétérogènes en interaction. Au jour d'aujourd'hui cette société est relativement statique. Cependant, la vision homogène multi-agent de AA4MM nous permet d'envisager dès maintenant l'utilisation des propriétés des systèmes multi-agents comme l'ouverture, l'auto-organisation ou encore l'émergence de propriétés pour la modélisation et la simulation des systèmes complexes. En résolvant d'un manière multi-agent homogène les contraintes des niveaux technique, dynamique, syntaxique et sémantique, AA4MM permet d'envisager la multi-modélisation et le couplage de simulation comme étant un système complexe adaptatif à part entière.

On peut dès lors imaginer une société de *m-agents* qui s'auto-organise et s'adapte aux besoins du modélisateur. La conception d'un modèle d'un système complexe reposerait alors sur un dialogue entre le modélisateur les *m-agents* et les données issues des observations du système. Le modélisateur poserait quelques contraintes sur le système étudié et les aspects qui l'intéressent et l'ensemble des *m-agents* lui proposerait automatiquement un multi-modèle particulier. On peut également voir la simulation comme une équipe de *m-agents* qui pourrait s'auto-reproduire afin d'explorer de manière concurrente les diverse possibilités offertes par le modèle ; voire d'en apprendre les configurations intéressantes afin d'améliorer la qualité de la simulation. Nous pourrions continuer la liste longtemps.

AA4MM n'est qu'une première pierre. En répondant à la question de la conception d'un modèle comme étant une société de modèles, de simulateurs et de logiciels en interaction, ce travail pose une question bien plus large : comment utiliser les propriétés des systèmes complexes pour les modéliser et les simuler ?

Reférences bibliographiques

http ://www.isi.edu/nsnam/ns/. URL http://www.isi.edu/nsnam/ns/.

Teletraffic engineering handbook. Technical report, ITU-D Study Group 2 Question 16/2, 2006. URL http://oldwww.com.dtu.dk/teletraffic/handbook.html.

N. Adam, K. M. Chandy, and J. Misra. Asychronous distributed simulation via a sequence of parallel computations. *Communications of the ACM, Vol 24, No*, 11, 1981.

E. Adar and B. Huberman. Free riding on gnutella. Technical report, Technical report, Xerox PARC, 10 Aug. 2000. URL citeseer.ist.psu.edu/adar00free.html.

F. Amblard and D. Phan. *Modélisation et Simulation Multi-agents, Application pour les Sciences de l'Homme et de la Société*. Hermes Lavoisier, 2006.

E. Amouroux, T.-Q. Chu, A. Boucher, and A. Drogoul. GAMA : An Environment for Implementing and Running Spatially Explicit Multi-agent Simulations. In A. Ghose, editor, *Agent Computing and Multi-Agent Systems*, volume 5044, pages 359–371. Springer Berlin / Heidelberg, 2009.

F. Bai and A. Helmy. A survey of mobility models in wireless adhoc networks. *Wireless Ad-Hoc Networks*, pages 1–29, June 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=?doi=10.1.1.107.7131.

J. J. Barton and V. Vijayaraghavan. UBIWISE, A Ubiquitous Wireless Infrastructure Simulation Environment. Technical report, Hewlett-Packard Laboratories Palo Alto, 2003.

I. Baumgart, B. Heep, and S. Krause. OverSim : A scalable and flexible overlay framework for simulation and real network applications. In *Ninth International Conference on Peer-to-Peer Computing (IEEE P2P)*, pages 87–88, Seattle, WA, USA, Sept. 2009. doi : 10.1109/P2P.2009.5284505. URL http://doc.tm.uka.de/2009/p2p09_oversim.pdf.

F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE - A White Paper. Technical report, Telecom Italia Lab, Sept. 2003.

P. Benjamin, M. Patki, and R. Mayer. Using ontologies for simulation modeling. In *WSC '06 : Proceedings of the 38th conference on Winter simulation*, pages 1151–1159. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

S. Bonneaud. *Des agents-modèle pour la simulation des systèmes complexes. Application à l'écosystémique des pêches.* PhD thesis, Université de Bretagne Occidentale., 2008.

V. Borrel, F. Legendre, M. D. De Amorim, and S. Fdida. Simps : using sociology for personal mobility. *IEEE/ACM Trans. Netw.*, 17(3) :831–842, 2009. ISSN 1063-6692. doi : http://dx.doi.org/10.1109/TNET.2008.2003337.

P. Bourgine, D. Chavalarias, and E. Perrier. French Roadmap for Complex Systems. 2008-2009 Edition. Réseaux National des Systèmes Complexes (RNSC), 2008.

L. Bourgois, A. Oulhaci, and J.-M. Auberlet. Simulation de déplacement de piétons : Vers un modèle de perception et de prédiction d'action chez autrui. In *Journées Francophones des Systèmes Multi-Agents*, 2010.

C. Bourjot, V. Chevrier, and V. Thomas. How social spiders inspired an approach for region detection. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 426–433, Bologne, Italie, July 2002.

M. Bouzid. *Contribution à la modélisation de l'interaction agent/environnement : Modélisation stochastique et simulation parallèle.* PhD thesis, Université Henri Poincaré - Nancy 1, 7 Nov. 2001.

R. Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1) :14–23, Mar. 1986. ISSN 0882-4967. doi : 10.1109/JRA.1986.1087032.

T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC) : Special issue on Mobile Ad Hoc Networking : Research, Trends and Applications*, 2 :483–502, 2002.

P. Cantot, J.-L. Igarza, D. Luzeaux, and R. Rabeau. *Simulation et modélisation des systèmes de systèmes*. Traité Informatique et Systèmes d'Information (IC2). Hermes, Lavoisier, 2009.

J. Cao, W. S. Cleveland, D. Lin, and D. X. Sun. Internet traffic tends to poisson and independent as the load increases. Technical report, Bell Labs, 2001.

H. Casanova, A. Legrand, and M. Quinson. SimGrid : a Generic Framework for Large-Scale Distributed Experiments. In *10th IEEE International Conference on Computer Modeling and Simulation - EUROSIM / UKSIM 2008*, Cambrige United Kingdom, 2008. IEEE. URL `http://hal.inria.fr/inria-00260697/PDF/uksim.pdf`.

K. M. Chandy and J. Misra. Asynchronous distributed simulation via a sequence of parallel computations. *Commun. ACM*, 24(4) :198–206, 1979. ISSN 0001-0782. doi : http://doi.acm.org/10.1145/358598.358613.

V. Chevrier and N. Fates. How important are updating schemes in multi-agent systems ? An illustration on a multi-turmite model. In *9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 533–540, Toronto, Canada, May 2010.

N. Conruyt, D. Sébastien, R. Courdier, D. David, N. Sébastien, and T. Ralambondrainy. Designing an Information System for the Preservation of the Insular Tropical Environment of Reunion Island. Integration of Databases, Knowledge Bases and Multi-Agent Systems by using Web Services. *Whitestein Series in Software Agent Technologies and Automatic Computing*, (61-90), 2009.

G. Desmeulles. *Réification de interactions pour l'expérience in virtuo de systèmes biologiques multi-modèles*. PhD thesis, Université de Bretagne Occidentale, 2006.

T. Disz, M. E. Papka, and R. Stevens. Ubiworld : An Environment Integrating Virtual Reality, Supercomputing, and Desigh. In *6th Heterogeneous Computing Workshop (HCW'97)*, 1997.

A. Doniec, R. Mandiau, S. Piechowiak, and S. Espié. A behavioral multi-agent model for road traffic simulation. *Eng. Appl. Artif. Intell.*, 21 :1443–1454, December 2008. ISSN 0952-1976. doi : 10.1016/j.engappai.2008.04.002. URL `http://dl.acm.org/citation.cfm?id=1454788.1454999`.

A. Drogoul, D. Vanbergue, and T. Meurisse. Multi-Agent based simulation : Where are the agents ? In J. S. Sichman, F. Bousquet, and P. Davidsson, editors, *Proceedings of the Third International Workshop on Multi-Agent-Based Simulation MABS 2002, Bologna, Italy*, LNAI 2581, pages 1–15. Springer Verlag, Berlin Heidelberg, July 2002.

D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-service resilience in peer-to-peer file sharing systems. In *SIGMETRICS '05 : Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 38–49, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-022-1. doi : http://doi.acm.org/10.1145/1064212.1064218.

M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. In *PINS '04 : Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 228–236, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-942-9. doi : http://doi.acm.org/10.1145/1016527.1016539.

J. Ferber. Les systèmes multi-agents : un aperçu général. *Techniques et sciences informatiques*, 16(8) : 979–1012, 1997.

J. Ferber and J. P. Müller. Influences and Reaction : a model of situated multiagent systems. In *Second International Conference on Multiagent Systems (ICMAS96)*, 1996.

P. A. Fishwick. *Handbook of dynamic system modeling.* Chapman & Hall/CRC, 2007. ISBN 1584885653.

P. A. Fishwick and B. P. Zeigler. A multimodel methodology for qualitative model engineering. *ACM Trans. Model. Comput. Simul.*, 2(1) :52–81, 1992. ISSN 1049-3301. doi : http://doi.acm.org/10.1145/132277.132280.

J. W. Fowler and O. Rose. Grand challenges in modeling and simulation of complex manufacturing systems. *Simulation*, 80 :9–469, 2004.

J. Fromm. *The emergence of complexity.* kassel university press, 2004.

R. M. Fujimoto. Parallel simulation : parallel and distributed simulation systems. In *Wsc '01 : proceedings of the 33rd conference on winter simulation*, pages 147–157, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7803-7309-X.

D. Gaïti and L. Merghem-Boulahia. *L'autonomie dans les réseaux*, chapter 5, pages 133–165. Hermes Lavoisier, 2006.

Y. Gangat, D. David, D. Payet, and R. Courdier. Modéliser et simuler l'aménagement énergétique d'un territoire par les systèmes multi-agents. In *Convergence des Réseaux, de l'Informatique, et du Multimédia pour les E-Services (CRIMES'09)*, 2009.

P. Garcia, C. Pairot, R. Mondejar, J. Pujol, H. Tejedor, and R. Rallo. *PlanetSim : A New Overlay Network Simulation Framework.* 2005. doi : 10.1007/11407386_10. URL `http://dx.doi.org/10.1007/11407386_10`.

N. Gaud. *Systèmes multi-agents holoniques : de l'analyse à l'implantation. Méta-modèle, méthodologie, et simulation multi-niveaux.* PhD thesis, Université de Franche Compté et Universite de Belfort-Montbéliard, 7 Dec. 2007.

N. Gaud, S. Galland, F. Gechter, V. Hilaire, and A. Koukam. Holonic Multilevel Simulation of Complex Systems. Application to real-time pedestrian simulation in virtual urban environment. *Simulation Modelling Practice And Theory (SIMPAT)*, 16(10) :1659–1676, Nov. 2008.

J. Gil-Guijano, G. Hutzler, and G. Louail. Accroche toi au niveau, j'enlève l'échelle. éléments d'analyse des aspects multiniveaux dans la simulation à base d'agents. *Revue d'Intelligence Artificielle*, 24 : 625–648, 2010.

M. P. Gleizes. Méthodes de développement de systèmes multi-agents. *Génie Logiciel*, (86) :2–7, 2008.

D. Gorgen, H. Frey, and C. Hiedels. Jane - the java ad hoc network development environment. In *ANSS '07 : Proceedings of the 40th Annual Simulation Symposium*, pages 163–176, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2814-7. doi : 10.1109/ANSS.2007.24.

V. Grimm. Ten years of individual-based modelling in ecology : what have we learned and what could we learn in the future ? *Ecological Modelling*, 115(2-3) :129–148, February 1999. doi : http://dx.doi.org/10.1016/S0304-3800(98)00188-4. URL `http://dx.doi.org/10.1016/S0304-3800(98)00188-4`.

V. R. S. Grimm. *Individual-based modeling and ecology.* Princeton University Press, 2005.

A. Grizard, L. Vercouter, T. Stratulat, and G. Muller. A peer-to-peer normative system to achieve social order. In *AAMAS06 Workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN'06)*, Hakodate, Japan, May 2006.

O. Gutknecht, J. Ferber, and F. Michel. Madkit : une expérience d'architecture de plate-forme multi-agent générique. In Hermes, editor, *8ème Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'2000)*, pages 223–236, La Réunion, 2000.

D. Hales. From selfish nodes to cooperative networks emergent link-based incentives in peer-to-peer networks. In *P2P '04 : Proceedings of the Fourth International Conference on Peer-to-Peer Computing (P2P'04)*, pages 151–158, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2156-8. doi : http://dx.doi.org/10.1109/P2P.2004.22.

D. Helbing. A mathematical model for the behavior of pedestrians. *Behavioral Science*, 36(4) :1099–1743, 1991.

D. Helbing, P. Molnar, I. J. Farkas, and K. Bolay. Self-organizing pedestrian movement. *Environment and Planning B : Planning and Design*, 28(3) :361–383, 2001. doi : 10.1068/b2697. URL `http://dx.doi.org/10.1068/b2697`.

J. Himmelspach and A. M. Uhrmacher. Plug'n simulate. In *ANSS '07 : Proceedings of the 40th Annual Simulation Symposium*, pages 137–143, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2814-7. doi : http://dx.doi.org/10.1109/ANSS.2007.34.

J. Himmelspach, M. Röhl, and A. M. Uhrmacher. Component-based models and simulation experiments for multi-agent systems in JAMES II. In *6th International Workshop on Agent Theory to Agent Implementation (AT2AI-6)*, 2008.

HLA Working Group and S. C. Symington. IEEE Standard for Modeling and Simulation (M&S) High Lavel Architecture (HLA) – Framework and Rules. IEEE-SA Standards Board, 11 Dec. 2000.

M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris. The Peersim simulator. `http://peersim.sf.net`.

Julien Siebert, Laurent Ciarletta, and Vincent Chevrier. Impact du comportement des utilisateurs dans les réseaux pair-à-pair (P2P) : modélisation et simulation multi-agents. In *16es Journées Francophones des Systèmes Multi-Agents 16es Journées Francophones des Systèmes Multi-Agents - JFSMA'08*, pages 129–138, Brest France, 10 2008. URL `http://hal.archives-ouvertes.fr/hal-00331919/en/`.

Julien Siebert, Laurent Ciarletta, and Vincent Chevrier. De l'intérêt du couplage de modèles pour appréhender les interactions utilisateurs-réseaux dynamiques. *Revue d'Intelligence Artificielle*, 2009. URL `http://hal.inria.fr/inria-00398679/en/`.

R. Kitio, O. Boissier, J. F. Hübner, and A. Ricci. Organisational artifacts and agents for open multi-agent organisations : "giving the power back to the agents". In *COIN'07 : Proceedings of the 2007 international conference on Coordination, organizations, institutions, and norms in agent systems III*, pages 171–186, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-79002-0, 978-3-540-79002-0.

F. Klein. *Contrôle d'un Système Multi-Agents Réactif par Modélisation et Apprentissage de sa Dynamique Globale*. PhD thesis, Université Nancy II, 4 Dec. 2009.

G. J. Klir. Architecture of systems problem solving (plenum press, new york, 1985), book abstract. In , pages 799–802.

F. Krief. *Les systèmes embarqués communicants : mobilité, sécurité, autonomie*. Lavoisier, ISBN 978-2-7462-1873-4, Sept. 2008. URL `http://hal.archives-ouvertes.fr/hal-00351746/en/`.

Y. Kubera, P. Mathieu, and S. Picault. Interaction-oriented agent simulations : From theory to implementation. In M. Ghallab, C. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*, pages 383–387. IOS Press, 2008. ISBN 978-1-58603-891-5. URL `http://www.ece.upatras.gr/ecai2008/`.

F. Kuhl, R. Weatherly, and J. Dahmann. *Creating computer simulation systems : an introduction to the high level architecture*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN 0-13-022511-8.

T. S. Kuhn. *The Structure of Scientific Revolutions*. University Of Chicago Press, 3rd edition, December 1996. ISBN 0226458083. URL `http://www.worldcat.org/isbn/0226458083`.

R. Kumar, D. D. Yao, A. Bagchi, K. W. Ross, and D. Rubenstein. Fluid modeling of pollution proliferation in p2p networks. *SIGMETRICS Perform. Eval. Rev.*, 34(1) :335–346, 2006. ISSN 0163-5999. doi : http://doi.acm.org/10.1145/1140103.1140316.

V. Kumar, L. Lin, D. Krajzewicz, F. Hrizi, O. Martinez, G. Sempere, J. Manuel, and R. Bauza. iTETRIS : Adaptation of ITS Technologies for Large Scale Integrated Simulation. In *3rd IEEE International Symposium on Wireless Vehicular Communications : IEEE WiVEC 2010*, 16 May 2010.

S. Kurkowski, T. Camp, and M. Colagrosso. Manet simulation studies : The incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9 :50–61, 2005a.

J. F. Kurose and K. W. Ross. *Computer Networking. A top-down approach. 4th Edition.* Pearson International Edition, 2008.

L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7) : 558–565, July 1978. ISSN 0001-0782. doi : 10.1145/359545.359563. URL `http://dx.doi.org/10.1145/359545.359563`.

T. Leclerc, L. Ciarletta, and A. Schaff. A stable linked structure flooding for mobile ad hoc networks with fault recovery. In *WWIC*, pages 204–215, 2010a.

U. Lee, M. Choiz, J. Choy, M. Y. Sanadidiy, and M. Gerla. Understanding pollution dynamics in p2p file sharing. In *5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, Santa Babara, CA, USA, February 2006.

F. Legendre, V. Borrel, M. Dias de Amorim, and S. Fdida. Reconsidering Microscopic Mobility Modeling for Self-Organizing Networks. *Network, IEEE*, 20(6) :4–12, 2006.

G. Lin, G. Noubir, and R. Rajaraman. Mobility models for ad hoc network simulation. In *IEEE INFOCOM 2004*, 2004.

D. Luzeaux and J.-R. Ruault. *Systèmes de systèmes Texte imprimé : concepts et illustrations pratiques.* Traité IC2 : information-commande-communication. Hermes Lavoisier, 2008.

K. Maeda, K. Sato, K. Konishi, A. Yamasaki, A. Uchiyama, H. Yamaguchi, K. Yasumoto, and T. Higashino. Getting Urban Pedestrian Flow from Simple Observation : Realistic Mobility Generation in Wireless Network Simulation. In *the 8th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM2005)*, 2005.

A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite : An approach to universal topology generation. 2001.

T. Meurisse. *Simulation multi-agent : du modèle à l'opérationnalisation.* PhD thesis, Université Paris 6, July 2004.

F. Michel. *Formalisme, outils et éléments méthodologiques pour la modélisation et la simulation multi-agents.* PhD thesis, Université des Sciences et Techniques du Languedoc, 21 Dec. 2004.

F. Michel. The irm4s model : the influence/reaction principle for multiagent based simulation. In *AAMAS '07 : Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–3, New York, NY, USA, 2007. ACM. ISBN 978-81-904262-7-5. doi : http://doi.acm.org/10.1145/1329125.1329289.

F. Michel, G. Beurier, and J. Ferber. The TurtleKit simulation platform : Application to complex systems. In A. Akono, E. Tonyé, A. Dipanda, and K. Yétongnon, editors, *Workshops Sessions, First International Conference on Signal & Image Technology and Internet-Based Systems SITIS' 05*, pages 122–128. IEEE, IEEE, november 2005.

J. A. Miller, G. T. Baramidze, A. P. Sheth, and P. A. Fishwick. Investigating ontologies for simulation modeling. In *ANSS '04 : Proceedings of the 37th annual symposium on Simulation*, page 55, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2110-X.

N. Minar, R. Burkhart, and C. Langton. The swarm simulation system : A toolkit for building multi-agent simulations, 1996. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.108.1436`.

M. L. Minsky. Matter, Mind and Models. In *International Federation of Information Processing Congress*, pages 45–49, 1965.

M. Morge and P. Mathieu. Mécanisme de rétribution pour les systèmes p2p d'échange de fichiers. *La revue d'Ingénierie des Systèmes d'Information (ISI), numéro spécial "Les systèmes d'information pair-è-pair"*, 12 :61–84, 2007.

G. Morvan, A. Veremme, and D. Dupont. Irm4mls : the influence reaction model for multi-level simulation. In *11th International wprkshop on Multi-Agent-Based Simulation (MABS) hosted at the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2010.

Y. Murakami, K. Minami, T. Kawasoe, and T. Ishida. Multi-Agent Simulation for Crisis Management. In *IEEE Workshop on Knowledge Media Networking*, 2002.

S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai. A survey of peer-to-peer network simulators. *Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK*, 2006a.

T. Navarrete Gutierrez, J. Siebert, L. Ciarletta, and V. Chevrier. Impact des dimensions spatiale et temporelle dans la modélisation d'un phénomène collectif de type "¡ free-riding "¿. In *18èmes Journées Francophones des Systèmes Multi-Agents - JFSMA'10*, Mahdia, Tunisia, Oct. 2010. URL `http://hal.inria.fr/inria-00534600/en`.

M. North, T. Howe, N. Collier, and J. Vos. "A Declarative Model Assembly Infrastructure for Verification and Validation,". In S. Takahashi, D. Sallach, and J. Rouchier, editors, *Advancing Social Simulation : The First World Congress*, 2007.

A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A meta-model for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 17(3) :432–456, Dec. 2008. ISSN 1387-2532. doi : 10.1007/s10458-008-9053-x. URL `http://www.springerlink.com/content/l2051h377k2plk07/`. Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

E. O'Neil. TATUS A ubiquitous Computing Simulator. Master's thesis, University of Dublin, Trinity College, Sept. 2004.

D. Payet, R. Courdier, and N. Sébastien. Environment as support for simplification, reuse and integration of processes in spatial MAS. In *IEEE International Conference on Information Reuse and Integration, IRI'06*, pages 127–131, Waikoloa, Hawaii, USA, Sept. 2006b.

A. Pereira and L. P. Reis. Agent-based Simulation of Ecological Models. In *5th Workshop on Agent-based SImulation*, 2004a.

A. Pereira and L. P. Reis. Agent-based Ecological Models Calibration - on the edge of a new approach. In *International Conference on Knowledge Engineering and Decision Support*, 2004b.

M. Piunti and A. Ricci. From Agents to Artifacts Back and Forth : Purposive and Doxastic Use of Artifacts in MAS. In *Sixth European Workshop on Multi-Agent Systems (EUMAS08)*, 2008.

G. Pongor. Omnet : Objective modular network testbed. In *MASCOTS '93 : Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, pages 323–326, San Diego, CA, USA, 1993. The Society for Computer Simulation, International. ISBN 1-56555-018-8.

G. Pujolle. *Les Réseaux-2008*. Eyrolles, 2008.

G. Quesnel. *Approche formelle et opérationnelle de la multi-modélisation et de la simulation des systèmes complexes. Apports pour la simulation des systèmes multi-agents*. PhD thesis, Université du littoral - Côte d'opale, 1 Dec. 2006.

G. Quesnel, R. Duboz, and É. Ramat. The virtual laboratory environment - an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Practice and Theory*, 17(4) :641–653, 2009a. ISSN 1569-190X. doi : DOI:10.1016/j.simpat.2008.11.003. URL `http://www.sciencedirect.com/science/article/B6X3C-4V42J96-1/2/27322d978d7b0b3dedea4fee25029645`.

S. F. Railsback, S. L. Lytinen, and S. K. Jackson. Agent-based Simulation Platforms : Review and Development Recommendations. *Simulation*, 82(9) :609–623, Sept. 2006.

E. Ramat. *Modélisation et Simulation Multi-agents, Application pour les Sciences de l'Homme et de la Société*, chapter Chapitre 2 : Simulation à évenements discrets, pages 49–71. Hermes Sciences, 2006.

A. Ricci, M. Viroli, and A. Omicini. Give agents their artifacts : the a&a approach for engineering working environments in mas. In *AAMAS '07 : Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–3, New York, NY, USA, 2007a. ACM. ISBN 978-81-904262-7-5. doi : http://doi.acm.org/10.1145/1329125.1329308.

G. F. Riley. The Georgia Tech Network Simulator. *ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools)*, Aug. 2003.

V. Schafer. D'une informatique centralisée aux réseaux généraux, le tournant des années 1970, http ://interstices.info/vers-reseaux-generaux. URL `http://interstices.info/vers-reseaux-generaux`.

N. Sebastien. *Distribution et Parallélisation de Simulations Orientées Agent*. PhD thesis, Université de La Réunion, 2009.

D. Servat, E. Perrier, J.-P. Treuil, and A. Drogoul. When agents emerge from agents : Introducing multi-scale viewpoints in multi-agent simulations. In *First International Workshop on MAS and ABMS*, 1998.

J. Siebert. Impact du comportement des utilisateurs sur le fonctionnement des réseaux pair-à-pair : modélisation et simulation multi-agent. Master's thesis, Université Henri Poincaré (UHP) Nancy 1, June 2007a.

J. Siebert, V. Chevrier, and L. Ciarletta. Modélisation multimodèle des réseaux dynamiques : cas des réseaux pair-à-pair. In *JDIR'08 - 9èmes Journées Doctorales en Informatique et Réseaux*, Villeneuve d'Ascq, France, 2008a.

J. Siebert, V. Chevrier, and L. Ciarletta. Entwined influences of users'behaviour and QoS : a multi-model approach. In *Autonomous Infrastructure, Management and Security*, 1 July 2008b.

P. Sommer. Design and Analysis of Realistic Mobility Models for Wireless Mesh Networks. Master's thesis, Communication Systems Group Computer Engineering and Networks Laboratory (TIK) Department of Information Technology and Electrical Engineering, Sept. 2007.

C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of Predictability in Human Mobility. *Science*, 327(5968) :1018–1021, 19 Feb. 2010.

S. Straßburger, A. Hamm, G. Schmidgall, and S. Haasis. Using HLA Ownership Management in Distributed Material Flow Simulations. In *European Simulation Interoperability Workshop*, London, June 2002.

T. Stratulat, J. Ferber, and J. Tranier. Masq : towards an integral approach to interaction. In , pages 813–820. ISBN 978-0-9817381-7-8.

A. S. Tanenbaum and M. Van Steen. *Distributed Systems : Principles and Paradigms*. Prentice Hall, 2001.

K. Teknomo, Y. Takeyama, and H. Inamura. Review on microscopic pedestrian simulation model. In *Proceedings Japan Society of Civil Engineering Conference*, 2000.

V. Thomas, C. Bourjot, V. Chevrier, and D. Desor. MAS and RATS Multi-agent simulation of social differentiation in rats' groups. Interest for the understanding of a complex biological phenomenon. In *International Workshop on Self-Organization and Evolution of Social Behaviour*, Monte Verita, Sept. 2002.

A. Tolk and J. A. Muguira. The Levels of Conceptual Interoperability Model. In *2003 Fall Simulation Interoperability Workshop*, Orlando, Florida, Sept. 2003.

H. Van Dyke Parunak, R. Savit, and R. L. Riolo. Agent-based modeling vs. equation-based modeling : A case study and users' guide. In *MABS*, pages 10–25, 1998. URL `http://citeseer.ist.psu.edu/parunak98agentbased.html`.

D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1) :5–30, 2007. ISSN 1387-2532. doi : http://dx.doi.org/10.1007/s10458-006-0012-0.

U. Wilensky. Netlogo Wolf sheep predation model, 1997b. URL `http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation`. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

U. Wilensky. Netlogo Shepherds model, 1998b. URL `http://ccl.northwestern.edu/netlogo/models/Shepherds`. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

U. Wilensky. Netlogo, 1999. URL `http://ccl.northwestern.edu/netlogo/`. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

U. Wilensky. Netlogo Rabbits grass weeds model, 2001. URL `http://ccl.northwestern.edu/netlogo/models/RabbitsGrassWeeds`. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

J. Winick and S. Jamin. Inet-3.0 : Internet topology generator. Technical report, University of Michigan.

B. P. Zeigler and T. I. Ören. Multifaceted, multiparadigm modeling perspectives : tools for the 90's. In *WSC '86 : Proceedings of the 18th conference on Winter simulation*, pages 708–712, New York, NY, USA, 1986. ACM. ISBN 0-911801-11-1. doi : http://doi.acm.org/10.1145/318242.318513.

B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of Modeling and Simulation*. Academic Press, January 2000. ISBN 0127784551.

X. Zeng, R. Bagrodia, and M. Gerla. Glomosim : A library for parallel simulation of large-scale wireless networks. In *in Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.

Résumé

Ce travail de thèse, mené sur deux thématiques : les réseaux ambiants et la simulation multi-agent, a pour contexte l'étude des interactions entre le fonctionnement des réseaux ambiants (réseaux pair-à-pair et réseaux mobiles ad hoc) et les comportements de leurs usagers (mobilité, partage d'un service, *etc.*). Pour étudier ces phénomènes, nous avons mis en lumière le besoin de réutiliser, de coupler et de faire interagir des outils de modélisation et de simulation provenant de domaines scientifiques différents (réseaux informatiques, simulations sociales) afin d'intégrer simultanément plusieurs niveaux d'abstraction dans l'étude menée. Cette problématique de réutilisation et de couplage de modèles (la multi-modélisation) et de simulation (interopérabilité de simulateurs) n'est pas propre aux seuls réseaux ambiants et celle-ci s'inscrit dans le domaine plus vaste de l'étude des systèmes complexes. Cette thèse se propose d'aborder les questions de réutilisation et de couplage des outils de modélisation et de simulation sous l'angle des systèmes multi-agents et d'inscrire les solutions à la fois d'ingénierie logicielle, de simulation distribuée, de multi-modélisation dans un cadre multi-agent générique et homogène : le méta-modèle AA4MM. L'idée fondatrice est de créer une société de modèles, de simulateurs et de logiciels de simulation (*blocs MSL*) afin d'intégrer plusieurs niveaux d'abstraction dans une même modélisation et simulation. AA4MM propose une approche multi-agent homogène qui permet de facilement réutiliser des *blocs MSL*, de les rendre interopérable et de les coupler de manière modulaire. AA4MM permet également au modélisateur de clairement spécifier les changements d'échelles. AA4MM propose une méthode de simulation totalement décentralisée. Au niveau des aspects théoriques du méta-modèle AA4MM, nous avons proposé une preuve formelle de l'algorithme de simulation décentralisée. Nous avons également appliqué AA4MM à un cas d'étude pédagogique en couplant ensemble plusieurs instances de NetLogo (un outil de simulation répandu). Nous avons également appliqué AA4MM aux réseaux ambiants et à notre question initiale sur les influences mutuelles entre performances du réseau et comportements des usagers. Dans ce contexte, nous avons pu réutiliser des simulateurs existants qui n'étaient pas prévus pour interagir afin d'aborder les phénomènes d'influences mutuelles entre performances des réseaux et comportements de leurs usagers.

Mots-clés: Multi-modélisation, couplage de simulation, méta-modélisation, systèmes multi-agent, réseaux ambiants

Abstract

This work has been done between the fields of ubiquitous networks and multi-agent based simulation. The main context is to study mutual influences existing between ubiquitous network performances and their users behaviours. We have highlighted the need for reusing and coupling modelling and simulation softwares together in order to simultaneously integrate several abstraction levels in the study. We target those needs by a multiagent approach and we propose a metamodel : AA4MM. The core idea in AA4MM is to build a society of models, simulators and simulation softwares that solves the core challenges of multimodelling and simulation coupling in an homogeneous perspective. AA4MM major contributions are the possibility to easily reuse, to make interoperable and modular existing heterogeneous models and softwares, to manage scale changes and a simulation algorithm fully decentralized. We apply this metamodel to the field of ubiquitous networks in order to target the question of mutual influences between networks performances and users behaviours.

Keywords: Multi-modelling, Multi-Simulation, Multiagent paradigm, Complex systems modeling and simulation, Ubiquitous network modeling