PADAWAN, un modèle multi-échelles pour la simulation orientée interactions

Sébastien Picault, Philippe Mathieu, Yoann Kubera

PADAWAN, un modèle multi-échelles pour la simulation orientée interactions

S. Picault sebastien.picault@lifl.fr
P. Mathieu philippe.mathieu@lifl.fr
Y. Kubera yoann.kubera@lifl.fr

Laboratoire d'Informatique Fondamentale de Lille
Université des Sciences et Technologies de Lille, France

Résumé

La conception de simulations multi-agents appliquées aux systèmes complexes pose entre autres le problème de la modélisation de comportements intervenant à des échelles spatiales, temporelles, comportementales différentes, chacune pertinente pour représenter un des aspects du phénomène étudié. Nous proposons ici un formalisme générique destiné à la représentation d'environnements multiples, disposant d'échelles spatio-temporelles propres, et auxquels on peut associer un ensemble de règles comportementales auxquelles se soumettent les agents présents dans ces environnements. Par ailleurs chaque environnement peut être encapsulé au sein d'un agent, lui-même situé dans un autre environnement. Cette uniformité de représentation est rendue possible grâce à l'approche orientée interaction pour la conception de simulation (IODA), qui établit une séparation entre agents et interactions, et ce de la modélisation jusqu'au code. Nous expliquons également comment ce modèle est implémenté et donnons quelques exemples d'utilisation.

Mots-clés : *Simulation multi-agents multi-échelles ; Environnements ; Modèles de comportement (interactions)*

Abstract

The design of multiagent simulations devoted to complex systems, addresses the issue of modelling behaviors that are involved at different space, time, behavior scales, each one being relevant to represent a feature of the phenomenon. We propose here a generic formalism intented to represent multiple environments, endowed with their own spatiotemporal scales and with behavioral rules for the agents they contain. In addition, an environment can be nested inside an agent, which itself is situated in another environment, etc. This uniform representation is made possible through the interaction-oriented approach for the design of agent simulations (IODA), which clearly separates agents from interactions from the modelling to the code. We also explain the implementation of this approach and give some concrete examples.

Keywords: *Multiagent, multiscale simulation ; Environments ; Behavior models (interactions)*

1 Introduction

La simulation multi-agents a depuis longtemps fait les preuves de son adéquation à traiter des systèmes complexes, et elle gagne peu à peu sur des solutions alternatives telles que les modèles équationnels, les réseaux bayésiens, et autres approches dans lesquelles les entités (le plus souvent, des variables) sont éloignées de celles en usage chez les thématiciens. L'intérêt de l'approche agent en la matière est de donner directement corps aux concepts qui sous-tendent les modèles d'origine, ainsi qu'au comportement des entités correspondantes. Sans doute cela ne se fait-il pas sans un risque de biais mais il n'en reste pas moins que la compréhension mutuelle entre experts du domaine et informaticiens est favorisée.

Toutefois, se pose de façon récurrente la question de la montée à l'échelle, dans le cas de simulations impliquant des masses d'entités, appartenant à de nombreuses familles et manifestant des comportements variés. Il n'est d'ailleurs pas certain qu'on puisse trouver une réponse générale à ce problème, car la question du large-échelle se résout en plusieurs sous-problèmes assez distincts, certains appelant des réponses logicielles, d'autres une modélisation adaptée, etc. Mentionnons entre autres :

– la gestion logicielle (temps d'exécution, mémoire...) d'un nombre d'agents très élevé ;
– la représentation de niveaux d'observations plus ou moins fins ;
– le choix de modéliser un sous-système par un seul agent ou par une organisation d'agents ;
– la représentation des comportements d'une entité selon qu'elle opère comme membre d'une organisation ou comme extérieure à cette organisation.

Dans nombre de cas, on se trouve confronté en fait à un problème qui ne porte pas tant sur

l'*échelle* de la simulation (la quantité d'agents) que sur la *coexistence d'échelles différentes* [1]. On voit d'ailleurs se développer des simulations multi-modèles [2, 4] qui utilisent des paradigmes différents selon le niveau d'observation à représenter, par exemple une simulation par agents à un niveau macroscopique et une approche par équations différentielles au niveau microscopique.L'inconvénient de ces méthodes est principalement le même que celui des modèles non-agents, à savoir le manque de *terrain commun* avec les experts dans la représentation des entités et des comportements, à quoi s'ajoute un risque accru de biais lié au couplage de modèles de natures différentes. Nous prônons au contraire une approche uniforme dans laquelle on puisse, du niveau macroscopique au niveau microscopique, ne manipuler que des agents et des comportements.

Les travaux menés depuis plusieurs années sur les simulations large-échelle dans l'équipe SMAC du LIFL nous ont conduit à proposer une approche *orientée interaction* (IODA) qui repose sur trois idées-clefs :
– *tout est agent*, au sens où toute entité pertinente est représentée par un agent dont on peut caractériser dynamiquement le niveau d'activité [9] ;
– les comportements sont des *interactions génériques* définies, de l'analyse au code, indépendamment des agents ;
– une interaction s'exécute lorsqu'un agent capable de l'effectuer (*source*) en rencontre un capable de la subir (*cible*) en vérifiant les conditions de réalisation de l'interaction.

Cette approche comporte un modèle formel de description des comportements et une méthodologie (IODA), ainsi qu'une implémentation (moteur JEDI) accompagnée d'un générateur de code (JEDI-Builder). Les détails en sont donnés par exemple dans [7]. Nous avons montré par ailleurs que cette approche permet d'expliciter des biais de simulation qui passent souvent inaperçus dans les méthodologies classiques [8] ; qu'elle facilite la révision d'hypothèses [12] ; qu'elle permet également de construire automatiquement des simulations à partir de variations sur un même modèle [6].

Dans cet article, nous nous proposons de décrire une formalisation des relations entre agents et environnement(s) qui permette de modéliser aussi facilement que possible la gestion d'échelles multiples dans une simulation par agents. Pour commencer, nous examinerons quelques modèles ou architectures couramment utilisés pour gérer des structures hiérarchiques ou l'appartenance d'agents à des environnement spatiaux ou sociaux multiples. Nous décrirons ensuite PADAWAN (pour « Pattern for Accurate Design of Agent Worlds in Agent Nests »), un modèle formel qui vise à dépasser les limitations des approches actuelles ou à les généraliser, et montrerons comment l'approche orientée interactions rend possible la spécification de comportements à chaque niveau d'observation considéré. Enfin nous expliquerons comment ce modèle est implémenté au sein de notre propre simulateur orienté interactions (JEDI) et donnerons quelques exemples de situations concrètes d'utilisation.

2 Quelques approches actuelles pour la gestion d'échelles ou de niveaux multiples

La simulation de systèmes complexes, qu'ils soient composés principalement d'agents réactifs comme en biologie cellulaire, ou plutôt d'agents cognitifs comme dans les sociétés artificielles, se doit de prendre en compte les divers *niveaux d'organisation* qui rendent compte des phénomènes étudiés. Aussi a-t-on vu apparaître très tôt des architectures destinées à représenter des *emboîtements* de systèmes et sous-systèmes (ce qui se traduit par des emboîtements d'agents ou d'environnements). Plus récemment, ce sont des problèmatiques *d'appartenance multiple* (à des groupes, à des espaces différents) qui ont pris le pas sur le simple problème d'encapsulation.

2.1 SWARM

La plateforme SWARM [10] est sans doute l'une des premières qui ait explicitement posé le problème des échelles multiples dans les simulations centrées individus. Elle propose une organisation récursive du système, qui se compose :
– de *swarms* contenant des agents et un ordonnanceur pour leurs actions ;
– d'*agents* contenus dans les swarms, et qui eux-mêmes peuvent *être un swarm*.

Cette approche permet d'emblée de remplacer un niveau de modélisation donné par le système sous-jacent, et ainsi de faire en sorte que le comportement d'un agent soit produit directement par le comportement collectif du swarm qui le constitue.

Toutefois, SWARM souffre de limitations sévères : c'est un modèle strictement hiérarchique

dont la structure est figée. De plus, ce n'est qu'une plateforme qui ne fournit aucun cadre conceptuel pour la modélisation. Enfin, l'idée qu'un agent-swarm n'ait pas d'autre comportement que la résultante de ses composants est limitante, car il arrive fréquemment qu'on puisse spécifier des comportements à des niveaux d'observation différents.

2.2 AGRE et MASQ

L'approche AGRE (Agents, Groupes, Rôles + Environnement) [5] apporte des outils conceptuels plus abstraits sans négliger le côté opérationnel des choses, en mettant notamment en parallèle l'environnement physique et les environnements sociaux (groupes et organisations) auxquels un agent peut prendre part. Ainsi, le *monde* multi-agent (organisations et monde physique) apparaît comme composé d'*espaces* (groupes et régions) eux-mêmes composés de *modes* (rôles et corps) assumés par des agents.

Dans AGRE, il est donc possible pour des agents d'interagir, via leur corps s'ils sont situés dans une même région de l'espace, ou via leurs rôles s'ils appartiennent à un même groupe. Il reste que le corps de l'agent (en tant que *présence de l'agent dans un environnement physique*) est supposé unique : point de vue certes intuitif, mais qui pose de fait des bornes à ce que l'approche permet de représenter. De plus, la distinction entre monde social et monde physique vient de la focalisation initiale de la méthode (AGR) sur les problèmes organisationnels dans un SMA. Or on pourrait aussi bien envisager un monde mémoriel (les souvenirs d'un agent), un monde simulé (l'imaginaire ou les prévisions d'un agent), etc. Au reste, la structuration physique du monde (nombre de niveaux d'emboîtements permis par exemple) n'est pas très clairement décrite.

Cette distinction entre physique et social reste fondamentale dans MASQ [13], qui propose par ailleurs une décomposition des comportements selon deux axes : intérieur/extérieur et individuel/collectif. Cela conduit ainsi à définir un « esprit » de l'agent, inaccessible de façon directe, couplé à plusieurs « corps » (l'un physique, les autres sociaux) qui sont en charge de la perception et de la réalisation du comportement effectif de l'agent.

Cette métaphore corps/esprit nous semble féconde en raison de sa capacité à restituer de manière assez intuitive (et néanmoins précise) une problématique opérationnelle qui est au centre de l'inscription d'un agent dans plusieurs environnements ; nous pensons même qu'elle doit être poussée un pas plus loin (cf. §4.1).

2.3 Les P-systèmes et le calcul de membrane

Les P-systèmes [11], bien qu'ils ne rentrent pas *stricto sensu* dans le cadre des approches par agents, présentent des caractéristiques intéressantes par rapport à la question de l'emboîtement de structures et de la différenciation comportementale.

Les P-systèmes sont en effet un modèle de calcul Turing-complet, inspiré de la biologie, qui repose sur les éléments suivants :
– des *membranes* (métaphore des membranes biologiques) contenant des symboles et des règles, et formant une structure arborescente ;
– des *symboles* (métaphore des espèces chimiques) qui sont contenus dans les membranes sous forme de multiensembles (ou sacs, i.e. couples symbole-quantité) : ces symboles peuvent être étiquetés pour leur permettre de franchir une membrane et un symbole spécial (δ) permet de dissoudre la membrane où il est placé ;
– des *règles* (métaphores des réactions chimiques) décrivant la transformation d'un sac de symboles en un autre (par exemple : $aab \rightarrow ac$) : ces règles sont placées dans les membranes et expriment la façon dont les symboles présents dans la même membrane peuvent être transformés.

Le calcul, non-déterministe, consiste à appliquer le plus grand nombre possible de règles dont les prémisses sont vérifiées (i.e. pour lesquels tous les symboles sont présents), quitte à choisir aléatoirement lorsque des règles sont en conflit pour la consommation de symboles identiques. Le résultat du calcul est le sac des symboles obtenus lorsque plus aucune règle ne peut s'activer.

Bien évidemment, la facilité à modéliser un phénomène par des P-systèmes décroît très vite avec la granularité des entités à représenter. De plus, les règles, au contraire de « véritables » réactions chimiques, ne disposent pas de constantes cinétiques : il y a donc assez loin de la métaphore originelle au modèle de calcul.

Toutefois, on peut souligner une idée clef de cette approche : les symboles n'ont aucun comportement en tant que tel, seul l'ensemble des règles présentes dans la membrane où se

trouvent ces symboles permet de définir leur fonction. Autrement dit, on a d'une part *séparation entre entités et comportements*, et d'autre part *définition locale des comportements*. Par ailleurs, si la structure membranaire reste arborescente, les extensions récentes des P-systèmes permettent des transformations de cette structure au cours de la simulation (création de membranes, divison, fusion, etc.).

2.4 Caractéristiques souhaitées

L'idée que nous défendons ici est qu'une approche générale de la simulation multi-échelles doit présenter les caractéristiques suivantes :

- La **dynamicité** : un agent doit pouvoir changer de niveau (d'environnement) à volonté ; de plus il ne doit pas y avoir de limitation de principe à la création ou à la dissolution d'un niveau ;
- La **localité** : le comportement d'un agent doit être le résultat de sa présence dans un environnement donné (l'espace qui définit ses conditions d'existence), lui-même associé à une échelle spatiale et à une échelle de temps ;
- L'**uniformité** : toutes les entités doivent avoir une structure logicielle similaire (autrement dit, toute entité pertinente doit être un agent [9]), de même que tous les comportements doivent être décrits à travers le même formalisme, et que la décomposition du monde en environnements doit être homogène ;
- La **généricité** : les connaissances relatives aux entités et à leurs comportements dans chaque environnement, ainsi qu'aux relations entre niveaux, doivent être exprimées dans un formalisme conceptuel et logiciel aussi accessible que possible à des non-informaticiens, puis injectées dans un moteur de simulation générique, et ce pour des domaines d'application aussi variés que possible ;
- La **réalité opérationnelle** : les éléments constitutifs du modèle utilisé lors de la conception doivent être réifiés logiciellement selon une méthode systématique (notamment afin de réduire le risque de biais).

3 Principes formels d'une simulation multi-échelles (PADAWAN)

Le modèle que nous défendons dans cet article, pour répondre aux critères énoncés ci-dessus, repose sur deux principes :

- permettre *l'encapsulation dynamique* d'un environnement par un agent, donc construire les bases d'une structure récursive relativement classique (SWARM, P-systèmes), mais non figée ;
- autoriser un agent à *être situé dans plusieurs environnements* simultanément, sans préjuger de ce que représentent ces environnements (monde physique, groupes, organisations, mémoire spatiale ou sociale, etc.), façon de généraliser les approches de type AGRE/MASQ.

Il s'appuie donc sur deux relations entre agents et environnements : la **situation** et l'**encapsulation**.

Dans la suite, nous désignerons par $\mathcal{E}$ l'ensemble des environnements utilisés dans une simulation, et par $\mathcal{A}$ l'ensemble des agents. Ces environnements doivent être vus comme des *espaces* de topologie quelconque, de dimensions quelconques. Nous allons montrer plus loin comment les doter en outre d'une échelle temporelle propre et de règles comportementales différenciées qui s'appliquent aux agents qu'ils contiennent.

3.1 Situation d'un agent dans un environnement

Nos agents sont *situés*, donc ils appartiennent à au moins un environnement, dans lequel ils peuvent percevoir et agir. Nous autorisons un agent à être situé dans plusieurs environnements, ce qui nous amène à définir la **relation de situation d'un agent dans un environnement** :

Définition 1 *Un agent $a \in \mathcal{A}$* ***est situé*** *dans un environnement $e \in \mathcal{E}$, noté :* $\mathbf{a} \lhd \mathbf{e}$*, lorsque a peut percevoir, être perçu, agir ou subir des actions, dans e.*

On notera dans la suite :

$$\forall a \in \mathcal{A}, \text{location}(a) = \{e \in \mathcal{E} | a \lhd e\}$$

$$\forall e \in \mathcal{E}, \text{content}(e) = \{a \in \mathcal{A} | a \lhd e\}$$

Le fait de travailler avec des agents situés impose : $\nexists a \in \mathcal{A} \mid \text{location}(a) = \emptyset$.

On appellera **agent simplement situé** tout agent $a \in \mathcal{A}$ situé dans un environnement unique. L'ensemble des agents simplement situés sera noté $\mathcal{S}$:

$$\mathcal{S} = \{a \in \mathcal{A} | \exists e \in \mathcal{E}, \text{location}(a) = \{e\}\}$$

De même, on appellera **agent multi-situé** tout agent $a \in \mathcal{A}$ situé dans plusieurs environnements simultanément. L'ensemble des agents-multi-situés sera noté $\mathcal{M}$: $\mathcal{M} = \mathcal{A} \setminus \mathcal{S}$

Les agents simplement situés et les agents-mutisitués sont bien entendu *caractérisés dynamiquement* : il n'y a donc pas de typage préalable et un agent peut changer de statut au cours de la simulation.

Un agent multi-situé peut représenter des situations très diverses : par exemple, il peut s'agir d'un agent jouant un rôle de frontière entre deux environnements physiques (une porte), qui doit donc percevoir ou être perçu par les autres agents de ces environnements. On peut aussi s'en servir pour représenter l'appartenance sociale d'un agent ou sa présence dans des espaces appartenant à des échelles différentes.

3.2 Encapsulation d'un environnement par un agent

La seconde relation du modèle représente la capacité, pour *a priori* n'importe quel agent, d'encapsuler un environnement (dans lequel vont pouvoir se situer d'autres agents). Il est conceptuellement important de maintenir la distinction entre agent (entité) et environnement (espace), mais d'une certaine façon on peut dire que ces agents « jouent le rôle » d'un environnement.

Définition 2 *Un agent $a \in \mathcal{A}$* ***encapsule*** *un environnement $e \in \mathcal{E}$, noté :* $\mathbf{a} \sim \mathbf{e}$*, lorsque a « contient » e. Un agent ne peut encapsuler qu'un seul environnement à la fois, et un environnement n'être encapsulé que par un seul agent.*

Par construction, on pose qu'il existe un environnement unique e^0 non encapsulé par un agent (e^0 correspond à « l'Environnement » unique d'un SMA classique). Cet environnement sera appelé « environnement racine » (ou « environnement zéro ») de la simulation.

On note $\mathcal{E}^\star = \mathcal{E} \setminus \{e^0\}$ l'ensemble des environnements encapsulés par des agents.

On appellera **agent régulier** tout agent n'encapsulant pas d'environnement, et **agent-compartiment** tout agent encapsulant un environnement. L'ensemble des agents-compartiments sera noté $\mathcal{C}$: $\mathcal{C} = \{a \in \mathcal{A} | \exists e \in \mathcal{E}^\star, a \sim e\}$ L'ensemble des agents réguliers sera noté $\mathcal{R}$: $\mathcal{R} = \mathcal{A} \setminus \mathcal{C}$

Par ailleurs on définit :

$$\forall e \in \mathcal{E}^\star, \text{host}(e) = !a \in \mathcal{C} | a \sim e$$

$$\forall a \in \mathcal{C}, \text{space}(a) = !e \in \mathcal{E}^\star | a \sim e$$

3.3 Utilisation conjointe des deux relations

En combinant la situation dans un environnement et l'encapsulation d'un environnement par un agent, on peut définir de nouveaux concepts et les bases d'une architecture multi-échelles.

Ainsi, la simple intersection $\mathcal{M} \cap \mathcal{C}$ regroupe les agents-compartiments qui sont également multi-situés. Cette situation correspond donc à un agent capable d'agir ou de percevoir dans plusieurs environnements, et encapsulant lui-même un environnement. Loin d'être une chimère, cette situation peut se révéler très utile pour la modélisation : par exemple pour représenter une protéine de transport située dans la membrane d'une cellule, avec une extrémité à l'intérieur et une à l'extérieur, et capable de contenir d'autres molécules ; ou encore, un ascenseur peut être vu comme un agent-compartiment qui est situé (accessible) dans plusieurs autres environnements (les étages). Nous appelons **agent-tube** un tel compartiment multi-situé.

Plus généralement, on peut caractériser la structure de la simulation au moyen de nouvelles relations : l'inclusion et l'hébergement.

Définition 3 ***Inclusion d'environnements****. Un environnement e_1 est* ***inclus*** *dans un environnement e_2 (noté $e_1 \subset e_2$) si et seulement si :*

$$\exists a \in \mathcal{C} | a \sim e_1 \wedge a \lhd e_2$$

Définition 4 ***Hébergement d'agents****. Un agent a_1 est* ***hébergé*** *par un agent a_2 (noté $a_1 \sqsubset a_2$) si et seulement si :*

$$\exists e \in \mathcal{E}^\star | a_2 \sim e \wedge a_1 \lhd e$$

Ces deux relations permettent de construire le **graphe d'environnements** (ou **graphe d'inclusion**) d'une simulation, graphe orienté défini par l'ensemble des relations d'inclusion ($\subset$) entre environnements. Ce graphe est dynamique puisqu'il représente les relations effectives de situation et d'encapsulation à chaque instant dans la simulation. Il peut aussi présenter des cycles dans le cas général. De même, le **graphe d'agents** (ou **graphe d'hébergement**) d'une simulation est le graphe orienté défini par l'ensemble des relations d'hébergement ($\sqsubset$) entre

agents. On peut également construire un graphe mixte en identifiant les environnements encapsulés à leur compartiment (cf. fig. 3).

Par extension, on note $e_1 \subseteq e_2$ (inclusion transitive) si et seulement si : $e_1 = e_2$ ou $e_1 \subset e_2$ ou $\exists e \in \mathcal{E} | e_1 \subseteq e \wedge e \subseteq e_2$ (i.e. s'il existe un chemin entre e_1 et e_2 dans le graphe d'inclusion) ; de même $a_1 \sqsubseteq a_2$ (hébergement transitif) si et seulement si : $a_1 = a_2$ ou $a_1 \sqsubset a_2$ ou $\exists a \in \mathcal{A} | a_1 \sqsubseteq a \wedge a \sqsubseteq a_2$.

Dans le cas général, tout environnement n'est pas nécessairement inclus au sens large dans e^0 (il suffit par exemple de définir deux agents tubes qui s'hébergent mutuellement). On montre aisément que, pour que ces situations paradoxales soient évitées, il suffit qu'à tout moment le graphe d'environnements de la simulation *ne comporte pas de cycles* (nous appellerons cette situation une **Simulation Topologiquement Régulière**). Cela revient à dire que la relation $\subseteq$ est une **relation d'ordre**. En conséquence, le graphe d'environnements est un demi-treillis de borne supérieure (de racine) e^0 ($e^0 = \sup_{\subseteq} \mathcal{E}$).

Dès lors, la notion quelque peu qualitative de **niveau** d'un environnement (ou du compartiment associé) peut alors être définie formellement comme la *distance minimale à* e^0, ce qui permet de caractériser une partie de l'organisation d'un système et de ses sous-systèmes.

4 Mise en œuvre dans l'approche orientée interactions

Nous allons montrer ici comment l'approche orientée interactions s'avère particulièrement fructueuse en combinaison avec les relations de situation et d'encapsulation formalisées ci-dessus, tout particulièrement lorsqu'on souhaite qu'un agent manifeste des comportements différents selon l'environnement où il est placé.

L'approche orientée interactions (IODA) qui sous-tend nos travaux [7] se traduit lors de la modélisation (puis au passage à l'implémentation) par les trois points suivants :
- toute entité pertinente est un agent (plus ou moins complexe) possédant des *primitives* de perception et d'action ;
- tout comportement est décrit par une *interaction générique* qui possède une réalité logicielle propre : en l'occurrence ces interactions sont des règles construites sur des primitives, et opérant simultanément sur un agent source (qui peut effectuer l'interaction) et un agent cible (qui peut la subir) ;
- les interactions sont affectées aux agents sources et cibles au moyen d'une *matrice d'interaction*.

Lors de chaque pas de temps de la simulation, le moteur de simulation recherche, pour chaque agent source possible et en fonction des agents voisins, les couples (interaction, cible) réalisables et en effectue un.

4.1 Notion de face d'un agent

Cette séparation que l'approche IODA introduit entre entités (agents) et comportements (interactions) est particulièrement utile lorsque l'on veut pouvoir spécifier des comportements différents selon le niveau où agit un agent. En effet, *il suffit essentiellement d'associer une matrice d'interaction à chaque environnement*. Nous allons expliquer ce mécanisme ci-dessous.

Dans la suite, nous proposons, dans le cas général d'un agent a tel que location$(a) = \{e_1, ..., e_n\}$, de décomposer l'agent a en une partie « centrale » non située, et en un ensemble de parties « périphériques » situées dans chaque environnement e_i. Cette distinction généralise la division esprit/corps de MASQ [13], mais cette terminologie semble suggérer une différence de nature entre ces parties de l'agent.

Nous préférons appeler **« noyau de** a **»** la partie non située (notée $a_{|\bullet}$) et **« face de** a **dans** e_i **»** (notée $a_{|e_i}$) la partie de a située dans e_i (soit en fait a vu comme agent du seul environnement e_i). À $a_{|e_i}$ sont associées des propriétés spécifiques à sa situation, comme une position dans e_i ainsi qu'un halo de perception qui permet de calculer les voisins de a *dans* e_i, i.e. les agents de e_i avec lesquels a peut tenter d'interagir selon la matrice d'interaction M_i de e_i.

Ainsi, on peut voir un agent issu du modèle multi-échelles comme la superposition d'un agent « désincarné » qui ne perçoit ni n'agit (il se limite à un ensemble d'état et à des fonctions de calcul, de décision) et d'un ou plusieurs organes sensori-moteurs qui chacun, pris individuellement dans l'environnement où ils sont situés, peuvent être traités tout à fait comme des agents d'une simulation ordinaire. Le pseudo-agent $a_{|e_i}$ est simplement soumis au principe de sélection d'interaction classique de IODA relativement à l'environnement e_i et à la matrice d'interaction M_i, la seule différence avec une approche « plate » (dans un environnement

unique) étant que les primitives exécutées par $a_{|e_i}$ ont accès à l'état et aux fonctions du pseudo-agent noyau $a_{|\bullet}$.

4.2 Gestion des échelles de temps et ordonnancement de la simulation

L'approche IODA, par défaut, s'applique à des simulations à temps discret, c'est-à-dire dans lesquelles on procède par divisions égales du temps en intervalles de durée δt appelés **pas de temps**. Le temps de la simulation est donc représenté par un intervalle entier $[0, T_{\max}]$; au pas de temps t_i, le temps écoulé depuis le début de la simulation est donc $t_i \delta t$. On note t la valeur du pas de temps courant.

Il est facile, dans ce cadre, d'associer à chaque environnement sa propre échelle de temps.

Définition 5 *On appelle **assignation temporelle** des environnements l'application :*

$$\begin{array}{rcl} \chi : \mathcal{E} & \rightarrow & \mathbb{N}^* \times \mathbb{N} \\ e & \mapsto & (N, \varphi) \end{array}$$

*où N et φ représentent la **période** et la **phase** attachées à l'environnement e.*

La période permet d'indiquer qu'il « se passe quelque chose » dans e tous les $N\delta t$; la phase, que e est « décalé » par rapport au début de la simulation d'un temps $\varphi\delta t$. Rien n'empêche que cette assignation change au cours de la simulation (par exemple pour tenir compte des variations d'activité d'une cellule en fonction de la température) : l'assignation temporelle est alors aussi fonction de t.

On peut alors dire qu'un environnement e est **activé au temps** t lorsque : $t \equiv \varphi (\text{mod } N)$. On désigne par $\mathcal{E}_{\text{act}}(t)$ l'ensemble des environnements activés au temps t.

Les agents susceptibles d'agir au temps t (i.e. agents « actifs » susceptibles d'effectuer une interaction, ou « labiles » susceptibles de changer d'état, cf. [9]) sont donc ceux situés dans les environnements activés au temps t. En pratique, comme chaque agent agit selon la matrice d'interaction de l'environnement où il se trouve, il faut recenser les faces situées dans des environnements activés, soit les couples $(a, e) \in \mathcal{A} \times \mathcal{E}$ activés à l'instant t : on les désigne par $\mathcal{A}_{\text{act}}(t)$.

$$\mathcal{A}_{\text{act}}(t) = \bigcup_{e \in \mathcal{E}_{\text{act}}(t)} \left(\bigcup_{a \in \text{content}(e)} \{(a, e)\} \right)$$

Pour la sélection d'interaction au temps t, les couples de $\mathcal{A}_{\text{act}}(t)$ sont soumis préalablement à une **politique d'ordonnancement** (par défaut : mélange aléatoire) qui va déterminer l'ordre dans lequel les faces correspondantes évaluent et choisissent les interactions à réaliser. On peut envisager diverses politiques d'ordonnancement, comme un tri portant sur les agents, ou sur les environnements, ou sur les deux. Entre autres, on peut mentionner un tri portant sur les environnements selon leur niveau (tri ascendant ou descendant) : ainsi, faire agir les niveaux les plus élevés en premier pourrait donner une approche « émergentiste » ; ou encore, un tri portant sur les environnements selon leur période (tri ascendant ou descendant).

Une fois établi l'ordre de $\mathcal{A}_{\text{act}}(t)$, il reste à opérer une sélection d'interaction classique sur les *faces* correspondantes, i.e. pour chaque couple (a, e) la face $a_{|e}$ est susceptible d'interagir comme source au plus une fois. Quant au noyau $a_{|\bullet}$, il n'effectue ni ne subit par lui-même aucune interaction : seule l'activité de ses faces est susceptible de l'affecter.

4.3 Spécificité des matrices d'interaction liée à l'encapsulation

Interactions hébergeur-hébergé. En général, un agent compartiment doit pouvoir interagir avec les agents qu'il héberge. Pour réaliser cela, il suffit d'ajouter dans la matrice d'interaction une **ligne générique « parent »** : elle sert à spécifier quelles interactions le compartiment peut effectuer avec l'environnement qu'il encapsule et avec les agents qu'il héberge (voir l'exemple §6).

Interaction hébergé-hébergeur. Réciproquement, un agent doit aussi pouvoir interagir avec l'agent compartiment qui l'héberge, ne serait-ce que pour pouvoir en sortir. Il suffit ici d'ajouter à la matrice d'interaction une **colonne générique « parent »** qui spécifie quelles interactions le compartiment peut subir de la part des agents qu'il héberge.

Grâce à ces deux extensions, l'uniformité de IODA est respectée car seule la matrice d'interaction est modifiée, sans aucun changement dans les algorithmes de sélection d'interactions.

4.4 Ajout de primitives spécifiques

Les interactions, telles que définies dans la méthode IODA, sont essentiellement des règles,

pourvues d'une part conditionnelle qui doit être vérifiée pour que l'interaction puisse avoir lieu, et d'une part procédurale qui décrit la séquence d'actions constituant l'interaction proprement dite : toutes deux s'expriment au moyen de *primitives*, i.e. de fonctions ou de procédures exécutées par l'agent source et/ou l'agent cible de l'interaction.

L'introduction des relations de situation et d'encapsulation dans l'approche orientée interactions va de pair avec l'ajout de primitives minimales permettant aux agents de construire des interactions relatives à la manipulation de ce système multi-échelles. Nous allons en donner quelques exemples, en montrant que leur sémantique peut être définie de façon très rigoureuse. Bien évidemment des primitives supplémentaires peuvent être développées selon les besoins spécifiques à chaque domaine d'application.

Primitives de test. En premier lieu, les agents disposent de fonctions booléennes permettant de les caractériser : agent régulier vs. compartiment, agent simplement situé vs. multi-situé, etc. Un agent peut également tester s'il est hébergé par un autre agent. Cela ne présente guère de difficulté.

Primitives d'action. Définir la sémantique des actions réalisables demande un peu de rigueur, notamment parce qu'il faut s'assurer que la définition vaut aussi bien pour le cas des agents multi-situés ou des agents compartiments que pour celui des agents simplement situés ou des agents réguliers. De plus, les primitives doivent garantir à tout moment qu'on reste bien dans le cadre d'une Simulation Topologiquement Régulière.

Ainsi, par exemple, la primitive « entrer dans un compartiment » `enter(a,c)` n'a de sens que si a et c sont situés dans au moins un environnement commun et que $c \not\sqsubseteq a$: dès lors, il faut modifier la situation de a en remplaçant tous ses environnements communs avec c par l'environnement encapsulé par c :
$\text{location}(a) \leftarrow (\text{location}(a) \setminus \text{location}(c)) \cup \{\text{space}(c)\}$

Dans certains cas, la primitive peut être paramétrée par la donnée d'une fonction. Par exemple, la division d'un compartiment c suppose de définir une *politique de répartition* π pour savoir où placer les agents qui étaient hébergés par ce compartiment. Pour exécuter `divide(c)`, on doit donc créer un compartiment c' situé lui-même dans $\text{location}(c)$ puis définir la politique :

$$\pi : \text{content}(\text{space}(c)) \rightarrow \{\text{space}(c), \text{space}(c')\}$$

Dès lors, pour tout agent $a \sqsubset c$, il reste à remplacer dans $\text{location}(a)$, $\text{space}(c)$ par $\pi(a)$ (qui vaut soit $\text{space}(c)$, soit $\text{space}(c')$). La politique par défaut est une répartition aléatoire.

Nous avons de même défini des primitives permettant de détruire (récursivement) un environnement ou un agent, de créer un agent dans des environnements spécifiés, d'entrer et de sortir d'un compartiment, de fusionner des compartiments, de dissoudre un compartiment (en situant les agents qu'il héberge dans les environnements où il était lui-même situé), de transformer un agent régulier en compartiment et vice-versa, de changer d'environnement en « traversant » un agent multisitué (porte), etc. Toutes ces primitives sont actuellement utilisables pour construire des interactions dans JEDI.

5 Implémentation au sein du moteur JEDI

Le modèle général décrit ci-dessus, limité aux relations d'encapsulation et de situation, se traduit initialement par le diagramme de classes simple de la figure 1.

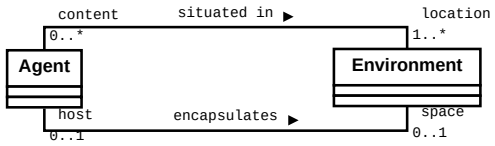

FIGURE 1 – Diagramme des classes montrant les relations introduites entre Agent et Environnement pour gérer les niveaux multiples.

En pratique, pour implémenter cette approche dans le simulateur orientée interactions JEDI (qui met en œuvre fidèlement les concepts de IODA, cf. [7]), nous avons réifié le concept de face des agents comme indiqué sur la figure 2. L'entité qui apparaît comme un « individu » (utilisée pour l'analyse et manipulée par l'utilisateur) dérive de la classe PADAWAN_Agent, mais toute mise en situation d'un agent dans un environnement crée en fait une face (PADAWAN_Face) qui se comporte comme un agent JEDI ordinaire dans son environnement.

L'ordonnanceur de JEDI a également été adapté pour calculer, à chaque pas de temps, les faces situées dans des environnements activables, puis leur appliquer une politique d'ordonnancement,

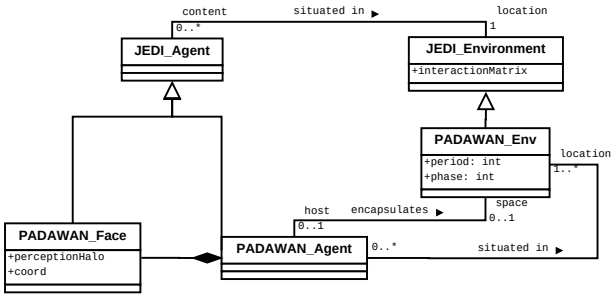

FIGURE 2 – Diagramme des classes montrant l'implémentation effective du modèle multi-échelles dans la méthode IODA (au sein de la plateforme JEDI). Les agents « faces » situés chacun dans un environnement unique sont l'équivalent d'agents JEDI classiques, mais l'entité véritablement utilisée pour l'analyse est leur agrégation autour d'un « noyau ».

et enfin leur faire réaliser la sélection d'interaction relativement à la matrice de leur environnement. Le choix de la politique d'ordonnancement (cf. § 4.2) est un paramètre du moteur JEDI (par défaut, ordre aléatoire).

Au cours d'un pas de temps, chaque face (en tant qu'agent JEDI « classique ») peut interagir au plus une fois comme source, et potentiellement plusieurs fois comme cible (nous rappelons qu'en pratique JEDI permet de *désactiver* un agent à l'issue d'une interaction, pour l'empêcher d'interagir comme source ou comme cible). Un agent PADAWAN possédant des faces dans deux environnements activés est donc susceptible d'être source de deux interactions (une dans chaque environnement).

Il incombe donc *aux interactions* (i.e. au concepteur de la simulation) d'assurer le cas échéant la cohérence des actions effectuées par l'agent, via ses faces, dans les environnements où il se situe. Si par exemple une face exécute une primitive « grossir » et une autre « maigrir », cela veut dire que deux interactions, comportant chacune l'une de ces primitives, étaient réalisables simultanément : selon l'implémentation donnée aux primitives, cela peut être une erreur de conception, mais aussi la fusion de forces antagonistes issues d'environnements différents, ou encore des transformations qui n'affectent que les faces et pas le noyau de l'agent (apparence de l'agent dans chaque environnement par exemple).

6 Exemple de modèle PADAWAN

Nous allons maintenant illustrer à travers un modèle concret (extrait d'un modèle mis en œuvre dans un projet industriel) comment une situation donnée peut être représentée. En l'occurrence, nous avons dans un centre commercial plusieurs magasins, qui peuvent eux-mêmes contenir d'autres magasins (point presse, sandwicherie...). Des clients peuvent naviguer dans l'ensemble du centre commercial à la recherche de leurs articles. Un graphe simplifié montrant les relations d'inclusion d'environnements et d'hébergement d'agents à un instant donné est présenté figure 3.

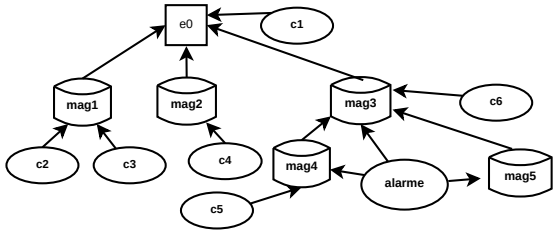

FIGURE 3 – Graphe mixte (inclusion et hébergement superposés) d'une simulation multi-échelle d'un centre commercial. Les clients (agents réguliers, c1 à c6) peuvent circuler à travers les emboîtements de magasins (compartiments mag1 à mag5) ; les magasins 3 à 5 sont en outre pourvu d'une alarme incendie commune (agent régulier multi-situé).

TABLE 1 – Une des matrices d'interaction utilisées. La présence de « Prendre » dans la ligne Client, colonne Article, signifie que l'interaction Prendre peut être effectuée par une instance de Client comme source, avec une instance d'Article comme cible. La colonne ∅ désigne les interactions réflexives (source = cible).

Source \ Cible	∅	*parent*	Client	Magasin	Article	Alarme
parent			Informer	Ouvrir Fermer	Solder Retirer	Arrêter
Client	Explorer	Régler Sortir		Entrer	Prendre	Actionner
Magasin		Verser%	Attirer			
Alarme	Sonner	Alerter				

Pour ce qui est des comportements, ils sont régis ici par deux matrices d'interaction : la première (tab. 1) est commune à e^0 et aux environnements encapsulés dans les magasins 1, 3, et 4 ; l'autre est utilisée dans les magasins 2 et 5 qui sont respectivement un restaurant et un loueur de DVD. Par rapport à une matrice d'interaction classique [7], PADAWAN introduit la ligne *parent* qui exprime les interactions qu'un compartiment (ici, un magasin) peut effectuer sur les agents qu'il héberge, ainsi que la colonne *parent* qui recense celles que les agents peuvent effectuer avec le compartiment qui les héberge (la ligne et la colonne *parent* sont simplement ignorées lorsque cette matrice est utilisée dans e^0). Ainsi, un client peut *Entrer* dans un magasin situé dans le même environnement que lui, et il peut *Sortir* d'un magasin dans lequel il se trouve. Un magasin peut également décider

de l'ouverture ou de la fermeture des magasins qu'il héberge. L'agent multi-situé Alarme peut être actionné dans mag3 ou dans mag4 par les clients : elle alerte alors les magasins où elle se trouve et se met à sonner. En cas de fausse alerte, un des compartiments peut l'arrêter.

La deuxième matrice est identique à la première, à deux exceptions près : on a remplacé Prendre par une interaction Commander (car les clients n'accèdent pas aux articles directement), et on a supprimé l'interaction Actionner. Les clients ne peuvent donc pas mettre en route l'alarme, mais elle sonnera si elle a été actionnée dans mag3 ou mag4. On a donc très facilement modulé les capacités comportementales des clients.

7 Conclusion et perspectives

Le modèle PADAWAN que nous avons présenté ici vise à simplifier la spécification de comportements d'agents situés dans des environnements opérant à des échelles spatiales ou temporelles différentes. Nous pouvons construire, à partir des relations de situation et d'encapsulation entre agents et environnements, un emboîtement dynamique qui ne présuppose pas de typage préalable des agents, ni d'usage particulier (physique, social ou autre) des environnements. Nous avons caractérisé des situations « régulières » qui englobent les SMA habituels. Nous avons également montré comment l'usage d'interactions séparées des agents permet d'exprimer simplement une variabilité comportementale.

Le modèle PADAWAN et son implémentation dans JEDI sont utilisés actuellement pour le développement d'un « Serious Game » (projet FormatStore) destiné à la formation de vendeurs (petits magasins ou supermarchés) ; nous donnerons plus de détails ultérieurement sur le fonctionnement de ce simulateur. Par ailleurs, dans la prolongation de travaux antérieurs [12], nous comptons appliquer également notre approche à la simulation en biologie, qui se prête tout spécialement à l'expérimentation de problématiques multi-échelles. À plus long terme, nous espérons que la convergence avec certaines approches comme AGRE/MASQ, ciblées sur les problèmes organisationnels, pourrait conduire à une élaboration théorique utilisable hors du champ de la simulation, par exemple pour la construction de systèmes ouverts ou participatifs.

Références

[1] G. An. Introduction of an agent-based multi-scale modular architecture for dynamic knowledge representation of acute inflammation. *Theoretical Biology and Medical Modelling*, 5(11), 2008.

[2] S. Bonneaud, P. Redou, D. Thébault, and P. Chevaillier. Multi-modélisation agent orientée patterns. application aux écosystèmes exploités. In Camps and Mathieu [3], pages 119–128.

[3] V. Camps and P. Mathieu, editors. *Actes des 15e Journées Francophones sur les Systèmes Multi-Agents*. Cépaduès éditions, 2007.

[4] D. David, D. Payet, A. Botta, G. Lajoie, S. Manglou, and R. Courdier. Un couplage de dynamiques comportementales : Le modèle DS pour l'aménagement du territoire. In Camps and Mathieu [3], pages 129–138.

[5] J. Ferber, F. Michel, and J. Báez. AGRE : Integrating environments with organizations. In Weyns *et al.*, editor, *Environments for Multi-Agent Systems : First International Workshop (E4MAS'2004). Revised Selected Papers*, volume 3374 of *LNAI*, pages 48–56, Berlin, 2005. Springer-Verlag.

[6] F. Gaillard, Y. Kubera, P. Mathieu, and S. Picault. Une forme de rétro-ingénierie pour systèmes multi-agents. In Zahia Guessoum, editor, *Actes des 17eme Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2009)*, 2009.

[7] Y. Kubera, P. Mathieu, and S. Picault. Interaction-oriented agent simulations : From theory to implementation. In Ghallab *et al.*, editor, *Proc. of the 18th European Conference on Artificial Intelligence (ECAI'08)*, pages 383–387. IOS Press, 2008.

[8] Y. Kubera, P. Mathieu, and S. Picault. How to avoid biases in reactive simulations. In Demazeau *et al.*, editor, *Proc. of the 7th Int. Conf. on Practical Applications of Agents and Multi-Agents Systems (PAAMS'2009)*, volume 55 of *Practical Advances in Intelligent and soft computing*, pages 100–109. Springer, 2009.

[9] Y. Kubera, P. Mathieu, and S. Picault. Everything can be agent ! In van der Hoek *et al.*, editor, *Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'2010)*, 2010.

[10] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The SWARM simulation system : a toolkit for building multi-agent simulations. Working Paper 96-06-042, Santa Fe Institute, Santa Fe, 1996.

[11] G. Păun and G. Rozenberg. A guide to membrane computing. *Theoretical Computer Science*, 287(1) :73–100, 2002.

[12] S. Picault, F. Corellou, C. Schwartz, and F.-Y. Bouget. Simulation multi-agent de réseaux génétiques : les rythmes circadiens d'*Ostreococcus tauri*. In Camps and Mathieu [3], pages 149–158.

[13] T. Stratulat, J. Ferber, and J. Tranier. MASQ – towards an integral approach to interaction. In Decker *et al.*, editor, *Proc. of the 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'09)*, pages 813–820, Budapest, Hungary, 2009.